



Big data techniques for real-time processing of massive data streams

TÉCNICAS BIG DATA PARA EL PROCESAMIENTO DE FLUJOS DE DATOS MASIVOS EN TIEMPO REAL

Author: **Laura Melgar García**

Advisors: Dra. Alicia Troncoso Lora and Dra. Cristina Rubio Escudero

Center of Postgraduate Studies
Pablo de Olavide University

Doctoral Thesis by compendium of publications
International Doctorate Mention
Seville, January 2023

A mi familia y amigos.

Doctoral Thesis supported by the predoctoral scholarship program FPU (*Formación de Profesorado Universitario*) granted by the Spanish Ministry of Universities (FPU19/03488), by the Fulbright predoctoral research scholarship at the New York University, by the Junta de Andalucía and by the research group PAIDI TIC-254: Data Science & Big Data Lab, of the Pablo de Olavide University.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text.

Laura Melgar García

Seville, January 2023

Agradecimientos

Me hace especial ilusión que las primeras líneas de esta Tesis Doctoral vayan dirigidas a mis dos directoras, Alicia Troncoso y Cristina Rubio. Dos directoras que han sido la conjunción perfecta para todas las etapas por las que he pasado en estos cuatro años. Alicia, gracias por tu total disponibilidad y tus consejos siempre acertados. Cristina, gracias por abrirme las puertas de la investigación y enseñarme esta profesión siempre desde el lado más real y sincero.

A mis directoras de tesis me gustaría agradecerles la oportunidad que me han dado para poder descubrir por mí misma que esta es la profesión a la que me gustaría dedicarme al finalizar esta etapa. Ellas son mis referentes. Creo que esa Laura de 10 años a la que le encantaba escribir sus historias y que estudiaba con una pizarra como si estuviese dando clases estaría muy orgullosa de la actual Laura. Tener la posibilidad de escribir mis propias "historias" científicas y dar clases en la Universidad es para mí trabajar en lo que me gusta y creo que no hay mayor privilegio profesional que esto.

También quiero aprovechar para agradecer a mis amigos y familiares la compañía y apoyo durante toda la etapa universitaria, ya que no lo hice en el TFG ni en el TFM. Puede ser que dentro de mí supiese que me quedaba una última oportunidad para hacerlo.

A mis amigos de toda la vida, ¡qué suerte poder decir que conservo mis amistades de cuando era pequeña, algunas desde los 4 años! Esther, María Amores, María Gonsi, Cris, Rocío, Pauli, Salva, Mario, Monti, Si echo la vista atrás siempre os veo. Soy muy afortunada de poder vivir tantas etapas con vosotros al lado.

A mis amigos de Ingeniería de la Salud por hacer que recuerde esa etapa de una forma muy especial y divertida. Reme, Cati, Olga, Marta, Carmen, Eva, Antonio, ..., la de buenos momentos que hemos pasado y, sobre todo, seguimos y seguiremos pasando.

A mis "chanaos" del Erasmus en Milán, Lau, Patxi, Vicent, Marta, Cris y un larguísimo etcétera, gracias por darme el año más divertido de mi vida. El "Once Erasmus, always Erasmus" se cumple y aunque ya no salgamos de martes a sábado, seguimos siendo familia. Gracias a todos los "ESNers" de Milán porque hicisteis que me llevase un regalo

para toda la vida: mi Andre, mi "ESNer preferito".

A mis amigos de París, ¡qué 14 meses más importantes pasé allí! Gracias a Pati por estar siempre conmigo, también a María José, a Paco, a los compañeros del Máster que me ayudaron inmensamente, a los de Aptus Health, sobre todo, a Thomas y a Anabelle gracias por acompañarme y entenderme siempre, y a mis amigos de Disney que hicieron que esa navidad y ese verano fueran mágicos. Gracias por hacer que Paris fuese casa.

A mis amigos de Nueva York de los que he aprendido tantísimo. Gracias a Vai por ser de las personas más buenas que conozco y estar siempre en la puerta de enfrente, a Lucrezia por nuestra complicidad, a mis brasileños que adoro Thales, Jorge, David y Amanda, y a Maryam, Florine, Beatrice, Sandra, Gonzalo, Todos los que hicisteis que viviese muy intensamente el caos de NYC y que esa ciudad sea siempre especial. Muchísimas gracias a la Comisión Fulbright España y a Claudio Silva por apostar por mí.

A mis compañeros y amigos del lab de la UPO, gracias por formar parte de esta etapa. Gracias a Paco por su apoyo constante y sus magníficas ideas, a Pepe porque llegar al lab y que te reciban siempre con una sonrisa vale oro, a Rubén por sus historias tan entretenidas, a Gualberto por su disponibilidad y buenos consejos y a Ángela, Andrés y Adrián que tienen un futuro fantástico por delante. Y, por supuesto, gracias a David porque tus podcasts realistas de WhatsApp me han motivado muchísimo y porque, sin duda, sin ti esta tesis no sería lo que es. También gracias a Manuel, Pedro, Belén, Alberto, José Mari, etc por acogerme tan bien en la US y porque espero que vayamos a comer caracoles muchas más veces.

A mi familia que quiero tanto. A mi yaya porque la mejor herencia que nos ha dado es su forma de ver la vida. Yaya, espero que estés contenta de tener, por fin, una nieta Doctora como tanto has querido siempre, aunque no sea de las que están en el hospital. A mis primos por los momentos tan buenos que pasamos en Monesterio, Chipiona o Marismillas. A mis titos y titas por demostrarme siempre tanto cariño.

A mi padre y a mi madre, por haberme dado siempre libertad y haber confiado en mí más que yo misma. No puedo sentirme más afortunada y agradecida por los padres que tengo y lo sabe todo el mundo. Mamá, gracias por haberme transmitido siempre tu positividad y fuerza y por volcarte siempre en todo lo que tenga que ver con Sandra y conmigo, como si te estuviese pasando a ti misma. Papá, gracias por haberme enseñado a ser siempre la mejor versión de mí misma sin tener que compararme con nadie y porque si alguien lee esta tesis y me hace las mejores preguntas que se pueden hacer, sé que serás tú. Aunque os lo diga muchas veces quiero que quede por aquí escrito: os quiero mucho.

A Sandri, mi hermanita, mi compañera de todo. El regalo más grande que tengo desde que nací y me lo hicieron mis padres. Gracias por ser la mejor hermana mayor que se puede tener. Ya lo sabes, para mí es un orgullo que me confundan contigo. A Antonio, gracias por ser tal cuál eres, el mejor compañero de vida para mi hermana. Y a toda la preciosa familia Loal, por ser siempre tan auténticos. Y por supuesto, Sandri y Antonio, gracias por hacerme tita porque nada me podría hacer más ilusión. Ese niño que viene en camino no sabe la suerte que tiene con sus padres.

A Andre, por estar siempre. Gracias por creer en mí y entenderme en todo momento, por tu forma de ser conmigo, tu forma de querer y de demostrarlo. Sin tu apoyo constante no habría podido escribir esta tesis. Me siento muy afortunada del "nosotros" que estamos creando. *Sei sempre speciale. Ti voglio bene.* También gracias a toda la familia Blanco-Brizzi, a los gigliesi y a los italo-madrileños por acogerme tan bien.

Para mí se cierra una etapa estupenda. Ha sido un privilegio trabajar en la investigación y enseñanza para descubrir cuántas cosas me quedan todavía por aprender y aportar mi granito de arena para que otros puedan aprender. ¡Gracias a todos!

Abstract

Machine learning techniques have become one of the most demanded resources by companies due to the large volume of data that surrounds us in these days. The main objective of these technologies is to solve complex problems in an automated way using data. One of the current perspectives of machine learning is the analysis of continuous flows of data or data streaming. This approach is increasingly requested by enterprises as a result of the large number of information sources producing time-indexed data at high frequency, such as sensors, Internet of Things devices, social networks, etc. However, nowadays, research is more focused on the study of historical data than on data received in streaming. One of the main reasons for this is the enormous challenge that this type of data presents for the modeling of machine learning algorithms.

This Doctoral Thesis is presented in the form of a compendium of publications with a total of 10 scientific contributions in International Conferences and journals with high impact index in the Journal Citation Reports (*JCR*). The research developed during the PhD Program focuses on the study and analysis of real-time or streaming data through the development of new machine learning algorithms. Machine learning algorithms for real-time data consist of a different type of modeling than the traditional one, where the model is updated online to provide accurate responses in the shortest possible time. The main objective of this Doctoral Thesis is the contribution of research value to the scientific community through three new machine learning algorithms. These algorithms are big data techniques and two of them work with online or streaming data. In this way, contributions are made to the development of one of the current trends in Artificial Intelligence.

With this purpose, algorithms are developed for descriptive and predictive tasks, i.e., unsupervised and supervised learning, respectively. Their common idea is the discovery of patterns in the data.

The first technique developed during the dissertation is a triclustering algorithm to produce three-dimensional data clusters in offline or batch mode. This big data algorithm is called *bigTriGen*. In a general way, an evolutionary metaheuristic is used to search for

groups of data with similar patterns. The model uses genetic operators such as selection, crossover, mutation or evaluation operators at each iteration. The goal of the *bigTriGen* is to optimize the evaluation function to achieve triclusters of the highest possible quality. It is used as the basis for the second technique implemented during the Doctoral Thesis.

The second algorithm focuses on the creation of groups over three-dimensional data received in real-time or in streaming. It is called *STriGen*. Streaming modeling is carried out starting from an offline or batch model using historical data. As soon as this model is created, it starts receiving data in real-time. The model is updated in an online or streaming manner to adapt to new streaming patterns. In this way, the *STriGen* is able to detect concept drifts and incorporate them into the model as quickly as possible, thus producing triclusters in real-time and of good quality.

The last algorithm developed in this dissertation follows a supervised learning approach for time series forecasting in real-time. It is called *StreamWNN*. A model is created with historical data based on the k -nearest neighbor or *KNN* algorithm. Once the model is created, data starts to be received in real-time. The algorithm provides real-time predictions of future data, keeping the model always updated in an incremental way and incorporating streaming patterns identified as novelties. The *StreamWNN* also identifies anomalous data in real-time allowing this feature to be used as a security measure during its application.

The developed algorithms have been evaluated with real data from devices and sensors. These new techniques have demonstrated to be very useful, providing meaningful triclusters and accurate predictions in real time.

Resumen

El gran volumen de datos que nos rodean en la actualidad ha derivado en que uno de los recursos más demandados por las empresas sea el uso de técnicas de aprendizaje automatizado o *machine learning*. El objetivo principal de estas tecnologías es resolver problemas complejos de forma automatizada a partir de los datos. Una de las perspectivas actuales del *machine learning* es el análisis de flujos de datos continuos o *data streaming*. Este enfoque está siendo cada vez más solicitado por las empresas debido a la gran cantidad de fuentes de información que producen datos indexados en el tiempo en alta frecuencia, como son los sensores, dispositivos del internet de las cosas, redes sociales, etc. Sin embargo, a día de hoy, la investigación se centra más en el estudio de datos en histórico que en los datos recibidos en *streaming* (a gran velocidad). Una de las principales causas es el enorme reto que presenta esta tipología de datos para el modelado de algoritmos de *machine learning*.

Esta Tesis Doctoral se presenta en la modalidad de compendio de publicaciones aportando un total de 10 contribuciones científicas en Congresos Internacionales y revistas con alto índice de impacto en el *Journal Citation Reports (JCR)*. La investigación desarrollada durante el programa de Doctorado se ha enfocado en el estudio y análisis de datos en tiempo real o en *streaming* mediante el desarrollo de nuevos algoritmos de *machine learning*. Los algoritmos de *machine learning* para los datos en tiempo real constan de un modelado distinto del tradicional, donde destaca la actualización del modelo de forma online para proporcionar respuestas precisas en el menor tiempo posible. El objetivo principal de esta Tesis Doctoral es la aportación de valor a la comunidad científica mediante 3 nuevos algoritmos de *machine learning*. Los tres algoritmos se encuadran en técnicas de big data y dos de ellos trabajan con datos online o en *streaming*. De esta manera, se contribuye al desarrollo de una de las tendencias actuales de la Inteligencia Artificial.

Con este propósito se desarrollan algoritmos para tareas descriptivas y predictivas, es decir, de tipo aprendizaje no supervisado y supervisado respectivamente. La idea común en todos los algoritmos es el descubrimiento de patrones en los datos.

La primera técnica desarrollada durante la Tesis Doctoral es un algoritmo de triclustering para conseguir agrupaciones de datos con tres dimensiones trabajando de forma offline o en *batch*. Este algoritmo de big data se llama *bigTriGen*. De forma general, se buscan grupos de datos con patrones similares usando una metaheurística evolutiva. El modelo usa operadores genéticos como los operadores de selección, cruce, mutación o evaluación en cada iteración. El objetivo del *bigTriGen* es la optimización de la función de evaluación para conseguir triclusters de la mayor calidad posible. El *bigTriGen* se ha usado como base para la segunda técnica implementada durante la Tesis Doctoral.

El segundo algoritmo se enfoca en la creación de grupos sobre datos con tres dimensiones que se reciben en tiempo real o en *streaming*. Se ha llamado *STriGen*. El modelado en *streaming* se lleva a cabo partiendo de un modelo *offline* o en batch que usa datos históricos. Cuando este modelo está creado, se empiezan a recibir datos en tiempo real. El modelo se actualiza de forma *online* o en *streaming* para adaptarse a los nuevos patrones de los datos. De esta manera, se consigue que el *STriGen* detecte cambios de deriva o *concept drift* y los incorpore al modelo lo más rápido posible, produciendo así triclusters en tiempo real y de buena calidad.

El último algoritmo desarrollado en la Tesis Doctoral es de tipo aprendizaje supervisado para predicción de series temporales en tiempo real. Se ha llamado *StreamWNN*. Se parte de un modelo con datos en histórico basado en el algoritmo de los k -vecinos cercanos o *KNN*. Una vez que el modelo está construido, se comienzan a recibir datos en tiempo real. El algoritmo proporciona predicciones de los datos futuros manteniendo el modelo siempre actualizado de forma incremental e incorporando además patrones del *streaming* identificados como novedades. El *StreamWNN* también identifica datos anómalos en tiempo real permitiendo así utilizar esta característica como medida de seguridad para la aplicación en la que se use.

Los algoritmos desarrollados se han evaluado con datos reales procedentes de dispositivos y sensores. Se ha demostrado la gran utilidad de estas nuevas técnicas que proporcionan triclusters significativos y realizan predicciones muy precisas en tiempo real.

Contents

I	Summary of the dissertation	1
1	Introduction	3
1.1	Structure of the dissertation	4
1.2	Research motivation	5
1.3	Research goals	7
1.4	Contributions	8
2	Research context	13
2.1	The streaming paradigm	14
2.1.1	Data streaming requirements	14
2.1.2	Data streaming computational approaches	16
2.2	Streaming pattern evolution discovery	17
2.2.1	Concept drift	17
2.2.2	Novelties in streams	18
2.2.3	Anomalies in streams	18
2.3	Apache Kafka	19
II	Research methodology	21
3	Methodology	23
3.1	Triclustering	24
3.1.1	Problem statement	24
3.1.2	bigTriGen	27
3.1.2.1	Fitness function	29
3.1.2.2	Genetic operators	30
3.1.2.3	Validation of the yielded triclusters	33
3.1.3	STriGen	34
3.1.3.1	Offline phase	35

3.1.3.2	Online phase	38
3.1.3.2.1	Incremental learning	39
3.1.3.2.2	Concept drift	42
3.1.3.3	Validation of the yielded triclusters	43
3.2	Forecasting	45
3.2.1	Problem statement	45
3.2.2	Offline phase	48
3.2.3	Online phase	51
3.2.3.1	Incremental learning	53
3.2.3.2	Novelties and anomalies	56
4	Applications	59
4.1	Triclustering applications to Smart Cities and medicine	60
4.1.1	Precision agriculture	60
4.1.1.1	Datasets description	60
4.1.1.2	Parameter tuning	62
4.1.1.3	Model performance	62
4.1.2	Seismogenic	65
4.1.2.1	Dataset description	65
4.1.2.2	Parameter tuning	66
4.1.2.3	Model performance	66
4.1.3	Environmental sensors	67
4.1.3.1	Dataset description	67
4.1.3.2	Parameter tuning	67
4.1.3.3	Model performance	67
4.1.4	Medical images	68
4.1.4.1	Dataset description	68
4.1.4.2	Parameter tuning	69
4.1.4.3	Model performance	69
4.2	Forecasting applications to energy electricity demand	70
4.2.1	Dataset description	70
4.2.2	Parameter tuning	72
4.2.3	Model performance	73
4.2.3.1	Incremental learning	74
4.2.3.2	Novelties and anomalies	74
4.2.3.3	Scalability and timely results	76

III List of publications 77

5 Publications 79

5.1	Journal and conferences articles	80
5.1.1	"Discovering spatio-temporal patterns in precision agriculture based on triclustering"	80
5.1.2	"High-content screening images streaming analysis using the STriGen methodology"	92
5.1.3	"Coronavirus Optimization Algorithm: A Bioinspired Metaheuristic Based on the COVID-19 Propagation Model"	96
5.1.4	"A new forecasting algorithm based on neighbors for streaming electricity time series"	112
5.1.5	"Discovering three-dimensional patterns in real-time from data streams: an online triclustering approach"	125
5.1.6	"Generating a seismogenic source zone model for the Pyrenees: a GIS-assisted triclustering approach"	146
5.1.7	"Nearest neighbors-based forecasting for electricity demand time series in streaming"	159
5.1.8	"A new big data triclustering approach for extracting three-dimensional patterns in precision agriculture"	171
5.1.9	"Streaming big time series forecasting based on nearest similar patterns with application to energy consumption"	183
5.1.10	"Nearest neighbors with incremental learning for real-time forecasting of electricity demand"	200
5.1.11	"A novel distributed forecasting method based on information fusion and incremental learning for streaming time series"	209
5.1.12	"Identifying novelties and anomalies for incremental learning in streaming time series forecasting"	233

IV Final remarks 257

6 Conclusions and future developments 259

6.1	Conclusions	260
6.2	Conclusiones	263
6.3	Future works	266
6.4	Trabajos futuros	268

Bibliography 273

List of Figures

1.1	Summary of the publications and research goals	11
2.1	Examples of the different possibilities to identify patterns in streaming . .	18
3.1	Representation of triclusters	26
3.2	Overview of the whole methodology of the <i>bigTriGen</i> algorithm.	28
3.3	Kafka architecture outline for <i>STriGen</i> algorithm.	34
3.4	Representation of streams.	38
3.5	Examples of the updating operations during the online phase.	41
3.6	Overview of the <i>STriGen</i> online phase for one tricluster.	44
3.7	Outline of the kafka architecture for the <i>StreamWNN</i> algorithm.	46
3.8	Overview of the whole methodology of the <i>StreamWNN</i> algorithm.	47
3.9	Sliding window for time series of the <i>StreamWNN</i> algorithm	48
4.1	<i>NDVI</i> samples for the studied maize crop	61
4.2	Yielded triclusters by <i>bigTriGen</i> for the maize plantation	63
4.3	Yielded triclusters by <i>bigTriGen</i> for the vineyard crop	64
4.4	Scalability analysis of the <i>bigTriGen</i>	65
4.5	Analysis of the energy electricity demand in Spain	71
4.6	Evolution of mean euclidean distance between neighbors in the online model as iterations pass for $h=144$	74
4.7	Second worst forecast day for no update execution during July 2015	75
4.8	Anomaly identified for $h=48$ during September 11 th 2013	76
4.9	Time versus iterations for each horizon for daily incremental + novelties .	76

List of Tables

4.1	TRIQ values for each tricluster of the maize dataset with <i>NDVI</i> index . .	63
4.2	TRIQ values for each tricluster of the vineyard dataset with <i>MSI</i> index . .	64
4.3	<i>STriGen</i> and baseline comparison for environmental sensors dataset	68
4.4	Performance metrics for each tricluster of the HCS images using <i>STriGen</i>	69
4.5	MAPE error metric (in percentage) for each type of update	73

Part I

Summary of the dissertation

1 | Introduction

THE main objective pursued during the PhD program is to contribute to scientific research in Artificial Intelligence through the implementation of new algorithms. This Chapter introduces the research conducted. Section 1.1 presents the organization of the dissertation document. Section 1.2 describes the motivation for the development of the PhD thesis. Section 1.3 presents the aims to be solved. Section 1.4 presents the scientific publications in which these aims have been addressed.

1.1 | Structure of the dissertation

The dissertation has been organized into four parts, each of which has been structured through chapters.

- **PART I: SUMMARY OF THE DISSERTATION.** This part has been described in Chapters 1 and 2.

In particular, Section 1.2 presents the current situation of the research topic and the main motivations for this dissertation. Section 1.3 defines the research goals. Section 1.4 details the scientific contributions published during the PhD program addressing these objectives.

The research work is contextualized in Sections 2.1, 2.2 and 2.3.

- **PART II: RESEARCH METHODOLOGY.** This part details the three new machine learning methodologies implemented during the PhD program in Chapter 3 and the results obtained from their applications to real datasets in Chapter 4.

On the one hand, Section 3.1 summarizes the two new algorithms developed to produce triclusters. The first one is described in Section 3.1.2 and has been called *bigTriGen* as it is a big data algorithm that creates triclusters based on genetic evolutionary heuristic. The second one is described in Section 3.1.3 and has been called *STriGen* since it is an algorithm to create triclusters based on genetic evolutionary heuristic in streaming or real time. Moreover, Section 3.2 summarizes the new model developed to perform real-time predictions based on the nearest neighbor algorithm. This algorithm has been called *StreamWNN*.

On the other hand, the applications of these algorithms to real datasets are in Sections 4.1 and 4.2. Specifically, the first Section describes the applications of the two new triclustering algorithms and the second one details the application of the new forecasting technique.

- **PART III: PUBLICATIONS.** This part contains the research papers published during the PhD program, detailed in Chapter 5. Publications are organized by date of publication, specifying the type of scientific article.
- **PART IV: FINAL REMARKS.** The last part is addressed in Chapter 6 in which some final conclusions are described, as well as future work to be carried out.

1.2 | Research motivation

Artificial Intelligence or AI was born with the idea of endowing machines with the same intelligent capabilities as human beings. The definition of intelligence in this context has been quite ambiguous for decades, going through different approaches. At present, the definition of intelligence for AI systems is mostly focused on creating systems that act rationally to achieve the best possible results.

Machine learning or ML is a branch of Artificial Intelligence that focuses on learning automatically from previous experiences without having to program tasks specifically. ML methods are based on algorithms that learn from data by looking for patterns to make decisions without human intervention.

For several years, AI and ML have been present in our daily lives in many aspects and fields. Some of them are: medicine, social networks, advertising, recommender systems, pattern recognition, Internet of Things, smart cities, cybersecurity, etc. All these applications are making these technologies become a reality from which a lot of information can be obtained to improve our quality of life.

Machine learning systems learn from data. For this reason, a large part of the efforts prior to the application of a ML algorithm are focused on obtaining a suitable dataset. Currently, a significant amount of our daily data is generated with a temporal component. For example, most electronic devices generate time-indexed data continuously at high speed.

Data streams are data that are received continuously at a high speed in instants of time that do not have to be equispaced. Speed is one of the main characteristics present in data that is receiving more and more attention thanks to applications of great socio-economic impact today within Smart Cities or Industry 4.0. In fact the current technological reality is moving towards the new concept of Fast Data, which aims to reduce the time between the arrival of a data and the extraction of information from it. According to a study by IBM [7], 88% of companies emphasize the need to analyze data in near real-time and that the future of data is in its being fast.

Traditional techniques for working with large volumes of data such as MapReduce and Hadoop with HDFS were born in the batch or offline processing paradigm, when processing was done periodically and in blocks. However, the current trend of working with data arriving in real-time and producing fast responses also affects traditional ML algorithms and models that must be developed following a different approach. Although

the great increase of interest from the industry and companies in data streams is a fact, most of the research today is still focused on working with static or batch data. Thus, our proposal focuses on one of the current trends in high-frequency temporal data such as continuous data flow to provide answers in real-time or in the shortest possible time.

During this dissertation, new machine learning algorithms are developed with online learning for predictive (supervised learning) and descriptive (unsupervised learning) tasks. In both types of learning, emphasis has been placed on detecting and dealing with new patterns in the flow of data received on a continuous basis, such as the concept drift, novelties and anomalies. To validate the proposed methodological developments, the results are oriented to Smart City data in order to contribute with quick solutions in areas such as precision agriculture, environmental sensors or sustainable and efficient energy.

1.3 | Research goals

The common objectives of this dissertation are to learn about and provide value to the scientific community. These large-scale objectives can be subdivided into the following research goals R.G.:

- **R.G. 1:** This objective is divided into two parts.

The first is the theoretical-practical study of the literature related to online learning or continuous data streams. Specifically, the research focuses on studying what data streams are, how to model these data in unsupervised and supervised learning problems and how to implement new algorithms.

The second part deals with the study of other areas of machine learning, besides from data streams, in order to be able to collaborate with other researchers.

- **R.G. 2:** Design and development of a machine learning model named *bigTriGen* to perform triclustering on three-dimensional dataset using big data techniques.

Model verification with precision agriculture and seismic data.

- **R.G. 3:** Design and development of a new algorithm for triclustering in three-dimensional data in real-time with online updating named *STriGen*.

Model verification with continuous data streams from environmental sensors and medical images.

- **R.G. 4:** Design and development of a time series forecasting model that works in real-time with online incremental learning by detecting new patterns called *StreamWNN*.

Model verification with energy electricity demand in Spain in real-time.

1.4 | Contributions

The techniques and results developed during the dissertation have been published in relevant journals and conferences in the area of knowledge.

In this Section, the publications have been sorted by year, including a quality indicator. Research papers published in journals have been rated according to the impact factor (IF) of the Journal Citation Reports *JCR*. Papers published in international conferences have been rated according to the GII-GRIN-SCIE (GGS) conference rating [19].

2020

In the first year of the Ph.D. program, the study of triclustering algorithms and the state of the art of data streams was initiated. As a result, a first version of the *bigTriGen* batch triclustering algorithm for the study of patterns in precision agriculture was published at an international conference [14]. This article received the invitation to the Special Issue published in the year 2022 [17]. A first version of the *STriGen* algorithm for producing triclusters that incrementally adapt to medical images was also published at an international conference [13]. In addition, several collaborations were performed: in an indexed journal [12], in an international conference [8] and in the congress of the Spanish society of gastroenterology, hepatology, nutrition and pediatrics [20].

- **SOCO 2020** [14]: Melgar-García L., Godinho M. T., Espada R., Gutiérrez-Avilés D., Brito I. S., Martínez-Álvarez F., Troncoso A., Rubio-Escudero C. "Discovering spatio-temporal patterns in precision agriculture based on triclustering". *15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020)*. pp. 226-236, 2020. Advances in Intelligent Systems and Computing, Vol. 1268. Springer International Publishing, Cham. doi: 10.1007/978-3-030-57802-2_22

Invitation for Special Issue in Neurocomputing journal.

- **ACM SAC 2020** [13]: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A. "High-content screening images streaming analysis using the STriGen methodology". *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC 2020)*. Association of Computing Machinery, pp. 537-539, 2020. doi: 10.1145/3341105.3374071

GGS class (rating): 2 (A-)

- **Big Data 2020** [12]: Martínez-Álvarez F., Asencio-Cortés G., Torres JF., Gutiérrez-Avilés D., Melgar-García L., Pérez- Chacón R., Rubio-Escudero C., Riquelme J. C., Troncoso A. "Coronavirus Optimization Algorithm: A Bioinspired Metaheuristic Based on the COVID-19 Propagation Model". *Big Data*, Vol. 8 (4), 308-322, 2020. doi: 10.1089/big.2020.0051

IF: 3.644, 15/108 Computer Science, Theory and Methods: Q1.

- **HAIS 2020** [8]: Jiménez-Herrera P., Melgar-García L., Asencio-Cortés G., Troncoso A. "A new forecasting algorithm based on neighbors for streaming electricity time series". *Hybrid Artificial Intelligent Systems (HAIS 2020)*. pp. 522-533, 2020. Lecture Notes in Computer Science, Vol. 12344. Springer International Publishing, Cham. doi: 10.1007/978-3-030-61705-9_43

Invitation for Special Issue in IGPL journal.

- **SEGHNP 2020** [20]: Román E., Rubio C., Melgar L., Ribes C., et. al. "Aplicación Guías ESPGHAN 2012: ¿En qué casos no las hemos aplicado?". *SEGHNP Libros de Trabajos 2020*

Classified as "Distinction work" [12 works of distinction out of the 211 accepted]

2021

In the second year of PhD, the research focused on continuing the development of the triclustering algorithms implemented in 2020 and initiating the development of the real-time prediction algorithm. As a result, the complete *STriGen* algorithm for discovering real-time triclusters was published in an indexed journal [16]. An application of the batch triclustering algorithm for seismic data was also published in another indexed journal [2]. Finally, the first work of the *StreamWNN* algorithm for online time series forecasting was published in a national congress where it won the best paper award in the general category of the congress [15].

- **Information Science 2021** [16]: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A. "Discovering three-dimensional patterns in real-time from data streams: an online triclustering approach". *Information Sciences*, Vol. 558, 174-193, 2021. doi: 10.1016/j.ins.2020.12.089

IF: 5.910 9/156 Computer Science, Information Systems: Q1.

- **Computers and Geosciences 2021** [2]: Amaro-Mellado J. L., Melgar-García L., Rubio-Escudero C., Gutiérrez-Avilés D. "Generating a seismogenic source zone model for the Pyrenees: a GIS-assisted triclustering approach". *Computers and*

Geosciences, Vol. 150, 104736, 2021. doi: 10.1016/j.cageo.2021.104736

IF: 2.991 42/109 Computer Science, Interdisciplinary Applications: Q2.

- **CAEPIA 20/21** [15]: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A. "Nearest neighbors-based forecasting for electricity demand time series in streaming". *XIX Conference of the Spanish Association for Artificial Intelligence (CAEPIA 20/21)*. pp. 185-195, 2021. Lecture Notes in Artificial Intelligence, Vol. 2882 LNAI. Springer International Publishing, Cham. doi: 10.1007/978-3-030-85713-4_18

First prize for the best paper in the general category of the congress.

2022

During the last year of the PhD program the investigation has been divided into two parts. On the one hand, the two Special Issues of the works developed in 2020 have been published. Specifically, the complete *bigTriGen* algorithm with its application to precision agriculture has been published in an indexed journal [17]. In addition, the Special Issue version of the collaboration [8] developed in 2020 has been published in another indexed journal [9]. On the other hand, the incremental learning version of *StreamWNN* has been published in an international conference [18].

- **Neurocomputing 2022** [17]: Melgar-García L., Gutiérrez-Avilés D., Godinho M. T., Espada R., Brito I. S., Martínez-Álvarez F., Troncoso A., Rubio-Escudero C. "A new big data triclustering approach for extracting three-dimensional patterns in precision agriculture". *Neurocomputing*, 500, 268-278, 2022. doi: 10.1016/j.neucom.2021.06.101

IF: 5.719 30/139 Computer Science, Artificial Intelligence: Q1

Special Issue of the SOCO 2020 conference.

- **Logic Journal of the IGPL 2022** [9]: Jiménez-Herrera P., Melgar-García L., Asencio-Cortés G., Troncoso A. "Streaming big time series forecasting based on nearest similar patterns with application to energy consumption". *Logic Journal of the IGPL*, 1367-0751, 2022. doi: 10.1093/jigpal/jzac017

IF:0.931 3/21 Logic: Q1.

Special Issue of the HAIS 2020 conference.

- **ICDM 2022** [18]: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A. "Nearest neighbors with incremental learning for real-time forecasting

of electricity demand". *IEEE International Conference on Data Mining (ICDM 2022)*.

GGs class (rating): 1 (A++)

2023

The research objectives of the *StreamWNN* are completed with two scientific papers already in under review in two indexed journals. In particular, the first article addresses the incremental learning approach in a more extensive way. The model is compared with other forecasting techniques in the literature, demonstrating a better performance in terms of time and accuracy. The second article focuses on the discovery of novelties and anomalies in the continuous streams. The current version of these contributions are in Sections 5.1.11 and 5.1.12.

- **Under review 1:** Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A. "A novel distributed forecasting method based on information fusion and incremental learning for streaming time series".
- **Under review 2:** Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A. "Identifying novelties and anomalies for incremental learning in streaming time series forecasting".

Figure 1.1 summarizes the mentioned publications and their relationship to the research goals detailed in Section 1.3.

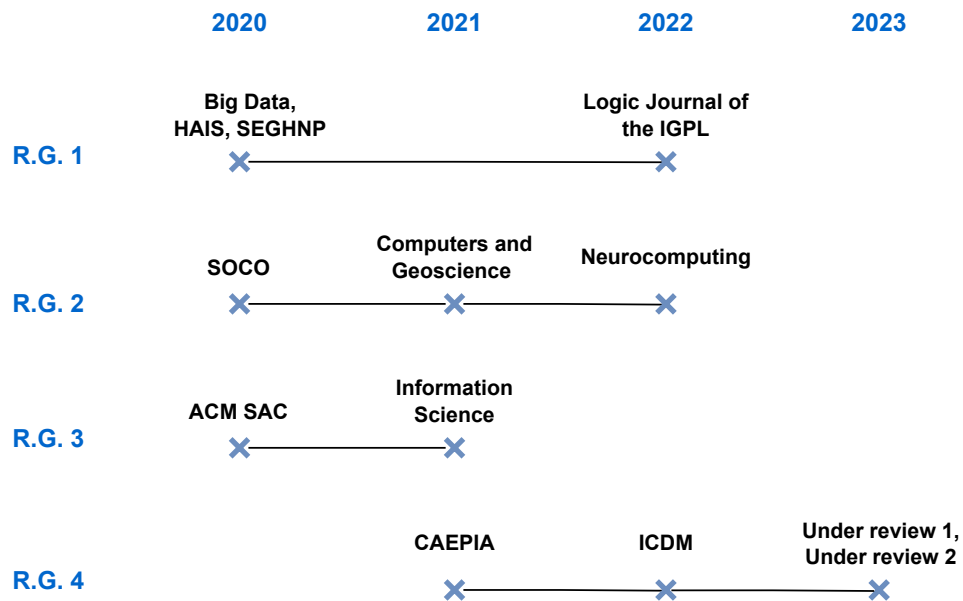


Figure 1.1: Summary of the publications and research goals

2 | Research context

DATA is one of the most relevant components of a machine learning application. Depending on the nature of the data, different answers can be achieved. This Chapter introduces the central concept of the dissertation: continuous data streams. Section 2.1 discusses the problem of continuous data streams indicating its characteristics and approaches for implementation in Sections 2.1.1 and 2.1.2, respectively. Section 2.2 presents a general overview of the types of patterns in continuous data streams. Section 2.3 describes the platform employed to manage data flows.

2.1 | The streaming paradigm

Advances in technology in recent years have resulted in an increasing amount of data being generated and stored. Since much of this data is generated by electronic devices, one of its main characteristics is that it tends to have a very high temporal frequency, giving rise to large volumes of information that are treated as continuous data flows or data streams. These flows can be potentially infinite. Consequently, their storage, processing and treatment are different from that used for static datasets. In addition, this type of data brings the possibility of performing real-time analysis and making quick decisions.

Nowadays, these enormous amounts of incoming data arrive from many different sources, such as Internet of Things (IoT), sensor networks, social media, medical devices or videos. This huge variety of data sources is boosting the interests of the Fast Data environment. In this paradigm, the main objective is to gain insights from the real-time analysis of a set of data. For example, the detection of fraudulent operations in banking transactions must be carried out as quickly as possible. The company Databricks confirmed this assumption by publishing the growing number of streaming jobs from its users. They stated that, in the last three years the number of streaming jobs went from thousands to more than four million per week [4].

In terms of machine learning models, the process of learning from incoming flows of data is different compared to that carried out for static datasets. In batch applications, it is necessary to re-train the entire model to introduce new data into the learning model. This approach is not effective for data streams. For data streams, the machine learning model has to be updated in real-time considering the incoming data. This is one of the requirements for the applications that work in streaming. All the main requisites for data streaming applications are discussed in Section 2.1.1.

2.1.1 | Data streaming requirements

Batch data processing works with available static data. In this case, data can be accessed as many times as necessary and without time or memory limitations. In contrast, streams of data are flows of continuous data that are potentially infinite and can undergo dynamic changes as time passes. In the streaming paradigm, the model has to produce responses right after receiving each instance and has to work with limited memory.

The main requirements that data streaming applications have to guarantee are [11]:

- Stream incoming data: Streaming data must be received in a specific manner, fulfilling different constraints. These constraints are: each data can be received only once, in the order of arrival and following the first-in, first-out scheme. In this way, no time is wasted in storing the data, contributing to a fast response time. Only data that is considered to be essential for the accurate future behavior of the execution will be stored. For example, if data indicating a new pattern in the flow is detected, this data will be stored in a buffer that will be refilled and emptied while monitoring the number of instances at all times.

In order to reduce as much as possible the waiting time between one response and another, methods that work with data streams must have mechanisms to deal with unexpected behavior data. This may be the case of: missing data, erroneous data or delayed data. One of these mechanisms could be, for example, to previously know the average of the batch data of each type. When the course of the program is blocked, it is necessary to wait a short additional time.

Depending on the type of computational approach of the model, data prior to the streaming phase will be used. Such data is called historical, offline or batch. They must not satisfy any of the above conditions. They are used to create summaries or offline model bases for real-time processing.

- Bounded memory: In general, data streams cannot be stored in memory because it is not known when they will stop arriving. In other words, it is possible to have an infinite continuous data flow, in which case it would be nearly infeasible to store all the data due to memory limitations. This condition is related to not wasting time storing the data in memory.

However, it is advisable to checkpoint the model using a secondary model in case of system failures. In this way the integrity of the system can be preserved.

- Low latency: The time interval between the reception of the data and the model response should be as short as possible.

Latency is also reflected in the time each iteration takes to fulfill the rest of the requirements. Therefore, the way of receiving data streams, the process of working with limited memory, the model update if necessary and the model response should be as fast as possible.

- Stream modeling: Regarding the modeling, there are two main conditions to take into account: to get a model that is always adapted to new data patterns and to provide the fastest possible response.

Streaming models are designed to avoid having to be completely retrained every time a new input arrives as this would make it impossible to meet the limited memory and time requirements. These models are continuously updated in a linear fashion according to the number of instances. The models must always be adjusted to the new patterns that arrive, eliminating outdated data. The update must be concrete and fast.

Each time a response must be produced, the model must be updated. The production of the response must also be concrete and fast.

It is essential for the modeling to be scalable. To this purpose, it is recommended to perform the operations in a distributed and multi-threaded manner.

These requirements must be met each time a new data stream arrives.

2.1.2 | Data streaming computational approaches

Traditional machine learning algorithms work with a batch or so-called offline approach, i.e., all data is available and static. Algorithms that work with streaming data follow an online learning approach. The concept of online or streaming learning is usually interchangeable, as used in this dissertation [11].

One computational approach to online learning is the incremental or continual learning. In this type of approach the model evolves by adapting to concept drift or new patterns in the data. Therefore, the model learns and adapts autonomously when new data arrives, mimicking human behavior. These algorithms have to add new information but they also have to deal with catastrophic forgetting, i.e., they also have to maintain previously learned concepts.

Another approach to learn from data streams is the offline/online. In this case, the algorithm usually starts working in offline or batch mode by creating a summary of the data or a base model without having to meet the requirements of the streaming paradigm. Then, the online phase begins and online learning starts. The online phase can include incremental or continual learning to keep the model updated.

2.2 | Streaming pattern evolution discovery

One of the intrinsic characteristics of data streams is that it is common for new patterns to emerge over time. Therefore, the behavior of the incoming data must be monitored and the model must be kept updated. Different types of phenomena can occur: concept drift in existing classes, emergence of new classes or detection of totally different data at a single point in time [1]. These categories correspond to concept drift, novelties and anomalies, defined in Sections 2.2.1, 2.2.2 and 2.2.3, respectively.

Each of these categories must address a different approach to discover the phenomena and deal with them. However, the common concept is that the input data does not contain the identification as concept drift, novelty, anomaly or none of them. Hence, to know whether or not a pattern has been correctly detected in the data, it is necessary to monitor its evolution. This Section defines and differentiates the three categories. The method for identifying and processing them will depend on the learning model selected.

Figure 2.1 illustrates a graphical classification example of the performance of a model over time. Figure 2.1a identifies the initial state of the model where two classes are shown. Figure 2.1b shows the adaptation of the model to concept drift by changing its decision boundary to differentiate the two classes at the following instant time. Figure 2.1c depicts the new class identified as a novelty and the changing of the decision boundary at the following instant time. Figure 2.1d shows the detection of an anomaly in the following instant time.

2.2.1 | Concept drift

Concept drift refers to the opposite assumption of stationary learning. Stationary learning focus on the idea of working with constant data distribution and constant patterns over time. In other words, concept drift describes data distribution changes throughout time. These changes can happen gradually, recurrently, abruptly or incrementally.

Due to concept drift, the model deteriorates producing less accurate responses and becoming outdated. It is very important to adapt the model to current data. The discovery and treatment of concept drift depends on the type of data and the type of algorithm.

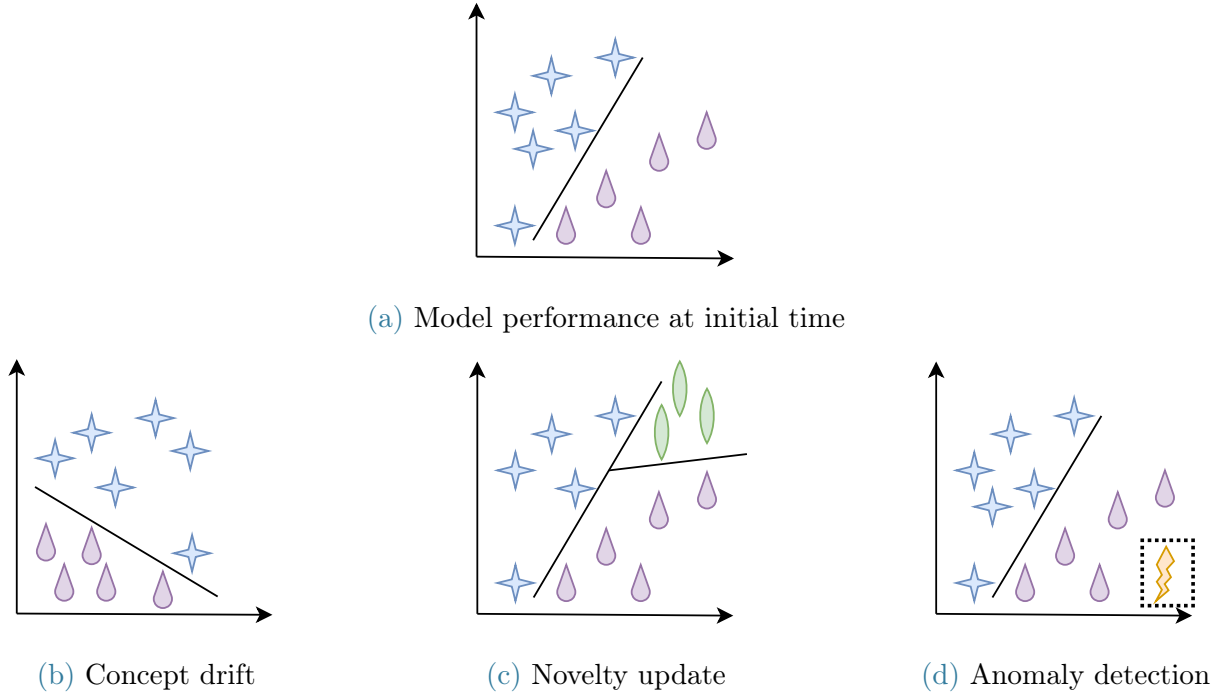


Figure 2.1: Examples of the different possibilities to identify patterns in streaming

2.2.2 | Novelties in streams

Novelties identify patterns in the stream data different from those present during the training phase. They define new classes that capture new behaviors. The main difference of novelties with respect to concept drift is that the first one are new classes and the second one entails a change in the concept associated with the classes present in the training.

When a novelty is discovered, it is important to add it to the model considering that it is a new class. Novelties are patterns that are expected to recur in the data over time. If model adjustment is not carried out, the performance of the model will be degraded when instances belonging to the new class appear.

2.2.3 | Anomalies in streams

Anomalies or outliers in data streams identify instances that occur very distantly and sporadically. Outliers are also unknown data that behave differently from the training data. They identify a totally unexpected pattern. In contrast, when a pattern is identified as a novelty it is expected that streaming instances of novelty class will be identified again.

Generally, anomalies are not included in the model but they trigger an alarm as soon as they are identified.

2.3 | Apache Kafka

Apache Kafka is a distributed open-source platform for managing data streams from various sources to send to users. This platform is scalable, fault tolerant and low latency. Thousands of companies are currently using Apache Kafka to create high-performance pipelines for streaming events [10].

Kafka consists of servers and clients that communicate via the TCP network protocol. On the one hand, Kafka runs as a cluster of one or more servers. The servers in charge of storage are called brokers and the servers for importing and exporting data are called connectors, as Kafka Connect. On the other hand, the clients allow writing applications and microservices for reading, writing and processing streams in a distributed and parallel way.

Apache Kafka follows the general communication procedure: a sender communicates a message to a receiver. Thus, the main terms of Apache Kafka are producers, events or messages and consumers.

Producers are client applications that write messages in Apache Kafka. More specifically, producers are said to publish streaming events. These events capture streams of facts that occur and consist of a key, a value and a date-time. These events are published by producers in topics to which consumers are subscribed. When the consumer detects that there is a new message in the topic to which it is subscribed, it can read and process that event in streaming. An important feature of Kafka that demonstrates its scalability is that producers and consumers are not paired, i.e., producers do not depend on how consumers are acting.

Events are published in topics. Topics are partitioned over a number of buckets in different Kafka brokers to increase the scalability of the application. These events remain in their topics for as long as determined in the Kafka configuration. A topic can have none, one or multiple producers, as well as consumers. Events are always kept in the order in which they were written.

During this dissertation Kafka has been used with the centralized ZooKeeper service to control the flexible and synchronized operation of Kafka's distributed systems.

Part II

Research methodology

3 | Methodology

STREAMING applications follow a different modeling than static or batch machine learning methods. This Chapter describes the three new algorithms that have been developed during this dissertation. Section 3.1 discusses the two new triclustering algorithms implemented. In particular, the algorithm *bigTriGen* for big data triclustering is in Section 3.1.2 and the new triclustering algorithm for streaming data called *STriGen* is in Section 3.1.3. Section 3.2 introduces the new algorithm called *StreamWNN* for time series forecasting in real-time.

3.1 | Triclustering

Two novel triclustering algorithms are presented in this Section. The traditional conceptualization of clustering, biclustering and triclustering can be found in Section 3.1.1. A new big data triclustering methodology named *bigTriGen* is introduced in Section 3.1.2. The second novel triclustering algorithm works in streaming mode. It is called *STriGen* and is described in Section 3.1.3.

3.1.1 | Problem statement

One of the best known machine learning techniques is the grouping of unlabeled data into sets of similar samples. This technique is commonly referred to as clustering. Clustering algorithms assemble data into a specific number of clusters or groups. Depending on the algorithm, the specific number of clusters can be known in advanced or not. However, the common feature of clustering techniques is that the clusters are formed by sets of data with a relationship between their samples, but these relationships are not known to the user. In other words, the resulting groups do not have a related label, they are simply *group*₁, *group*₂, etc.

Traditional clustering algorithms are applied to datasets of one or two dimensions. These dimensions will be referred to as instances (rows in a matrix) and features (columns in a matrix), hereafter. A cluster is a group of instances over all features, i.e., for each yielded cluster, the grouped instances share a behavior pattern across all features. The next Formula represents the cluster k which consists of n instances and all F features of the data or D^F . The $2D$ of the example cluster refers to a two-dimensional dataset.

$$C_k^{2D} = \{(i_1, \dots, i_n), D^F\} \quad D^F = f_{f=1}^F \quad (3.1)$$

Biclustering is an evolution of the clustering algorithm applied to a two-dimensional dataset that provides clusters formed by a set of instances over a set of features. An example of the k produced bicluster with n instances and h features is:

$$B_k^{2D} = \{(i_1, \dots, i_n), (f_1, \dots, f_h)\} \quad (3.2)$$

The definition of the triclustering technique is motivated by the path that many real-world applications are currently following. It is becoming more common for data to

be recorded over time from different sources such as sensors, Internet of Things (IoT) devices or social media, providing time series data. Therefore, in the analysis performed for this kind of data it is very important to take into account time as another dimension of the dataset so that the evolution of the data over time can be considered. In this way, triclustering algorithms are able to create groups of data with similar behaviors over time. Triclustering uses three-dimensional datasets consisting of instances (rows in a matrix), features (columns in a matrix) and times (depth in a matrix). A three-dimensional dataset D^{3D} can be defined as:

$$\begin{aligned} D^{3D} &= \{D^I, D^F, D^{TP}\} \\ D^I &= \{i_1, \dots, i_I\} \\ D^F &= \{f_1, \dots, f_F\} \\ D^{TP} &= \{t_1, \dots, t_{TP}\} \end{aligned} \quad (3.3)$$

which is composed of a total of I instances, F features and TP time points. The instance-feature pair of the i -th instance and the j -th feature can be presented as a sequence of time-indexed values as: $D^{TP}(i, j) = \{v_{t_1}, v_{t_2}, \dots, v_{t_{TP}}\}$.

Therefore, a tricluster is group of instances over a group of features over a group of time points. The tricluster k of a three-dimensional dataset with n instances, h features and y time points is defined as:

$$\begin{aligned} T_k &= \{T^I, T^F, T^{TP}\} \quad \text{with} \quad T^I \subset D^I, T^F \subset D^F, T^{TP} \subset D^{TP} \\ &= \{(i_1, \dots, i_n), (f_1, \dots, f_h), (t_1, \dots, t_y)\} \end{aligned} \quad (3.4)$$

Each tricluster can have a different number of instances, features and time points. The time series of a tricluster have similar behaviors in terms of values (instances-features pairs) or trend. This homogeneity behavior pattern BP is summarized as:

$$BP(T_k(i_X, f_X)) \sim BP(T_k(i_Z, f_Z)) \quad \forall i_X, i_Z \in T^I, \forall f_X, f_Z \in T^F \quad (3.5)$$

A graphical representation of the three-dimensional dataset D^{3D} , its components in slices and a tricluster is in Figure 3.1.

Clustering techniques, including biclustering and triclustering, can be performed following different approaches. One of them is the evolutionary clustering approach, also known as genetic clustering. In the genetic algorithms paradigm, a complete evolutionary process is carried out for each tricluster to be obtained. With these evolutionary meta-heuristics, similar behavior patterns BP are expected to be found. The general

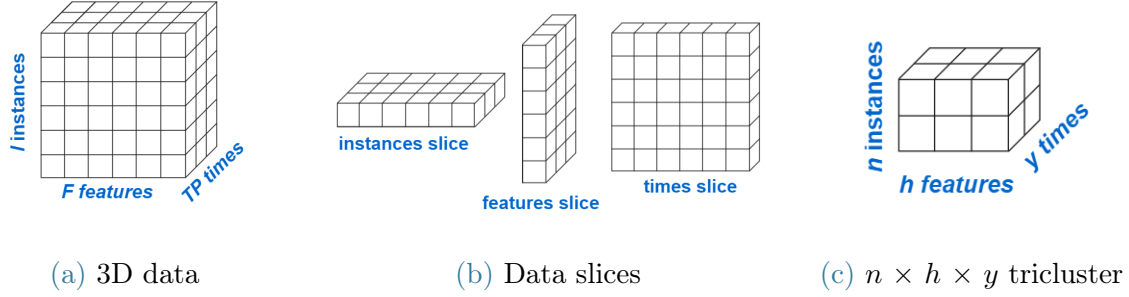


Figure 3.1: Representation of triclusters

methodology of this type of algorithms has two main actors: a population and several individuals. A population contains individuals and an individual is a potential solution of the algorithm. If a genetic triclustering is applied to a three-dimensional dataset, an individual is a tricluster T_k with a set of instances, a set of features and a set of time points.

Evolutionary clustering algorithms evolve the population of individuals by means of genetic operators. The most common genetic operators are: generation of an initial population, selection, crossover, mutation and evaluation. These individuals evolve by optimizing a fitness function. In other words, evolutionary clustering algorithms represent the process of natural selection by choosing the fittest individuals for reproduction. Thus, the offspring of the next generation are the individuals with the highest value of the fitness function.

Clustering techniques follow an unsupervised learning approach. This characteristic makes the validation process of the obtained tricluster complicated, since the expected results are not known in advance. There are several ways to approach the validation process. Some of the most common approaches are: computing a metric to calculate how homogeneous a cluster is, calculating how heterogeneous the samples in different clusters are, or validating the results with an expert in the area.

3.1.2 | bigTriGen

In this Section, a new big data triclustering algorithm is presented. It is based on the genetic algorithm paradigm. These three features give the name *bigTriGen* to the algorithm, i.e., a big data algorithm for triclustering based on genetic meta-heuristics.

The goal of the *bigTriGen* algorithm is to discover a triclustering model $M_{D^{3D}}$ from a three-dimensional dataset D^{3D} . The triclustering model can be described as:

$$M_{D^{3D}} = \{T_1, T_2, \dots, T_N\} \quad (3.6)$$

with N being the total number of triclusters found and each tricluster T_k following the representation in Formula 3.4. Each T_k is a tricluster composed of a set of instances, features and time points that have similar behavior pattern BP . The *bigTriGen* has to meet specific requirements in order to demonstrate that it is a big data algorithm.

The *bigTriGen* evolves a population of individuals using genetic operators over a given number of generations optimizing an evaluation function, also known as fitness function. The algorithm uses genetic operators to obtain the best solutions in each generation. The main parameters of the *bigTriGen* algorithm are:

- N : Number of triclusters to be mined or number of individuals to be discovered.
- G : Number of generations of the genetic evolutionary process.
- In : Size of the initial population.
- Sel : Fraction of the population that promoted to the next generation.
- Mut : Probability of mutation.
- w_{msl} , w_s and w_{ov} : Weights for the MSL , for the tricluster area size and for the overlapping components of the fitness function, respectively. They are defined in detail in Section 3.1.2.1.
- w_{gr} , w_{pe} and w_{sp} : Weights for the GRQ , for the Pearson correlation and for the Spearman correlation components of the $TRIQ$ quality measure, respectively. They are defined in detail in Section 3.1.2.3.

In addition, *bigTriGen* has control parameters for the minimum and maximum number of instances, features and times that can compose the tricluster.

A flow diagram of the whole *bigTriGen* methodology is in Figure 3.2.

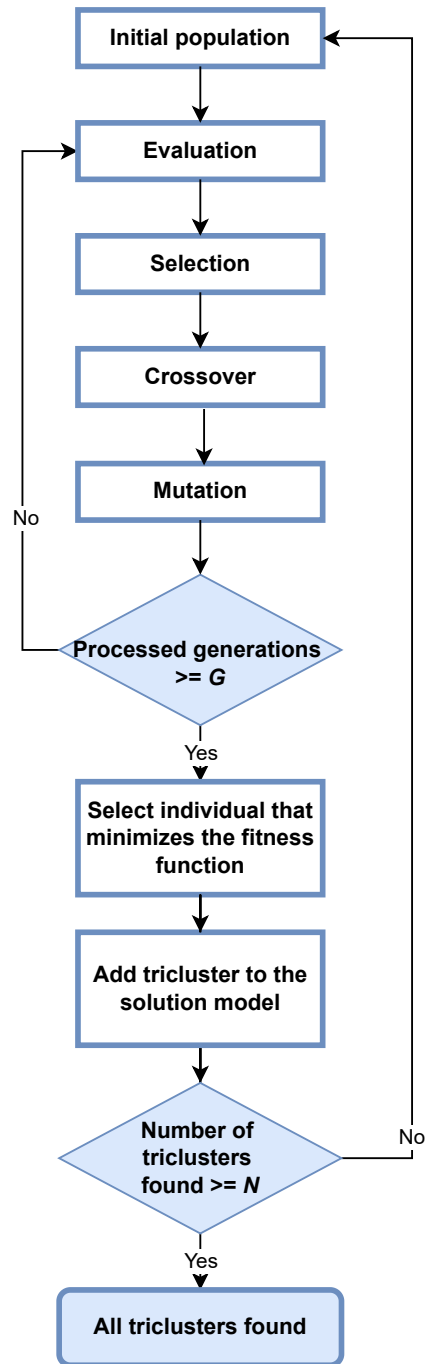


Figure 3.2: Overview of the whole methodology of the *bigTriGen* algorithm.

bigTriGen is developed in Scala 2.12 with Apache Spark 2.3.4. The implementation is based on the Apache Spark DataFrame object to take advantage of this data structure and distribute the application across the nodes of the cluster where it is deployed.

3.1.2.1. Fitness function

In *bigTriGen* the function to optimize is the fitness function. Triclusters with the best fitness function promote since they provide the best possible solutions.

The fitness function is based on the multi slope measure or *MSL* measure. *MSL* measures the resemblances of the slope angles formed by the components of the tricluster at each time point. It is based on the differences between the angles that a series forms with the X-axis every two points (the slope of a straight line). In other words, the *MSL* provides a quantification of how similar the behavior patterns *BP* of a tricluster are.

MSL depends on the multiangular comparisons of the three graphic representations of a tricluster. Henceforth, AC_{multi} defines the multiangular comparison term and TRI_{xop} defines the graphic representation of a tricluster with x , o and p being either instances I , features F or time points T . Therefore, the three graphic representations of a tricluster are: TRI_{ift} , TRI_{itf} and TRI_{tif} . In particular, a graphic representation TRI_{xop} analyzes the evolution of the expression levels of x with o as the outlines by creating one panel for each p . For example, TRI_{itf} creates a panel for each feature and in each panel times are on the X-axis with each corresponding instance as the outline. The $AC_{multi}(TRI_{xop})$ considers the average of the differences of angles vectors formed by x and o for each panel p . The *MSL* measure of a tricluster T_k is defined as the average of its three graphic representations as defined in the following Formula. The complete description of the *MSL* is in [5].

$$MSL(T_k) = \frac{1}{3} \left(AC_{multi}(TRI_{ift}) + AC_{multi}(TRI_{itf}) + AC_{multi}(TRI_{tif}) \right) \quad (3.7)$$

The fitness function depends not only on the *MSL* measure but also on a control mechanism to balance the size of the triclusters and the overlap between them. The fitness function of a tricluster T_k is defined by:

$$Fitness_Function(T) = \frac{w_{msl} * \left(MSL(T_k) / (2\pi) \right) + w_s * S(T^I, T^F) + w_{ov} * OV(T_k, M_{D^{3D}})}{w_{msl} * w_s * w_{ov}} \quad (3.8)$$

where $MSL(T_k)$ is the *MSL* measure of the tricluster, $S(T^I, T^F)$ is the size of the area demarcated by the instances and features of the tricluster and $OV(T_k, M_{D^{3D}})$ is the degree of overlap of the tricluster with the remaining triclusters in the model $M_{D^{3D}}$. The rest of the parameters represent the weights given by the user prior to the execution of the

algorithm: w_{msl} , w_s and w_{ov} , for MSL , size of the area demarcated by the instances and features of the tricluster and overlap respectively. A default setting of these weights is to fix w_{msl} at 0.8 and distribute 0.2 between w_s and w_{ov} .

3.1.2.2. Genetic operators

The *bigTriGen* uses genetic operators to evolve the individuals. These genetic operators are the following:

Initial population The algorithm starts by creating In individuals. These individuals are subsets of instances, features and time points that are randomly selected from the dataset.

Depending on the application of the *bigTriGen*, it is of interest or not to create a contiguous subspace of the data in a tricluster. For example, if the algorithm is applied on a bioinformatic problem to discover genes and conditions that have a homogeneous pattern, it would make sense that the initial population operator could create individuals with instances and features that are not contiguous. However, if the *bigTriGen* is applied to a field where efficient precision agriculture is under consideration, this genetic operator must provide individuals that have a set of contiguous instances, a set of contiguous features and a set of contiguous time points. In the case of precision agriculture for a field, the algorithm performs this type of approach since the farmer cannot provide a specific treatment for each point in the field, but for some contiguous meters in the field. This approach is called squared initial population.

The initial population operates differently when searching for the first solution tricluster than for the rest of the $N - 1$ solution triclusters. In the first one, individuals are generated completely randomly. For the remaining solutions, each individual from the initial random population is checked and if an individual has more than 20% of its components in the already found solution triclusters, this individual will be randomly generated again until there is no overlap. The following conditional Formula represents the two ways of creating the initial population $initial_pop_k$ of the k tricluster T_k :

$$initial_pop_k = \begin{cases} random_population_k, & \text{if } k = 1 \\ \text{control } \{sols_T \cap random_population_k\} & \text{otherwise} \end{cases} \quad (3.9)$$

where $random_population_k$ refers to the complete random selection of instances, features and time points for the individuals of the $initial_pop_k$, $sols_T$ are the solution triclusters already found and $control$ refers to the process of checking for no more than 20% of overlapping components.

Evaluation The evaluation operator is implemented by means of the fitness function. This operator is very important when promoting the best individuals to the next generation, as well as for selecting the final solution triclusters.

The *bigTriGen* fitness function is described in Formula 3.8.

Selection A tournament selection algorithm is chosen for this genetic operator. This operator separates the individuals randomly into three groups. Each group has its individuals sorted by fitness.

The individuals that promote from this selection operation to the next generation are called parents. The number of parents to be selected from one generation to the next one is defined by the control parameter Sel , which is a percentage value over the total number of individuals In .

$$num_parents = \lfloor In \times Sel \rfloor \quad (3.10)$$

Therefore, the algorithm selects from each of the three groups created by the tournament selection operator the individuals with the highest fitness function. Specifically, the number of individuals to be chosen from each group is one third of the total number of parents or $num_parents$.

Crossover This operator of the *bigTriGen* creates new individuals from the crossover of the parents. The individuals that have been promoted from the selection operator are called parents. Specifically, two parents create two new mixed individuals called offspring or children.

The first child is composed of the instances of the first parent, the features of the second parent and the time points of the first parent. The second child is created in the opposite way. Therefore, considering P_1 and P_2 as the parents with their corresponding components and CH_1 and CH_2 as the children, the following Formula describes the implemented crossover operator:

$$\begin{aligned} P_1 &= \{P_1^I, P_1^F, P_1^{TP}\} \quad \text{and} \quad P_2 = \{P_2^I, P_2^F, P_2^{TP}\} \\ CH_1 &= \{P_1^I, P_2^F, P_1^{TP}\} \quad \text{and} \quad CH_2 = \{P_2^I, P_1^F, P_2^{TP}\} \end{aligned} \quad (3.11)$$

The number of crossovers $num_crossover$ to be performed is established in the following Formula:

$$\begin{aligned} num_children &= In - \lfloor In \times Sel \rfloor = In - num_parents \\ num_crossovers &= \lfloor num_children/2 \rfloor + (num_children \bmod 2) \end{aligned} \quad (3.12)$$

Once all $num_crossovers$ crossovers are computed, children are sorted by best fitness function and $num_children$ are selected. At this point, the population consists of In individuals: $num_parents$ promoted from the selection operator and $num_children$ from the crossover operator.

Mutation The individuals selected from the crossover operator, known as children, are eligible to be mutated. The selection or not of a child to be mutated is controlled by the percentage parameter Mut . Each child receives a random percentage value $probability_mut$. Children with $probability_mut$ less than Mut are mutated.

Mutation is performed by adding or removing a random instance, feature or time point of the child. One of these six operations is computed. The operation is chosen randomly. This alteration of the children is only performed if the maximum and minimum number of instances, features and time points of the individual can be conserved.

Depending on the purpose of the application of the algorithm, i.e, working with contiguous or non-contiguous data, the mutation operator can also change an existing instance, feature or time point to a random one, in that way, three more operations can be performed.

The mutation operator returns the offspring that have not been mutated and the children that have been mutated. The goal of this operator is to guarantee variability for the following generations.

The individuals that promote to the next generation are the parents, that are the chosen individuals from the selection operator, and the mutated and not mutated children, that are the children obtained after the mutation operator.

3.1.2.3. Validation of the yielded triclusters

When working with real datasets, the ground truth of the solutions found by the *bigTriGen* is not known. A common procedure to evaluate the results of this type of unsupervised learning algorithms is to measure the similarity of the data in a tricluster. The *TRIQ* measure is the triclustering quality measure selected for the *bigTriGen*. This measure is extensively discussed in [6].

TRIQ provides a quality measure that considers the similarity of the triclusters patterns using a weighted normalization of the *MSL* angle value, denoted $GRQ(T_k)$. It is a quality measure of the graphic representation of a tricluster T_k using the multi slope measure *MSL* (Formula 3.7):

$$GRQ(T_k) = 1 - \frac{MSL(T_k)}{2\pi} \quad (3.13)$$

In addition, *TRIQ* takes into account the level of correlation of the time series of the tricluster using weighted Pearson and Spearman correlation values. The *TRIQ* measure of the tricluster T_k is described in the following Formula:

$$TRIQ(T_k) = \frac{w_{gr} * GRQ(T_k) + w_{pe} * PEQ(T_k) + w_{sp} * SPQ(T_k)}{w_{gr} + w_{pe} + w_{sp}} \quad (3.14)$$

where $GRQ(T_k)$ is the measure corresponding to the graphical representation of T_k and $PEQ(T_k)$ and $SPQ(T_k)$ are the Pearson and Spearman correlation quality values of T_k , respectively. The weighted parameters w_{gr} , w_{pe} and w_{sp} refer to the *GRQ*, Pearson correlation and Spearman correlation. These parameters are typically configured with 0.4 for the w_{gr} and the remaining 0.6 distributed between the w_{pe} and w_{sp} .

Another way to validate the yielded triclusters produced from the *bigTriGen* for real-world datasets is to perform a visual analysis of the patterns discovered over time. Pattern behavior analysis should be performed by an expert in the type of data. Depending on the type of data, other analyses can be carried out. For example, when working with agricultural crops fields, it is very useful for the expert to create relationships between the tricluster patterns, their specific area in the crop and the treatments that were applied to it.

3.1.3 | STriGen

In this Section, a new online triclustering algorithm is presented. The algorithm is called *STriGen* where the 'S' stands for the streaming or online mode, 'Tri' for triclustering and 'Gen' for its based genetic evolutionary heuristic. The objective of *STriGen* is to find groups of similar behavioral patterns in three-dimensional streaming datasets.

The *STriGen* addresses the streaming mode using the offline-online computational approach. During the first phase, the algorithm creates a sketch model or summary model of the historical data, without having to satisfy any of the streaming requirements. The second phase is an online phase that starts whenever the offline model is computed. During the online phase, flows of new data are continuously arriving to the model. These flows of new data are called data streams. The triclusters of the offline model evolve during the online phase keeping the model always updated and identifying and processing outdated data effects, called concept drift.

The *STriGen* is built on Apache Spark 2.3.4 with HDFS file system on Hadoop 2.7.7. The algorithm uses Apache Kafka 2.11 as streaming platform with a single pipeline and the Kappa Architecture. An overview of the workflow of the offline-online phase of the algorithm is illustrated in Figure 3.3. Once the offline phase is finished, kafka producers start receiving online data. Producers publish the online data in the kafka topic where a queue of data is created. The kafka consumers process the data in the queue using a first-in first-out basis. Consumers feed the streams to the online phase of the *STriGen*. Accurate real-time triclusters are obtained thanks to the developed online phase which updates the current model. When there are no more data streams, producers publish a specific message so that kafka consumers can understand the algorithm has to finish.

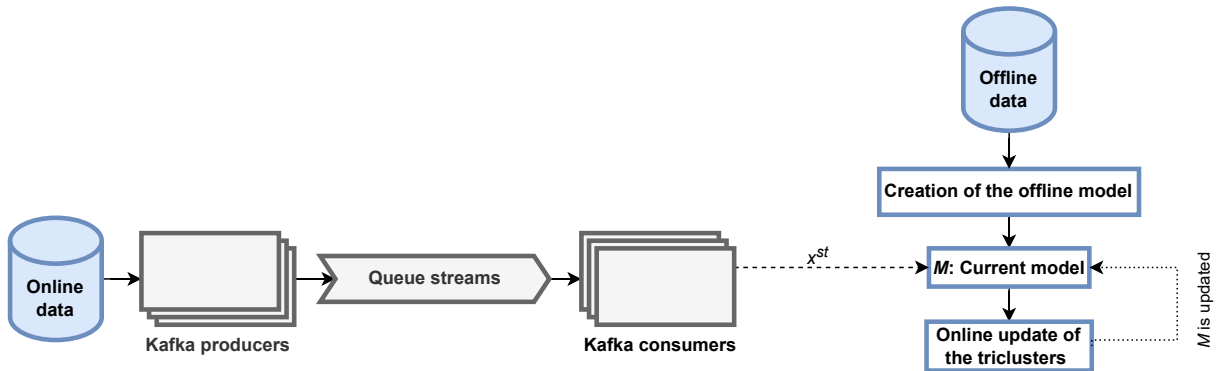


Figure 3.3: Kafka architecture outline for *STriGen* algorithm.

The two phases of the *STriGen* algorithm are described in Section 3.1.3.1 and 3.1.3.2.

3.1.3.1. Offline phase

The offline phase of the *STriGen* processes data in static or batch mode without meeting the requirements of streaming algorithms. The offline phase follows the same methodology of the *bigTriGen* algorithm, described in Section 3.1.2. Therefore, *STriGen* takes into account the same control parameters, the same main actors (individuals and population) and the same fitness function formula as *bigTriGen*.

The objective of the offline phase of the *STriGen* is to create a summary model of the components of each mined tricluster. The natural selection process is implemented by selecting for reproduction the individuals with the best fitness functions. The fitness function uses the multi slope measure *MSL* defined in Formula 3.7. In addition, the fitness function controls the size of the triclusters with respect to their instances, features and time points, and the overlap of the triclusters. The complete description of the fitness function can be found in Formula 3.8.

The genetic operators used to evolve the individuals of a population are: creation of an initial population, evaluation, selection, crossover and mutation:

Initial population The *STriGen* algorithm starts by creating an initial population of In individuals that are subsets of random instances, features and time points.

The creation of the individuals in the population is completely random for the first generation. For the rest of generations, this offline phase creates a percentage of the individuals randomly and the others are created taking into account the unexplored areas of the previous solution triclusters.

This control of the overlapping individuals is slightly different from the one of the *bigTriGen*. This minor modification causes the *STriGen* to include one more control parameter called *Ale*. *Ale* is the percentage of individuals In that have to be randomly created for the rest of the $N - 1$ generations. Therefore, $100 - Ale$ is the percentage of individuals to be created from the unexplored areas in the solutions already found.

Evaluation Once a population is created, it is evaluated to measure the quality of each of individual. The evaluation operator is the same as the one of *bigTriGen* and uses the same fitness function defined in Formula 3.8.

Selection The selection operator is the tournament selection with the same methodology as the one described for the *bigTriGen*. In summary, individuals are divided

randomly into three groups. The tournament selection promotes one third of $In \times Sel$ individuals from each group, considering Sel a fraction of the population. The promoted individuals are called parents and are those with the best fitness function.

Crossover The crossover operator of the *STriGen* includes a different approach from that of the *bigTriGen*. It is implemented by means of a random one-point crossover technique. This technique mixes the instances and features of the two selected individuals or parents to create two new individuals or children. This mixing occurs only at a specific index location that depends on a new percentage control parameter *Cross*. The one-point crossover with the index location as *loc* is represented as:

$$\begin{aligned}
 P_1 &= \{P_1^I, P_1^F, P_1^{TP}\} \quad \text{and} \quad P_2 = \{P_2^I, P_2^F, P_2^{TP}\} \\
 CH_1 &= \{(i_{1_P1}, \dots, i_{loc_P1}, i_{loc+1_P2}, \dots, i_{n_P2}), \\
 &\quad (f_{1_P1}, \dots, f_{loc_P1}, f_{loc+1_P2}, \dots, f_{h_P2}), P_1^{TP}\} \\
 &\quad \text{and} \\
 CH_2 &= \{(i_{1_P2}, \dots, i_{loc_P2}, i_{loc+1_P1}, \dots, i_{n_P1}), \\
 &\quad (f_{1_P2}, \dots, f_{loc_P2}, f_{loc+1_P1}, \dots, f_{h_P1}), P_2^{TP}\}
 \end{aligned} \tag{3.15}$$

with P_1 and P_2 referring to the parents and CH_1 and CH_2 to the crossed individuals or children. Each parent can have a different number of instances and features, but the total number of samples is represented as n instances and h features. For example, i_{loc_P1} is the instance at index *loc* of P_1 and f_{h_P1} is the last feature of the first parent. Time points are not crossed, as the sketch model has to provide a tricluster solution with contiguous data in time to evolve it in real-time.

Mutation The mutation operator is performed by inserting, removing or changing instances, features or time points, following the same methodology as the one described for the *bigTriGen*.

Algorithm 3.1 represents the methodology of the offline phase considering the control parameters N and G defined by the user where N is the number of triclusters to be mined and G the number of generations of the evolutionary process.

Algorithm 3.1 Offline phase

```

1: for  $k = 1$  to  $N$  do
2:   Initialize population
3:   Evaluate population
4:   for  $g = 1$  to  $G$  do
5:     Select individuals
6:     Crossover individuals
7:     Mutation individuals
8:     Evaluate new population
9:      $g \leftarrow g+1$ 
10:  end for
11:  Select individual that minimizes the fitness function
12:   $T_k$  is made up of the components of the selected individual
13:   $k \leftarrow k+1$ 
14: end for
15: return Summary model  $M$  with triclusters components

```

In general, the offline phase of the *STriGen* is very similar to the *bigTriGen* methodology. However, two main differences should be noted. On the one hand, their aims are different. The aim of the *STriGen* is to provide an accurate sketch or summary model to build a good insight for the online or streaming phase of the algorithm. The *bigTriGen* is a complete algorithm by itself and, therefore, aims to generate the best triclustering model for its data. On the other hand, triclusters' time points in the sketch model of the *STriGen* offline phase have to be continuous. In addition, they have to take into account the sliding window parameter described in the next Section 3.1.3.2. This parameter focuses on keeping the triclusters with the most recent w samples, generating up to date triclusters.

3.1.3.2. Online phase

The online phase starts when the offline summary model is computed. From this time onward, data streams may start to arrive. A stream data is a set of I instances and F features for a single time point, considering I and F as the same values of the offline dataset D^{3D} (Formula 3.3). The stream sample at time z is st_z and consists of I instances at time z and F instances at time z . Therefore, a stream st_z can be defined as:

$$st_z = \{(i_1, \dots, i_I)_z, (f_1, \dots, f_F)_z\} \quad (3.16)$$

As streams can be infinite, the model uses a sliding window of w data that defines the w most recent streams samples to consider. w is an integer number between 2 and the total number of streams at the current moment z . Figure 3.4 depicts the case where w is 3. In this illustration, only the 3 most recent streaming samples are taken into account at time z .

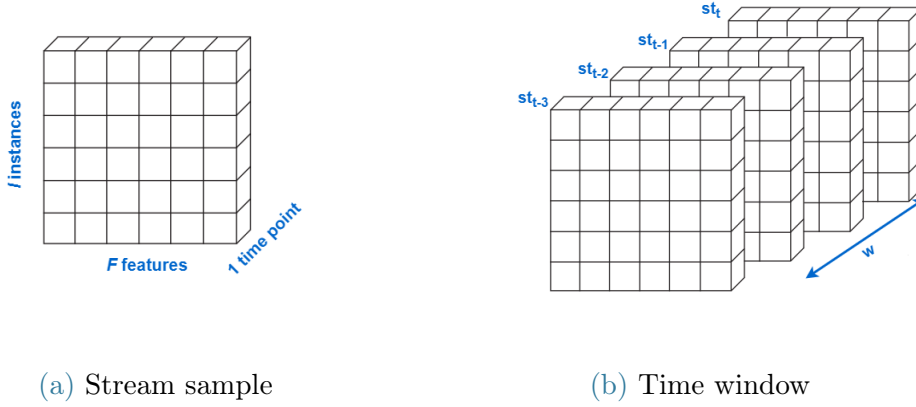


Figure 3.4: Representation of streams.

Considering the sliding window, each time a new stream arrives at the time instant z , the first task of the online phase is to remove all triclusters components that include samples up to the instant point $z - w - 1$. This is a very useful and common procedure for machine learning techniques in streaming mode. Thus, the tricluster contains the most recent data at any time.

In order to provide good quality triclusters in real time, triclusters have to be updated quickly during the online phase. The following two sections are presented to achieve this.

The first one performs the incremental update of the model. The second one describes the procedure for detecting a concept drift and the update of the involved tricluster components. In this way, the model is always formed by the triclusters adapted to the most recent w streams without recomputing a whole batch algorithm.

The quality of each tricluster is evaluated in real-time to have a measurement of its behavior pattern evolution. This evaluation is calculated using the graphical quality measure GRQ of a tricluster, defined in Formula 3.13. The GRQ is part of the $TRIQ$ validation measure. A tricluster has higher graph quality as smaller the MSL value is, which is minimized by the fitness function. Therefore, the $GRQ(T_k)$ metric of the tricluster T_k can be a value between 0 and 1, where 1 identifies a T_k with the highest graph quality, which is the target to be pursued in this algorithm. This quality metric helps to identify whether a tricluster needs to include new concepts, i.e., instances or features that can identify a concept drift, or not. In the case that this happens, previous samples could be removed from the tricluster to add those that are more recent and provide more quality information.

3.1.3.2.1. Incremental learning

The incremental learning approach of the *STriGen* makes the model evolve and include knowledge from the new stream sample considering, at most, the previous z streams.

Four different scenarios are performed and evaluated with the corresponding GRQ value of the new tricluster. The model selects the option with the highest GRQ value that corresponds to the best graphical quality of each evolved tricluster. These updates help the model to discover new components that may be of interest to the tricluster.

The first two possible updates of the incremental learning approach are related to the already existing components in the tricluster T_k for the sliding window w . The other two possible updates include the mutation operator that allows the tricluster to evolve by deleting, adding or changing instances or features.

The first option to update the model refers to the case in which the tricluster has as its most recent time instant the point $z - 1$, it is: T_k includes samples from the previous stream. In this case, the update is performed including for the new stream at time z the same existing instances and features in T_k , considering the window of streams w . This update is illustrated in Figure 3.5a.

The second possible update is a particular case of the first one. This update is

performed whenever the tricluster does not include the instant $z - 1$ in its components. Two options are carried out: either by adding only the next instant point of T_k or by adding all the instant points until z . Figure 3.5b shows an example of this update where the sliding window is 5, so at most 5 previous streams can be included in the tricluster components. In the first option, the tricluster contains only three time points, i.e., $z - 4$ to $z - 2$. The second option includes all possible w time points, i.e., $z - 4$ to z , as did the first possible update depicted in Figure 3.5a.

The third and fourth possible updates execute a random mutation operator. This operator is performed to avoid obtaining a local optimum instead of the desired global optimum, due to the fact that the offline phase is a *NP* hard problem and so the first two updates might not be enough. The mutation operator is applied on the T_k obtained from the above-mentioned updates. The genetic operator deletes, changes or adds instances or features to T_k randomly. The *STriGen* firstly chooses the type of mutation randomly: delete an existing sample, change an existing sample to a new sample or add a new sample. The called sample can be an instance or a feature. The second random choice of the *STriGen* consists of randomly selecting the specific instance or feature in which the mutation is performed. It is important to take into account the following requirements:

- If the deleting option on the feature f_z is randomly selected, the $z - th$ feature is removed from all the instances and time points of the existing tricluster T_k .
- If the adding option on the feature f_z that is not in the tricluster is randomly selected, the $z - th$ feature is added for all instances and time points of the existing tricluster T_k .
- For the changing option, the delete option is executed first and then, the add one.

The methodology of instance mutation is the same as that of the features but considering the instance i_z . These two options are illustrated in Figures 3.5c and 3.5d for feature mutation and instance mutation respectively.

In summary, if the last point in the previous tricluster is $z - 1$, there will be 2 triclusters to evaluate. The first one will be the result of option number 1 and the second one will be the one resulting from the mutation applied on it. However, if the last point in the previous tricluster is not $z - 1$, there will be 4 triclusters to evaluate. These will be the 2 resulting from update number 2 and the mutation of each of them.

The incremental learning approach of the stream z ends up evaluating the possible triclusters with the *GRQ* measure. The tricluster that maximizes *GRQ* is selected and the model is updated with its components. This process is performed for each of the N

triclusters to find.

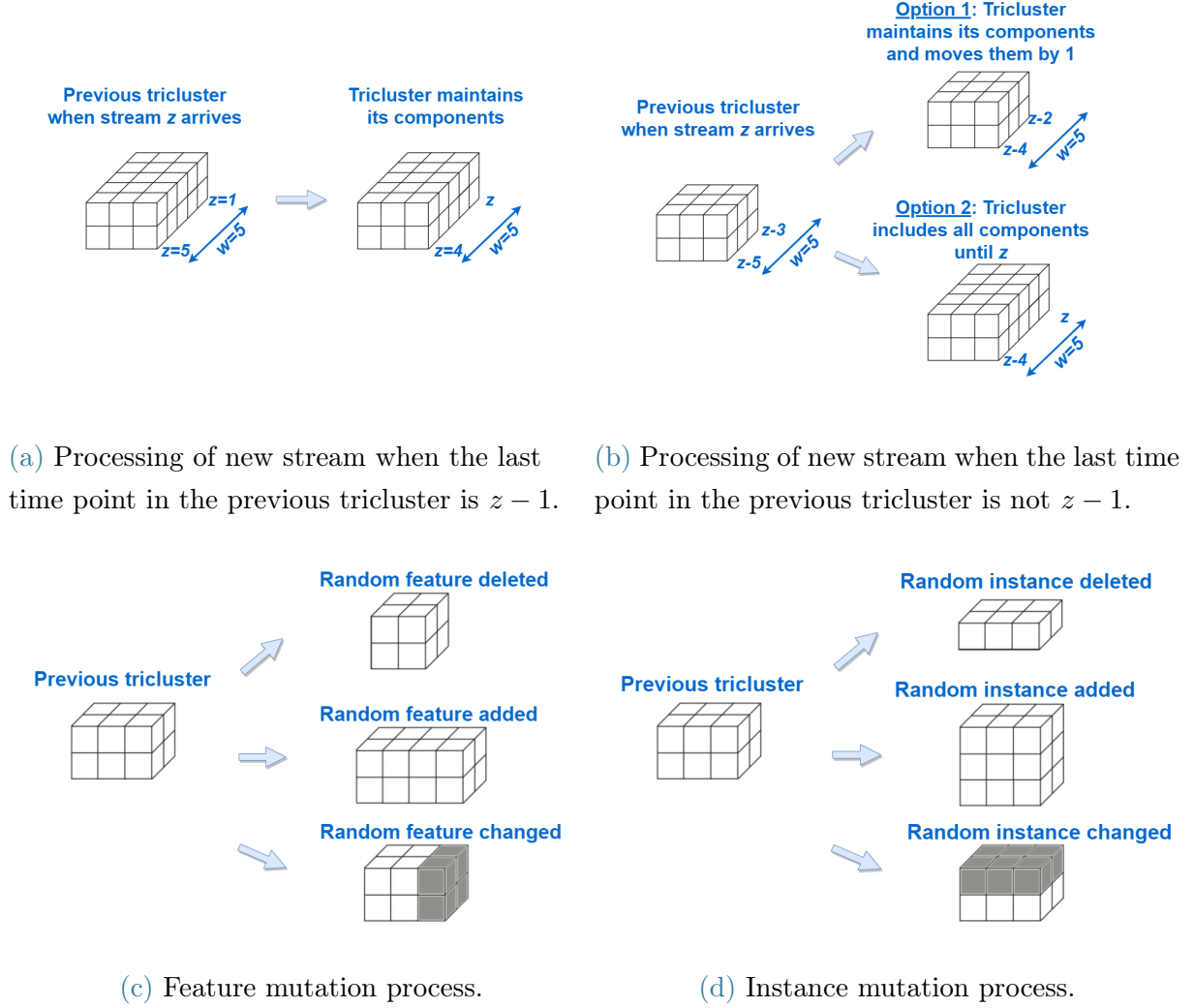


Figure 3.5: Examples of the updating operations during the online phase.

Algorithm 3.2 represents the methodology of the incremental learning of the *STriGen*. The statements within the if-condition refer to the second option presented. The sentences within the else-condition identify the case process in which the last time point of the tricluster is $z-1$, i.e., the first possible option explained.

Algorithm 3.2 Incremental update of the triclusters.

```

1:  $z \leftarrow$  current time
2:  $z-t \leftarrow$  last time of the tricluster  $TRI$ 
3: if  $z - t \neq z-1$  then
4:    $newTRI_1 \leftarrow \text{add}(z - t + 1, TRI)$ 
5:    $newTRI_2 \leftarrow \text{mutate}(newTRI_1)$ 
6:    $newTRI_3 \leftarrow \text{add}([z - t + 1, \dots, z], TRI)$ 
7:    $newTRI_4 \leftarrow \text{mutate}(newTRI_3)$ 
8:    $[\text{bestTRI}, \text{bestGRQ}] \leftarrow \text{evaluate}(newTRI_1, newTRI_2, newTRI_3, newTRI_4)$ 
9: else
10:   $newTRI_1 \leftarrow \text{add}(z, TRI)$ 
11:   $newTRI_2 \leftarrow \text{mutate}(newTRI_1)$ 
12:   $[\text{bestTRI}, \text{bestGRQ}] \leftarrow \text{evaluate}(newTRI_1, newTRI_2)$ 
13: end if
14: return Updated tricluster with bestTRI and bestGRQ

```

3.1.3.2.2. Concept drift

Concept drift occurs whenever an existing concept changes or a new one appears in the incoming stream data. In the *STriGen* online concept drift is detected with the abrupt decrease of the quality measure *GRQ*. Its meaning is that tricluster components are outdated or obsolete.

Two control parameters must be set prior to the start of the *STriGen* execution for the algorithm to identify and resolve the concept drift in real-time. These two parameters are *minGRQ* and *numIt*. The first is a threshold to identify an abrupt decrease in the *GRQ* as a concept drift. The second is the maximum number of iterations the tricluster can undergo to adjust to the new concept.

In case the *GRQ* of the current tricluster is less than *minGRQ*, a concept drift is identified in the tricluster. In such a scenario, the update of the incremental learning approach is not enough. The parameter *GRQ* allows *STriGen* to enter a loop for a maximum of *numIt* iterations or until the *GRQ* of the current tricluster is greater than *minGRQ*. A mutation is performed at each iteration and the evaluation of the *GRQ* is made afterwards.

Therefore, the *STriGen* adjusts rapidly to small changes but also to abrupt changes by detecting new tricluster components or by making the current ones disappear. This update is described in Algorithm 3.3.

Algorithm 3.3 Additional updating.

```

1: iter  $\leftarrow$  0
2: while currentGRQ <  $minGRQ$  and iter <  $numIt$  do
3:   TRI  $\leftarrow$  mutate(TRI)
4:   currentGRQ  $\leftarrow$  evaluate(TRI)
5:   iter  $\leftarrow$  iter+1
6: end while
7: return TRI

```

The complete online phase of the *STriGen* is in Algorithm 3.4.

Algorithm 3.4 Online phase.

```

S  $\leftarrow$  {}
while new data stream at time  $z$  do
  for each TRI in current model do
    TRI  $\leftarrow$  remove( $z - w - 1$ , TRI)
    [newTRI, currentGRQ] = updating(TRI) //Algorithm 3.2
    if currentGRQ <  $minGRQ$  then
      updatedTRI  $\leftarrow$  additionalUpdating(newTRI) //Algorithm 3.3
    else
      updatedTRI  $\leftarrow$  newTRI
    end if
    S  $\leftarrow$  S  $\cup$  updatedTRI
  end for
end while
return Set of updated triclusters  $S$ 

```

An overview of the online phase for a tricluster is illustrated in Figure 3.6. The workflow is executed N times in parallel for each tricluster $\{T_1, \dots, T_N\}$. The online phase continues to run until no more stream data st_z arrives.

3.1.3.3. Validation of the yielded triclusters

The validation of the triclusters solutions found by the *STriGen* is performed with the analysis of the evolution over time of the $GRQ(T_k)$ and the $TRIQ(T_k)$, defined in Formulas 3.13 and 3.14 respectively. Given that this is an online algorithm, it is of interest to study the quality metrics throughout time and not only at the end of the execution, as is case for the *bigTriGen* batch algorithm. Therefore, further analysis can be performed on the model update for small and large changes in the streams.

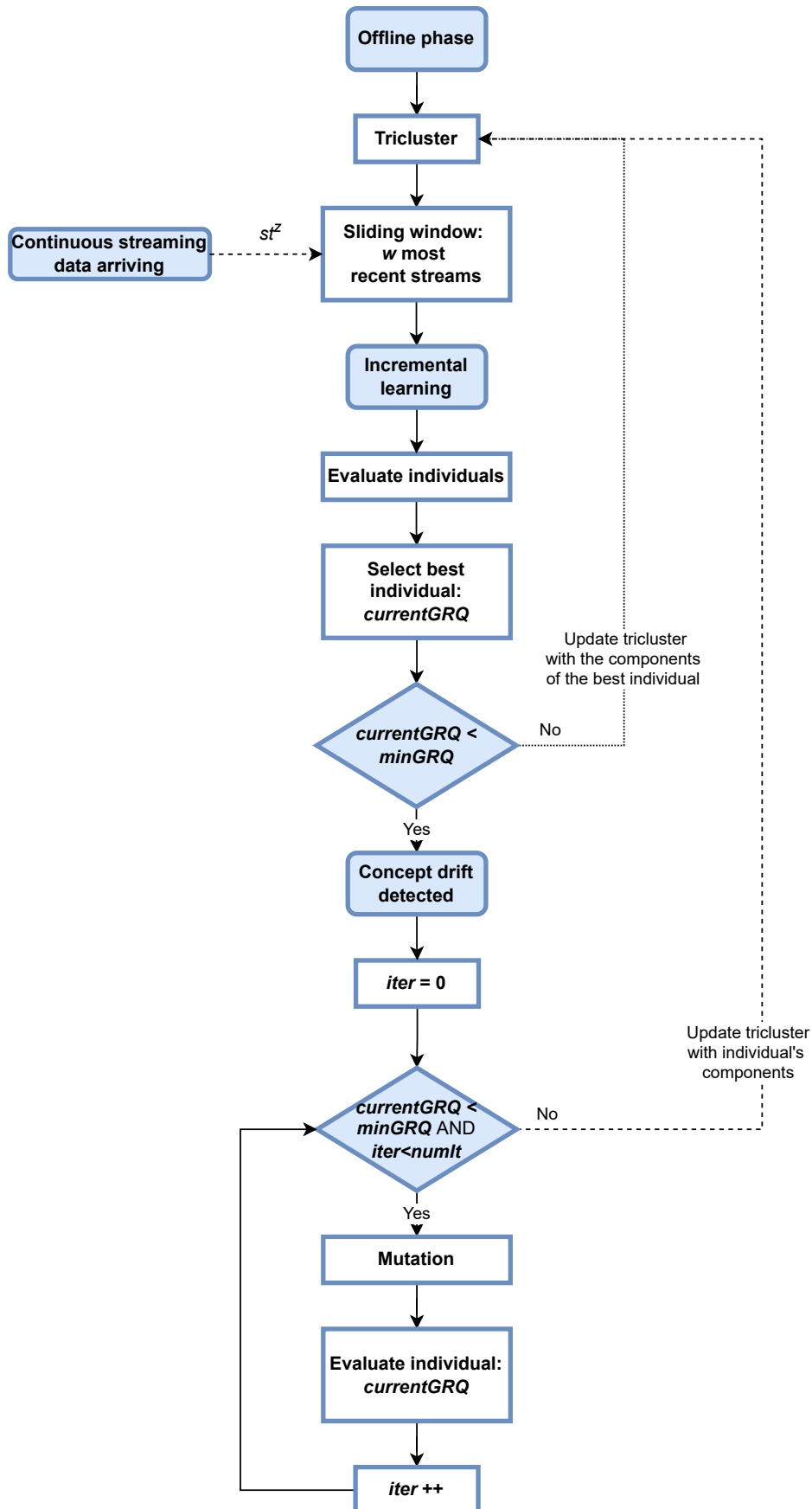


Figure 3.6: Overview of the *STriGen* online phase for one tricluster.

3.2 | Forecasting

In this Section, a novel time series algorithm for real-time forecasting called *StreamWNN* is presented. An overview of the developed methodology is in Section 3.2.1. The algorithm is composed of an offline phase followed by an online phase, defined in Section 3.2.2 and Section 3.2.3 respectively.

3.2.1 | Problem statement

The traditional K -nearest neighbor (KNN) is a well-known machine learning algorithm designed to predict the class of a sample by taking into account the classes of its K closest instances. This algorithm can be applied in classification and regression models, depending on the data type of the classes.

The traditional KNN algorithm is usually applied to time series problems. An instance of a time series is represented by its past w features and its next h classes, with w standing for past window and h for prediction horizon. When working with time-dependent data, it is important to keep the data chronologically ordered. Therefore, a time series X_t can be defined as a set of chronologically ordered values as follows:

$$X_t = \{(x^1, y^1), \dots, (x^T, y^T)\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \quad (3.17)$$

where the time series X_t is transformed into T instances composed of features and classes. The i – th instance is represented as (x^i, y^i) with x^i referring to the w past features and y^i to the next h classes. The traditional KNN algorithm applied to time series computes the forecast \hat{y}^i of the next h classes by considering the K nearest instances to x^i in terms of distance.

The KNN methodology focuses on the idea of finding similar patterns in the data. This approach is used to develop a new time series forecasting algorithm that works in real-time. This new algorithm is denoted *StreamWNN* which stands for Stream Weighted Nearest Neighbors, as it works in stream processing by providing real-time forecasts and is based on a weighted version of the KNN .

The *StreamWNN* addresses the streaming mode using a two-phase approach, i.e., an offline phase followed by an online phase. The sole purpose of the first phase is to provide a distributed big data model based on the KNN algorithm. The offline phase does not have to satisfy any of the requirements of algorithms running in real-time. All

these conditions must be fulfilled during the second phase of the *StreamWNN*. The online phase begins when the offline base model is computed. From this moment on, new incoming instances arrive in real-time. First, the streaming features of the st -th instance x^{st} arrive and the *StreamWNN* performs the forecast of the next h classes in streaming obtaining \hat{y}^{st} . Data streams do not stop arriving. Once the actual h classes y^{st} arrive, an error metric is calculated between the real value and the prediction. The online phase is repeated as long as new data continues to be received. The model can be updated in two different ways: following an incremental learning approach using an online buffer and including stream novelties in the model. These two options are detailed in Section 3.2.3.1 and in Section 3.2.3.2 respectively.

The proposed algorithm is built on Apache Spark 2.3.4 and uses the HDFS file system in Hadoop 2.7.7 and the Kappa Architecture in Apache Kafka 2.11 as streaming platform, i.e., a single pipeline is specifically designed for the job. Figure 3.7 illustrates the kafka architecture for the *StreamWNN* algorithm. Consumers are subscribed to the topics where the kafka producers publish the new incoming data streams. The queue in the topics follows a first-in-first-out basis so that all the streams are processed. When no more data comes online, the producer publishes a specific message which makes the kafka consumers understand that the algorithm is finished.

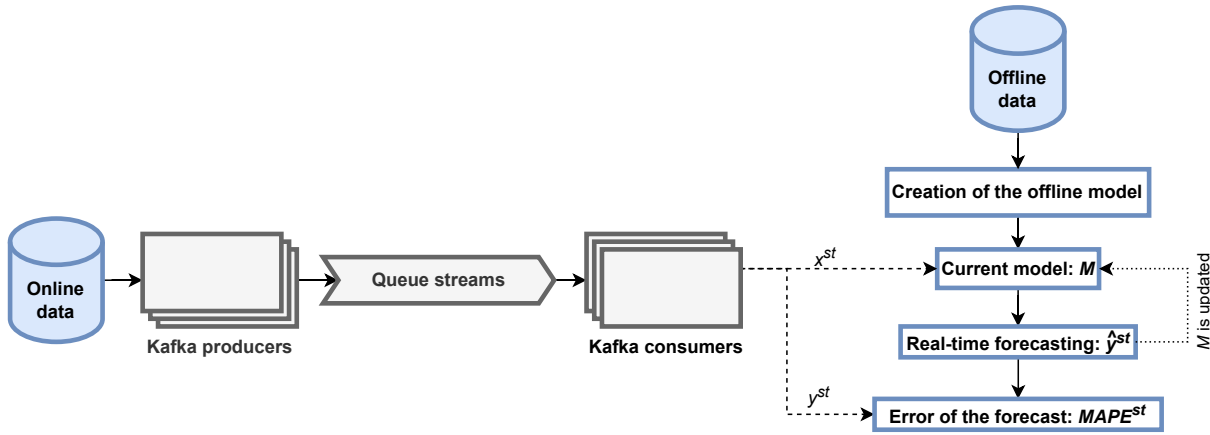
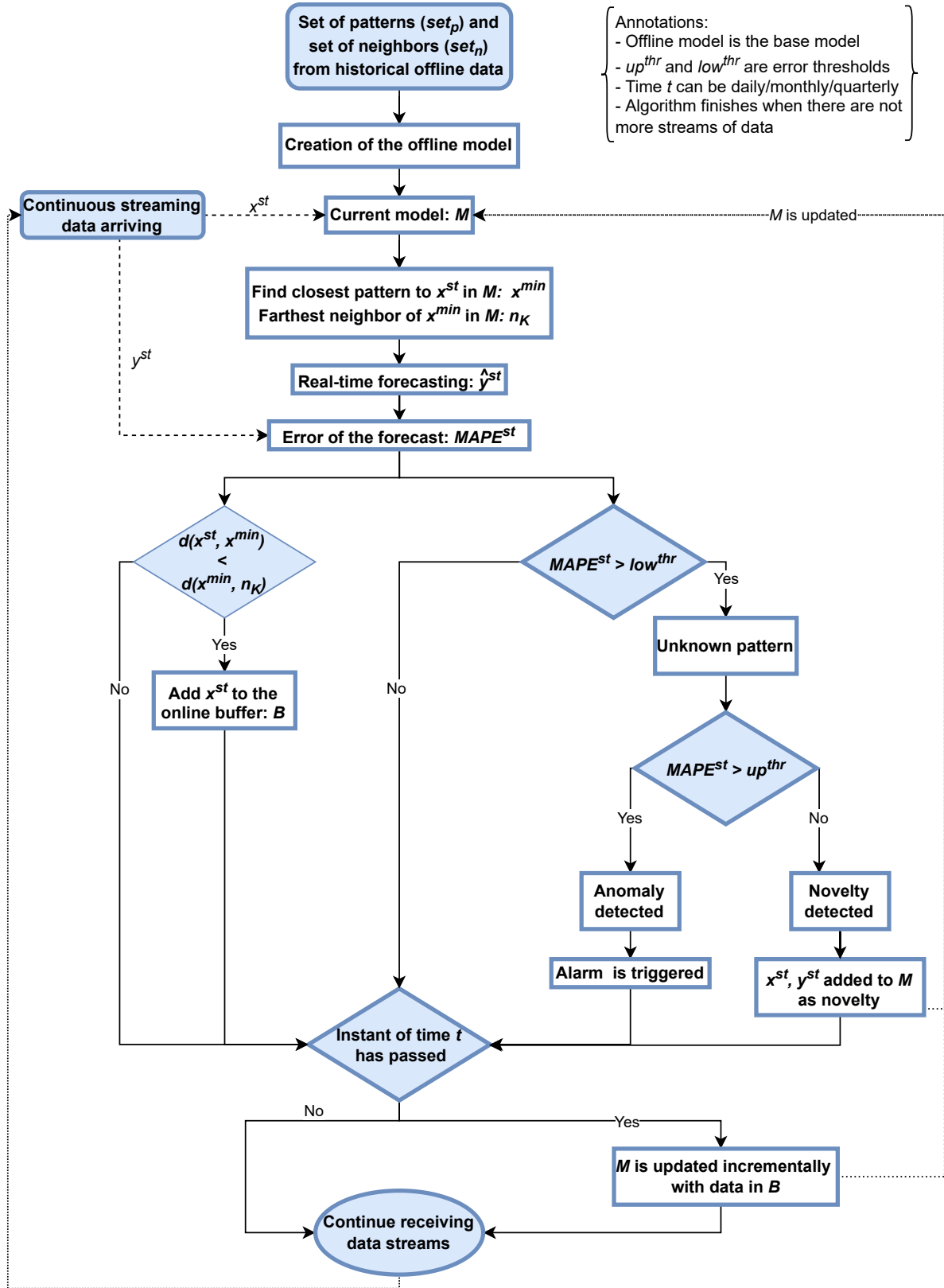


Figure 3.7: Outline of the kafka architecture for the *StreamWNN* algorithm.

A flow diagram of the whole *StreamWNN* methodology is depicted in Figure 3.8.

Figure 3.8: Overview of the whole methodology of the *StreamWNN* algorithm.

3.2.2 | Offline phase

The first phase of the *StreamWNN* is the offline phase. In *StreamWNN* the model is created only once during its offline phase and in a distributed and efficient way in batch processing. The next phase of the algorithm deals with streaming data using this offline base model which is computed with historical data. Considering $X_{historical}$ as the time series of the historical data, $X_{historical}$ is divided into 70% and 30% in a chronologically ordered manner. The first split forms the set of neighbors (set_n) while the remaining 30% of the data forms the set of patterns (set_p). This data separation is represented as:

$$\begin{aligned} X_{historical} &= \{(x^1, y^1), \dots, (x^H, y^H)\} = \{(set_n), (set_p)\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \\ set_n &= \{(x^1, y^1), \dots, (x^N, y^N)\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \\ set_p &= \{(x^{N+1}, y^{N+1}), \dots, (x^{P+N+1}, y^{P+N+1})\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \end{aligned} \quad (3.18)$$

where (x^{P+N+1}, y^{P+N+1}) is the same as (x^H, y^H) since set_p consists of P instances, set_n consists of N instances and there are H instances in $X_{historical}$. Instances are generated following a sliding window for the classes, i.e., there are no gaps between any of the y^i . Figure 3.9 illustrates an example of the sliding window with 100 time points as the number of values in the past window or w , and 50 time points for the next horizon classes or h . In the illustrated example, x^1 covers $\{x_1, \dots, x_{99}\}$ and y^1 covers $\{x_{100}, \dots, x_{149}\}$, x^2 covers $\{x_{50}, \dots, x_{149}\}$ and y^2 covers $\{x_{150}, x_{199}\}$ and x^3 covers $\{x_{100}, \dots, x_{199}\}$ and y^2 covers $\{x_{200}, x_{249}\}$.

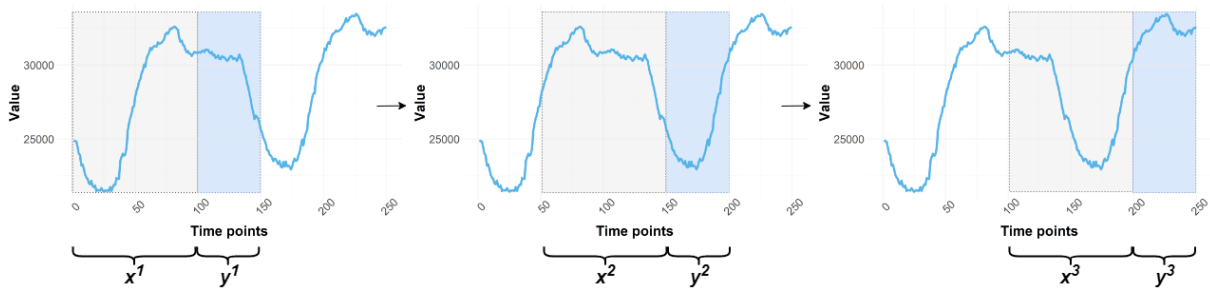


Figure 3.9: Sliding window for time series of the *StreamWNN* algorithm

The offline model is based on the traditional K -nearest neighbor methodology. The offline part of the *StreamWNN* creates a model in which each instance of the set_p is associated with its K nearest instances from the set_n . The euclidean distance is the

distance metric selected to quantify how close two features are:

$$d(x^i, x^t) = \sqrt{\sum_{l=1}^{|w|} \left(x^i(l) - x^t(l) \right)^2} \quad (3.19)$$

where $x^i(l)$ is the l -th value of the i -th feature of the set of patterns set_p and $x^t(l)$ is the l value of the t -th feature of the set of neighbors set_n . The K closest features x^t in set_n of the x^i are included in the model. Therefore, considering that the set_p consists of P instances, the model has P unique patterns samples. It may happen that not all series in the set_n are in the model since some of them may not be the nearest neighbor of any of the P instance patterns in the set_p .

Once the K closest neighbors of each pattern x^i are found, the prediction of its next classes y^i is performed taking into account the classes of the K neighbors. The Formula to calculate the offline predictions is the following:

$$\hat{y}^i(l) = \frac{1}{\sum_{j=1}^K \alpha_j} \sum_{j=1}^K \alpha_j \left(y_{n_j}(x^i)(l) \right) \quad 1 \leq l \leq h \quad (3.20)$$

with $y_{n_j}(x^i)(l)$ representing the l -th class value of the j -th neighbor of x^i and α_j defining the inverse squared euclidean distance between x^i and its j -th neighbor $x_{n_j}(x^i)$:

$$\alpha_j = \frac{1}{d(x^i, x_{n_j}(x^i))^2} \quad (3.21)$$

The offline model is defined as:

$$M = \left\langle \left(x^i, y^i \right), \left\langle \left(x_{n_1}(x^i), y_{n_1}(x^i) \right), \dots, \left(x_{n_K}(x^i), y_{n_K}(x^i) \right) \right\rangle \right\rangle \quad (3.22)$$

where (x^i, y^i) refers to the i -th instance from the set_p and $(x_{n_j}(x^i), y_{n_j}(x^i))$ to its j -th instance neighbor from the set_n .

With the offline predictions \hat{y}^i , an average error metric of the performance of the model can be computed. The actual next classes y^i of x^i are known since the offline model works in batch processing. The error metrics used are the Mean Absolute Percentage Error *MAPE* and the Mean Absolute Error *MAE*. These metrics are calculated for each

sample in the model.

$$MAPE_i = \frac{1}{h} \sum_{l=1}^h \left| \frac{y^i(l) - \hat{y}^i(l)}{y^i(l)} \right| \times 100 \quad (3.23)$$

$$MAE_i = \frac{1}{h} \sum_{l=1}^h |y^i(l) - \hat{y}^i(l)| \quad (3.24)$$

Once the offline model is created, $MAPE_{offline}$ and $MAE_{offline}$ are computed as the average of all $MAPE_i$ and MAE_i in the model.

The offline phase is represented in Algorithm 3.5.

Algorithm 3.5 Offline phase

```

1:  $w \leftarrow$  Window of past features
2:  $h \leftarrow$  Prediction horizon or next values to predict
3:  $K \leftarrow$  Number of neighbors to select
4: distances  $\leftarrow [ ]$ 
5:  $M \leftarrow [ ]$ 
6: for each  $x^i$  in  $set_p$  do
7:   for each  $x^t$  in  $set_n$  do
8:     distances  $\leftarrow$  add(distances,  $d(x^i, x^t)$ );
9:   end for
10:   $\langle (x_{n_1}(x^i), y_{n_1}(x^i)), \dots, (x_{n_K}(x^i), y_{n_K}(x^i)) \rangle \leftarrow$  K_smallest(distances,  $K$ )
11:   $M \leftarrow$  add( $M$ ,  $\langle (x^i, y^i), \langle (x_{n_1}(x^i), y_{n_1}(x^i)), \dots, (x_{n_K}(x^i), y_{n_K}(x^i)) \rangle \rangle$ )
12:   $\hat{y}^i \leftarrow$  predict( $M, x^i$ )
13:   $MAPE_i \leftarrow$  MAPE( $y^i, \hat{y}^i$ )
14:   $MAE_i \leftarrow$  MAE( $y^i, \hat{y}^i$ )
15: end for
16:  $MAPE_{offline} \leftarrow$  mean( $MAPE_i$ )
17:  $MAE_{offline} \leftarrow$  mean( $MAE_i$ )
18: return Offline model  $M$ ,  $MAPE_{offline}$ ,  $MAE_{offline}$ 

```

3.2.3 | Online phase

The second phase of the *StreamWNN* starts when the offline model is computed. From this moment on, the algorithm can receive flows of data that must be processed in real-time to obtain forecasts as fast as possible. Flowing data or streams are published in kafka topics where a queue is created following the first-in-first-out criterion. The kafka consumer collects groups of features (w instances) taking into account the sliding window requirement for the classes. Thus, the past window of the $st - th$ streaming instance is x^{st} .

Online predictions of the h next classes of x^{st} are performed considering the current model. Distances between the incoming stream x^{st} and all the P feature patterns x^i in the model are calculated. The stream is associated only with its closest x^i in the model, which is called x^{min} from now on.

$$x^{min} = \arg \min_{x^i \in M} d(x^i, x^{st}) \quad (3.25)$$

Online predictions \widehat{y}^{st} are calculated taking into account two components. The first one is the classes of its nearest pattern y^{min} . The second component is the classes of the neighbors of x^{min} in the model. This last component can be represented as $y_{n_j}(x^{min})$ for the $j - th$ neighbor of x^{min} . The complete online forecast formula is described as:

$$\widehat{y}^{st}(l) = \frac{1}{\sum_{j=1}^K \alpha_j + \alpha_{min}} \times \left(\sum_{j=1}^K \alpha_j \left(y_{n_j}(x^{min})(l) \right) + \alpha_{min} \left(y^{min}(l) \right) \right) \quad 1 \leq l \leq h \quad (3.26)$$

with α_{min} referring to the inverse squared euclidean distance between x^{min} and x^{st} :

$$\alpha_j = \frac{1}{d(x^{min}, x^{st})^2} \quad (3.27)$$

The consumer continues to receive data streams and collects them. The actual classes y^{st} are the next h samples received after x^{min} . The online error metrics $MAPE_{st}$ and MAE_{st} are computed following Formula 3.23 and Formula 3.24 respectively, with y^{st} and its prediction \widehat{y}^{st} . When no more streams are available, the online phase ends. Prior to the end of the algorithm, the average $MAPE_{online}$ and MAE_{online} metrics are performed.

When working with data streams, new behaviors are likely to appear in the data.

Data evolves over time and therefore the model must adapt and incorporate these new behaviors to produce accurate responses. There are four different types of implementations of the online phase of the *StreamWNN*: (1) Do not update of the model, (2) Update the model following an incremental learning approach, (3) Add novelties to the model, (4) Update the model with incremental learning and add novelties into the model.

In the first implementation the model remains the same as the one obtained from the offline phase. The second implementation updates the neighbors $(x_{n_j}(x^i), y_{n_j}(x^i))$ of the model with streaming data. This implementation is described in Section 3.2.3.1. The third implementation discovers data that behaves differently compared to the samples in the model. This option identifies anomalies in the streaming data that may trigger an alarm. The whole procedure for identifying novelties and anomalies can be found in Section 3.2.3.2. These two implementations can be computed at the same time in order to take advantage of both updates.

The online part of the *StreamWNN* is in Algorithm 3.6.

Algorithm 3.6 Online phase including neighbor updating

```

1:  $M \leftarrow$  Offline model from Algorithm 3.5
2: distances  $\leftarrow [ ]$ 
3: for each  $x^{st}$  that arrives do
4:   for each  $x^i$  in  $M$  do
5:     distances  $\leftarrow$  add(distances,  $d(x^i, x^{st})$ )
6:   end for
7:    $x^{min} \leftarrow$  K_smallest (distances, 1)
8:    $d^{min} \leftarrow d(x^{min}, x^{st})$ 
9:    $\widehat{y}^{st} \leftarrow$  predict ( $M, x^{st}$ )
10:   $MAPE_{st} \leftarrow$  MAPE( $y^{st}, \widehat{y}^{st}$ )
11:   $MAE_{st} \leftarrow$  MAE( $y^{st}, \widehat{y}^{st}$ )
12:   $M \leftarrow$  Check if  $M$  can be updated with Algorithm 3.7 and 3.8
13: end for
14:  $MAPE_{online} \leftarrow$  mean( $MAPE_{st}$ )
15:  $MAE_{online} \leftarrow$  mean( $MAE_{st}$ )
16: return updated  $M$  model, forecasts  $\widehat{y}^{st}$ ,  $MAPE_{online}$  and  $MAE_{online}$  errors

```

3.2.3.1. Incremental learning

Typically in batch processing each time a change in the data distribution is discovered, the entire model is trained again to include knowledge of this new trend. However, in stream processing a complete retraining of the model is not possible since the algorithm has to provide adequate and real-time responses. Incremental learning is a machine learning paradigm that continuously learns from incoming data streams. In other words, it does not assume that all the patterns that can provide information to generate an accurate model are in the offline historical data. The incremental learning setting of the *StreamWNN* detects different behaviors in the data from those already included in the model and adds these new distributions of data to the current model incrementally. The forecasts provided by the *StreamWNN* incremental learning update considers the knowledge included in the incoming streams.

The *StreamWNN* update during its online incremental learning approach focuses on the fitting of the associated neighbors of each instance in the model, i.e., the $j - th$ neighbor $(x_{n_j}(x^i), y_{n_j}(x^i))$ of the $i - th$ instance (x^i, y^i) in the model is fitted. The model modifies the neighbors of (x^i, y^i) but always keeping K associated instances for each of them. Therefore, it can be said that the model is internally updated since it does not increase the number of instances but only modifies the ones that it had before. The incremental learning is performed according to the idea that there may be new data distributions in the streams that are better neighbors of (x^i, y^i) than the already existing neighbors in the model $(x_{n_j}(x^i), y_{n_j}(x^i))$. Neighbors of the model have a great influence on the performance of online forecasts.

The methodology of the incremental learning approach for the *StreamWNN* follows the same steps as the ones mentioned in the general online phase without updates, i.e., x^{st} data representing the $st - th$ feature instance from the stream flow is collected and its online forecast $\widehat{y^{st}}$ is performed. Once the forecast is obtained, a condition involving the distances is checked. Distances are computed as:

$$\begin{aligned} d^{min} &= d(x^{min}, x^{st}) \\ d^K &= d(x^{min}, x_{n_K}(x^{min})) \end{aligned} \tag{3.28}$$

where x^{min} is the nearest feature pattern in the current model (see Formula 3.25) of x^{st} and $x_{n_K}(x^{min})$ is the farthest (K) neighbor of x^{min} in the current model.

In the case where d^{min} is less than d^K , it can be understood that the incoming data x^{st} is a more accurate neighbor than $x_{n_K}(x^{min})$. Changing the latter neighbor to x^{st}

provides more accurate online forecasts. Whenever this condition is satisfied, a buffer of online data B is filled as follows:

$$B = \{(x^{min}, x^{st}, d^{min})\} \quad (3.29)$$

The online buffer is filled as many times as necessary. However, the model is updated with knowledge in B only when a specific requirement is satisfied. This requirement must be chosen prior to the execution of *StreamWNN* and depends on the data used. It can be a time-based or a threshold-based option. For example, if the selected option is a daily incremental learning, the update based on the buffer B will be performed every day. The buffer will be checked after collecting all the data for a day and, if the buffer is not empty, the incremental learning update will be performed. The requirement that depends on a threshold is usually set considering the error metrics of the offline model. In this case, whenever the \widehat{y}^{st} provides a larger error than the defined threshold, the incremental learning update is carried out with the instances in the buffer.

Whenever the specific requirement is met, the *StreamWNN* checks the buffer B and the model. For each i -th instance of the model, all its neighbors in the model and the possible neighbors in the buffer are collected:

$$neighbors_x^i = \{x_{n_1}(x^i), \dots, x_{n_k}(x^i)\} \cup \{x^{st}(x^i) \in B\} \quad (3.30)$$

where the first part of the union identifies the neighbors of x^i in the model and the second part are the streams that are in B associated with x^i . Since each instance in the model needs to have K neighbors, the K $neighbors_x^i$ with the smallest distance to x^i are the new neighbors. In other words, the model is updated to include the closest K neighbors of x^i .

The following Formula is an example of the 40 - th instance of the model that has been updated incrementally. The streams $x^{st_{10}}$ and x^{st_5} were in the buffer B associated with x^{40} . Its $K=3$ closest $neighbors_x^{40}$ ordered from closest to farthest are: $x^{st_{10}}$, its first historical neighbor x_{n_1} and x^{st_5} :

$$M = \left\langle \left(x^{40}, y^{40} \right), \right. \\ \left. \left\langle \left(x_{st_{10}}(x^{40}), y_{st_{10}}(x^{40}) \right), \left(x_{n_1}(x^{40}), y_{n_1}(x^{40}) \right), \left(x_{st_5}(x^{40}), y_{st_5}(x^{40}) \right) \right\rangle \right\rangle \quad (3.31)$$

The methodology of the incremental learning approach of the *StreamWNN* is in Algorithm 3.7.

Algorithm 3.7 Neighbor Updating

```

1:  $update_{moment} \leftarrow \text{select}(\text{time-based or threshold-base option})$ 
2:  $B \leftarrow [ ]$ 
3:  $distances_{M \cup B} \leftarrow [ ]$ 
4: if  $d^{min} < d(x^{min}, x_{n_K}(x^{min}))$  then
5:    $B \leftarrow \text{add}(B, \langle x^{min}, x^{st}, d^{min} \rangle)$ 
6:   if  $update_{moment}$  and  $B$  not empty then
7:      $neighbors\_x^{min} \leftarrow \{x_{n_1}(x^{min}), \dots, x_{n_k}(x^{min})\} \cup \{x^{st}(x^{min}) \in B\}$ 
8:      $distances_{M \cup B} \leftarrow \text{add}(distances_{M \cup B}, d(x^{min}, neighbors\_x^{min}))$ 
9:      $new\_neighbors\_x^{min} \leftarrow K\_smallest(distances_{M \cup B}, K)$ 
10:     $M \leftarrow \text{update}(M, \langle x^{min}, y^{min} \rangle, new\_neighbors\_x^{min})$ 
11:   end if
12: end if
13: return updated  $M$  model

```

3.2.3.2. Novelties and anomalies

Novelties and anomalies are highly correlated terms that identify patterns different from the known concepts of the model. Although novelties and anomalies can be classified together as "unknown patterns", each identifies a different behavior of the data and should therefore be treated independently. Novelties and anomalies in data streams are very interesting to study. Actions taken in real-time thanks to their identification, can modify the course of a given execution and avoid erroneous operations.

In *StreamWNN* a x^{st} is considered an unknown pattern if its error obtained between the forecast $\widehat{y^{st}}$ and its actual class y^{st} is greater than a defined threshold. This identification follows an unsupervised learning approach, since the dataset has time series data without any label concerning the identification or not of an unknown pattern. The only way to determine if an unknown pattern is correctly identified is to monitor the evolution of the error metric over time and to check the identified patterns in real-time.

When an unknown pattern is identified, the model performs a different approach depending on an error-based threshold. Therefore, two thresholds are set prior to the *StreamWNN* execution to identify novelties and anomalies. These two thresholds are an upper one (up^{thr}) and a lower one (low^{thr}). The values of these thresholds are defined after an analysis of the errors obtained once the offline model has been computed. Thus, they change depending on the dataset used.

Anomalies are defined as data that do not conform to the expected behaviors of the incoming data. One way to identify anomalies using k -nearest neighbors based models is to identify whether a pattern occurs far from its closest neighbors [3]. Following this assumption, in *StreamWNN* anomalies are stream data that get a large error. The meaning of a large forecast error is that its behavior is very different from the expected one. Therefore, x^{st} is identified as an anomaly if the online error of its h future values is larger than up^{thr} . The detection of an anomaly triggers a real-time alarm in the algorithm. This alarm can influence the modification of the following incoming forecasts manually by the user.

On the other hand, novelties are unknown patterns that represent new and emergent patterns in the data. Novelties should be added to the model since data with similar behavior is expected to appear in the input streams. Novelties are identified when an instance gets an error greater than low^{st} but less than up^{thr} .

The classification of an unknown pattern as an anomaly or a novelty is briefly described in the conditional Formula 3.32.

$$x^{st} = \begin{cases} Anomaly & \text{if } \widehat{y^{st}} > up^{thr} \\ Novelty & \text{if } low^{thr} < \widehat{y^{st}} < up^{thr} \\ \text{Not an unknown pattern} & \text{otherwise} \end{cases} \quad (3.32)$$

Upon identification of a novelty the model is updated by adding (x^{st}, y^{st}) as a new instance. Its K neighbors are selected as its K closest instances from the entire offline historical data, i.e., both set_p and set_n . The idea behind this is that new streaming behaviors can include neighbors that were not selected as K closest neighbors for any of the offline patterns. The model is said to be externally updated with novelties because its dimensions increase.

An update representation of the model M composed of p instances and with x^{st} identified as novelty is:

$$M = \left\langle \begin{aligned} &\left\langle (x^1, y^1), \left\langle (x_{n_1}(x^1), y_{n_1}(x^1)), \dots, (x_{n_K}(x^1), y_{n_K}(x^1)) \right\rangle \right\rangle, \\ &\dots, \\ &\left\langle (x^p, y^p), \left\langle (x_{n_1}(x^p), y_{n_1}(x^p)), \dots, (x_{n_K}(x^p), y_{n_K}(x^p)) \right\rangle \right\rangle, \\ &\left\langle (x^{st}, y^{st}), \left\langle (x_{n_1}(x^{st}), y_{n_1}(x^{st})), \dots, (x_{n_K}(x^{st}), y_{n_K}(x^{st})) \right\rangle \right\rangle \end{aligned} \right\rangle \quad (3.33)$$

Algorithms 3.8 and 3.9 describe the identification of unknown patterns in the stream incoming data and the methodology for updating the model with novelties, respectively.

Algorithm 3.8 Unknown patterns identification

```

1:  $M \leftarrow$  Current model
2:  $low^{thr} \leftarrow$  select
3:  $up^{thr} \leftarrow$  select
4: for each  $x^{st}$  that arrives do
5:    $\hat{y}^{st} \leftarrow$  predict( $M, x^{st}$ ) (see Equation 3.26)
6:    $MAPE_{st} \leftarrow$  MAPE( $y^{st}, \hat{y}^{st}$ )
7:   if  $MAPE_{st} > low^{thr}$  then
8:      $x^{st}$  is an unknown pattern
9:     if  $up^{thr} > MAPE_{st}$  then
10:       $x^{st}$  is a novelty
11:       $M \leftarrow$  Add novelty to the model with Algorithm 3.9
12:     else
13:       Anomaly identification
14:       Alert created by the model
15:     end if
16:   end if
17: end for
18: return updated  $M$  model and identification of unknown patterns

```

Algorithm 3.9 Model update with novelties

```

1:  $M \leftarrow$  Current model
2: distances  $\leftarrow [ ]$ 
3:  $set_{offline} \leftarrow set_n \cup set_p$ 
4:  $x^{st} \leftarrow$  Novelty
5: for each  $x^i$  in  $set_{offline}$  do
6:   distances  $\leftarrow$  add(distances,  $d(x^{st}, x^i)$ )
7: end for
8:  $\{n_1(x^{st}), \dots, n_K(x^{st})\} \leftarrow$  K_smallest(distances,  $K$ )
9:  $M \leftarrow$  add( $M, < (x^{st}, y^{st}), < (x_{n_1}(x^{st}), y_{n_1}(x^{st})), \dots, (x_{n_K}(x^{st}), y_{n_K}(x^{st})) >>$ )
10: return updated  $M$  model with novelty

```

The same execution of the *StreamWNN* can perform the incremental learning approach and the novelties approach. The first approach updates neighbors in the model and the second approach identifies anomalies in real-time and updates the model with novelties. The first one is an internal update and the second one is an external update.

4 | Applications

THE application of machine learning algorithms to real datasets can improve our quality of life. This is one of the main goals of this dissertation. The three new machine learning algorithms described in Chapter 3 are applied to real datasets and their results are presented in this Chapter. On the one hand, the application of the *bigTriGen* and *STriGen* triclustering algorithms to Smart Cities and medical projects is summarized in Section 4.1. On the other hand, the results of the *StreamWNN* forecasting algorithm for the Spanish energy electricity demand are presented in Section 4.2.

The complete results achieved in each of these applications can be consulted in the scientific articles published during the PhD program.

4.1 | Triclustering applications to Smart Cities and medicine

This Section describes the applications made with *bigTriGen* and *STriGen* algorithms. In particular, Section 4.1.1 describes the application of *bigTriGen* to precision agriculture. Section 4.1.2 presents the application of the *bigTriGen* to find seismogenic triclusters. Regarding the *STriGen*, Section 4.1.3 describes the application of this algorithm to real environmental sensor data. Section 4.1.4 summarizes the results found for the application of *STriGen* to medical images data. These applications are described in detail in the scientific papers [2, 13, 14, 16, 17].

4.1.1 | Precision agriculture

This Section describes the application of the *bigTriGen* algorithm to two different crop image datasets using vegetation indices to improve precision agriculture. In addition, the parameters used and the performance of the image set are discussed.

4.1.1.1. Datasets description

The paradigm of agriculture has been evolving in recent decades. Crops are being studied and treated following a site-specific management which has given rise to the concept of precision agriculture. Not managing the crop in an uniform way can provide both economic and environmental advantages. Effective implementation of precision agriculture requires knowledge and characterization of spatial variability within the field.

The analysis of a field area by means of vegetation indices allows the identification of different characteristics that are often invisible to the naked-eye. These indices are algebraic combinations of the measured canopy multispectral reflectance of different wavelength bands. This reflectance can be detected remotely using aerial or satellite imagery. They allow the evaluation of both quantitative and qualitative measures of crops such as cover, vigor, growth, type of quality. There are several famous vegetation indices. Each index provides different information about the crop.

Two different crops located in Baixo Alentejo, Portugal, are studied using different vegetation indices. The *bigTriGen* provides interesting triclusters for each of the analyzed sites. The first crop is a 63.82 ha maize plantation and the second is a 5.15 ha vineyard

crop.

The maize plantation is monitored between April 2018 and September 2018, coinciding with the sowing and harvesting of corn crops. A total of nineteen satellite images are extracted from the Sentinel 2 Mission with a time interval between them of five, ten and fifteen days. In this experimentation, the Normalized Differential Vegetation Index $NDVI$ is used. This index provides a value between 1.0 and -1.0 related with the content of the studied vegetation. The 1.0 value corresponds to denser and healthier areas, while -1.0 represents the poorer and most stressed areas. Figure 4.1 represents three sample images of the $NDVI$ values in the research site. The $NDVI$ calculation depends on the near-infrared band NIR and the band for the visible or Red regions. Its formula is:

$$NDVI = \frac{NIR - Red}{NIR + Red} \quad (4.1)$$

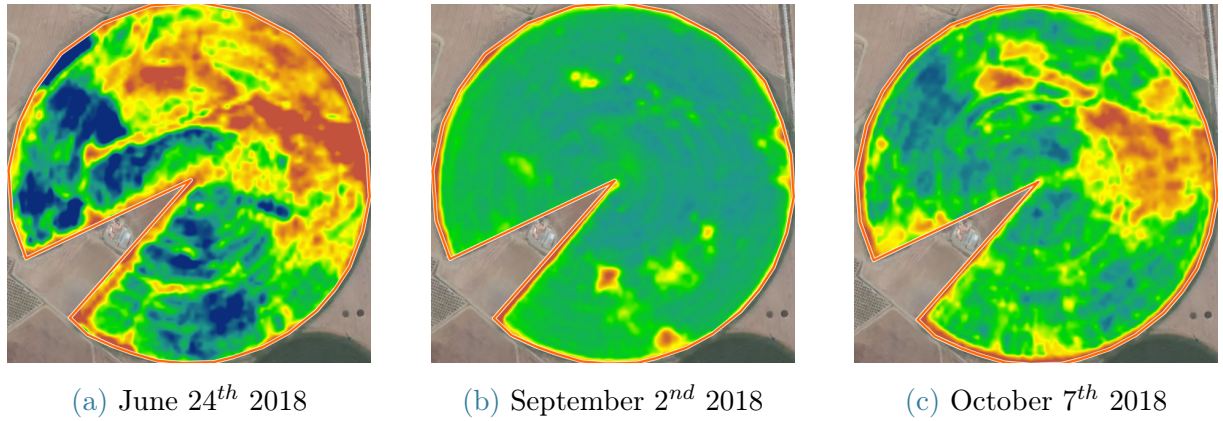


Figure 4.1: $NDVI$ samples for the studied maize crop

The second crop studied with *bigTriGen* is a vineyard crop. The algorithm provides precision viticulture triclusters. The area is monitored during 2018, 2019 and 2020 from May to October which are the months corresponding to the vineyard session. Image data are extracted from the Sentinel 2 Mission using the QGIS software. In this experimentation, four more indices are analyzed along with $NDVI$. The Soil-Adjusted Vegetation Index $SAVI$ and the Enhanced Vegetation Index EVI provide some improvements to the $NDVI$ index for this particular type of crop. They introduce a correlation factor L for soil brightness. The EVI also includes two more coefficients C_1 and C_2 for the atmospheric resistance and a *Blue* band. In addition, the Moisture Stress Index MSI and the Green Normalized Difference Vegetation Index $GNDVI$ are studied including two more bands: the middle-infrared MIR and the *Green* band, respectively. Each of these indices is analyzed specifically for the information that they provide for

vineyard crops. They are discussed in more detail in the following Sections. The formulas for these crops are the following:

$$SAVI = \frac{NIR - Red}{NIR + Red + L} \quad (4.2)$$

$$EVI = 2.5 \times \frac{NIR - Red}{NIR + C_1 \times Red - C_2 \times Blue + L} \quad (4.3)$$

$$MSI = \frac{MIR}{NIR} \quad (4.4)$$

$$GNDVI = \frac{NIR - Green}{NIR + Green} \quad (4.5)$$

4.1.1.2. Parameter tuning

Prior to defining the *bigTriGen* configuration parameter values, it is necessary to run the algorithm several times with different settings for each parameter. For both experimentation image dataset, the parameters with the best fit are: $N=4$, $G=10$, $In=200$, $Sel=0.8$ and $Mut=0.1$. Weights are fixed to: $w_{msl}=0.8$, $w_s=0.1$, $w_{ov}=0.1$, $w_{gr}=0.4$, $w_{ps}=0.3$ and $w_{sp}=0.3$.

Regarding the vegetation indices, the coefficients of the *SAVI* and *EVI* indices, defined in Formulas 4.2 and 4.3, are set according to the values that they usually used when applied to vineyard crops. L is fixed to 0.5, C_1 is 6 and C_2 is 7.5.

4.1.1.3. Model performance

The yielded triclusters are validated by an agriculture expert of the specific crop type of each dataset. In addition, the *TRIQ* measure provides a numeric value to determine the quality of the tricluster, following the computation defined in Formula 3.14.

All four discovered triclusters from the maize crop dataset obtained high *TRIQ* values, confirming the accurate quality of the triclusters. The *TRIQ* quality values for this experiment can be found in Table 4.1. Figure illustrates the evolution of the *NDVI* index over time for each of the instances and features of each tricluster. Figure 4.2 shows that tricluster components have a similar behavior over time which is very important to validate them graphically. The field's farmer also validated the triclusters using additional information. For example, the *NDVI* changes in triclusters T_1 , T_2 and T_3 are related to the use of fertilizers and the increase of amount of water in the irrigation process. The *NDVI* recovery from the initial values after mid September in the T_3 zones is related to

the application of fungicide by the farmers in August. The crop zones in T_4 present a constant low $NDVI$ over time.

T_1	T_2	T_3	T_4
0.803	0.753	0.827	0.742

Table 4.1: TRIQ values for each tricluster of the maize dataset with $NDVI$ index

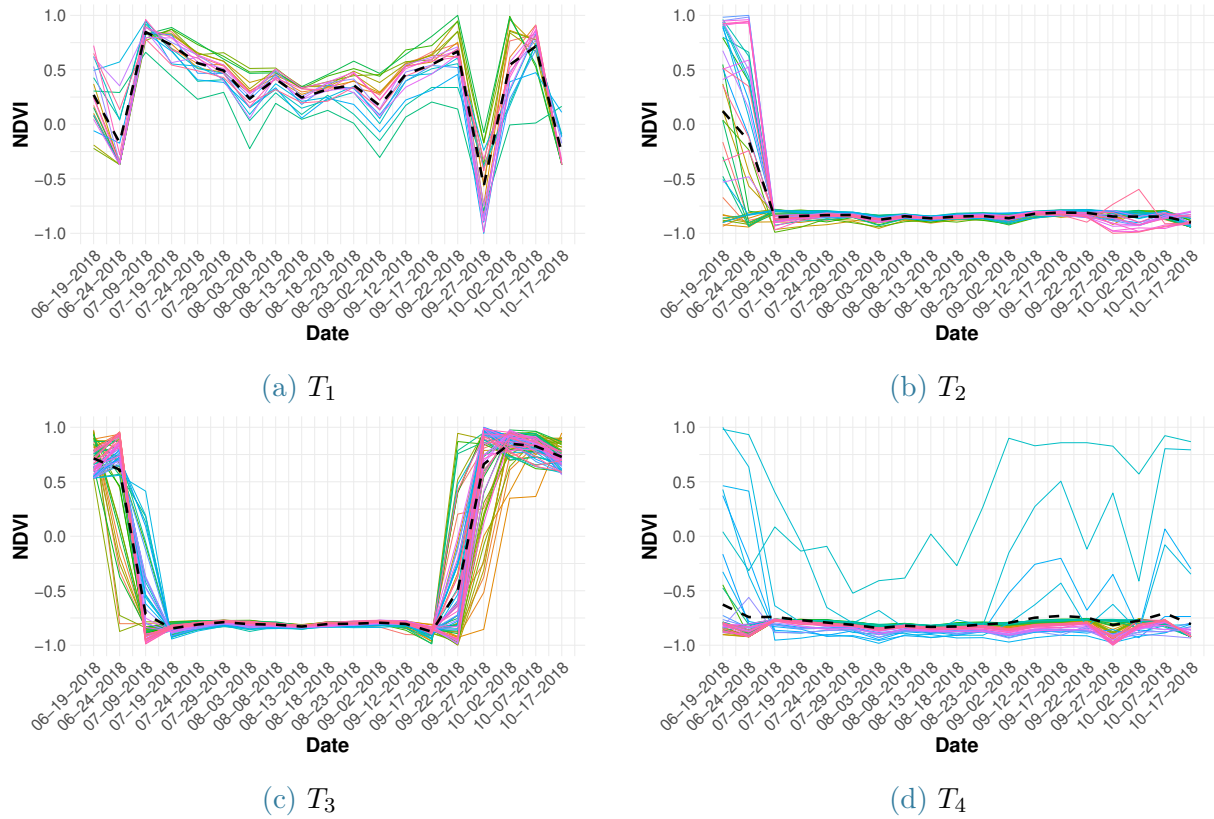


Figure 4.2: Yielded triclusters by *bigTriGen* for the maize plantation

For the second dataset, five vegetation indices are studied for the vineyard session during 2018. Triclusters found by *bigTriGen* using the $NDVI$ index showed a great uniformity among all the zones found, i.e., the pattern behavior is uniform for all the triclusters. The same assessment can be made with $SAVI$, EVI and $GNDVI$ indices with the 2018 images. However, the triclusters found with the MSI vegetation index allow identifying different trends in the water stress behavior. This index usually varies between 0.4 and 2, with higher values representing higher water stress and lower soil moisture. The $TRIQ$ quality measure of the yielded triclusters is in Table 4.2. High $TRIQ$ values and an expert of this type of crop allow the validation the triclusters. The analysis of the productivity of each field zone can be performed considering the triclusters.

The *MSI* evolution over time for each tricluster is shown in Figure 4.3.

T_1	T_2	T_3	T_4
0.8799	0.9365	0.9153	0.8321

Table 4.2: TRIQ values for each tricluster of the vineyard dataset with *MSI* index

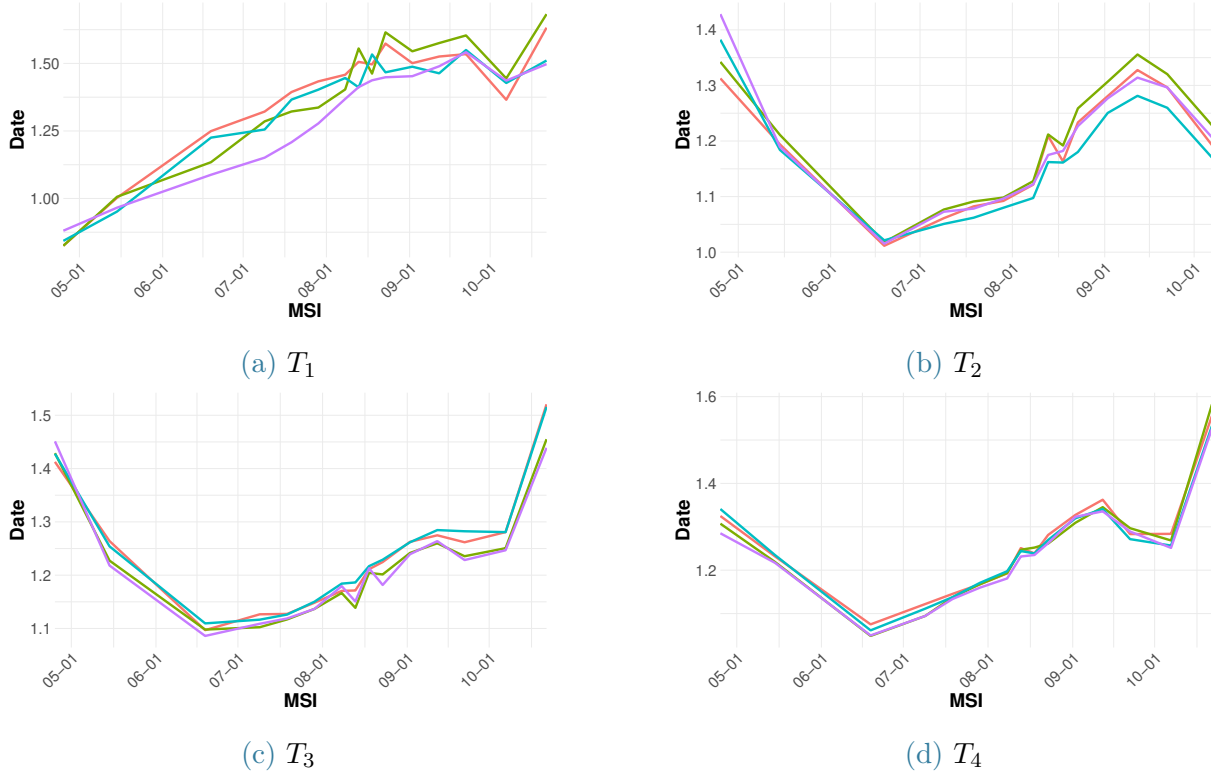


Figure 4.3: Yielded triclusters by *bigTriGen* for the vineyard crop

The demonstration of *bigTriGen* scalability is carried out with twenty-four experiments using 12, 24, 36 and 48 cores. In addition, the length of the vineyard dataset is multiplied by 1, 2, 4, 6, 16 and 32, resulting in six datasets of increasing length. Figure 4.4 presents the execution time of each experiment where a linear increase can be identified demonstrating the scalability of the algorithm.

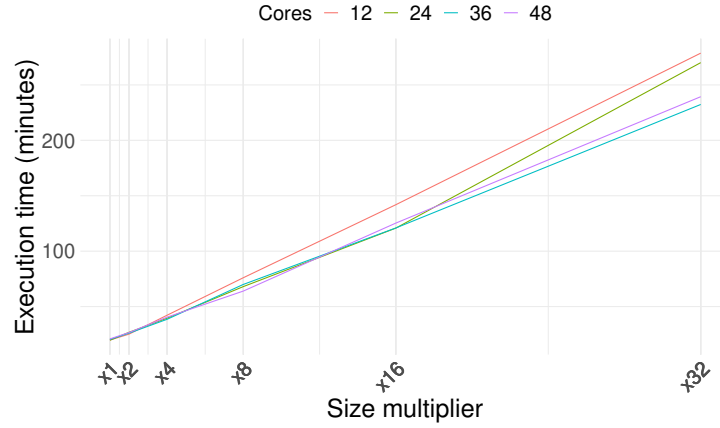


Figure 4.4: Scalability analysis of the *bigTriGen*

4.1.2 | Seismogenic

This Section describes the methodology followed to generate a seismogenic source zone using *bigTriGen*. The description of the data used, the selected parameters and the performance of the execution are described.

4.1.2.1. Dataset description

Seismogenic source zones are essential for applying seismic hazard calculations, especially in regions with a low or a moderate seismicity. In this application, the *bigTriGen* has been used to create a seismogenic source zone based on an updated, reviewed, declustered, extensive and homogeneous earthquake catalog. The catalog includes 3500 earthquake events in the Pyrenees from 1978 to 2019 of magnitude equal or greater than 2.5.

Data must be preprocessed to fit the *bigTriGen* modelling. In this case, the catalog is sorted into 30×20 cells. Each cell represents an area of approximately $16.5 \text{ km} \times 16.5 \text{ km}$. The dataset employed consists of x , y and f coordinates. Relative latitude is represented by the x coordinate and the relative longitude by the y coordinate. However, as time between earthquake events is widely spaced, the third dimension of the data is not the time but the following features of the recorded cell: $\{M_{max}, D, M_{2.9}, M_{3.3}, M_{3.7}, M_{4.1}, E\}$. These features represent:

- M_{max} : Maximum earthquake magnitude recorded in the cell.
- D : Mean epicenters depth of the earthquakes recorded in the cell.
- $M_{2.9}$: Number of earthquakes with a magnitude equal or larger than 2.9 recorded

in the cell.

- $M_{3.3}$: Number of earthquakes with a magnitude equal or larger than 3.3 recorded in the cell.
- $M_{3.7}$: Number of earthquakes with a magnitude equal or larger than 3.7 recorded in the cell.
- $M_{4.1}$: Number of earthquakes with a magnitude equal or larger than 4.1 recorded in the cell.
- E : Total number of earthquakes occurring.

Following this definition, the cell $C_{4,2,M_{3.3}}$ identifies the number of earthquakes with magnitude equal or greater than 3.3 in the cell with relative latitude 4 and relative longitude 2.

After the execution of the *bigTriGen* algorithm with this seismic dataset, N triclusters are found. Each yielded tricluster identifies an area with similar behavior patterns discovered by the algorithm. This area corresponds to a set of relative latitudes and a set of relative longitudes extracted from the x and y coordinates of each preprocessed cell.

4.1.2.2. Parameter tuning

After running the *bigTriGen* several times, the parameters that fit the best to the dataset are: $N=8$, $G=30$, $In=20$, $Sel=0.8$, $Mut=0.5$. Weights are fixed to: $m_{msl}=0.8$, $w_s=0.1$, $w_{ov}=0.1$, $w_{gr}=0.4$, $w_{ps}=0.3$ and $w_{sp}=0.3$.

4.1.2.3. Model performance

In this particular application, the delineated zones are evaluated by the seismic characterization of each zone. An seismogenesis expert studied each tricluster using seismic parameters such as the b – value to estimate the relationship between small and large earthquakes which is related to the physics of the place, i.e., the lower the value is, the more energy can be accumulated. According to the expert, another relevant parameter that demonstrated the good quality of the solution triclusters is the annual rate parameter normalized or $AR3$. This parameter corresponds to the number of events that exceed a threshold per year. The graphical representation of these parameters allows the evaluation of the zones identified by the *bigTriGen*. They are seismically different from each other and cover the vast majority of the region's epicenter.

4.1.3 | Environmental sensors

STriGen is applied to an environmental sensor dataset. The description of the real dataset, the parameters used and the performance of the model are described in this Section.

4.1.3.1. Dataset description

Nowadays, data streaming modelling is gaining importance due to the increasing number of sources providing continuous and massive flows of data. Some of the most common data streaming sources in these days are the social media, medical devices, videos or Internet of Things (IoT). There are several applications of machine learning models on this type of data. For example, obtaining triclusters from IoT environmental sensors can help develop a more efficient management of specific areas in real-time.

For the application of the *STriGen* algorithm, a real-world dataset from seven environmental sensors in twelve different areas of Malaga, Spain, has been selected. These sensors record atmospheric pressure, precipitation, relative humidity, solar radiation, temperature, wind direction and wind speed. The dataset consist of 5000 time points.

4.1.3.2. Parameter tuning

The *STriGen* has two phases: an offline phase which creates a base tricluster model followed by an online phase where the triclusters are obtained in real-time. In this particular application, the first phase uses the first 500 time points of the sensors dataset. The remaining 4500 time points are used to obtain real-time triclusters and analyze the patterns found.

After a tuning process, the traditional tricluster parameters are set to: $N=3$, $G=200$, $In=400$, $Sel=0.7$ and $Mut=0.4$. Weights are set to: $w_{msl}=0.8$, $w_s=0.1$, $w_{ov}=0.1$, $w_{gr}=0.4$, $w_{ps}=0.3$ and $w_{sp}=0.3$. The *STriGen* specific offline parameters are set to: $Ale=0.2$ and $Cross=0.4$. The *STriGen* online parameters are fixed at: $w=20$, $minGRQ=0.975$ and $minIt$ to 35.

4.1.3.3. Model performance

The components of each yielded tricluster vary with time as streams arrive at the model. In this application, the quality measure used is the *GRQ* value. The mean *GRQ* of the three triclusters for the entire streaming phase is 0.9468, denoting a very high representation quality measure.

Since it is a real dataset, the ground truth is not known. For this particular application, the performance of the model is evaluated by comparing *STriGen* results with those of a baseline algorithm on the same dataset. The selected baseline algorithm is a version of the *STriGen* in which the best tricluster is not selected during the online phase, but a random tricluster among all possibilities. The mean and standard deviation *GRQ* measurements of the *STriGen* and the baseline algorithm are reported in Table 4.3. Results demonstrate that *STriGen* is well adapted to the evolution of the data streams since the *STriGen* results are higher than those of the baseline.

Algorithm	T_1		T_2		T_3	
	Mean	Std	Mean	Std	Mean	Std
STriGen	0.9398	0.0294	0.9390	0.0289	0.9614	0.0299
Baseline	0.7653	0.0314	0.7682	0.0302	0.7665	0.3533

Table 4.3: *STriGen* and baseline comparison for environmental sensors dataset

4.1.4 | Medical images

This Section presents the application of the *STriGen* streaming algorithm to medical High-Content Screening (HCS) images. This Section describes the dataset used, the configuration parameters and the performance of the algorithm.

4.1.4.1. Dataset description

High-Content Screening is a biological technique that investigates the whole cellular processes by providing insights into the biological circuits of genes in biology. HCS combines an automated image acquisition and analysis of intact cells exposed to some chemical or genomic perturbations that alter cells' phenotype. Cell screening is performed in parallel with measurements of multiple fluorescent readouts. During the automated image analysis phase of this technique, measurement of subcellular locations and fluorescence color intensity during different complex cellular events is performed both temporally and spatially. The analysis of HCS images in real-time can provide fast and accurate information about cells, which has proven very useful for detecting DNA, cytokinesis, apoptosis or cell division.

STriGen is applied to a dataset of 406 HCS images of cervical cancer cells or HeLa cells. In particular, the dataset presents the reaction of HeLa cells to Transferring receptors. These receptors are often used to target cancer cells since they enhance site-specific therapies. The application of *STriGen* to HCS can help to discover the fluorescent

marker color, cell component shape, spatial situation, distribution of color pixels in a specific area to analyze interactions or co-occurrences and time evolution of a cell. Images must be preprocessed prior to the execution of the streaming algorithm by creating a 3D matrix with coordinates of each pixel of RGB image representing the color features of the fluorescent marker.

4.1.4.2. Parameter tuning

For this application, the offline phase of the *STriGen* is performed with the first 10 streams. Therefore, the remaining 396 images are received as stream data in the online phase of the algorithm.

After the tuning process, the configuration parameters are $N=3$, $G=300$, $In=600$, $Sel=0.7$, $Mut=0.4$. Weights are set to the same values as in the previous application of the *STriGen*, i.e., $w_{msl}=0.8$, $w_s=0.1$, $w_{ov}=0.1$, $w_{gr}=0.4$, $w_{ps}=0.3$ and $w_{sp}=0.3$. The specific offline parameters of the *STriGen* are set at: $Ale=0.1$ and $Cross=0.4$. The online parameters that provide the most accurate results are: $w=3$, $minGRQ=0.90$ and $minIt=20$.

4.1.4.3. Model performance

During the HCS image preprocessing phase, the ground truth is obtained and stored in files. This is achieved from the actual pixel location of each cellular component. The ground truth allows the calculation of traditional performance measurements such as accuracy or f1 score for each tricluster in a specific time point. Table 4.4 shows the obtained accuracy and f1 score values for this experiment, demonstrating the good performance of the model.

Metric	T_1	T_2	T_3
Accuracy	0.9076	0.9083	0.9098
F1 score	0.7026	0.7041	0.6892

Table 4.4: Performance metrics for each tricluster of the HCS images using *STriGen*

The execution time of the streaming part of the algorithm is very important. The online phase provides updated triclusters in an average of 16 seconds. Providing real-time quality triclusters for HeLa cells with *STriGen*. This can help detect external agents added to the exposed cell as substances, drugs or antibodies. In addition, the reaction of the cell can be continuously analyzed by comparing how triclusters components change.

4.2 | Forecasting applications to energy electricity demand

This Section describes the main experimentation carried out with the *StreamWNN* algorithm. The dataset used is introduced in Section 4.2.1 and the selected parameters are in Section 4.2.2. An overview of the results obtained is in Section 4.2.3. This application is described in detail in the scientific papers found in [15, 18].

4.2.1 | Dataset description

Electricity demand forecasting is proving to be very useful for supply chain planning in the energy sector, i.e., energy generation, storage and distribution. Energy demand data are continuous flows of data, also known as data streams, coming from smart meters. The *StreamWNN* is applied to the electricity demand data of Spain with the main goal of improving the efficiency of the electricity demand management in real-time. The algorithm processes and gets responses as new data arrives. In this type of data it is very common for changes to occur over time, so that new patterns, new trends and new behaviors can appear in real-time. In this way, the *StreamWNN* is able not only to provide forecasts in real-time, but also to adapt and adjust the model to these new arriving data, obtaining accurate and timely responses.

The electrical energy consumption of Spain is recorded over time providing one sample every 10 minutes. In particular, in this work the energy demand in megawatt (MW) and the date and time of the measured value are used. This experimentation is carried out on 497,832 samples corresponding to the whole time series data over 9 years and 6 months, starting on January 1st 2007 and ending on June 21st 2016. This set is selected because several benchmark algorithms used exactly this period of the data. These algorithms are used to perform comparisons with the results obtained.

Specifically, once the dataset is preprocessed, it is divided into two sets of data: offline and online or stream data, which correspond to approximately 70% and 30% of the whole data chronologically ordered. The offline data is divided into the oldest 70% of the chronologically sorted data for the set_n and the remaining 30% for the set_p . This division makes the set_n contain the electricity demand values from January 1st 2007 to August 23rd 2011. On the other hand, the set_p contains instances from August 24th 2011 through August 19th 2013.

The algorithm predicts almost 3 years in a real-time manner, i.e., the stream or online dataset covers from August 20th 2013 to June 21st 2016. This incoming flow of data is simulated for the *StreamWNN* since the goal is to study the behavior of the algorithm for a specific set of stream data. Specifically, the data set intended to be used for the online part imitates the behavior of continuous data streams. Each sample of the online set is published in the kafka topic by the kafka producer every few seconds, randomly calculated between 0.1 and 5 seconds.

The whole experimentation is executed in a cluster located in the Data Science and Big Data Laboratory in Pablo de Olavide University. The cluster is made up of 4 nodes: 3 slaves and 1 master. It has 4 Processor Intel (R) Core (TM) i7-5820K CPU with 48 cores and 120 GB of RAM memory.

Figure 4.5 illustrates the average behavior of the electrical energy demand in Spain. In particular, Figure 4.5a represents the mean demand by hour for the whole dataset, Figure 4.5b depicts the average demand by week days and hours and Figure 4.5c shows the mean demand by month and hours. Figure 4.5d represents the average demand by month and its evolution over the years.

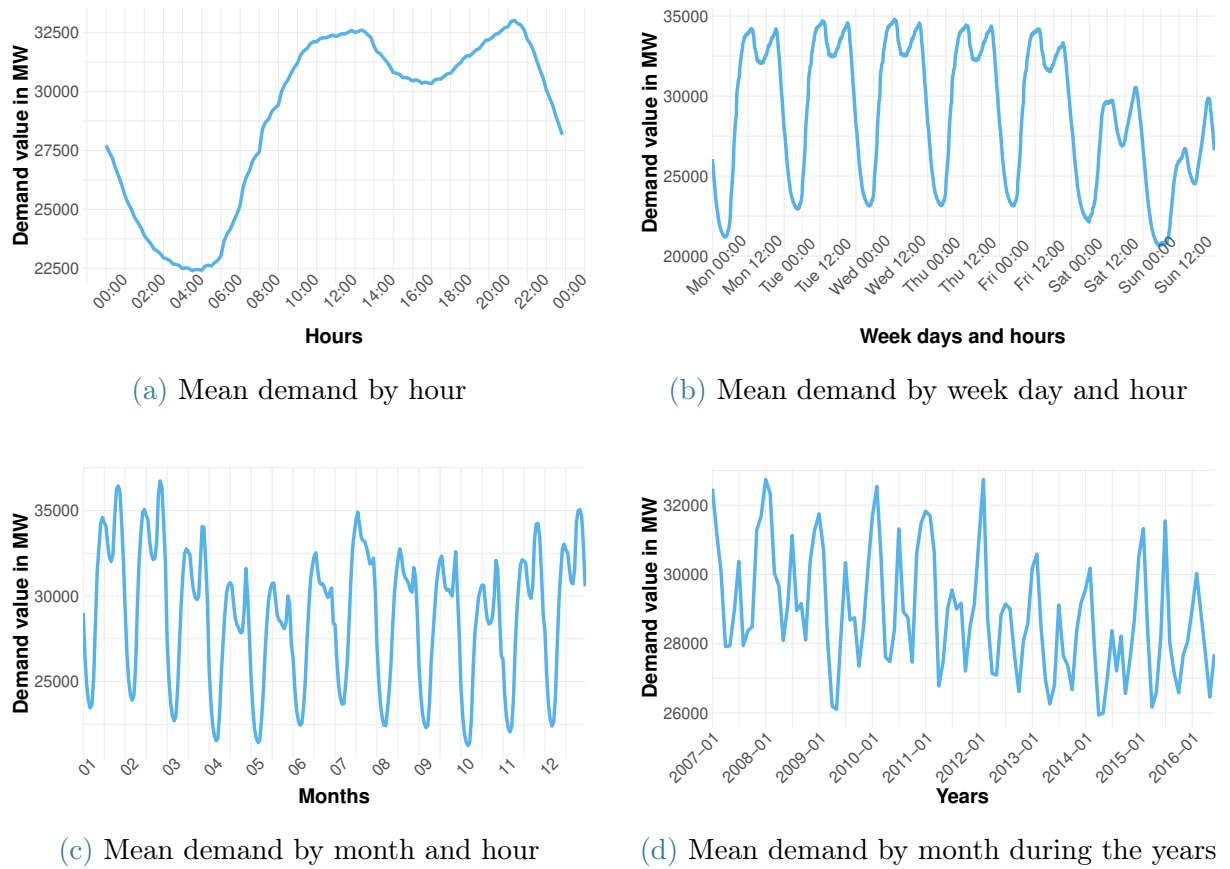


Figure 4.5: Analysis of the energy electricity demand in Spain

4.2.2 | Parameter tuning

The *StreamWNN* is executed with 4 different sets of control parameters to study the influence of the time window w , prediction horizon h and number of neighbors K on the results achieved. These parameters are those considered optimal in several benchmark algorithms using the same Spanish electricity energy demand dataset and are the ones used for the *StreamWNN*:

1. For prediction horizon $h=24$ (4 hours): $w=144$ and $K=4$
2. For prediction horizon $h=48$ (8 hours): $w=288$ and $K=2$
3. For prediction horizon $h=72$ (12 hours): $w=576$ and $K=4$
4. For prediction horizon $h=144$ (24 hours): $w=864$ and $K=4$

The online incremental learning approach of the *StreamWNN* is performed with five different values to study its influence on the results:

- Daily
- Monthly
- Quarterly
- \widehat{y}^{st} larger than $MAPE_{offline}$
- \widehat{y}^{st} larger than $MAPE_{offline} + \sigma_MAPE_{offline}$

with $MAPE_{offline}$ referring to the average $MAPE$ of the offline model and $\sigma_MAPE_{offline}$ to its standard deviation. Therefore the model can perform an incremental learning approach with either a time-based option or a threshold-based option. Each time this requirement is accomplished during the execution, the model can be updated in an incremental manner considering the information in the buffer.

The parameters required for the identification of novelties and anomalies are two thresholds: up^{thr} and low^{thr} . Both of them are based on error metrics obtained from the offline model. This approach is performed in conjunction with the daily incremental learning as it is the option that provides the best results. After a tuning process, the two thresholds for anomalies and novelties detection are:

- $low^{thr} = MAPE_{offline}$
- $up^{thr} = MAPE_{offline} + 3 \times \sigma_MAPE_{offline}$

4.2.3 | Model performance

The metrics used to evaluate the performance of the algorithm are the mean absolute percentage error $MAPE$ defined in Formula 3.23 and the mean absolute error MAE in Formula 3.24. These metrics are expressed in percent and in megawatt MW, respectively.

Table 4.5 presents a summary of the results obtained for each prediction horizon during the streaming or online phase. The Table includes the results for the approach without update of the *StreamWNN*, the incremental learning approach with online daily basis, the novelties update and the two last options executed together, named "online daily + novelties" in the Table 4.5. Four $MAPE$ metrics are calculated for each implementation: its mean value, its standard deviation σ and the best and worst $MAPE$ achieved. Results demonstrate that the most accurate performance is obtained with the incremental learning approach together with the identification of novelties.

Type of update	Mean	σ	Best	Worst
No update	2.4288	2.0745	0.2464	33.0031
Online daily	2.1982	2.0138	0.2198	33.0031
Novelties	2.3139	2.1403	0.2464	21.5966
Online daily + novelties	2.0703	1.9974	0.2463	22.8013

(a) $h=24$

Type of update	Mean	σ	Best	Worst
No update	2.7617	2.0842	0.4101	31.2720
Online daily update	2.5499	2.0878	0.3207	31.2720
Novelties update	2.6486	2.0339	0.4101	31.2720
Online daily + novelties	2.4296	2.0886	0.2685	31.2720

(b) $h=48$

Type of update	Mean	σ	Best	Worst
No update	3.3535	2.8200	0.6002	33.3860
Online daily update	3.1350	2.8039	0.4493	33.3860
Novelties update	3.2692	2.7467	0.6002	33.3860
Online daily + novelties	2.9914	2.7813	0.4493	33.3860

(c) $h=72$

Type of update	Mean	σ	Best	Worst
No update	3.8466	3.6137	0.6548	29.3278
Online daily update	3.5741	3.5292	0.6267	27.0206
Novelties update	3.7585	3.5271	0.6548	29.3278
Online daily + novelties	3.4099	3.4238	0.6302	29.3278

(d) $h=144$

Table 4.5: MAPE error metric (in percentage) for each type of update

4.2.3.1. Incremental learning

The incremental learning approach is analyzed with the five possibilities defined above: daily, monthly, quarterly, forecast \widehat{y}^{st} greater than $MAPE_{offline}$ and forecast \widehat{y}^{st} greater than $MAPE_{offline}$ plus its standard deviation. The best results for all four prediction horizons are obtained with the daily update. The possibility that provides the second best results is the threshold based on $MAPE_{offline}$.

The behavior of this paradigm is based on the evolution of the average euclidean distance between the neighbors in the online model as iterations pass. Figure 4.6 depicts this evolution over iterations for the $h=144$, the behavior of the rest of prediction horizons is similar. When there is no model update, the distance remains the same for all the iterations. In contrast, when the online model is updated with incremental learning, this distance decreases. The update that achieves the lowest average distance is the daily update, which is the same update that achieves the lowest error metrics. Time-dependent updates reduce this average distance in a stepwise fashion.

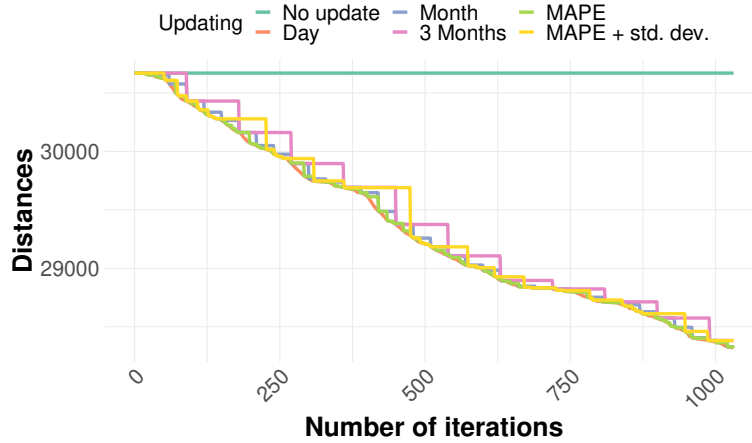


Figure 4.6: Evolution of mean euclidean distance between neighbors in the online model as iterations pass for $h=144$

4.2.3.2. Novelties and anomalies

In the case of the identification of a novelty, the *StreamWNN* online model is updated by including the novelty in it. An interesting example of this update with the Spanish electricity demand is July 2015 which ranked as the warmest July in the historical series. This results in the model not having any similar patterns in the historical data, so updating the model with novelties is key to obtaining accurate results.

Figure 4.7 illustrates the actual demand and forecast results achieved for July 21st

2015. This day was the second worst forecast day for the *StreamWNN* application with no online update. The *MAPE* error achieved without updating matches the error achieved for the daily incremental learning option. This average error is greater than 8%. However, the daily incremental update together with the novelties update provides an error of only 2.9197% which corresponds to a very good improvement of the forecast values.

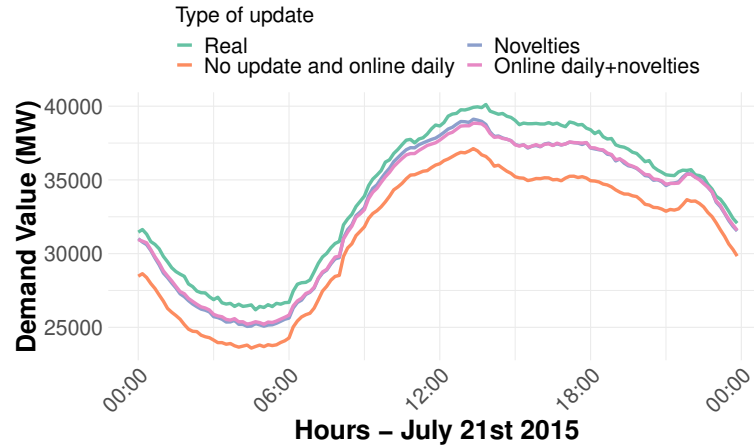


Figure 4.7: Second worst forecast day for no update execution during July 2015

In contrast, anomalies need to be identified in order to trigger an alarm so that the user can act in real-time and prevent from future problems. Correcting the energy demand in the next seconds after an anomaly is detected can help to adjust the demand quickly.

An example of a detected anomaly is depicted in Figure 4.8. It presents the actual energy demand and the forecast of the eight hours that triggered the alarm as an anomaly for the experiment with $h=48$. The anomaly was detected on a Wednesday morning, i.e., September 11th 2013 from 8:00am to 3:50pm. The actual energy demand is much lower than the expected for a weekday at that time. If the alarm had been taken into account, the energy demand for the rest of the day could have been modified for the following hours avoiding more energy demand than necessary. It is important to consider that the forecast of the same day for the following two years does not trigger any alarm which supports the idea that September 11th, 2013 is correctly classified as an anomaly and is not a novelty.

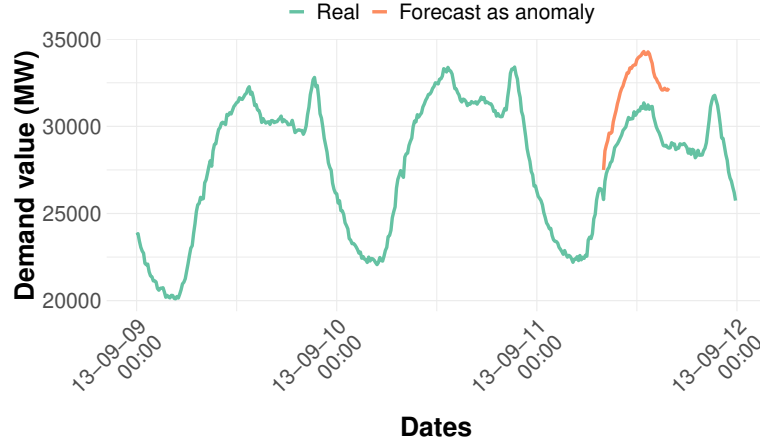


Figure 4.8: Anomaly identified for $h=48$ during September 11th 2013

4.2.3.3. Scalability and timely results

The online execution of the *StreamWNN* has to provide accurate and timely forecasts. Figure 4.9 demonstrates the scalability of the algorithm for the online daily incremental learning along with novelties update. The other update types and the no update option have a similar behavior. It can be said that the model is scalable since the execution time increases linearly with iterations for all prediction horizons. The number of iterations to be computed at each horizon for the same stream dataset is different, since w and h are different in each case. In addition, this assumption influences the execution time difference between $h=72$ and 144 compared to $h=48$ and 24, since the model must be updated with a larger set of data for the two first horizons than for the other two. The average time to perform an iteration is approximately 10 seconds for the first two horizons and 6.66 seconds for the last two.

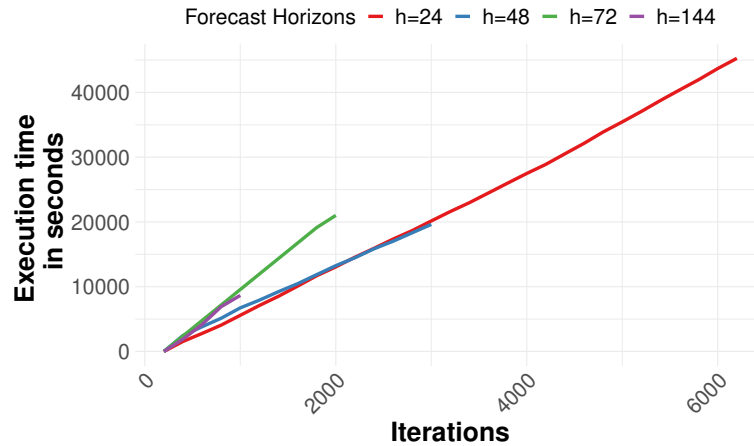


Figure 4.9: Time versus iterations for each horizon for daily incremental + novelties

Part III

List of publications

5 | Publications

SCIENTIFIC contributions published in scientific papers during the PhD program are presented in this Chapter in Section 5.1. The contributions have been sorted in chronological order.

5.1 | Journal and conferences articles

5.1.1 | "Discovering spatio-temporal patterns in precision agriculture based on triclustering"

Authors: Melgar-García L., Godinho M. T., Espada R., Gutiérrez-Avilés D., Brito I. S., Martínez-Álvarez F., Troncoso A., Rubio-Escudero C.

Publication type: Conference article.

Conference: 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020).

Publication: Advances in Intelligent Systems and Computing, Springer International Publishing, Cham.

Year: 2020.

Volume: 1268

Pages: 226-236

DOI: 10.1007/978-3-030-57802-2_22



Discovering Spatio-Temporal Patterns in Precision Agriculture Based on Triclustering

Laura Melgar-García¹(✉), Maria Teresa Godinho^{2,3}, Rita Espada⁴,
David Gutiérrez-Avilés¹, Isabel Sofia Brito^{5,6}, Francisco Martínez-Álvarez¹,
Alicia Troncoso¹, and Cristina Rubio-Escudero⁷

¹ Data Science & Big Data Lab, Pablo de Olavide University, 41013 Seville, Spain
{lmelgar,dgutavi,fmaralv,atrolor}@upo.es

² Department of Mathematical and Physical Sciences, Polytechnic Institute of Beja,
Beja, Portugal
mtgodinho@ipbeja.pt

³ Center for Mathematics, Fundamental Applications and Operations Research,
University of Lisboa, Lisbon, Portugal

⁴ Associação dos Agricultores do Baixo Alentejo, Beja, Portugal
rita.espada.25@gmail.com

⁵ Department of Engineering, Polytechnic Institute of Beja, Beja, Portugal
isabel.sofia@ipbeja.pt

⁶ Instituto de Desenvolvimento de Novas Tecnologias - Centre of Technology
and Systems, Lisbon, Portugal

⁷ Department of Computer Languages and Systems, University of Seville,
Seville, Spain
crubioescudero@us.es

Abstract. Agriculture has undergone some very important changes over the last few decades. The emergence and evolution of precision agriculture has allowed to move from the uniform site management to the site-specific management, with both economic and environmental advantages. However, to be implemented effectively, site-specific management requires within-field spatial variability to be well-known and characterized. In this paper, an algorithm that delineates within-field management zones in a maize plantation is introduced. The algorithm, based on triclustering, mines clusters from temporal remote sensing data. Data from maize crops in Alentejo, Portugal, have been used to assess the suitability of applying triclustering to discover patterns over time, that may eventually help farmers to improve their harvests.

Keywords: Triclustering · Spatio-temporal patterns · Precision agriculture · Remote sensing

1 Introduction

It is a well-established fact that shortage of natural resources endangers our future. Public awareness of these problems urges local authorities to intervene and impose tight regulations on human activity. In this environment, reconciling economic and environmental objectives in our society it is mandatory.

Precision agriculture (PA) has an important role in the pursuit of such aspiration, as the techniques used in PA permit to adjust resource application to the needs of soil and crop as they vary in the field. In this way, specific-site management (that is the management of agricultural crops at a spatial scale smaller than the whole field) is a tool to control and reduce the amount of fertilizers, phytopharmaceuticals and water used on site, with both ecological and economic advantages. Indeed, being able to characterize how crops behave over time, extracting patterns and predicting changes is a requirement of utmost importance for understanding agro-ecosystems dynamics [1].

One of the major concerns associated to the shortage of natural resources is the enormous consumption of water associated to farming activities. Water is a scarce resource worldwide and this problem is particularly acute in the South of Europe, where the Alentejo (Portugal) and Andalusia (Spain) regions are located. Both regions are mainly agriculture-dependent and thus farmers and local authorities are apprehensive about the future.

In this paper, an algorithm is proposed to delineate management zones by measuring the variability of crop conditions within the field through the analysis of time series of geo-referenced vegetation indices, obtained from satellite imagery. In particular, the well-known normalized difference vegetation index (NDVI), indicator for vegetation health and biomass, is used to analyze how the crop varies over time in order to find patterns that may help to improve its production. There are more vegetation indices as GNDVI, SAVI, EVI or EVI2 [2,3] which should be used in extended works.

A triclustering method, based on an evolutionary strategy called TriGen [4] has been applied to a set of satellite images indexed over time from a particular maize crop in Alentejo, Portugal. Although the method was originally designed to discover gene behaviors over time [5], it has also been applied to other research fields such as seismology [6]. The TriGen is a genetic algorithm, and therefore the fitness function is a key aspect since it leads to the discovery of triclusters of different shapes and aspects. The multi-slope measure (MSL) [7], the three-dimensional mean square residue (MSR3D) [8] and the least squared lines (LSL) [9] are the available fitness functions to mine triclusters in TriGen. Furthermore, the TRIclustering quality (TRIQ) index [10] was proposed to validate the results obtained from the aforementioned fitness functions.

The rest of the paper is structured as follows. In Sect. 2, the recent and related works are reviewed and the process of data acquisition and preprocessing is described. In Sect. 3 the proposed algorithm and its adaption to this particular problem are described. In Sect. 4 the results are presented and discussed. Finally, in Sect. 5, the conclusions of this work and point directions for future work are presented.

228 L. Melgar-García et al.

2 Related Works

This section reviews the most recent and relevant works published in the field of spatio-temporal patterns in precision agriculture.

The spatio-temporal pattern discovery issues for satellite time series images are discussed in [11]. The authors introduced how to perform an automatic analysis of these patterns and the problem of determining its optimal number. Unfortunately, these questions are still open issues in the literature and it is unlikely that a general consensus can be reached in the near future.

The estimation of spatio-temporal patterns of agricultural productivity in fragmented landscapes using AVHRR NDVI time series was analyzed in [12]. Four different approaches were applied to eight years of Australian crops, including calculation of temporal mean and standard deviation layers, spatio-temporal key NDVI patterns, different climatic variables and relationships between productivity and production.

In Fung et al. [13], the authors proposed a novel spatio-temporal data fusion model for satellite images using Hopfield Neural Networks. Synthetic and real datasets from both Hong Kong and Australia, respectively, were used to assess the method performance, showing remarkable results and outperforming some of other existing methods.

The use of convolutional neural networks (CNN) is being currently applied in a wide range of spatio-temporal patterns discovery applications [14]. Hence, Tan et al. [15] enhanced an existing CNN model for image fusion by proposing a new network architecture and a novel loss function. Results showed superior performance in terms of accuracy and robustness. Ji et al. [16] proposed a 3D CNN dealing with multi-temporal satellite images. In this case, the method was designed for crop classification. After discussing the results achieved, outperforming existing well-established methods, the authors claimed that it is especially suitable for characterizing crop growth dynamics.

An ensemble model for making spatial predictions of tropical forest fire susceptibility using multi-source geospatial data can be found in [17]. The authors evaluated the Lao Cai region, Vietnam, through several indices including NDVI.

Bui et al. [18] proposed an approach based on deep learning for predicting flash flood susceptibility. Real data from a high frequency tropical storm area were used to assess its performance.

Clustering-based approaches with application to precision agriculture can also be found in the literature. Thus, clustering tools for integration of satellite imagery and proximal soil sensing data are described in [19]. In particular, a novel method was introduced with the aim of determining areas with homogeneous parts in agricultural fields.

The application of triclustering to georeferenced satellite images time series can be also found in [20]. However, the authors addressed a different problem: the patterns analysis of intra-annual variability in temperature, using daily average temperature retrieved from Dutch stations spread over the country.

3 Methodology

This section introduces the *TriGen* algorithm, the methodology used to extract behavior patterns from satellite images along with the time points when they were taken. This methodology is applied to a 3D dataset (composed of rows, columns, and depths) that represents the X-axis coordinates (rows) and the Y-axis coordinates (columns) of each satellite image taken at a particular instant (depth). *TriGen* is a genetic algorithm that minimizes a fitness function to mine subsets of X-axis coordinates, Y-axis coordinates, and time points, called triclusters, from 3D input datasets. The *NDVI* values in the yielded subsets of $[X, Y]$ coordinates along with the subset of time points, share similar behavior patterns.

In general terms, *TriGen* is explained from two main concepts, presented in the following sections: the triclustering model applied to the case study (Sect. 3.1) and the inputs, output and algorithm workflow of *TriGen* (Sect. 3.2).

3.1 Triclustering

The case study presented has been modeled as a triclustering problem, in which 3-dimensional patterns are extracted from an original dataset. Prior to explaining this development, it is necessary to distinguish between two types of dataset:

- D_{2D} (2-dimensional dataset): a matrix with a set of instances (rows) and a set of features (columns).
- D_{3D} (3-dimensional dataset): a 3D matrix with a set of instances (rows) and features (columns), taken at a particular time points (depths).

Clustering algorithms are applied to D_{2D} datasets performing a complete partition it; for each yielded clusters, the values of the grouped instances share a behavior pattern through all features. In contrast, the triclustering algorithms work with D_{3D} datasets and group not only subsets of instances, but also subsets of features and time points. In this case, for each yielded tricluster, the values of grouped instances for the particular grouped features share a behavior pattern through a group of time points.

Thus, for this case study, the application of the *TriGen* algorithm to a D_{3D} dataset of satellite images where the instances are the Y coordinates of the space, the features are the X coordinates of the area and, the time points are the moment at the images where taken, will yield a set of triclusters representing, each of them, a behavior pattern of *NDVI*, for a particular subspace (subset of Y and X coordinates) through a specific set of times (subset of time points).

3.2 The *TriGen* Algorithm

In order to mine the triclusters from the D_{3D} dataset of satellite images, the *TriGen* algorithm is applied. *TriGen* is based on the genetic algorithm paradigm;

230 L. Melgar-García et al.

therefore, it evolves a population of individuals employing genetic operators during a specific number of generations to optimize an evaluation function.

The inputs of *TriGen* are two: the D_{3D} dataset of satellite images and the initial configuration of the genetic process. The parameters that can be set are the number of triclusters to mine (N), the number of generations of the genetic process (G), the size of the initial population (I), the fraction of population that promoted to the next generation (Sel) and, the probability of mutation (Mut). A complete analysis of the influence of these parameters in the performance of the algorithm can be consulted in [4, 7, 8].

Each individual in the genetic process is represented as a tricluster and composed of a subset of instances of D_{3D} , a subset of features of D_{3D} and, a subset of time points of D_{3D} ; the individuals (triclusters) with the best fitness function value are the output of the algorithm.

The genetic operators allow for searching among the individuals to obtain better solutions for each generation. For the *TriGen* algorithm, the description of them is the following:

- Initial population. The individuals are generated with three methods. The first method consists in a random selection of the elements of the individuals. The second one, considering the rows and columns of D_{3D} as a geographical area, performs a random selection of a rectangular sub-area and time points. The last one selects the elements of the individuals taking into account the rows, columns, and time points of D_{3D} visited in already extracted solutions in order to explore the most number of elements of D_{3D} .
- Evaluation. This operator applies the fitness function to the population in order to assess the quality of each individual. The fitness function used in the present case study is *MSL*.
- Selection. A tournament selection algorithm is applied to promote the individuals with the best evaluation to the next generation. The rest of individuals in the next population are generated by crossing and mutations.
- Crossover. Two individuals are combined to generate another two ones. The crossover used is the one point crossing. Each of the three elements of the two involved individuals (parents), are split in two and the four parts are combined two new individuals (offspring).
- Mutation. This operator modifies an individual to obtain variability in the next generation. Three actions have been used: insertion of a new coordinate $[X, Y]$ or time point, deletion of an existing coordinate $[X, Y]$ or time point and change of an existing coordinate $[X, Y]$ or time point.

4 Results

This section reports and discusses the results achieved after the application of the proposed methodology to a particular dataset. Thus, Sect. 4.1 describes the high resolution remote sensing imagery used in this study and Sect. 4.2 introduces the validation function used to evaluate the quality of the triclusters obtained. Finally, Sect. 4.3 reports the spatio-temporal patterns obtained and discusses its physical meaning.

4.1 Dataset Description

Located in the Baixo Alentejo region of Portugal, the site under study is a 63.82 ha maize plantation, with center at coordinates $(38^{\circ}08'12''N, 7^{\circ}53'42''W)$, as shown in Fig. 1. The site was monitored between sowing (April of 2018) and harvesting (September of the same year) and it is characterized by a set of nineteen images retrieved at time intervals of five, ten and fifteen days, from the Sentinel 2 Mission. The research site was irrigated using a central pivot irrigation system.



Fig. 1. Location of the research site.

Vegetation indices are, by definition, algebraic combinations of the measured canopy reflectance of different wavelength bands [21]. The use of Vegetation Indices in this context is based on the fact that healthy and unhealthy plants reflect light differently. Due to this difference, crop canopy multispectral reflectance, which is detectable remotely through aerial or satellite imagery, can be used to monitor the state of the crop [22]. For these reasons, one of the most widely used indices is applied to the images: the Normalized Differential Vegetation Index (NDVI). The NDVI can be calculated as follows:

$$NDVI = \frac{NIR - Red}{NIR + Red}, \quad (1)$$

where *Red* and *NIR* stand for the spectral reflectance measurements acquired in the red (visible) and near-infrared regions, respectively, and $NDVI \in [-1, 1]$.

As pointed out in [23], the NDVI index has proven to be quite useful in monitoring variables such as crop nutrient deficiency, final yield in small grains, and long-term water stress. All these variables are very important to the case study presented here. Figure 2 illustrates how the NDVI of the target area varies over time, including images at six different chronologically ordered time stamps.

232 L. Melgar-García et al.

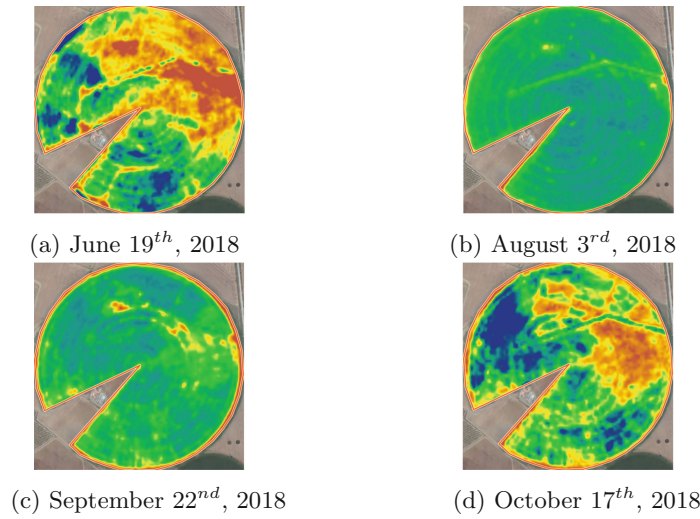


Fig. 2. Sample NDVI values for the research site, chronologically ordered.

4.2 Behaviour Patterns Quality, the *TRIQ* Measure

The *TRIQ* index has been used in order to measure the quality of the yielded triclusters in this case study, that is, the quality of the behavior pattern that a tricluster depicts. *TRIQ* measures the quality of a tricluster based on three elements: the similarity of the behavior patterns of the grouped $[X, Y]$ points along with the grouped time points and the Pearson's and Spearman's correlation indexes between all the $[X, Y]$ time series of the tricluster. *TRIQ* values rank in the $[0, 1]$ interval; *TRIQ* is a measure to maximize. A full description, definition, development, and performance of *TRIQ* can be consulted in [10].

4.3 Discovery of Spatio-Temporal Patterns in Maize Crops

TriGen analyzes the evolution of NDVI indices in each specific area and discovers triclusters of similar behavior patterns. Thus, the dataset with the NDVI indices of the satellite images over time is the first input of the algorithm.

TriGen has some configuration parameters, above-mentioned in Sect. 3.2. The algorithm has been run several times with different settings for each parameter. The configuration parameters that fit the best to these images are: $G = 10$, $I = 200$, $Sel = 0.8$ and $Mut = 0.1$. The number of triclusters to find is 4 and the fitness function used is *MSL*. Therefore, these values are the second input of the algorithm.

Each of the 4 discovered triclusters has a *TRIQ* measure. The first one has a *TRIQ* of 0.803, the second has 0.753, the third has 0.827 and the fourth has 0.742. These high values lead to confirm the good quality of all the triclusters. However, this measure itself does not guarantee the meaningfulness of the triclusters discovered. In order to interpret the evolution of the triclusters in an

accurate way, field's farmers provided additional information about the plantations site-specific conditions, such as irrigation or fungicide, for the same period. This information confirmed that triclusters were meaningful also in geophysical terms.

The triclusters discovered are represented in Figs. 3a, 3b, 3c and 3d. Each graph represents the evolution of the NDVI of the selected $[X, Y]$ components over time. The black dashed line added in each graph represents the mean value of all components. Triclusters components share a similar behavior. The first tricluster corresponds to areas with high NDVI values that remain almost constant over time. The components of the second tricluster are fields that start with a high NDVI and experiment a sudden decrease for the rest of the dates studied. The beginning of the third tricluster is similar to the previous one but with a recovery of the initial values after mid September. The last tricluster is formed by areas with constant low NDVI over time.

The changes of the NDVI values identified by triclusters 1, 2 and 3 during the first samples seem to be related with the use of fertilizers and the increase of the amount of water for the irrigation process. The third tricluster and some components of the first one show a change in their behaviour at mid September. It could be related to the application of fungicide by the farmers during August.

The proposed algorithm contributes in finding areas of similar crop conditions over the NDVI vegetation index using satellite images in different times. In addition, as TriGen includes the time dimension, the evolution over time of

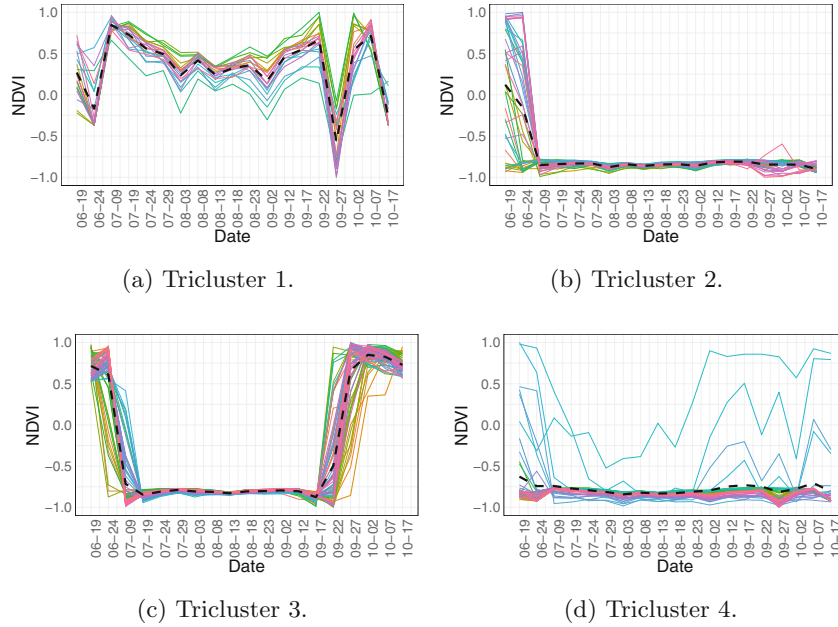


Fig. 3. Triclusters found by *TriGen* in 2018.

234 L. Melgar-García et al.

each tricluster's features can be analyzed. Nevertheless, the interpretation of the results needs the validation of a specialist as the TRIQ measure does not consider neither geographical nor environmental features.

5 Conclusions

The suitability of applying triclustering methods to discover spatio-temporal patterns in precision agriculture has been explored in this work. In particular, a set of satellite images from maize crops in Alentejo, Portugal, has been analyzed in terms of its NVDI temporal evolution. Several patterns have been found, identifying zones with tendency to obtain greater production and others in which human interventions are required to improve the soil properties. Several issues remain unsolved and are suggested to be addressed in future works. First, these patterns may help to identify the most suitable moments to apply fertilizers or pesticides. Second, the forecasting of maize production could be done based on such patterns. Third, additional crop production features such as amounts and characteristics of the fertilizers, phytopharmaceuticals and water used throughout the season (moister probes placed 30 cm underground were used to access the soil need for water before irrigation, when needed), would help to discover more robust patterns. Fourth, more images records during more years and a specific measure to assess the quality and meaning of precision agriculture triclusters would improve the application of the proposed algorithm to agricultural production. Fifth, more vegetation indices should be used.

Acknowledgements. The authors would like to thank the Spanish Ministry of Economy and Competitiveness for the support under project TIN2017-88209 and Fundação para a Ciência e a Tecnologia (FCT), under the project UIDB/04561/2020. The authors would also like to thank António Vieira Lima for giving access to data and Francisco Palma for his support to the whole project.

References

1. Tan, J., Yang, P., Liu, Z., Wu, W., Zhang, L., Li, Z., You, L., Tang, H., Li, Z.: Spatio-temporal dynamics of maize cropping system in Northeast China between 1980 and 2010 by using spatial production allocation model. *J. Geog. Sci.* **24**(3), 397–410 (2014)
2. Jurecka, F., Lukas, V., Hlavinka, P., Semerádova, D., Zalud, Z., Trnka, M.: Estimating crop yields at the field level using landsat and modis products. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis* **66**, 1141–1150 (2018)
3. Jiang, Z., Huete, A., Didan, K., Miura, T.: Development of a two-band enhanced vegetation index without a blue band. *Remote Sens. Environ.* **112**, 3833–3845 (2008)
4. Gutiérrez-Avés, D., Rubio-Escudero, C., Martínez-Álvarez, F., Riquelme, J.C.: Tri-gen: A genetic algorithm to mine triclusters in temporal gene expression data. *Neurocomputing* **132**, 42–53 (2014)

5. Melgar, L., Gutiérrez-Avilés, D., Rubio-Escudero, C., Troncoso, A.: High-content screening images streaming analysis using the STriGen methodology. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pp. 537–539 (2020)
6. Martínez-Álvarez, F., Gutiérrez-Avilés, D., Morales-Esteban, A., Reyes, J., Amaro-Mellado, J.L., Rubio-Escudero, C.: A novel method for seismogenic zoning based on triclustering: application to the Iberian peninsula. *Entropy* **17**(7), 5000–5021 (2015)
7. Gutiérrez-Avilés, D., Rubio-Escudero, C.: MSL: a measure to evaluate three-dimensional patterns in gene expression data. *Evol. Bioinform.* **11**, 121–135 (2015)
8. Gutiérrez-Avilés, D., Rubio-Escudero, C.: Mining 3D patterns from gene expression temporal data: a new tricluster evaluation measure. *Sci. World J.* **2014**, 1–16 (2014)
9. Gutiérrez-Avilés, D., Rubio-Escudero, C.: LSL: a new measure to evaluate triclusters. In: *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine*, pp. 30–37 (2014)
10. Gutiérrez-Avilés, D., Giraldez, R., Gil-Cumbreras, F.J., Rubio-Escudero, C.: TRIQ: a new method to evaluate triclusters. *BioData Min.* **11**(1), 15 (2018)
11. Radoi, A., Datcu, M.: Spatio-temporal characterization in satellite image time series. In: *Proceedings of the International Workshop on the Analysis of Multitemporal Remote Sensing Images*, pp. 1–4 (2015)
12. Hill, M.J., Donald, G.E.: Estimating spatio-temporal patterns of agricultural productivity in fragmented landscapes using AVHRR NDVI time series. *Remote Sens. Environ.* **84**(3), 367–384 (2003)
13. Fung, C.H., Wong, M.S., Chan, P.W.: Spatio-temporal data fusion for satellite images using Hopfield neural network. *Remote Sens.* **11**(18), 2077 (2019)
14. Kamilaris, A., Prenafeta-Boldú, F.: A review of the use of convolutional neural networks in agriculture. *J. Agric. Sci.* **156**(3), 312–322 (2018)
15. Tan, Z., Di, L., Zhang, M., Guo, L., Gao, M.: An enhanced deep convolutional model for spatiotemporal image fusion. *Remote Sens.* **11**(18), 2898 (2019)
16. Ji, S., Zhang, C., Xu, A., Shi, Y., Duan, Y.: 3D convolutional neural networks for crop classification with multi-temporal remote sensing images. *Remote Sens.* **10**(1), 75 (2018)
17. Tehrany, M.S., Jones, S., Shabani, F., Martínez-Álvarez, F., Bui, D.T.: A novel ensemble modeling approach for the spatial prediction of tropical forest fire susceptibility using logitboost machine learning classifier and multi-source geospatial data. *Theoret. Appl. Climatol.* **137**, 637–653 (2019)
18. Bui, D.T., Hoang, N.-D., Martínez-Álvarez, F., Ngo, P.-T.T., Hoa, P.V., Pham, T.D., Samui, P., Costache, R.: A novel deep learning neural network approach for predicting flash flood susceptibility: a case study at a high frequency tropical storm area. *Sci. Total Environ.* **701**, 134413 (2020)
19. Saifuzzaman, M., Adamchuk, V., Buelvas, R., Biswas, A., Prasher, S., Rabe, N., Aspinall, D., Ji, W.: Clustering tools for integration of satellite remote sensing imagery and proximal soil sensing data. *Remote Sens.* **11**(9), 1036 (2019)
20. Wu, X., Zurita-Milla, R., Izquierdo-Verdiguier, E., Kraak, M.-J.: Triclustering geo-referenced time series for analyzing patterns of intra-annual variability in temperature. *Ann. Am. Assoc. Geogr.* **108**, 71–87 (2018)

236 L. Melgar-García et al.

21. Schueller, J.: A review and integrating analysis of spatially-variable control of crop production. *Fertil. Res.* **33**, 1–34 (1992)
22. Xue, J., Su, B.: Significant remote sensing vegetation indices: a review of developments and applications. *J. Sens.* **17**, 1353691 (2017)
23. Govaerts, B., Verhulst, N.: The normalized difference vegetation index (NDVI) GreenSeeker™ handheld sensor: toward the integrated evaluation of crop management. *CIMMYT* (2010)

5.1.2 | "High-content screening images streaming analysis using the STriGen methodology"

Authors: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A.

Publication type: Conference article.

Conference: The 35th ACM/SIGAPP Symposium on Applied Computing (SAC 2020).

Publication: Association of Computing Machinery.

Year: 2020.

Pages: 537-539

DOI: 10.1145/3341105.3374071

Ranking: GGS class (rating): 2 (A-)

High-Content Screening images streaming analysis using the STriGen methodology

Laura Melgar-García

Data Science & Big Data Lab, Pablo de Olavide University
Seville, Spain
lmelgar@upo.es

David Gutiérrez-Avilés

Data Science & Big Data Lab, Pablo de Olavide University
Seville, Spain
dgutavi@upo.es

Cristina Rubio-Escudero

Department of Computer Science, University of Seville
Seville, Spain
crubioescudero@us.es

Alicia Troncoso

Data Science & Big Data Lab, Pablo de Olavide University
Seville, Spain
atolor@upo.es

ABSTRACT

One of the techniques that provides systematic insights into biological processes is High-Content Screening (HCS). It measures cells phenotypes simultaneously. When analysing these images, features like fluorescent colour, shape, spatial distribution and interaction between components can be found. STriGen, which works in the real-time environment, leads to the possibility of studying time evolution of these features in real-time. In addition, data streaming algorithms are able to process flows of data in a fast way. In this article, STriGen (Streaming Triclustering Genetic) algorithm is presented and applied to HCS images. Results have proved that STriGen finds quality triclusters in HCS images, adapts correctly throughout time and is faster than re-computing the triclustering algorithm each time a new data stream image arrives.

CCS CONCEPTS

• **Information systems** → **Clustering**; **Data stream mining**; • **Computing methodologies** → **Genetic algorithms**; • **Applied computing** → *Molecular evolution*; *Imaging*;

KEYWORDS

Real-time, Triclustering, Genetic operators, High-Content Screening

ACM Reference Format:

Laura Melgar-García, David Gutiérrez-Avilés, Cristina Rubio-Escudero, and Alicia Troncoso. 2020. High-Content Screening images streaming analysis using the STriGen methodology. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20)*, March 30–April 3, 2020, Brno, Czech Republic. ACM, New York, NY, USA, Article , 3 pages. <https://doi.org/10.1145/3341105.3374071>

1 INTRODUCTION

Nowadays, one of the biggest challenges in biology is understanding genes and their biological circuits. Due to that, techniques that

investigate complete cellular processes are becoming more relevant, as High-Content Screening (HCS). HCS combines an automated imaging and analysis of intact cells exposed to some perturbations (chemical or genomic) that alter their phenotype [11].

HCS is made by many steps that can take too much time if they are not effectively done. On the other hand, these days, stream computing trends are rising, i.e., algorithms that react in the fastest way to provide information from data in real time. Consequently, the processing and analysis of HCS images in a streaming environment could give quick information from them.

In [8] the TriGen algorithm is presented as a Triclustering algorithm that discovers groups of 3D datasets throughout instances, attributes and time. In [12] STriGen algorithm is introduced. STriGen is a new incremental learning method that finds groups of similar behaviour patterns in 3D stream data continuously. In this paper, STriGen algorithm is applied to HCS images to get information from images in real-time.

The article is structured as follows: STriGen and HCS methodologies are presented in Section 2; the experimental setup and the yielded results in Section 3; and finally the conclusions are in Section 4.

2 METHODOLOGY

2.1 STriGen methodology

STriGen is a new incremental learning method that creates triclusters (based on TriGen algorithm [8]) and keeps them updated taking into account the knowledge from previous streams and upgrading its learning method. STriGen meets all 4 Data Streaming requirements, i.e., data has to be processed in the order of its arrival and one by one; the learning model has to be updated incrementally; the model has to deal with small amount of memory referring to the huge quantity of data and it has to be fast.

STriGen algorithm is inspired in the offline/online approach of stream algorithms. Consequently, STriGen starts with an execution of the TriGen modified to treat stream data as static data with W data, where W is the maximum number of data streams that can be used in an iteration of the STriGen algorithm. Afterwards, the algorithm processes each new data stream and the model updates incrementally and quickly basing itself on the new streams in order to provide the upgraded triclusters.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SAC '20, March 30–April 3, 2020, Brno, Czech Republic

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-6866-7/20/03.

<https://doi.org/10.1145/3341105.3374071>

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

L. Melgar-García et al.

During this second phase of STriGen, it tries to extend the actual triclusters throughout time removing the "oldest" time point to keep always a maximum of W data points, a regular procedure in Data Streaming algorithms [6]. In addition to the extension of the actual triclusters over time, the learning model adjusts itself incrementally depending on the GRQ (GRaphical Quality) measure, part of one of the TriGen fitness functions [7].

More specifically, the STriGen learning model makes mutations into triclusters, i.e., adding, deleting and/or changing both instances and/or attributes, to be updated. These operations are quicker than re-training TriGen each time a new stream arrives to keep the learning model updated. Mutations allow to find current and global real solutions as STriGen depends on the results from the first execution of TriGen that can change every time it is executed. In this way, triclusters are mutated until their GRQ values are higher than $minGRQ$ or until the number of iterations made is higher than $numIt$. In this way, STriGen tries to include or remove some current tricluster's components in order to keep most accurate components. These 2 parameters and also the "window" W parameter and a minimum GRQ threshold (that can delete the oldest time included in the tricluster when its GRQ is smaller than this) take different values depending on the dataset, to be able to adjust to small or/and abrupt changes in streams.

When the dataset is synthetic or when the resulting triclusters are known in advance, a validation process to compare founded and real triclusters is done with accuracy and F1 Score measures. Datasets that are neither synthetic nor with known triclusters in advance, are evaluated depending on the GRQ value.

2.2 High-Content Screening methodology

High-Content Screening or Analysis (HCS or HCA) is gaining importance. HCS combines automated imaging acquisition and image analysis using, most frequently, automated fluorescence microscopy [4]. During the HCS process intact cells are incubated with substances that alter their phenotype in a desired way [3]. These cells are screened and multiple fluorescence readouts are measured in parallel. This process provides big volume of data with high biological information content. Subcellular locations and fluorescence colour intensity during different complex cellular events, in terms of space and time, are measured with the automated image analysis phase of HCS [5]. HCS images have been useful to detect and study DNA, cytokinesis, cell division, cell migration, apoptosis, mitosis, and more cellular events of target components.

HCS processes involve different tasks as cell preparation and labelling, image acquisition, image analysis and data management [9]. In terms of data challenges, HCS has 2 principal issues: data storage and data processing. One of the main interesting points of HCS is the simultaneous analysis of images that requires a high computing power and quickly computer network connections [1].

2.3 STriGen application to High-Content Screening images

Applying STriGen to HCS images allows to discover the best features to group to get information from cells. Actual numerical features extracted from HCS images are: 1) fluorescent marker colour; 2) cell component's shape; 3) spatial situation; 4) distribution of

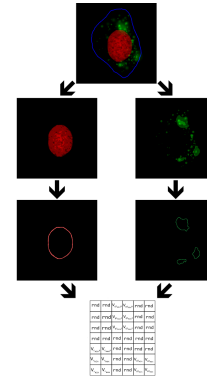


Figure 1: Example of preprocessing phase of HCS images

pixel-colour in a cell region of interest to study interactions or co-occurrences [11]. Moreover, evolution throughout time is added to these 4 features when applying STriGen to HCS.

The type of images used in this experiment are individual intact cells fluorescent microscopy-based HCS images. Images are processed in order to create a dataset that fits STriGen requirements. Firstly, each RGB image is transformed into a 3D matrix representing the coordinates of each image pixel in decimal numbers. In other words, values that represent images colours are the pixel-colour or fluorescent-marker colour features mentioned above. A filter is passed through every image to not include any image that present noise due to optic aberrations, microscopy issues or even bad acquisition of the image. Secondly, data is prepared to fit STriGen dataset specifications, i.e., taking into account: detecting the colours specified to analyse and detecting areas limits (for areas with more than an user fixed value). In addition, random values between 0 and 1000 are added in order to make STriGen able to ignore the background of images. A graphical example of this methodology is in Fig. 1.

3 RESULTS AND DISCUSSION

In this section, the results obtained by the application of the STriGen algorithm to a HCS images dataset are presented.

STriGen has been applied to a set of HCS images from [2] of HeLa cells (cervical cancer cells taken from a woman in the 50s that can divide themselves an unlimited number of times in well preserved laboratory conditions [10]). For this experiment, the dataset selected is the one that presents the reaction of HeLa cells to Transferrin receptors (usually applied as cancer cell target because they enhances site-specific therapies). The quality of the resulting triclusters is evaluated with the GRQ measure. In addition, for this experiment, a dataset with the desired STriGen founded triclusters has been created in order to compute F1 Score and Accuracy measures to check the performance of the algorithm.

460 HCS images similar to the above image in Fig. 1 have been used for this experiment with black for the background image, blue for the cell border, red for the nuclear DNA and green for the endosome compartment. Cell borders (blue colour) has been ignore because they do not provide extra information about HCS features.

High-Content Screening images streaming analysis using the STriGen methodology SAC '20, March 30-April 3, 2020, Brno, Czech Republic

Table 1: STriGen configuration parameters

Execution	$minGRQ$	$thresholdGRQ$	$numIt$
1	0.95	0.80	10
2	0.88	0.70	15
3	0.90	0.75	20

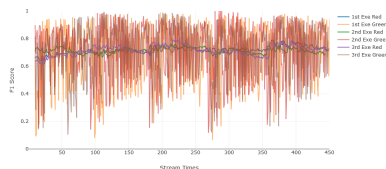


Figure 2: F1 Score results

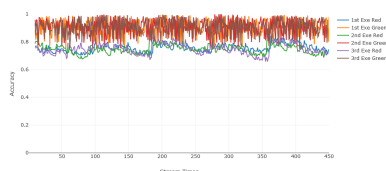


Figure 3: Accuracy results

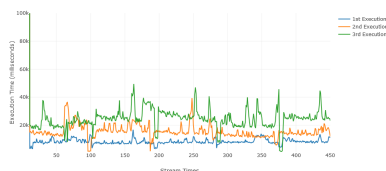


Figure 4: STriGen execution time

STriGen has been executed 3 times due to the fact that the first triclusters results depend on the first triclusters founded by TriGen. In that way, configuration parameters take different values in each execution to see the influence of them in the results (see Table 1) excepting W that has as maximum value 3.

STriGen performance results are in the following figures: Fig. 3 shows accuracy values for all triclusters and Fig. 2 shows F1 score values. Figures show that the algorithm performs in an accurate way and can find components correctly. F1 score of the green triclusters in all 3 executions varies a lot, it is due to the fact that green areas spread through cells and are in continuous movement, however the mean accuracy value in all 3 executions is 0.908. Red areas are more stable in both measures because they are in a similar position in all streams.

Apart from these measures, another important parameter that represents the good performance of the algorithm is time execution. This experiment has been done in a computer with an i7-5820K

3.3GHz processor and 48GB RAM memory. The first phase of STriGen (that is just one execution of TriGen with the first 10 streams) takes a mean of 8.9 minutes to execute completely. Afterwards, the Data Streaming phase starts and each new image stream is processed and provides founded triclusters in a mean of 16 seconds.

In general, the quality measures are mostly equal in the 3 executions. However, it can be seen that the execution time of the 3rd execution of STriGen is much smaller, due mostly to the small value of $numIt$.

4 CONCLUSIONS

The STriGen algorithm performs with good results when dealing with stream data as we have seen in Section 3. It is faster than executing the TriGen algorithm when a new stream arrives and so the evolution of data throughout time can be analysed in real-time. It proves that the learning model updates incrementally making mutations and taking into account the W most recent streams.

The application of HCS images into the Data Streaming environment with STriGen leads the possibility of obtaining information about the evolution of HCS features, i.e. components colour, shape, location and distribution, throughout time in real-time. A possible application of this experiment would be the fact that STriGen could detect when external agents like substances, drugs, antibodies, etc are added to the exposed cell and how the cell reacts to them, e.g., changing triclusters components. It allows to do a continuous analysis of the cell in real-time.

ACKNOWLEDGMENTS

Authors thank the Spanish Ministry of Economy and Competitiveness for the support under the project TIN2017-88209-C2-1-R and TIN2017-88209-C2-2-R.

REFERENCES

- [1] M. Bickle. 2008. High-content screening : A new primary screening tool ? 11, 11 (2008).
- [2] CellOrganizer Project [n. d.]. CellOrganizer project from Carnegie Mellon University. <http://www.cellorganizer.org/2d-hela/>
- [3] D. Cronk. 2013. Chapter 8 - High-throughput screening. (2013), 95 – 117. <https://doi.org/10.1016/B978-0-7020-4299-7.00008-1>
- [4] R. Flaumenhaft. 2007. 3.07 - Chemical Biology. (2007), 129 – 149. <https://doi.org/10.1016/B0-08-045044-X/00080-8>
- [5] G. Galea and J. C. Simpson. 2013. Chapter 17 - High-Content Screening and Analysis of the Golgi Complex. 118 (2013), 281 – 295. <https://doi.org/10.1016/B978-0-12-417164-0.00017-3>
- [6] M. Ghemoune, M. Lebbah, and H. Azzag. 2016. State-of-the-art on clustering data streams. *Big Data Analytics* 1, 1 (2016), 1–27. <https://doi.org/10.1186/s41044-016-0011-3>
- [7] D. Gutiérrez-Avilés, R. Giraldez, F.J. Gil-Cumbreras, and C. Rubio-Escudero. 2018. TRIQ: A new method to evaluate triclusters. *BioData Mining* 11, 1 (2018), 1–29. <https://doi.org/10.1186/s13040-018-0177-5>
- [8] D. Gutiérrez-Avilés, C. Rubio-Escudero, F. Martínez-Álvarez, and J.C. Riquelme. 2014. TriGen: A genetic algorithm to mine triclusters in temporal gene expression data. *Neurocomputing* 132 (2014), 42–53. <https://doi.org/10.1016/j.neucom.2013.03.061>
- [9] S. Lee and B. J. Howell. 2006. [25] - High-Content Screening: Emerging Hardware and Software Technologies. 414 (2006), 468 – 483. [https://doi.org/10.1016/S0076-6879\(06\)14025-2](https://doi.org/10.1016/S0076-6879(06)14025-2)
- [10] B.P. Lucey, W.A. Nelson-Rees, and G.M. Hutchins. 2009. Henrietta Lacks, HeLa cells, and cell culture contamination. *Archives of Pathology and Laboratory Medicine* 133, 9 (2009), 1463–1467.
- [11] F. Heigwer M. Boutros and C. Laufer. 2015. Microscopy-based High-content Screening. *Cell* 163, 6 (2015), 1314–1325. <https://doi.org/10.1016/j.cell.2015.11.007>
- [12] L. Melgar-García, D. Gutiérrez-Avilés, and C. Rubio-Escudero. 2019. Discovering Behavior Patterns in Big Data Streaming Environments : The STriGen Methodology. (2019). Manuscript submitted for publication.

5.1.3 | "Coronavirus Optimization Algorithm: A Bioinspired Metaheuristic Based on the COVID-19 Propagation Model"

Authors: Martínez-Álvarez F., Asencio-Cortés G., Torres JF., Gutiérrez-Avilés D., Melgar-García L., Pérez- Chacón R., Rubio-Escudero C., Riquelme J. C., Troncoso A.

Publication type: Journal article.

Journal: Big Data.

Year: 2020.

Volume: 8 (4)

Pages: 308-322

DOI: 10.1089/big.2020.0051

IF: 3.644, 15/108 Computer Science, Theory and Methods.

Quartil: Q1.

ORIGINAL ARTICLE

Coronavirus Optimization Algorithm: A Bioinspired Metaheuristic Based on the COVID-19 Propagation Model

F. Martínez-Álvarez,^{1,*} G. Asencio-Cortés,¹ J. F. Torres,¹ D. Gutiérrez-Avilés,¹ L. Melgar-García,¹ R. Pérez-Chacón,¹
C. Rubio-Escudero,² J. C. Riquelme,² and A. Troncoso¹

Abstract

This study proposes a novel bioinspired metaheuristic simulating how the coronavirus spreads and infects healthy people. From a primary infected individual (patient zero), the coronavirus rapidly infects new victims, creating large populations of infected people who will either die or spread infection. Relevant terms such as re-infection probability, super-spreading rate, social distancing measures, or traveling rate are introduced into the model to simulate the coronavirus activity as accurately as possible. The infected population initially grows exponentially over time, but taking into consideration social isolation measures, the mortality rate, and number of recoveries, the infected population gradually decreases. The coronavirus optimization algorithm has two major advantages when compared with other similar strategies. First, the input parameters are already set according to the disease statistics, preventing researchers from initializing them with arbitrary values. Second, the approach has the ability to end after several iterations, without setting this value either. Furthermore, a parallel multiviral version is proposed, where several coronavirus strains evolve over time and explore wider search space areas in less iterations. Finally, the metaheuristic has been combined with deep learning models, to find optimal hyperparameters during the training phase. As application case, the problem of electricity load time series forecasting has been addressed, showing quite remarkable performance.

Keywords: metaheuristics; soft computing; deep learning; big data; coronavirus

Introduction

The severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is a new respiratory virus, causing coronavirus disease 2019 (COVID-19), first discovered in humans in December 2019, that has spread across the globe, having reportedly infected >4 million people so far.¹ Much remains unknown about the virus, including how many people who may have very mild, asymptomatic, or simply undocumented infections and whether they can transmit the virus or not.² The precise dimensions of the outbreak are hard to evaluate.³

Bioinspired models typically mimic behaviors from the nature and are known for their successful application in hybrid approaches to find parameters in machine learning model optimization.⁴ Viruses can infect

people and these people can either die, infect other people, or simply recover after the disease. Vaccines and the immune defense system typically fight the disease and help to mitigate their effects while an individual is still infected. This behavior is typically modeled by an *SIR* model, consisting of three types of individuals: *S* for the number of susceptible, *I* for the number of infectious, and *R* for the number of recovered.⁵

Metaheuristics must deal with huge search spaces, even infinite for the continuous cases, and must find suboptimal solutions in reasonable execution times.⁶ The rapid propagation of the coronavirus along with its ability to cause infection in most of the countries in the world impressively fast has inspired the novel metaheuristic proposed in this study, named coronavirus optimization algorithm (CVOA). A parallel version

¹Data Science and Big Data Lab, Pablo de Olavide University, Seville, Spain.

²Department of Computer Science, University of Seville, Seville, Spain.

*Address correspondence to: F. Martínez-Álvarez, Data Science and Big Data Lab, Pablo de Olavide University, Seville ES-41013, Spain, E-mail: fmaralv@upo.es

is also proposed to spread different coronavirus strains and achieve better results in less iterations.

The main CVOA advantages regarding other similar approaches can be summarized as follows:

- (1) Coronavirus statistics are not currently known with precision by the scientific community and some aspects are still controversial, like the reinfection rate.⁷ In this sense, the infection rate, the mortality rate, the spreading rate, or the reinfection probability cannot be accurately estimated so far, due to several issues such as the lack of tests for asymptomatic people. However, the current state of the pandemic suggests certain values, as reported by the World Health Organization (WHO).⁸ Therefore, CVOA is parametrized with the actual reported values for rates and probabilities, preventing the user from performing an additional study on the most suitable setup configuration.
- (2) CVOA can stop the solutions exploration after several iterations, with no need to be configured. That is, the number of infected people increases over the first iterations; however, after a certain number of iterations, the number of infected people starts decreasing, until reaching a void infected set of individuals.
- (3) The coronavirus high spreading rate is useful for exploring promising regions more thoroughly (intensification), whereas the use of parallel strains ensures that all regions of the search space are evenly explored (diversification).
- (4) Another relevant contribution of this study is the proposal of a new discrete and of dynamic length codification, specifically designed for combining long short-term memory (LSTM) networks with CVOA (or any other metaheuristic).

There is one limitation to the current approach. Since there is no vaccine currently, it has not been included in the procedure to reduce the number of candidates to be infected. This fact involves an exponential increase of the infected population in the first iterations and, therefore, an exponential increase of the execution time for such iterations. This, however, is partially solved with the implementation of social isolation measures to simulate individuals who cannot be infected during a particular iteration.

A study case is included in this work that discusses the CVOA performance. CVOA has been used to find the optimal values for the hyperparameters of an LSTM architecture,⁹ which is a widely used model for artificial recurrent

neural network (RNN), in the field of deep learning.¹⁰ Data from the Spanish electricity consumption have been used to validate the accuracy. The results achieved verge on 0.45%, substantially outperforming other well-established methods such as random forest (RF), gradient-boost trees (GBT), linear regression (LR), or deep learning optimized with other metaheuristics. The code, developed in Python with a discrete codification, is available in the Supplementary Material section (along with an academic version in Java for a binary codification).

Finally, the need to further study the performance of well-established fitness functions¹¹ is acknowledged. However, given the relevance that this pandemic is acquiring throughout the world and the remarkable results achieved when combined with deep learning, this study is shared with the hope that it inspires future research in this direction.

The rest of the article is organized as follows. Related Works section discusses related and recent studies. The methodology proposed is introduced in Methodology section. Hybridizing Deep Learning with CVOA section proposes a discrete codification to hybridize deep learning models with CVOA and provides some illustrative cases. A sensitivity analysis on how populations are created and evolved over time is discussed in CVOA Sensitivity Analysis section. The results achieved are reported and discussed in Results section. Finally, the conclusions drawn and future study suggestions are included in Conclusions and Future Works section.

Related Works

There are many bioinspired metaheuristics to solve optimization problems. Although CVOA has been conceived to optimize any kind of problems, this section focuses on optimization algorithms applied to hybridize deep learning models.

It is hard to find consensus among the researchers on which method should be applied to which problem, and, for this reason, many optimization methods have been proposed during the past decade to improve deep learning models. In general, the criterion for selecting a method is its associated performance from a wide variety of perspectives. Low computation cost, accuracy, or even implementation difficulty can be accepted as one of these criteria.

The virus optimization algorithm was proposed by Liang and Cuevas-Juárez in 2016¹² and later improved by Liang et al.¹³ However, as many other metaheuristics, the results of its application are highly dependent on its initial configuration. In addition, it

simulates generic viruses, without adding individualized properties for particular viruses. The results achieved indicate that its usefulness is beyond doubt.

One of the most extended metaheuristics used to improve deep learning parameters is genetic algorithms (GAs). Hence, an LSTM network optimized with GA can be found in Chung and Shin.¹⁴ To evaluate the proposed hybrid approach, the daily Korea Stock Price Index data were used, outperforming the benchmark model. In 2019, a network traffic prediction model based on LSTM and GA was proposed in Chen et al.¹⁵ The results were compared with pure LSTM and autoregressive integrated moving average, reporting higher accuracy.

Multiagents systems have also been applied to optimize deep learning models. The use of particle swarm optimization (PSO) can be found in Liu et al.¹⁶ The authors proposed a model based on kernel principal component analysis and back propagation neural network with PSO for midterm power load forecasting. The hybridization of deep learning models with PSO was also explored in Fernandes-Junior and Yen¹⁷ but, this time, the authors applied the methodology with image classification purposes.

Ants colony optimization (ACO) models have also been used to hybridize deep learning. Thus, Desell et al.¹⁸ proposed an evolving deep RNNs using ACO applied to the challenging task of predicting general aviation flight data. The study in ElSaid et al.¹⁹ introduced a method based on ACO to optimize an LSTM RNNs. Again, the field of application was flight data records obtained from an airline containing flights that suffered from excessive vibration.

Some articles exploring the cuckoo search (CS) properties have been published recently as well. In Srivastava,²⁰ CS was used to find suitable heuristics for adjusting the hyperparameters of another LSTM network. The authors claimed an accuracy superior to 96% for all the data sets examined. Nawi et al.²¹ proposed the use of CS to improve the training of RNN to achieve fast convergence and high accuracy. Results obtained outperformed those than other metaheuristics.

The use of the artificial bee colony (ABC) optimization algorithm applied to LSTM can also be found in the literature. Hence, an optimized LSTM with ABC to forecast the bitcoin price was introduced in Yuliyono and Girsang.²² The combination of ABC and RNN was also proposed in Bosire²³ for traffic volume forecasting. This time the results were compared with standard backpropagation models.

From the analysis of these studies, it can be concluded that there is an increasing interest in using meta-

heuristics in LSTM models. However, not as many studies as for artificial neural networks can be found in the literature and, none of them, based on a virus propagation model. These two facts, among others, justify the application of CVOA to optimize LSTM models.

Methodology

This section introduces the CVOA methodology. Thus, Steps section describes the steps for a single strain. Remarks for a Parallel CVOA Version section introduces the modifications added to use CVOA as a parallel version. Suggested Parameters Setup section suggests how the input parameters must be set. Pseudocodes section includes the CVOA pseudocodes.

Steps

Step 1. Generation of the initial population. The initial population consists of one individual, the so-called patient-zero (*PZ*). As in the coronavirus pandemic, it identifies the first human being infected. If no previous local minima has been found, a random initialization for the *PZ* is suggested.

Step 2. Disease propagation. Depending on the individual, several cases are evaluated:

- (1) Each infected individual has a probability of dying (*P_DIE*), according to the COVID-19 death rate. Such individuals cannot spread the disease to new individuals.
- (2) The individuals who do not die will cause infection to new individuals (intensification). Two types of spreading are considered, according to a given probability (*P_SUPERSREADER*):
 - (a) Ordinary spreaders. Infected individuals will infect new individuals according to a regular spreading rate (*SPREADING_RATE*).
 - (b) Super-spreaders. Infected individuals will infect new individuals according to a super-spreading rate (*SUPERSPREADING_RATE*).
- (3) There is another consideration, since it is needed to ensure diversification. Both ordinary and super-spreader individuals can travel and explore very different solutions in the search space. Therefore, individuals have a probability of traveling (*P_TRAVEL*) to propagate the disease to solutions that may be quite different (*TRAVELER_RATE*). In case of not being a traveler, new solutions will change according to an *ORDINARY_RATE*. Note that one individual can be both super-spreader and traveler.

Step 3. Updating populations. Three populations are maintained and updated for each generation.

- (1) Deaths. If any individual dies, it is added to this population and can never be used again.
- (2) Recovered population. After each iteration, infected individuals (after spreading the coronavirus according to the previous step) are sent to the recovered population. It is known that there is a reinfection probability ($P_{REINFECTION}$). Hence, an individual belonging to this population could be reinfected at any iteration provided that it meets the reinfection criterion. Another situation must be considered since individuals might be isolated, as if they were following social distancing recommendations. For the sake of simplicity, it is considered that an isolated individual is sent to the recovered population when the isolation probability is met ($P_{ISOLATION}$).
- (3) New infected population. This population gathers all individuals infected at each iteration, according to the procedure described in the previous steps. It is possible that repeated new infected individuals are created at each iteration and, consequently, it is recommended to remove such repeated individuals from this population before the next iteration starts running.

Step 4. Stop criterion. One of the most interesting features of the proposed approach lies on its ability to end without the need of controlling any parameter. This situation occurs because the recovered and dead populations are constantly growing as time goes by, and the new infected population cannot infect new individuals. It is expected that the number of infected individuals increases for a certain number of iterations. However, from a particular iteration on, the size of the new infected population will be smaller than that of the current size because recovered and dead populations are too big, and the size of the infected population decays over time. In addition, a preset number of iterations ($PANDEMIC_DURATION$) can be added to the stop criterion. The social distancing measures also contribute to reach the stop criterion.

Remarks for a parallel CVOA version

It must be noted that it is very simple to use CVOA in a multivirus version since it can be implemented as a population-based algorithm, when considering the pandemic as a set of intelligent agents each of them

evolving in parallel. In contrast to trajectory-based metaheuristics, population-based metaheuristics enhances the diversification in the search space.

For this case, a new variable must be defined, *strains*, which determines the number of strains that will be launched in parallel. Each strain can explore different regions and can be differently configured so that each of them intensifies with their own rates.

Several considerations must be done for this case:

- (1) Every strain is run independently, following the steps in the previous section.
- (2) A wise strategy must be followed to generate *PZs* for each strain. For instance, it is suggested the generation of *PZs* is evenly spaced or, at least, with high Hamming distances. That way, the exploration of distinct regions of the search space is facilitated (diversification).
- (3) The interaction between the different strains is done by means of dead and recovered populations, which must be shared by all the strains. Operations over these populations must be handled as concurrent updates.²⁴
- (4) New infected populations, on the contrary, are different for each strain and no concurrent operations are required.
- (5) This version may help to simulate different rates for different strains. That way, if there is any initial information about the search space, some strains could be more focused on diversification and some others on intensification.

Depending on the hardware resources and how busy they are, every strain may evolve at different speeds. This situation should not pose any problems since it is known that the pandemic evolves at different rates and starts at different time stamps depending on region of the world.

Last, another application can be found for this parallel version. CVOA simulates an SIR model and consequently, any other global pandemic can be modeled by using the specific rates. Different pandemics could be run in parallel.

Suggested parameters setup

Since CVOA simulates the COVID-19 propagation, most of the rates (propagation, isolation, or mortality) are already known. This fact prevents the researcher from wasting time in selecting values for such rates and turns the CVOA into a metaheuristic quite easy to execute.

However, it must be noted that the current rates are still changing and it is expected they will vary over time, as the pandemic evolves. Maybe these values will not be stable until 2021 or even 2022. The suggested values have been retrieved from the World Health Organization²⁵ and are discussed hereunder:

- (1) *P_DIE*. An infected individual can die with a given probability. The case fatality ratio²⁶ varies by location, age of person infected, and the presence of underlying health conditions but, currently, this rate is set to $\sim 5\%$ by the scientific community.²⁷ Therefore, $P_DIE = 0.05$.
- (2) *P_SUPERSPREADER*. It is the probability that an individual spreads the disease to a greater number of healthy individuals. It is believed that this situation affects to a 10% of the infected population,²⁸ therefore, $P_SUPERSPREADER = 0.1$. After this condition is validated, two situations can be found:
 - (a) *ORDINARY_RATE*. If the infected individual is not a super-spreader, then the infection rate (also known as reproductive number, R_0) is 2.5. It is suggested that this rate is controlled by a random number in the range $[0, 5]$.
 - (b) *SUPERSPREADER_RATE*. If the infected individual turns out to be a super-spreader, then up to 15 healthy individuals can be infected. It is suggested that this rate is controlled by a random number in the range $[6, 15]$.
- (3) *P_REINFECTION*. This is a very controversial issue, since the scientific community does not agree on whether a recovered individual can be retested positive or not. As claimed by the WHO, no study has evaluated whether the presence of antibodies to COVID-19 confers immunity to subsequent infection by this virus in humans.²⁹ Some tests performed in South Korea suggest a rate of 2% according to the Korea Centers for Disease Control and Prevention.³⁰ Therefore, $P_REINFECTION = 0.02$, but this value will be re-evaluated, for sure, in the near future.
- (4) *P_ISOLATION*. This value is uncertain because countries are taking different measures for social isolation. This parameter helps to reduce the exponential growth of the infected population after each iteration. In other words, this parameter helps to reduce R_0 and it is crucial to ensure

the pandemic ends. Therefore, a high value must be assigned to this probability. It is suggested that $P_ISOLATION \geq 0.7$, since this value ensures $R_0 < 1$ (please refer to Fig. 5 to see Discussion section).

- (5) *P_TRAVEL*. This probability simulates how an infected individual can travel to any place in the world and can infect healthy individuals. It is known that almost a 10% of the population travel during a week (simulated time for every iteration),³¹ so $P_TRAVEL = 0.1$.
- (6) *SOCIAL_DISTANCING*. It is the number of iterations without social distancing measures. Since the populations grow exponentially at the beginning of the pandemic, this value must be carefully selected and must be set according to the size of the problem. Empirical values that suit for any codification vary from 7 to 12, so it is suggested that $7 \leq SOCIAL_DISTANCING \leq 12$.
- (7) *PANDEMIC_DURATION*. This parameter simulates the duration of the pandemic, that is, the number of iterations. Currently, these data are unknown so this number can be adjusted to the size of the problem. It is suggested that $PANDEMIC_DURATION = 30$.
- (8) *strains*. This parameter should be adjusted according to the size of the problem and the hardware availability, and it is difficult to suggest a value suitable for all situations. But a tentative initial value could be 5, in an attempt to simulate one different strain per continent. Therefore, $strains = 5$. Another important decision that must be made is how to initialize every *PZ* associated with the strains. When just one strain is considered, *PZ* is suggested to be randomly initialized. However, with $strains > 1$ the user should search for orthogonal *PZs* and to uniformly distribute them in the search space. This strategy should help to cover bigger search spaces in less iterations and to explore individuals with maximal distances.

Pseudocodes

This section provides the pseudocode of the most relevant functions for the CVOA, along with some comments to better understand them.

Function CVOA. This is the main function and its pseudocode can be found in Algorithm 1. Four lists must be maintained: dead, recovered, infected (the

CVOA: CORONAVIRUS OPTIMIZATION ALGORITHM

313

current set of infected individuals), and new infected individuals (the set of new infected individuals, generated by the spreading of the coronavirus from the current infected individuals).

The initial population is generated by means of the patient zero (PZ), which is a random solution.

The number of iterations is controlled by the main loop, evaluating the duration of the pandemic (preset value) and whether there is still any infected individual. In this loop, every individual can either die (it is sent to the dead list) or infect, thus enlarging the size of the new infected population. This infection mechanism is coded in function *infect* (see Function *infect* section).

Once the new population is formed, all individuals are evaluated and whether any of them outperforms the best current one, the latter is updated.

Algorithm 1: Function CVOA

```

1: define infectedPopulation, newInfectedPopulation as set of Individual
2: define dead, recovered as list of Individual
3: define PZ, bestIndividual, currentBestIndividual, aux as Individual
4: define time as integer
5: define bestSolutionFitness, currentBestFitness as real
6: time  $\leftarrow$  0
7: PZ  $\leftarrow$  InfectPatientZero()
8: infectedPopulation  $\leftarrow$  PZ
9: bestIndividual  $\leftarrow$  PZ
10: while time < PANDEMIC_DURATION AND sizeof (infectedPopulation) > 0 do
11:   dead  $\leftarrow$  die(infectedPopulation)
12:   for all  $i \in$  infectedPopulation do
13:     aux  $\leftarrow$  infect( $i$ , recovered, dead)
14:     if notnull(aux) then
15:       newInfectedPopulation  $\leftarrow$  aux
16:     end if
17:   end for
18:   currentBestIndividual  $\leftarrow$  selectBestIndividual(newInfectedPopulation)
19:   if fitness(currentBestIndividual) > bestIndividual then
20:     bestIndividual  $\leftarrow$  currentBestIndividual
21:   end if
22:   recovered  $\leftarrow$  infectedPopulation
23:   clear(infectedPopulation)
24:   infectedPopulation  $\leftarrow$  newInfectedPopulation
25:   time  $\leftarrow$  time + 1
26: end while
27: return bestIndividual

```

Function *infect*. This function receives an infected individual and returns the set of new infected individuals. Two additional lists, recovered and dead, are also received as input parameters since they must be updated after the evaluation of every infected individuals. The pseudocode is shown in Algorithm 2.

Two conditions are evaluated to determine the number of new infected individuals (use of *SPREADER*-

RATE or *SUPERSPREADER_RATE*) or how different the new individuals will be (*ORDINARY_RATE* or *TRAVELER_RATE*). The implementation on how these new infected individuals are encoded according to such rates is carried out in the function *newInfection*.

Algorithm 2: Function infect

```

Require: infected as of Individual; recovered, dead as list of Individual
1: define R1, R2 as real
2: define newInfected as list of Individual
3: R1  $\leftarrow$  RandomNumber()
4: R2  $\leftarrow$  RandomNumber()
5: if R1 < P_TRAVEL then
6:   if R2 < P_SUPERSPREADER then
7:     newInfected  $\leftarrow$  newInfection (infected, recovered, dead, SPREADER_RATE, ORDINARY_RATE)
8:   else
9:     newInfected  $\leftarrow$  newInfection (infected, recovered, dead, SUPERSPREADER_RATE, ORDINARY_RATE)
10:  end if
11: else
12:   if R2 < P_SUPERSPREADER then
13:     newInfected  $\leftarrow$  newInfection (infected, recovered, dead, SPREADER_RATE, TRAVELER_RATE)
14:   else
15:     newInfected  $\leftarrow$  newInfection (infected, recovered, dead, SUPERSPREADER_RATE, TRAVELER_RATE)
16:   end if
17: end if
18: return newInfected

```

Function *newInfection*. Given an infected individual, this function generates new infected individuals according to the spreading and traveling rates. This function also controls that the new infected individuals are not already in the dead list (in such case, this new infection is ignored) or in the recovered list (in such case, the *P_REINFECTION* is applied to determine whether the individual is reinfected or whether it remains in the recovered list). In addition, it considers that the new potential infected individual might be isolated, which is controlled by *P_ISOLATION*. Although the use of an extra list could be implemented, it has been decided to treat these individuals as recovered. Therefore, if an isolated individual is attempted to be infected, it is added to the recovered list.

The effective generation of the new infected individuals must be carried in the function *replicate*, whose pseudocode is not provided because it depends on the codification and the nature of the problem to be optimized. This function must return a set of new infected individuals, according to the aforementioned rates. Specific information on how this codification and replication is done for LSTM models is provided in Hybridizing Deep Learning with CVOA section.

The pseudocode for the described procedure can be found in Algorithm 3.

Algorithm 3: Function newInfection

Require: infected *as list of Individual*; recovered, dead *as list of Individual*

```

1: define R3, R4 as real
2: define newInfected as list of Individual
3: R3  $\leftarrow$  RandomNumber()
4: R4  $\leftarrow$  RandomNumber()
5: aux  $\leftarrow$  replicate(infected, SPREAD_RATE, TRAVELER_RATE)
6: for all  $i \in$  aux do
7:   if  $i \notin$  dead then
8:     if  $i \notin$  recovered then
9:       if  $R4 > P\_ISOLATION$  then
10:        newInfected  $\leftarrow i$ 
11:       else
12:        recovered  $\leftarrow i$ 
13:       end if
14:     else if  $R3 < P\_REINFECTION$  then
15:       newInfected  $\leftarrow i$ 
16:       remove  $i$  from recovered
17:     end if
18:   end if
19: end for
20: return newInfected

```

Function *die*. This function is called from the *main* function. It evaluates all individuals in the infected population and determines whether they die or not, according to the given P_DIE . Those meeting this condition are sent to the dead list. Algorithm 4 describes this procedure.

Algorithm 4: Function die

Require: infectedPopulation *as list of Individual*

```

1: define dead as list of Individual
2: define R5 as real
3: for all  $i \in$  infectedPopulation do
4:   R5  $\leftarrow$  RandomNumber()
5:   if  $R5 < P\_DIE$  then
6:     dead  $\leftarrow i$ 
7:   end if
8: end for
9: return dead

```

Function *selectBestIndividual*. This is an auxiliary function used to find the best fitness in a list of infected individuals. Its pseudo code is given in Algorithm 5.

Hybridizing Deep Learning with CVOA

This section describes the codification proposed for an individual, to hybridize deep learning with CVOA. The term hybridize is used in this context as the combination of two computational techniques (deep learning and CVOA) so that the best hyperparameter values are discovered. This strategy is very common in machine learning for optimizing models during the training process.^{32–34}

Algorithm 5: Function selectBestIndividual

Require: infectedPopulation *as list of Individual*

```

1: define bestIndividual as Individual
2: define bestFitness as real
3: bestFitness  $\leftarrow$  MINVALUE
4: for all  $i \in$  infectedPopulation do
5:   if fitness( $i$ )  $>$  bestFitness then
6:     bestFitness  $\leftarrow$  fitness( $i$ )
7:     bestIndividual  $\leftarrow i$ 
8:   end if
9: end for
10: return bestIndividual

```

Hence, the individual codification shown in Figure 1 has been implemented to apply CVOA to optimize deep neural network architectures.

As is shown in Figure 1, each individual is composed of the following elements. The element LR encodes the learning rate used in the neural network algorithm. It can take a value from 0 to 5 and its corresponding decoded values are 0, 0.1, 0.01, 0.001, 0.0001, and 0.00001.

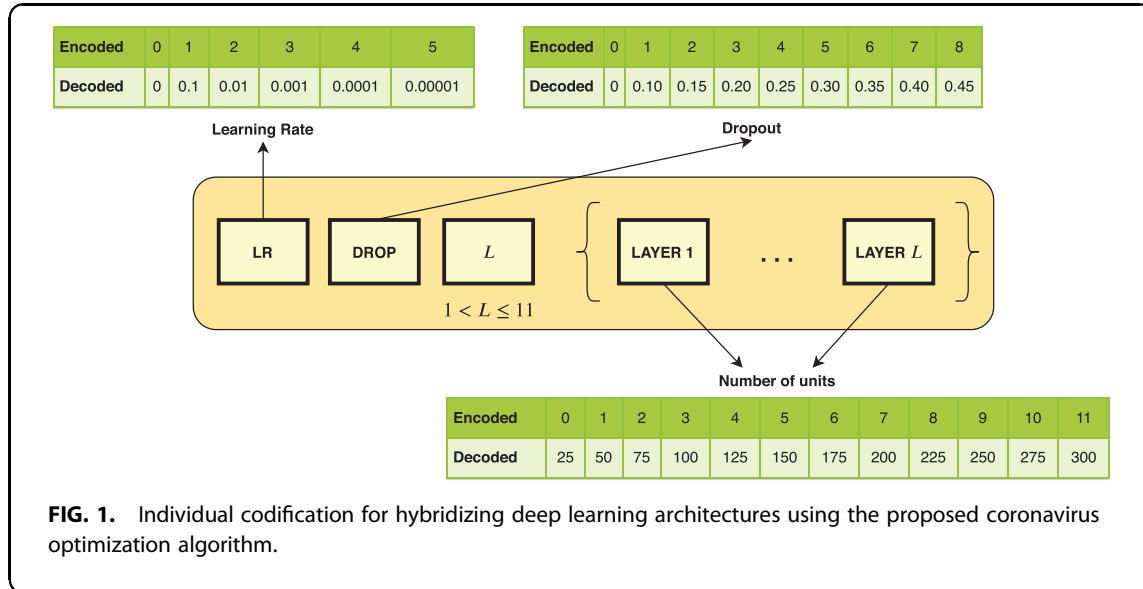
The element DROP encodes the dropout rate applied to the neural network. It can take values from 0 to 8 that correspond to 0, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, and 0.45, respectively. The dropout rate is distributed uniformly for all the layers of the network. That is, if the dropout is 0.4 and the network has four layers, then the 10% (0.1) of the neurons of each layer will be removed.

The element L of the individual stores the number of layers of the network. It is restricted to $1 < L \leq 11$. The first layer is referred to the input layer of the neural network. The rest of layers are hidden layers. The output layer is excluded from the codification. Therefore, the optimized network can contain from 1 to 10 hidden layers.

The proposed individual codification has a variable size. Thus, its size depends on the number of layers indicated in the element L . Consequently, a list of elements (LAYER 1, ..., LAYER L) are also included in the individual, which encode the number of units (neurons) for each network layer. Each of these elements can take values from 0 to 11, and their corresponding decoded values range from 25 to 300, with a step of 25.

PZ generation

The PZ, as it has been described previously, is the individual of the first iteration in the CVOA algorithm. After the hybridization proposed, a random individual is created considering the codification already defined.



In first place, a random value for the learning rate of the PZ is generated. Specifically, a number between 0 and 5 is generated randomly in a uniform distribution. Such limits are indicated in Figure 1, according to the possible encoded values of the learning rate element. The same process is carried out to produce a random value for the dropout element. In such case, a random number between 0 and 8 is generated.

In second place, a random number of layers are generated for the element L of PZ. Such number of layers is a random number between 2 and 11. Note that the first layer is reserved for the input layer of the neural network, as it has been discussed before.

In last place, for each one of the L layers, a random number of units is generated between 0 and 11, covering the possible encoded values for the number of units previously defined (Fig. 1).

Infection procedure

The infection procedure described here corresponds to the functionality of *replicate()*, introduced in line 5 of Algorithm 3. This procedure takes an individual as input and returns an infected individual according to the following procedure.

The first step is to determine the element L of the infected individual that will be mutated. The probability of such mutation that occurs has been set to $\frac{1}{3}$ so that every element has the same probability to mutate. If the mutation occurs, then the element L of the individual is

modified according to the process described in Single Position Mutation section.

If the element L (the number of layers of the network) changes, then the elements encoding the different layers within the individual (LAYER 1, ..., LAYER L) must be resized accordingly. Such resizing process is explained in Individual Resizing Process section.

The second step is to determine how many elements of the individual will be infected. If the *TRAVELER_RATE* < 0 , then the number of infected elements is generated randomly from 0 to the length of the individual (excluding the element L). Else, the *TRAVELER_RATE* indicates itself the number of infected elements.

As third step, once the number of infected elements of the individual is determined, a list of random positions is generated. For example, if three positions of the individual must be changed, then the random positions affected could be, for instance, referred to the elements {DROP, LAYER 2, LAYER 4}.

Finally, the selected positions of the individual are mutated. Such mutation is described in Single Position Mutation section.

Individual resizing process

When an individual is infected at the position of the element L , the list of elements that encodes the number of units per layer (LAYER 1, ..., LAYER L) must be resized accordingly.

In the case that the new number of layers after the infection is lower than its previous value, then the last leftover elements are removed. For instance, if the initial individual is $\{2, 0, 4\}\{3, 2, 1, 6\}$ (four layers), the element $L=4$ is infected and the new value is $L=2$, then the resulting individual will be $\{2, 0, 2\}\{3, 2\}$.

In the case that the new number of layers after the infection is higher than its previous value, the new random elements are added at the end of the individual. For instance, if the initial individual is $\{2, 0, 4\}\{3, 2, 1, 6\}$ (four layers), the element $L=4$ is infected and the new value is $L=6$, then the resulting individual could be $\{2, 0, 6\}\{3, 2, 1, 6, 0, 4\}$.

Single position mutation

The process carried out to change the value of a specific element of an individual is described in this section.

First, a signed amount of change $C \in \{-2, -1, +1, +2\}$ is randomly determined using the following criteria. A random real number P between 0 and 1 is generated using a uniform distribution. If $P < 0.25$, then the amount of change will be $C = -2$. Else if $P < 0.5$, then the amount of change will be $C = -1$. Else if $P < 0.75$, then the amount of change will be $C = +1$. Else, the amount of change will be $C = +2$.

Once the amount of change is determined, the new value for the infected element is computed. If its previous value is V , then the new value after the single position mutation will be $V' = V + C$. If the new value V' exceeds the limits defined for the individual codification, such value is set to the maximum or minimum allowed value accordingly.

CVOA Sensitivity Analysis

This section discusses several aspects about the sensitivity of CVOA to different configurations. Hence, Sensitivity to the Number of Strains section evaluates the evolution of the populations for a different number of strains. Sensitivity to the Parameters section assesses the performance when other well-known viruses are modeled. Finally, Sensitivity to the Social Distancing Measures section provides information about R_0 and how it varies when social distancing measures change.

Sensitivity to the number of strains

This section provides an overview on how populations evolve over time and how the search space is explored, when a different number of strains are used.

A binary codification has been used, with 20 bits, to conduct this experimentation. A simple fitness function has been evaluated, $f(x) = (x - 15)^2$, because the

goal of this section is to evaluate the growth of the populations, and not to find challenging optimum values. This function reaches the minimum value at $x=15$, that is, $f(15)=0$.

According to Suggested Parameters Setup section, the following configuration has been used: $P_DIE=0.05$, $P_ISOLATION=0.8$, $P_SUPERSREADER=0.1$, $P_REINFECTION=0.02$, $SOCIAL_DISTANCING=8$, $P_TRAVEL=0.1$, and $PANDEMIC_DURATION=30$.

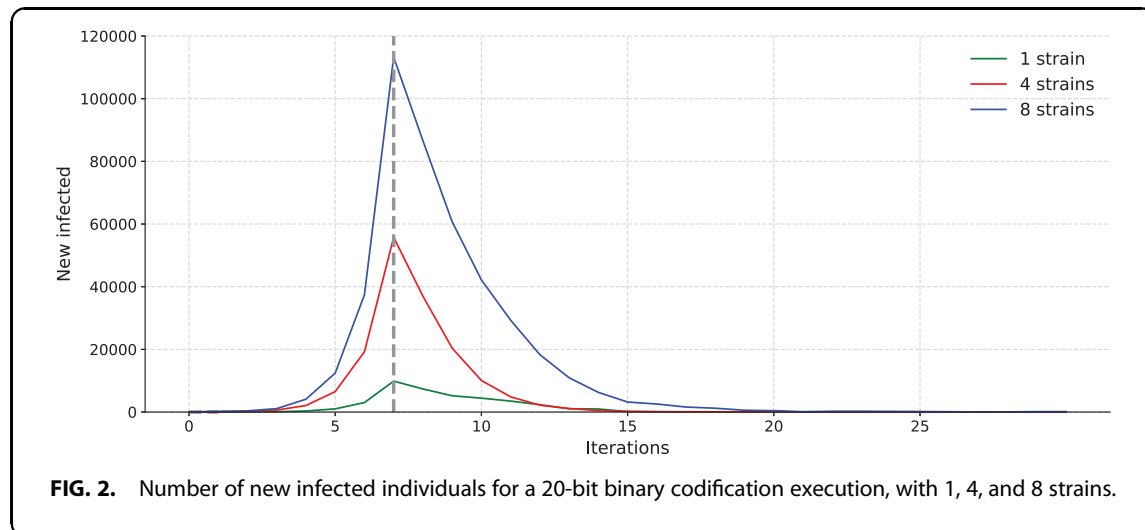
Every experiment has been launched 50 times and, on average, the optimum value was found during the iteration number 13, 6, and 3, for 1, 4, and 8 strains, respectively.

Figure 2 illustrates the evolution of the new infected population over time, for 1, 4, and 8 strains. The number of new infected people increases exponentially during the first $SOCIAL_DISTANCING=8$ iterations because $R_0 > 0$ but, from iteration 9 onward, an acute decrease is reported because R_0 becomes <0 . This fact is controlled by $P_ISOLATION=0.8$ (a deeper study on R_0 and $P_ISOLATION$ can be found in Sensitivity to the Social Distancing Measures section). It must be noted that iteration 0 (PZ infection) counts as a regular iteration.

Figures 3 and 4 show the accumulated number of recovered people and accumulated deaths, respectively. Note that deaths and recovered individuals cannot be infected again (except for the individuals in the recovered list that can be reinfected with a given probability, $P_REINFECTION$). These two curves are a direct consequence of the number of new infected people, so, once the number of new infections decreases or even disappears, these values remain almost constant. Also, it can be observed that $P_ISOLATION=0.8$ after $SOCIAL_DISTANCING=8$ iterations help to flatten the curves. A directly proportional relationship is reported between the number of strains and the number of explored individuals at the end of the pandemic.

Four main conclusions can be drawn from the analysis of these figures:

- (1) The number of new infected individuals, accumulated recovered, and deaths is directly proportional to the number of strains.
- (2) The higher the number of strains, the lower the number of iterations that are required to reach the optimal value.
- (3) The number of individuals evaluated increases at each iteration on an almost linear basis, as the number of strains increases. In case no random numbers were generated, the relationship



would be directly proportional, that is, four strains would evaluate four times the number of individuals than one strain would do.

- (4) To reach the optimum values, the search space explored is smaller as the number of strains increases. This is due to the generation of PZ evenly spaced, which makes easier to explore wider areas.

Sensitivity to the parameters

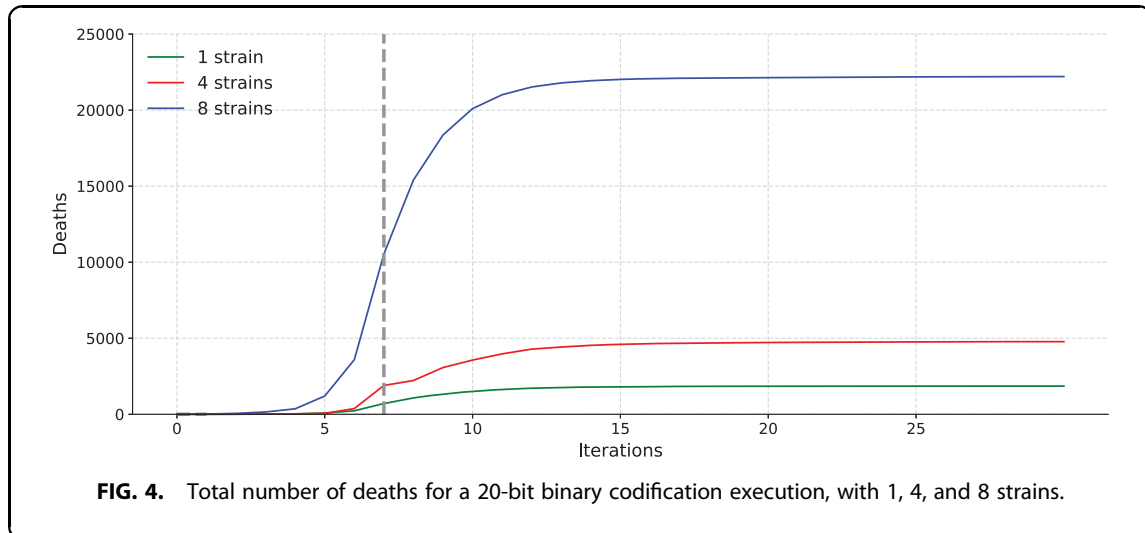
Several well-known viruses with deep impact in human beings' health are modeled in this section, to assess the CVOA robustness to different input parameter values.

Middle East respiratory syndrome (MERS), SARS, influenza (seasonal strains), and Ebola have been selected, with the parametrization given in Table 1. It is worth mentioning that the modeling of each virus requires much research and an approximate parametrization has been used, according to the references in the rightmost column.

All experiments have been conducted with 4 strains and 30 iterations. The viruses with vaccines have been simulated by using $P_{ISOLATION} = 0.95$ after five iterations, since this feature is not implemented in CVOA.

Table 2 summarizes the percentage of search space explored and the best fitness found, on average.





Codifications of 10, 20, 30, 40, and 50 bits have been used, with associated search spaces of length 1024, $1.05\text{E}+6$, $1.07\text{E}+09$, $1.10\text{E}+12$, and $1.13\text{E}+15$, respectively. Several findings are revealed:

- (1) CVOA finds the optimal values even for the longest codification (50 bits) and it is done by exploring a similar search space size as the other configurations do.
- (2) SARS is the second best parametrization, reaching remarkable fitness even for 50 bits. But it required the evaluation of a greater number of individuals and, therefore, the execution time was greater as well.
- (3) MERS obtained the poorest results in terms of fitness but it explored a smaller space search. This situation may be explained due to the low associated reproductive number ($R_0 < 1$).
- (4) Influenza has obtained slightly worse results in terms of fitness than CVOA but with less solutions explored. This configuration may be useful to obtain satisfactory results in a reduced execution time.

- (5) The high death fatality rate of Ebola prevents from exploring most of the search space. This fact makes difficult to visit optimal values. However, results for 40 bits are satisfactory in terms of fitness. For 50 bits, its use is discouraged considering the poor fitness value reached.

It can be concluded that variations in the input parameter values lead to results varying both in fitness and in execution time. This feature is very useful for the CVOA parallel version, since strains with different rates and probabilities can be simultaneously launched. That is, strains aiming at diversifying can be combined with strains aiming at intensifying.

Sensitivity to the social distancing measures

In this section, an analysis on how *P-ISOLATION* modifies R_0 is conducted. The purpose is to discover when $R_0 < 1$, situation in which the pandemic prevalence declines. A study with a 10-bit to 50-bit codification has been done as well as using different number of strains (1, 4, and 8).

Figure 5 illustrates how R_0 varies for a 40-bit codification, with probabilities of isolation ranging from 0 to 1, and with 1, 4, and 8 strains. Quite similar behaviors have been achieved for all codifications.

From the analysis of this figure, several conclusions are drawn:

- (1) R_0 is linear and inversely proportional to *P-ISOLATION*.

Table 1. Parametrization for other viruses

Disease	R_0	Fatality rate (%)	Vaccine	Super-spreaders	References
SARS	1.4–2.5	11	No	Yes	35,36
MERS	0.3–0.8	34.4	No	Yes	28,35,37
Influenza	0.9–2.1	0.1	Yes	No	38
Ebola	1.5–1.9	50	Yes	No	39,40

MERS, Middle East respiratory syndrome; SARS, severe acute respiratory syndrome.

Table 2. CVOA performance with different configurations

Disease	10 bits		20 bits		30 bits		40 bits		50 bits	
	Explored (%)	Fitness	Explored (%)	Fitness	Explored (%)	Fitness	Explored (%)	Fitness	Explored (%)	Fitness
SARS	57.32	0	0.54	0	6E-03	1	1E-05	4	3E-08	252
MERS	20.34	0	0.04	16	1E-02	36	1E-05	112	2E-09	3210
Influenza	13.23	0	0.02	0	8E-04	2	1E-06	14	1E-08	310
Ebola	62.93	0	0.44	0	7E-02	4	2E-05	15	1E-09	810
COVID-19	15.63	0	0.21	0	1E-03	0	1E-05	0	2E-08	0

COVID-19, coronavirus disease 2019; CVOA, coronavirus optimization algorithm.

- (2) The same negative slope is shown, with variations no higher than $10E-2$ on average for all codifications and number of strains.
- (3) R_0 is <1 with $P_ISOLATION$ values close to 0.65 (and higher). This fact involves a decline of the infectious disease.

Results

This section reports the results achieved by hybridizing a deep learning model with CVOA. Study Case: Electricity Demand Time Series Forecasting section describes the study case selected to prove the effectiveness of the proposed algorithm. Data Set Description section describes the data set used. Performance Analysis section discusses the results achieved and includes some comparative methods.

Study case: electricity demand time series forecasting

The forecasting of future values fascinates the human being. To be able to understand how certain variables evolve over time has many benefits in many fields.

Electricity demand forecasting is not an exception, since there is a real need for planning the amount to be generated or, in some countries, to be bought.

The use of machine learning to forecast such time series has been intensive during the past years.⁴¹ But, with the development of deep learning models, and, in particular of LSTM, much research is being conducted in this application field.⁴²

Data set description

The time series considered in this study is related to the electricity consumption in Spain from January 2007 to June 2016, the same as used in Torres et al..⁴³ It is a time series composed of 9 years and 6 months with a 10-minute sampling frequency, resulting in 497,832 measures.

As in the original article, the prediction horizon is 24, that is, this is a multistep strategy with $h=24$. The size of samples used for the prediction of these 24 values is 168. Furthermore, the data set was split into 70% for the training set and 30% for the test set, and in addition,

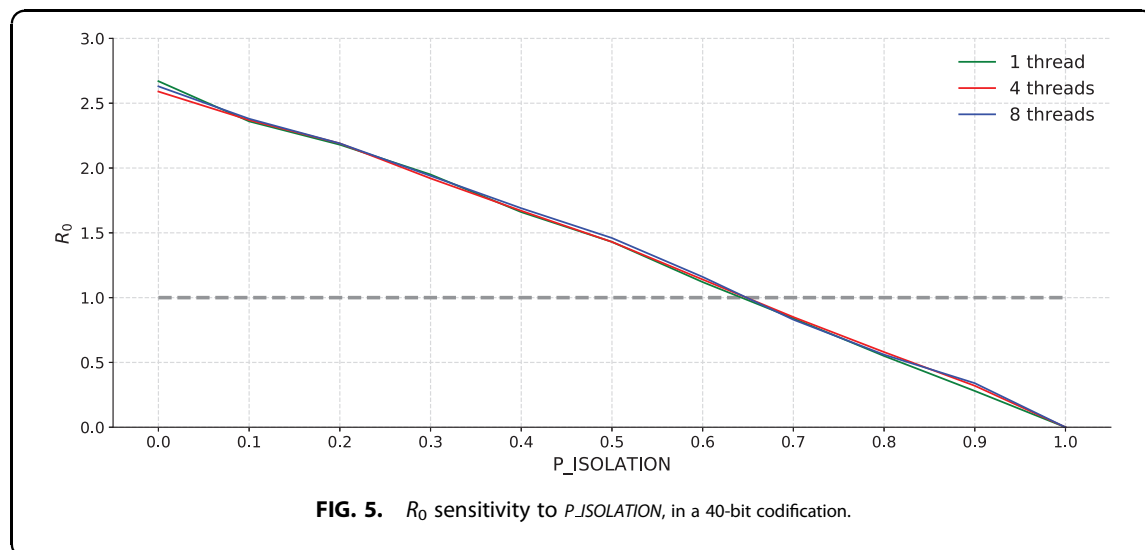


FIG. 5. R_0 sensitivity to $P_ISOLATION$, in a 40-bit codification.

Table 3. Results in terms of MAPE for LSTM-CVOA compared with other well-established methods

Method	MAPE (%)
LR	7.34
DT	2.88
GBT	2.72
RF	2.20
DNN-GS	1.68
DNN-RS	1.57
DNN-RS-LP	1.36
DNN-CVOA	1.18
LSTM-GS	1.22
LSTM-RS	0.84
LSTM-RS-LP	0.82
LSTM-CVOA	0.47

Bold indicates the best results for the proposed method in the article (LSTM-CVOA).

CVOA, coronavirus optimization algorithm; DNN, deep neural network; DNN-CVOA, CVOA has been combined with DNN; DNN-GS, DNN optimized with a grid search; DNN-RS, DNN optimized with random search; DNN-RS-LP, DNN smoothed with a low-pass filter; DT, decision tree; GBT, gradient-boosted trees; LR, linear regression; LSTM, long short-term memory; LSTM-CVOA, CVOA has been combined with LSTM; LSTM-GS, LSTM optimized with a grid search; LSTM-RS, LSTM optimized with random search; LSTM-RS-LP, LSTM smoothed with a low-pass filter; MAPE, mean absolute percentage error; RF, random forest.

a 30% of the training set has also been selected for the validation set, to find the optimal parameters. The training set covers the period from January 1, 2007, at 00:00 to August 20, 2013, at 02:40. Therefore, the test set comprises the period from August 20, 2013, at 02:50 to June 21, 2016, at 23:40.

Performance analysis

This section reports the results obtained by hybridizing LSTM with CVOA, by means of the codification proposed in Hybridizing Deep Learning with CVOA section, to forecast the Spanish electricity data set described in Data Set Description section.

LR, decision tree, GBT, and RF models have been used with parametrization setups according to those studied in Galicia et al.^{44,45} A deep neural network optimized with a grid search (DNN-GS) according to Torres et al.⁴³ has also been applied. Another deep neural network, but optimized with random search (DNN-RS) and smoothed with a low-pass filter (DNN-RS-LP),⁴⁶ has also been applied. Furthermore, CVOA has been combined with DNN (DNN-CVOA), using the same codification as in LSTM.

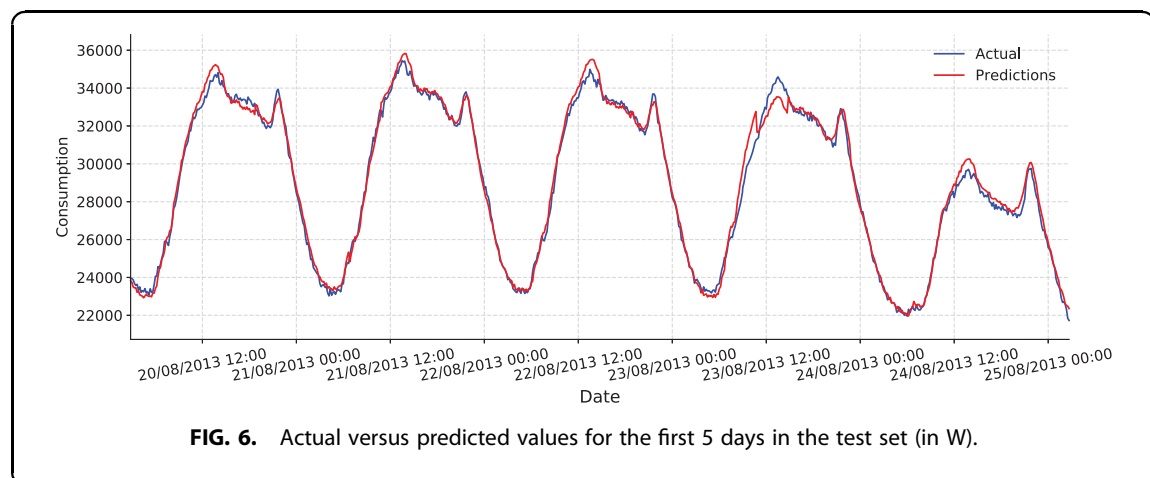
These results along with those of LSTM, and combinations with GS, RS, RS-LP, and CVOA, are summarized in Table 3, expressed in terms of the mean absolute percentage error. It can be observed that LSTM-CVOA outperforms all evaluated methods that have showed particularly remarkable performance for this real-world data set. In addition, DNN-CVOA outperforms all other DNN configurations, which confirms the superiority of CVOA with reference to GS, RS, and RS-LP.

Another relevant consideration that must be taken into account is that the compared methods generated 24 independent models, each of them for every value forming h . So, it would be expected that LSTM-CVOA performance increases if independent models are generated for each of the values in h .

These results have been achieved with the following codification: $\{4,0,8\}\{9,7,2,7,2,7,10,7\}$. The decoded architecture parameters are listed below:

- (1) Learning rate: $10E-04$.
- (2) Dropout: 0.
- (3) Number of layers: 8.
- (4) Units per layer: [250, 200, 75, 200, 75, 200, 275, 200]

Finally, Figure 6 depicts the first 5 predicted days versus their actual values, expressed in watts.

**FIG. 6.** Actual versus predicted values for the first 5 days in the test set (in W).

Conclusions and Future Studies

This study has introduced a novel bioinspired meta-heuristic, based on the COVID-19 pandemic behavior. On the one hand, CVOA has three major advantages. First, its high relation to the coronavirus spreading model prevents users from making any decision about the input values. Second, it ends after a certain number of iterations due to the exchange of individuals between healthy and dead/recovered lists.

In addition, a novel discrete and dynamic codification has been proposed to hybridize deep learning models. On the other hand, it exhibits some limitations. Such is the case for the exponential growth of the infected population as time (iterations) goes by.

Furthermore, a parallel version is proposed so that CVOA is easily transformed into a multivirus meta-heuristic, in which different coronavirus strains search for the best solution in a collaborative way. This fact allows to model every strain with different initial setups (higher *DEATH_RATE*, for instance), sharing recovered or dead lists.

Additional experimentation must be conducted to assess its performance on standard *F* functions and find out the search space shapes in which it can be more effective.

As for future study, some actions might be taken to reduce the size of the infected population after several iterations, which grows exponentially. In this sense, a vaccine could be implemented. This case would involve adding to the recovered list, at a given *VACCINE_RATE* healthy individuals. This rate will remain unknown until a vaccine is developed.

Another suggested research line is using dynamic rates. For instance, the observation of the preliminary effects of the social isolation measures in countries such as China, Italy, or Spain suggests that the *INFECT_RATE* could be simulated as a Poisson process, but more time and country recoveries are required to confirm this trend.

For the multistep forecasting problem analyzed, it would be desirable to generate independent models for each of the values that form the prediction horizon *h*.

Finally, further research has to be conducted to assess the CVOA performance when applied to other fields and combined with other networks.

Supplementary Material

Along with this article, an academic version in Java for a binary codification is provided, with a simple fitness

function in a GitHub repository (https://github.com/DataLabUPO/CVOA_academic). The master branch includes a simple implementation, whereas the sets branch provides an optimized version with a command line interface. In addition, the code in Python for the deep learning approach is also provided, with a more complex codification and the suggested implementation, according to the pseudocode provided (https://github.com/DataLabUPO/CVOA_LSTM).

Author Disclosure Statement

No competing financial interests exist.

Funding Information

The authors thank the Spanish Ministry of Economy and Competitiveness for the support under project TIN2017-88209-C2.

References

1. Velavan TP, Meyer CG. The COVID-19 epidemic. *Trop Med Int Health*. 2020;25:278–280.
2. Li R, Pei S, Chen B, et al. Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (SARS-CoV-2). *Nature*. 2020;368:489–493.
3. Giordano G, Blanchini F, Bruno R, et al. Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy. *Nat Med*. 2020;26:855–860.
4. Del Ser J, Osaba E, Molina D, et al. Bio-inspired computation: Where we stand and what's next. *Swarm Evol Comput*. 2019;48:220–250.
5. Tolic D, Kleineberg K, Antulov-Fantulin N. Simulating SIR processes on networks using weighted shortest paths. *Sci Rep*. 2018;8:6562.
6. Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. *Inf Sci*. 2013;237:82–117.
7. Tay MZ, Poh CM, Rénia L, et al. The trinity of COVID-19: immunity, inflammation and intervention. *Nat Rev Immunol*. 2020;20:363–374.
8. World Health Organization. 2019. Available online at <https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019> (last accessed March 20, 2020).
9. Kelotra A, Pandey P. Stock market prediction using optimized deep-ConvLSTM model. *Big Data*. 2020;8:5–24.
10. De-Cnudde S, Ramon Y, Martens D, Provost F. Deep learning on big, sparse, behavioral data. *Big Data*. 2019;7:286–307.
11. Glover F, Kochenberger GA. *Handbook of metaheuristics*. New York: Springer, 2003.
12. Liang YC, Cuevas-Juárez JR. A novel metaheuristic for continuous optimization problems: Virus optimization algorithm. *Eng Optim*. 2016;48:73–93.
13. Liang YC, Cuevas-Juárez JR. A self-adaptive virus optimization algorithm for continuous optimization problems. *Soft Comput*. 2020. [Epub ahead of print]; DOI: 10.1007/s00500-020-04730-0.
14. Chung H, Shin K-S. Genetic algorithm-optimized long short-term memory network for stock market prediction. *Sustainability*. 2018;10:3765.
15. Chen J, Xing H, Yang H, Xu L. Network traffic prediction based on LSTM networks with genetic algorithm. *Lect Notes Electr Eng*. 2019;550:411–419.
16. Liu Z, Sun X, Wang S, et al. Midterm power load forecasting model based on kernel principal component analysis and back propagation neural network with particle swarm optimization. *Big Data*. 2019;7:130–138.
17. Fernandes-Junior FE, Yen GG. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm Evol Comput*. 2019;49:62–74.

18. Desell T, Clachar S, Higgins J, Wild B. Evolving deep recurrent neural networks using ant colony optimization. *Lect Notes Comput Sci.* 2015; 9026:86–98.
19. ElSaid A, ElJamiy F, Higgins J, et al. Using ant colony optimization to optimize long short-term memory recurrent neural networks. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 13–20.
20. Srivastava D, Singh Y, Sahoo A. Auto tuning of RNN hyper-parameters using cuckoo search algorithm. In: *Proceedings of the International Conference on Contemporary Computing*, 2019, pp. 1–5.
21. Nawi NM, Khan A, Rehman MZ. A new optimized cuckoo search recurrent neural network (CSRNN). In: *Proceedings of the International Conference on Robotic, Vision, Signal Processing & Power Applications*, 2014, pp. 335–341.
22. Yuliyono AD, Girsang AS. Artificial bee colony-optimized LSTM for bitcoin price prediction. *Adv Sci Technol Eng Syst J.* 2019;4:375–383.
23. Bosire A. Recurrent neural network training using ABC algorithm for traffic volume prediction. *Informatica.* 2019;43:551–559.
24. Dhar V, Sun C, Batra P. Transforming finance into vision: Concurrent financial time series as convolutional net. *Big Data.* 2019;7:276–285.
25. World Health Organization. 2020. Coronavirus disease 2019 (COVID-19): Situation report 74. Technical report, WHO. Available online at <https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200403-sitrep-74-covid-19-mp.pdf> (last accessed May 9, 2020).
26. Ghani AC, Donnelly CA, Cox DR, et al. Methods for estimating the case fatality ratio for a novel, emerging infectious disease. *Am J Epidemiol.* 2005;162:479–486.
27. Mizumoto K, Chowell G. Estimating risk for death from 2019 novel coronavirus disease, China, January–February 2020. *Emerg Infect Dis.* 2020;26:1251–1256.
28. Wu JY, Leung K, Leung GM. Nowcasting and forecasting the potential domestic and international spread of the 2019-nCoV outbreak originating in Wuhan, China: A modelling study. *Lancet.* 2020;396: 689–697.
29. World Health Organization. 2020. Immunity passports in the context of COVID-19. Technical report, WHO. Available online at <https://www.who.int/news-room/commentaries/detail/immunity-passports-in-the-context-of-covid-19> (last accessed April 29, 2020).
30. Korea Centers for Disease Control and Prevention. 2020. Coronavirus Disease-19. Available online at https://www.cdc.go.kr/cdc_eng/ (last accessed May 9, 2020).
31. González MC, Hidalgo CA, Barabási AL. Understanding individual human mobility patterns. *Nature.* 2008;453:779–782.
32. Calvet L, Armas JD, Masip D, Juan AA. Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs. *Math Open.* 2017;15:261–280.
33. Darwish A, Hassanien AE, Das S. A survey of swarm and evolutionary computing approaches for deep learning. *Artif Intell Rev.* 2020;53: 1767–1812.
34. Devikanniga D, Vetrivel K, Badrinath N. Review of meta-heuristic optimization based artificial neural networks and its applications. *J Phy: Conf Ser.* 2019;1362:012074.
35. Trilla A. One world, one health: The novel coronavirus COVID-19 epidemic. *Med Clin.* 2020;154:175–177.
36. World Health Organization. 2003. Consensus document on the epidemiology of severe acute respiratory syndrome (SARS). Technical report, WHO. Available online at <https://www.who.int/csr/sars/en/WHOconsensus.pdf> (last accessed May 10, 2020).
37. World Health Organization. 2019. Middle East respiratory syndrome coronavirus (MERS-CoV). Technical report, WHO. Available online at <https://www.who.int/emergencies/mers-cov/en/> (last accessed May 10, 2020).
38. Coburn BJ, Wagner BG, Blower S. Modeling influenza epidemics and pandemics: insights into the future of swine flu (H1N1). *BMC Med.* 2009;7:30.
39. Khan A, Naveed M, Dur e Ahmad M. Estimating the basic reproductive ratio for the Ebola outbreak in Liberia and Sierra Leone. *Infect Dis Poverty.* 2015;4:13.
40. World Health Organization. 2020. Ebola virus disease. Technical report, WHO. Available online at <https://www.who.int/news-room/fact-sheets/detail/ebola-virus-disease> (last accessed May 10, 2020).
41. Martínez-Álvarez F, Troncoso A, Asencio-Cortés G, Riquelme JC. A survey on data mining techniques applied to electricity-related time series forecasting. *Energies.* 2015;8:13162–13193.
42. Bedi J, Toshniwal D. Deep learning framework to forecast electricity demand. *Appl Energy.* 2019;238:1312–1326.
43. Torres JF, Galicia A, Troncoso A, Martínez-Álvarez F. A scalable approach based on deep learning for big data time series forecasting. *Integr Comp Aided Eng.* 2018;25:335–348.
44. Galicia A, Torres JF, Martínez-Álvarez F, Troncoso A. Scalable forecasting techniques applied to big electricity time series. *Lect Notes Comput Sci.* 2019;10306:165–175.
45. Galicia A, Talavera-Llames RL, Troncoso A, et al. Multi-step forecasting for big data time series based on ensemble learning. *Knowl Based Syst.* 2019;163:830–841.
46. Torres JF, Gutiérrez-Avilés D, Troncoso A, Martínez-Álvarez F. Random hyper-parameter search-based deep neural network for power consumption forecasting. *Lect Notes Comput Sci.* 2019;11506:259–269.

Cite this article as: Martínez-Álvarez F, Asencio-Cortés G, Torres JF, Gutiérrez-Avilés D, Melgar-García L, Pérez-Chacón R, Rubio-Escudero C, Riquelme JC, Troncoso A (2020) Coronavirus optimization algorithm: a bioinspired metaheuristic based on the COVID-19 propagation model. *Big Data* 8:4, 308–322, DOI: 10.1089/big.2020.0051.

Abbreviations Used

ABC	=	artificial bee colony
ACO	=	ants colony optimization
COVID-19	=	coronavirus disease 2019
CS	=	cuckoo search
CVOA	=	coronavirus optimization algorithm
DNN	=	deep neural network
DNN-CVOA	=	CVOA has been combined with DNN
DT	=	decision tree
GAs	=	genetic algorithms
GBTs	=	gradient-boosted trees
GS	=	grid search
LR	=	linear regression
LSTM	=	long short-term memory
MAPE	=	mean absolute percentage error
RF	=	random forest
RS	=	random search
MERS	=	Middle East respiratory syndrome
PSO	=	particle swarm optimization
SARS	=	severe acute respiratory syndrome
SARS-CoV-2	=	SARS coronavirus 2
WHO	=	World Health Organization

5.1.4 | "A new forecasting algorithm based on neighbors for streaming electricity time series"

Authors: Jiménez-Herrera P., Melgar-García L., Asencio-Cortés G., Troncoso A.

Publication type: Conference article.

Conference: Hybrid Artificial Intelligent Systems (HAIS 2020).

Publication: Lecture Notes in Computer Science, Springer International Publishing, Cham.

Year: 2020.

Volume: 12344

Pages: 522-533

DOI: 10.1007/978-3-030-61705-9_43



A New Forecasting Algorithm Based on Neighbors for Streaming Electricity Time Series

P. Jiménez-Herrera, L. Melgar-García, G. Asencio-Cortés, and A. Troncoso^(*)

Division of Computer Science, Universidad Pablo de Olavide, 41013 Seville, Spain
pjimher@alu.upo.es, {lmelgar,guaasecor,ali}@upo.es

Abstract. This work presents a new forecasting algorithm for streaming electricity time series. This algorithm is based on a combination of the K-means clustering algorithm along with both the Naive Bayes classifier and the K nearest neighbors algorithm for regression. In its offline phase it firstly divide data into clusters. Then, the nearest neighbors algorithm is applied for each cluster producing a list of trained regression models, one per each cluster. Finally, a Naive Bayes classifier is trained for predicting the cluster label of an instance using as training the cluster assignments previously generated by K-means. The algorithm is able to be updated incrementally for online learning from data streams. The proposed algorithm has been tested using electricity consumption with a granularity of 10 min for 4-h-ahead predicting. Our algorithm widely overcame other four well-known effective online learners used as benchmark algorithms, achieving the smallest error.

Keywords: Forecasting · Real time · Streaming data · Electricity time series · Nearest neighbors.

1 Introduction

The current technological context has two main aspects of research and development. On the one hand, an industrial aspect, where the boom in advanced connection of devices or Internet of Things (IoT) is changing the means of production and service management systems, leading our society to a new industrial revolution known as Industry 4.0. And on the other hand, the data, as a consequence of the enormous amount of data that is generated daily in our society, coming from many different origins, including IoT between them, and leading us to a new technological revolution based on the analysis of large scale data known as Big Data.

Of the three V's that initially defined Big Data (speed, variety and volume), volume is perhaps the characteristic in which researchers have made the greatest effort. Almost all the technology developed in recent years allows to obtain approximate solutions to problems derived from the dimensionality of the data.

However, speed, despite being a feature present in many problems of data analysis, has not had the same impact, or rather, it is beginning to have it at present. Although the analysis of streaming data has been studied in the last decade, in very few cases forecasting techniques have been developed, which allow to have an updated model that is capable of giving a response in real time to obtain forecasts.

In this work, a new forecasting algorithm based on the nearest similar pattern for streaming electricity time series, named StreamNSP, is proposed. The algorithm firstly determines different patterns in historical data. Once the data stream is received, the algorithm predicts the pattern to which the data stream just arrived belongs. Then, the prediction is obtained using the nearest neighbor among the data with the same pattern to data stream. The performance of the proposed method has been tested on a real-world related to energy consumption. Finally, the results have been compared to that of the well-known prediction algorithms for streaming data.

The rest of the paper is structured as follows. Section 2 reviews of the existing literature related to the forecasting algorithms for streaming energy data. In Sect. 3 the proposed methodology to forecast streaming time series is introduced. Section 4 presents the experimental results corresponding to the prediction of the energy consumption. Finally, Sect. 5 closes the paper giving some final conclusions.

2 Related Work

Although time series forecasting have been extensively studied in the literature, there are very few works on time series forecasting for streaming big data environments. In general, the methods for predicting time series can be classified into classical methods based on Box and Jenkins [1], such as ARIMA and GARCH; and machine learning methods, such as support vector machines, nearest neighbors techniques and artificial neural networks. For a taxonomy of these techniques applied to energy time series forecasting, the reader is referred to [2].

In the last few years, some other techniques have been developed to forecast time series in the context of big data. In [3] the energy consumption in several buildings of a public university is predicted by applying a distributed k-means algorithm in Spark. The study in [4] shows that suitable accuracy predictions can be achieved by applying a deep learning algorithm to electricity consumption data in Spain. On the other hand, in [5] different scalable methods such as decision tree, gradient boosted trees and random forest, are used to also predict the electricity consumption in Spain.

One of the main challenges today is real-time decision making based on the analysis of continuous data streams usually coming from sensors in an IoT context within the new edge computing paradigm or smart grids. A new representation of energy data streams was proposed in [6] for purpose of detecting outlier consumers by applying clustering techniques in smart grids. In [7] the authors presented an energy prediction system based on an edge computing architecture.

524 P. Jiménez-Herrera et al.

In particular, an online deep neural network model adapted to the characteristics of IoT data was implemented for energy prediction. In [8] a new methodology for real-time forecasting of energy demand using weather predicted data was proposed. In [9] neural networks with different backpropagation algorithms was also proposed for real-time energy consumption forecasting using climate data. A combination of an online clustering algorithm with a neural-network based predictive model was presented for electricity load forecast in [10].

After a thorough review, it can be concluded that very few papers have been published to predict online streaming electricity time series and there is still a lot of research to be done.

3 Methodology

3.1 Overview

The StreamNSP algorithm has been developed for streaming of time series data, and it is based on a combination of the K-means clustering algorithm [11] along with both the Naive Bayes (NB) classifier [12] and the K nearest neighbors (KNN) algorithm [13] for regression.

The general idea behind the proposed forecasting algorithm is to take a training set in a offline phase and firstly divide it in clusters using the K-means algorithm. Then, the KNN algorithm is applied for each cluster producing a list of trained prediction models, one per each cluster. Finally, the NB classifier is trained for predicting the cluster label of an instance using as training the cluster assignments previously generated by K-means.

Once the model of StreamNSP is generated in the offline phase, it is tested and then updated online using data streams. The methodology carried out to train, test and compare StreamNSP with a set of benchmark algorithms is graphically described in the Fig. 1.

In first place, a previous process of data preparation is performed from the historical time series data. This process is described in Sect. 3.2. After the preparation of data, a model is produced for each forecasting horizon, using both the StreamNSP algorithm and a set of benchmark algorithms. Each model is evaluated using a prequential or interleaved test-then-train evaluation. Finally, a comparison of error metrics was performed. The same evaluation procedure is

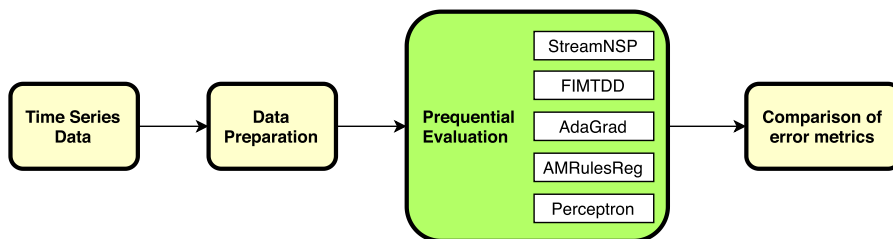


Fig. 1. Overview of the processes within the proposed methodology.

carried out for both StreamNSP and benchmark algorithms. The offline phase of StreamNSP is described in Sect. 3.3 and its online phase in Sect. 3.4. The benchmark algorithms are described in Sect. 4.3.

The algorithm StreamNSP was implemented in the Java programming language (Oracle Java 1.8.0_152-b16 SE for 64 bits) and adapted to be compatible for the MOA framework [14]. All experiments for testing and benchmarking StreamNSP were automated using the API of the MOA framework (version 19.04).

3.2 Data Preparation

The proposed forecasting algorithm is based on attributes (features) and a single numeric class to predict. However, time series data in a streaming is a sequence of numeric values. For such reason, the following data preparation process was made before to train and test the model. The Fig. 2 describes visually the procedure carried out to transform the time series data into a set of training and test for each prediction horizon. Specifically, a set of w lagged variables, a_1, \dots, a_w , were extracted from a time series x_1, \dots, x_n . These values reflect lagged windows of size w from the time series, as it can be seen in Fig. 2. These variables will act as input attributes for the model. Along with these attributes, a last column y representing the class was added to the so called propositional tables for each forecasting horizon from 1 to h , where h is the number of future values to predict. These column is a future value, in the time series, with respect to the past values window.

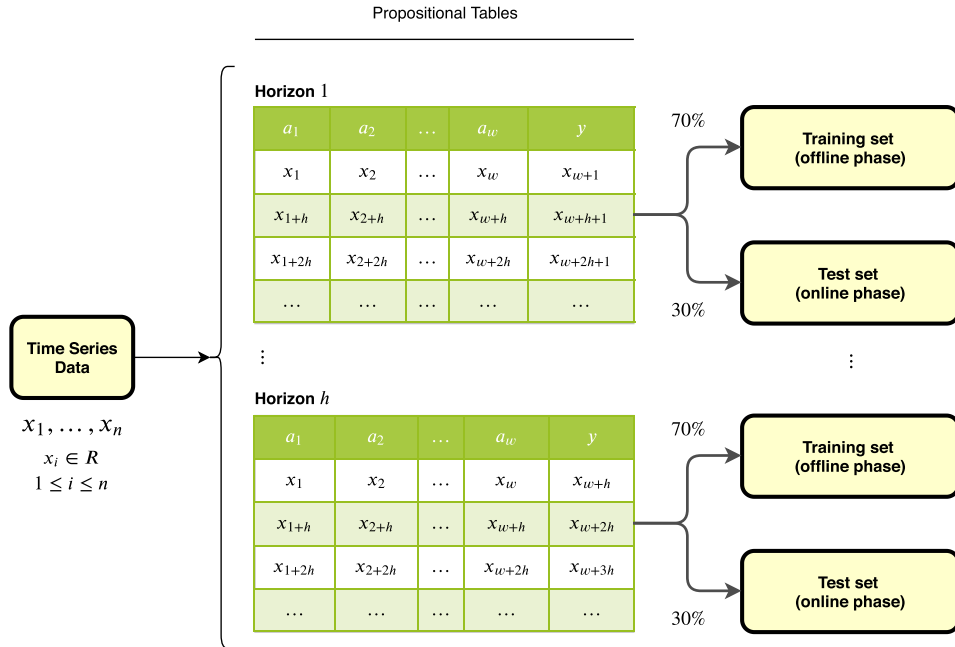


Fig. 2. Data preparation process.

Finally, for each horizon, both training and test subsets were taken from propositional tables, maintaining the original temporal order. The training set will be used to train the model in its offline phase, while the test set will be used to test and update the model in its online phase.

3.3 Offline Phase

The procedure carried out by the proposed StreamNSP algorithm in its offline phase is described in Fig. 3. As it can be seen in such figure, given a training set, each instance is extracted one by one according to its temporal order. Such instance is composed by its attributes a_1, \dots, a_w and its numeric class y .

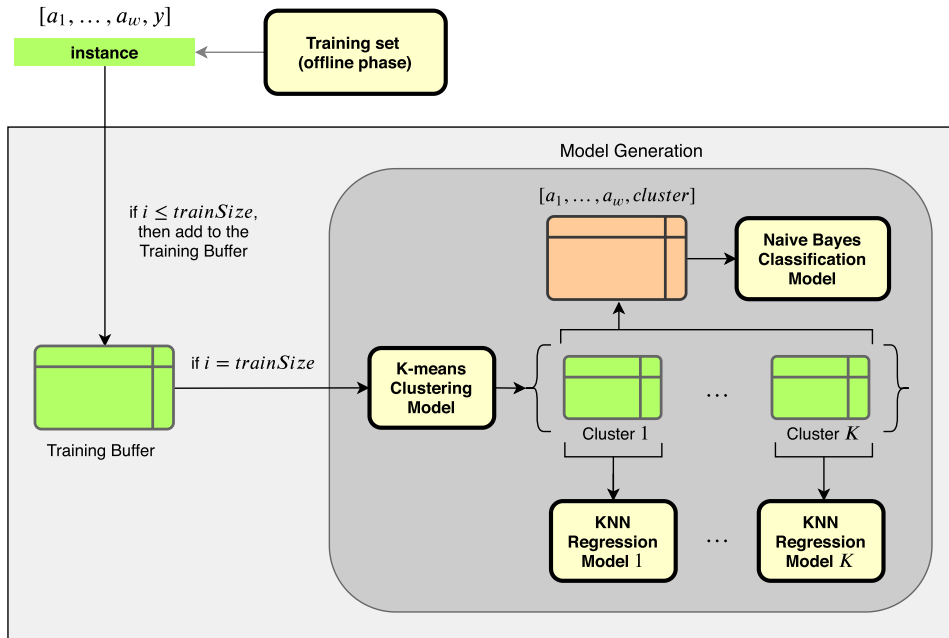


Fig. 3. Offline phase of the StreamNSP algorithm. (Color figure online)

Each instance from training set is stored in an internal training buffer, as it is shown in Fig. 3. Once such buffer is completed (i.e. when the number of instances i is equal to the training size $trainSize$) then such buffer is given to the K-means clustering algorithm. In this work, $K = 3$ clusters was set to model low, medium and high consumption.

As a result of the clustering, each instance is stored separately according to its assigned cluster. Then, a KNN regression model is trained and stored for each cluster. In this work, three KNN regression models were stored ($K = 3$). The algorithm KNN used was configured for one nearest neighbor search.

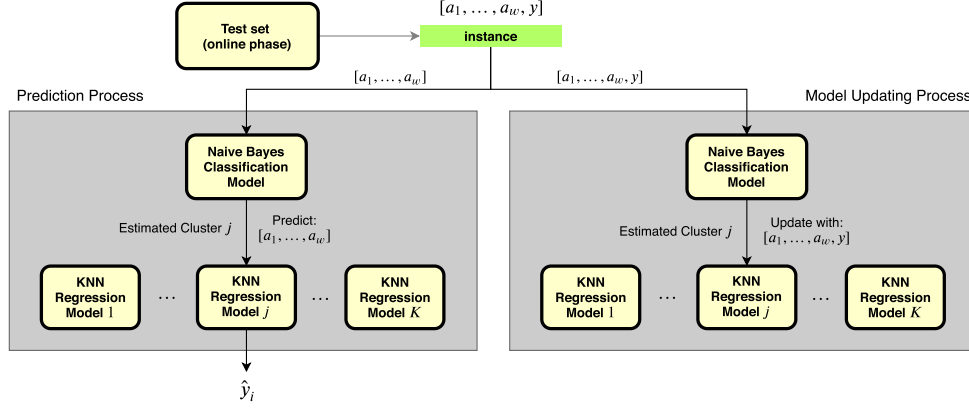


Fig. 4. Online phase of the StreamNSP algorithm.

A table containing the attributes a_1, \dots, a_w of all training instances along with their cluster assignments (the *cluster* label) is also made and stored (table coloured in orange in Fig. 3). Finally, a Naive Bayes model is trained using such table, with the aim to predict the cluster label of further test instances.

3.4 Online Phase

The procedure carried out by the proposed StreamNSP algorithm in its online phase is described in Fig. 4. As it can be seen, each instance is extracted one by one in online streaming from a test set. Such instance is composed of its attributes a_1, \dots, a_w and its numeric class y . The online phase is divided in two steps, due to the prequential evaluation performed. First, the step of predicting and, after, the step of updating the model.

The prediction step consists in receiving online the attributes a_1, \dots, a_w of the instance and use the Naive Bayes classification model previously trained. Such model returns the estimated cluster to which the test instance could belong (let be the cluster j). Then, the proper KNN regression model (the model j) is used to predict the numeric class \hat{y}_i of the instance.

The model updating step consists in receiving both the attributes a_1, \dots, a_w and the actual class y of the instance. Using the Naive Bayes classification model, an estimated cluster is produced (the same cluster as the prediction step, because the instance attributes are the same). Finally, the regression model j is updated using both the attributes and class of the instance.

4 Experimentation and Results

4.1 Dataset

The time series considered in this study is related to the electricity consumption in Spain from 1 January 2007 at 00:00 to 21 June 2016 at 23:50. It is a time

528 P. Jiménez-Herrera et al.

series of 9 years and 6 months with a high sampling frequency (10 min), resulting in 497,832 measures in total.

The time series was processed as described in Sect. 3.2 for a window of 144 lagged values corresponding to one day and a horizon of 24 values corresponding to 4 h. Thus, the past day is used to predict the next 4 h. After the preprocessing, we have removed the first 144 rows and the last 24 rows, in order to avoid empty values in the instances.

4.2 Evaluation Metrics

The MAE, RSME and MAPE errors have been used to evaluate the results of the proposed StreamNSP algorithm and the benchmark algorithms.

The MAE is the mean absolute error, which is the average of the absolute differences between actual and predictions. The MAE is defined in Eq. (1), where y_i are actual values, \hat{y}_i predicted values and n is the number of predicted samples.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (1)$$

The RMSE is the root mean square error, which represents the square root of the second sample moment of the differences between predicted values and observed values or the quadratic mean of these differences. The RMSE metric is defined in Eq. (2).

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2)$$

Finally, the third evaluation metric is the MAPE, which is calculated as the average absolute percent error, that is, the average of actual values minus predicted values divided by actual values as can be seen in Eq. (3).

$$MAPE = \frac{100}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (3)$$

4.3 Benchmark Algorithms

In order to compare the forecasting results of the algorithm StreamNSP with other suitable approaches in the literature, four regression algorithms for online learning from data streaming were selected.

The algorithm FIMTDD [15] learns a regression tree and it is able to address time-changing data streams. The algorithm has mechanisms for drift detection and model adaptation, which enable it to maintain accurate and updated regression models at any time. The algorithm observes each example only once at the speed of arrival, and maintains at any-time a ready-to-use tree model. The tree leaves contain linear models induced online. The number of instances a leaf

should observe between split attempts was set to 200. The threshold below which a split will be forced to break ties was set to 0.05.

The algorithm AdaGrad [16] is an online optimizer for learning linear regression from data streams that incorporates knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning. The adaptation feature of this algorithm derives into strong regret guarantees, which for some natural data distributions achieve high performance. No regularization was used for the linear regression model and the learning rate was set to 0.01.

The algorithm AMRulesReg [17] is an adaptive model that is able to generate decision rules for regression from data streams. In this model, the antecedent of a rule is a conjunction of conditions on the attribute values, and the consequent is a linear combination of attribute values. Each rule uses a Page-Hinkley test [18] to detect changes in the process generating data and react to changes by pruning the rule set.

Finally, the algorithm Perceptron [19] is based on the algorithm Hoeffding Trees [20], but instead of using Naive Bayes models at the leaf nodes it uses perceptron regressors. The perceptron regressors use the sigmoid activation function instead of the threshold activation function and optimize the squared error.

4.4 Results

The forecasting results obtained for StreamNSP and the benchmark algorithms using the electricity time series data stream are presented and discussed in this section.

Table 1 shows the MAE, RMSE and MAPE for StreamNSP, AdaGrad, FIMTDD, Perceptron and AMRulesReg. These errors have been calculated and averaged for all predicted horizons. As it can be seen that our StreamNSP algorithm achieved better results in MAE, RMSE and MAPE than the other four benchmark learners. The MAE error of StreamNSP has been compared with the best MAE result of the other four benchmark learners, in this case AMRulesReg, and it provides an error improvement of 66.958 MW. The difference between the RMSE for AMRulesReg and StreamNSP algorithms is also high, in particular 144,315 higher for AMRulesReg. In addition, AMRulesReg obtained a MAPE of 0.313% worse than StreamNSP. Note that AdaGrad algorithm seems to be not suitable for predicting energy consumption data, because of its high errors for all the computed metrics.

Table 2 shows the errors for each prediction horizon. As it can be seen, the growth of both MAE, RMSE and MAPE errors for the proposed StreamNSP algorithm is lower and more stable across the predicted horizons than that of the other four benchmarks. The increase of the MAPE across the 24 predicted horizons is 0.816%. This value is computed by the difference between the maximum MAPE (2.336 for the horizon 24) and the minimum (1.52 for the horizon 1). The errors were always increasing across the horizons.

Table 1. Comparison of the errors ordered by MAPE.

Algorithm	MAE	RMSE	MAPE (%)
AdaGrad	3242.434	3872.137	12.052
FIMTDD	1332.544	1751.836	4.911
Perceptron	612.676	986.839	2.297
AMRulesReg	590.401	944.853	2.211
StreamNSP	523.443	800.538	1.898

Table 2. Errors of the StreamNSP algorithm for each predicted horizon.

Horizon	MAE	RMSE	MAPE (%)	Horizon	MAE	RMSE	MAPE (%)
1	421.573	568.969	1.520	13	527.449	753.941	1.907
2	428.810	578.613	1.554	14	534.718	776.120	1.942
3	435.304	590.439	1.571	15	544.480	798.651	1.976
4	443.158	604.724	1.595	16	552.460	823.934	2.008
5	454.045	621.032	1.637	17	560.636	854.281	2.050
6	466.106	635.736	1.674	18	580.300	915.053	2.116
7	470.065	640.124	1.687	19	593.585	959.358	2.166
8	473.147	644.658	1.698	20	604.592	996.627	2.214
9	477.315	650.281	1.713	21	614.681	1031.443	2.245
10	483.942	660.731	1.742	22	622.031	1051.802	2.269
11	492.637	682.344	1.783	23	627.828	1063.401	2.292
12	510.162	722.867	1.845	24	643.615	1129.542	2.336

Figure 5 shows the MAPE for each predicted horizon for both the StreamNSP and the benchmark methods. It can be observed that the StreamNSP exhibits the highest stability of MAPE across the predicted horizons with respect to the other benchmark learners. For the first 10 horizons, AMRulesReg and Perceptron methods provided a lower MAPE than StreamNSP. However, StreamNSP has the lowest average error for all horizons. StreamNSP has an error difference among horizons of 0.816%, while AMRulesReg and Perceptron have 3.217% and 3.467%, respectively. For horizons higher than 10, StreamNSP overcame the rest of algorithms, being the best method for higher horizons.

Figure 6 shows the best forecasted day according to the minimum MAPE obtained for the StreamNSP. This figure includes the actual and predicted values for each hour of the day (with 10 min of interval). This day corresponds to October 29, 2015. The x-axis is the time and the y-axis is the target variable (energy consumption in MW). The prediction fits very well at all times and it does not shows large errors for any time.

Figure 7 shows the forecasted day with the biggest MAPE for the StreamNSP. It can be seen high differences between the actual and predicted values. It seems

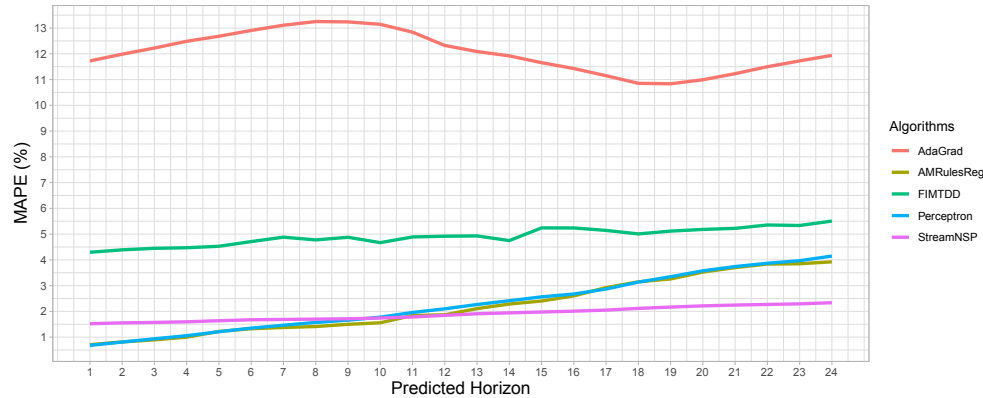


Fig. 5. Comparison of the MAPE for all algorithms along the predicted horizon.

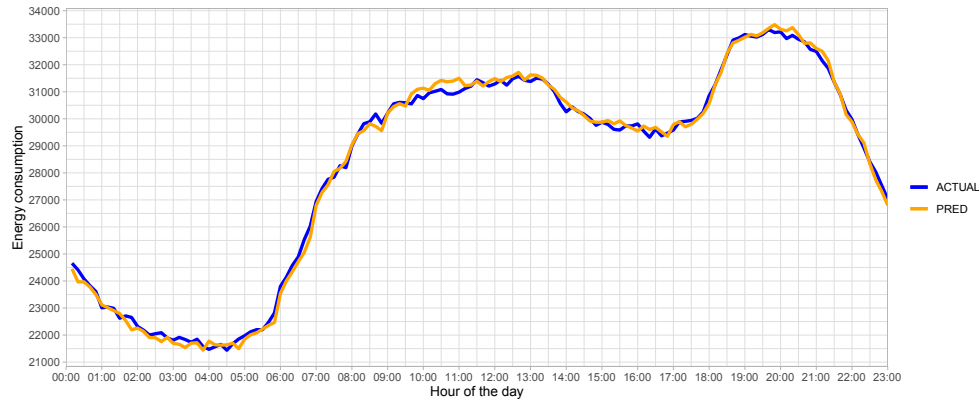


Fig. 6. Best forecasted day for the StreamNSP (day with the smallest MAPE).

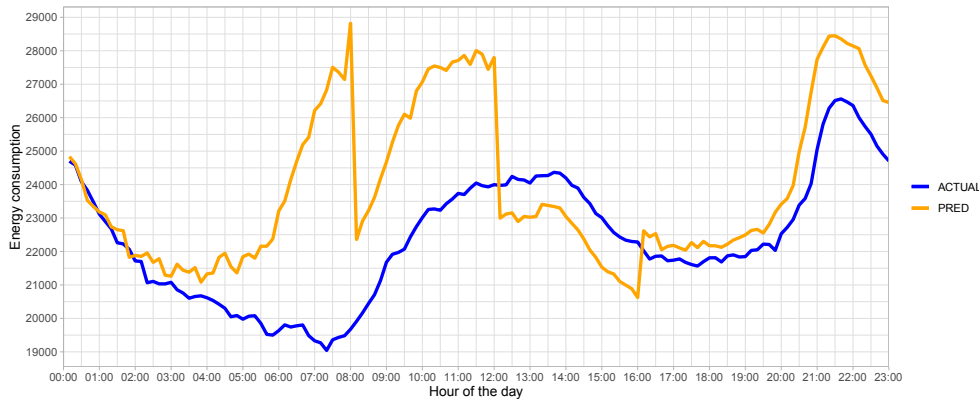


Fig. 7. Worst forecasted day for the StreamNSP (day with the maximum MAPE).

532 P. Jiménez-Herrera et al.

that the nearest neighbour selected to make these predictions may not represent correctly the energy consumption for such day. This day corresponds to May 1, 2014, which is a holiday in Spain (Labor day).

5 Conclusions

In this work the new StreamNSP forecasting algorithm has been proposed for online learning on streaming data. StreamNSP has an offline phase to obtain the prediction model using the historical data. This phase consists of splitting the training data into clusters using the K-means algorithm. Then, a nearest neighbors algorithm is applied for each cluster producing a list of trained regression models, one per each cluster. In addition to that, a Naive Bayes classifier is trained for predicting the cluster label of an instance using as training the cluster assignments previously generated by K-means. The algorithm can be updated incrementally for online learning from data streams including new instances into the model corresponding to its estimated cluster. StreamNSP has been tested using the electricity consumption with a granularity of 10 min for predicting a prediction horizon of four hours. The algorithm widely overcame other four online learners, such as AdaGrad, FIMTDD, Perceptron and AMRulesReg, achieving an average MAPE of 1.89% instead of 2.21%, 2.29%, 4.91% and 12.05% obtained by the other algorithms. Moreover, the StreamNSP has obtained the most accurate predictions for large forecasting horizons (11 or more values ahead).

As future work, other base algorithms will be tested for the clustering, classification and regression inner components of the StreamNSP. Furthermore, a sensitivity study of the number of clusters used in StreamNSP will be performed.

Acknowledgements. The authors would like to thank the Spanish Ministry of Science, Innovation and Universities for the support under the project TIN2017-88209-C2-1-R.

References

1. Box, G., Jenkins, G.: Time Series Analysis: Forecasting and Control. John Wiley, Hoboken (2008)
2. Martínez-Álvarez, F., Troncoso, A., Asencio-Cortés, G., Riquelme, J.C.: A survey on data mining techniques applied to electricity-related time series forecasting. *Energies* **8**(11), 13162–13193 (2015)
3. Pérez-Chacón, R., Luna-Romera, J.M., Troncoso, A., Martínez-Álvarez, F., Riquelme, J.C.: Big data analytics for discovering electricity consumption patterns in smart cities. *Energies* **11**, 683 (2018)
4. Torres, J.F., Galicia, A., Troncoso, A., Martínez-Álvarez, F.: A scalable approach based on deep learning for big data time series forecasting. *Integr. Comput.-Aided Eng.* **25**(4), 335–348 (2018)
5. Galicia, A., Torres, J.F., Martínez-Álvarez, F., Troncoso, A.: A novel spark-based multi-step forecasting algorithm for big data time series. *Inf. Sci.* **467**, 800–818 (2018)

6. Laurinec, P., Lucká, M.: Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting. *Data Mining Knowl. Disc.* **33**(2), 413–445 (2018). <https://doi.org/10.1007/s10618-018-0598-2>
7. Luo, H., Cai, H., Yu, H., Sun, Y., Bi, Z., Jiang, L.: A short-term energy prediction system based on edge computing for smart city. *Fut. Gener. Comput. Syst.* **101**, 444–457 (2019)
8. Kwak, Y., Seo, D., Jang, C., Huh, J.-H.: Feasibility study on a novel methodology for short-term real-time energy demand prediction using weather forecasting data. *Energy Build.* **57**, 250–260 (2013)
9. Ahmad, T., Chen, H.: A review on machine learning forecasting growth trends and their real-time applications in different energy systems. *Sustain. Cities Soc.* **54**, 102010 (2020)
10. Gama, J., Rodrigues, P.P.: Stream-based electricity load forecast. In: *Proceedings of the Knowledge Discovery in Databases*, pp. 446–453 (2007)
11. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297 (1967)
12. John, G.H., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pp. 338–345 (1995)
13. Aha, D., Kibler, D.: Instance-based learning algorithms. *Mach. Learn.* **6**, 37–66 (1991)
14. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. *J. Mach. Learn. Res.* **11**, 1601–1604 (2010)
15. Ikononovska, E., Gama, J., Džeroski, S.: Learning model trees from evolving data streams. *Data Mining Knowl. Disc.* **23**(1), 128–168 (2011)
16. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**, 2121–2159 (2011)
17. Almeida, E., Ferreira, C., Gama, J.: Adaptive model rules from data streams. In: *Proceedings of the Machine Learning and Knowledge Discovery in Databases*, pp. 480–492 (2013)
18. Basseville, M.: Detecting changes in signals and systems-a survey. *Automatica* **24**(3), 309–326 (1988)
19. Bifet, A., Holmes, G., Pfahringer, B., Frank, E.: Fast perceptron decision tree learning from evolving data streams. In: *Proceedings of the Advances in Knowledge Discovery and Data Mining*, pp. 299–310 (2010)
20. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: *Proceedings of the Knowledge Discovery on Databases*, pp. 97–106 (2001)

5.1.5 | "Discovering three-dimensional patterns in real-time from data streams: an online triclustering approach"

Authors: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A.

Publication type: Journal article.

Journal: Information Sciences.

Year: 2021.

Volume: 558

Pages: 174-193

DOI: 10.1016/j.ins.2020.12.089

IF: 5.910 9/156 Computer Science, Information Systems.

Quartil: Q1.



Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Discovering three-dimensional patterns in real-time from data streams: An online triclustering approach



Laura Melgar-García^a, David Gutiérrez-Avilés^a, Cristina Rubio-Escudero^b, Alicia Troncoso^{a,*}

^a Data Science & Big Data Lab, Pablo de Olavide University, ES-41013 Seville, Spain

^b Department of Computer Science, University of Seville, Avda. Reina Mercedes s/n, Seville 41012, Spain

ARTICLE INFO

Article history:

Received 22 July 2020

Received in revised form 13 November 2020

Accepted 30 December 2020

Available online 26 January 2021

Keywords:

Data streaming

Patterns

Real-time

Triclustering

ABSTRACT

Triclustering algorithms group sets of coordinates of 3-dimensional datasets. In this paper, a new triclustering approach for data streams is introduced. It follows a streaming scheme of learning in two steps: offline and online phases. First, the offline phase provides a summary model with the components of the triclusters. Then, the second stage is the online phase to deal with data in streaming. This online phase consists in using the summary model obtained in the offline stage to update the triclusters as fast as possible with genetic operators. Results using three types of synthetic datasets and a real-world environmental sensor dataset are reported. The performance of the proposed triclustering streaming algorithm is compared to a batch triclustering algorithm, showing an accurate performance both in terms of quality and running times.

© 2021 Elsevier Inc. All rights reserved.

1. Introduction

Currently, research efforts are focused on developing new methodologies that consider continuous and massive flows of data from a wide variety of sources with the objective of analyzing and taking advantage of them. Therefore, the term Data Streaming that defines these in motion data takes relevance in this context. Nowadays, there are a vast variety of data streaming sources such as Internet of Things (IoT), social media, medical devices, or videos. When dealing with continuous flows of data and real-time characteristics, traditional machine learning methodologies (i.e., clustering, regression, classification) have to be adapted to this new environment [1].

There are two main computational approaches to develop machine learning models for data streaming [1]. On the one hand, incremental learning is found, where the model evolves and adapts incrementally to concept drift [2] in data streams. On the other hand, the offline-online learning emerges, where the model is divided into an offline phase in order to create a summary (or sketch) of the data without time execution restrictions and an online phase to update the synopsis data in real-time as fast as possible.

Among all the streaming analysis tasks, the behavior pattern extraction becomes relevance as it is the base of a vast number of current real-time applications such as customer analysis, fraud detection, or sentimental analysis [3]. Different types of algorithms are found in the literature depending on the type of patterns to be searched. Clustering [4] extracts similar behaviour patterns over all the features analyzed. Biclustering [5] appears as an evolution of clustering since patterns can

* Corresponding author.

E-mail address: atrolor@upo.es (A. Troncoso).

be extracted from a subset of instances and features. Triclustering [6] allows for considering a third dimension in the dataset, usually time, finding groups of elements with similar behaviour throughout three-dimensional datasets.

In this work, we present a new online triclustering algorithm for data streaming, called STriGen. The STriGen is a learning method based on genetic algorithms, which combines an offline and online phase, to discover collections of resembling patterns in 3D data streams in real time. We show the results obtained from the application of STriGen to three synthetic datasets of different complexity and to a real dataset collected from seven environmental sensors. The results obtained for the synthetic datasets are validated by means of comparing the triclusters found to the real ones, providing quality evaluation measures such as the accuracy and F1 score. The triclusters obtained by the STriGen using the real dataset can not be validated as the synthetic ones since the real triclusters are not known, and therefore we use the Graphical Quality (GRQ) measure described in [7].

The rest of the paper is structured as follows: a summary of the previous research related to the paper's topic is presented in Section 2; Section 3 describes how the offline and online part of the proposed algorithm works; the experimental setting carried out and the results obtained are shown in Section 4; and, finally, the conclusions and future works are provided in Section 5.

2. Related works

In this Section, the current state of the art related to the proposed research is presented. The STriGen algorithm is associated to two main areas: the streaming analysis and the triclustering approach.

Regarding the streaming analysis topic, several proposals of new machine learning approaches adapted to this emerging environment can be found in the literature. In [8], an online version of the support vector machine model was developed to predict air pollutant levels using streaming time series. The authors in [9] presented a streaming version of the linear discriminant analysis algorithm for dimension reduction. In [10], the authors developed a streaming analytics system for large-scale data. An adaptive ensemble learning for online activity recognition from data streams was presented in [11]. Furthermore, efforts to develop tools to facilitate the adaption of classic machine learning models to the streaming environment have been carried out. In this sense, a general adaptive incremental learning framework for streaming data analysis was performed in [12] and, an API to apply machine learning algorithms to data streams, well-known as SAMOA, was introduced in [13].

In streaming environments, both the prompt anomaly detection and the generation of a continuous flow of data are highly explored fields. The authors in [14] presented a real-time methodology to detect cybercrimes related to credit card frauds; another anomaly detection algorithm for streams of data was developed in [15]. In an overview regarding the pattern discovery task in streaming, the authors in [16] proposed a methodology to discover patterns in multiple time-series in streaming. A classification of streaming features based on an emerging-pattern approach was presented in [17]. The authors presented SPADE in [18], which is a shape-based pattern detection method for streaming time-series.

Also, new online machine learning approaches have emerged in the area of evolving systems in clustering due to the rise of the streaming analysis field. In [19], the authors developed a fuzzy-rule-based model with the capability to adapt to the new streams of data and to deal with missing values efficiently. The authors in [20] presented an evolving optimal granular system to perform the approximation of functions such as time-series models, classification or regression functions, demonstrating its outperform when comparing with other evolving methods in multivariate problems. The authors in [21] presented a new approach for online regression and system identification problems in data streams, based on evolving fuzzy systems. An updating of the previous methodology was developed in [22], where the authors presented a new rule splitting framework for generalized evolving fuzzy systems, demonstrating the outperform of this new algorithm against the original version. In [23], the author presented a new online incremental clustering algorithm based on a dynamic cluster merging method, which consisted in the calculation of the covariance matrix of the clusters susceptible to be joined. Furthermore, in [20], the authors developed other evolving systems specializing in evolving fuzzy rule-based models and neuro-fuzzy networks in online and real-time environments. A complete survey of evolving systems dealing with streaming data can be found in [24].

In the streaming analysis field, an important research area is the adaptation of existing methodologies to the streaming environment. Thus, some algorithms are adapted to the streaming environment using incremental learning. For example, K-Means methods for data streaming modify just the calculation process of the closest mean to the new point instead of applying the whole algorithm every time a new point arrives [25]. We can also find a mixed online/offline strategy, which is advantageous as streaming concerns just the online phase. In the offline phase, traditional algorithms are applied to the data without meeting all the requirements of data streaming. During the online phase, selected streaming data are structured, keeping only a statistic summary of them and not the full data. StreamKM++ [26] and CluStream [27] are some examples of this online/offline methodology.

Finally, the triclustering topic emerges as a methodology for gene expression data time series. The goal in triclustering is the same as in clustering, that is, minimizing the intra-triclustering distance and maximizing the inter-triclustering distance. There are different triclustering algorithms approaches depending on their behaviour: iterative searches, distribution param-

eter identification, pattern mining, evolutionary multi-objective optimization, among other options [28]. The authors introduced an algorithm based on the symmetry property of the triclusters in [29]. Lately, an extended and generalized version of the previous proposal was presented in [30]. An evolutionary computation approach was proposed in [31], where the fitness function was defined as a multi-objective measure. Another proposal based on genetic algorithms was developed in [32], where the authors mined the optimal shifting and scaling patterns from 3D-microarrays. The triclustering genetic-based algorithm, TriGen, also used multi-objective optimization approach [6]. The authors in [33] developed a triclustering algorithm based on a statistical methodology for 3D short gene expression data time series datasets. The triclustering techniques have been also applied to other areas. In particular, a triclustering algorithm was used to depict seismogenic zoning over the Iberian Peninsula in [34]. The authors applied triclustering techniques in high-content screening images to identify its components in [35]. Moreover, triclustering techniques have been successfully applied to medical environments, like in [36], where triclustering was applied in combination with random forest to predict the need for non-invasive ventilation in ALS patients. A survey of several existing triclustering algorithms can be found in [37].

3. Materials and methods

This Section describes the proposed algorithm STriGen. Section 3.1 defines and specifies the main characteristics of the triclusters. The STriGen algorithm is presented in Section 3.2, including the type of data that it needs, the description of its two phases and the validation of the resulting triclusters.

3.1. Tricuster definition

Triclustering algorithms are an evolution of clustering algorithms [38]. Specifically, clustering is applied on 2-dimensional datasets and triclustering on 3-dimensional datasets. In both cases, data are a set of instances (rows in a matrix) and features (columns in a matrix) and, just in triclustering, data also contain a set of time points (depths in a matrix).

In particular, a cluster is defined as a group of instances over all features. However, a tricluster is defined as a group of instances over a group of features and over a group of time points. For example, in a 3-dimensional dataset with L instances, K features and P time points, a resulting tricluster is a subset composed of $j \leq L$ instances, $h \leq K$ features and $y \leq P$ time points, respectively. An example tricluster formed by $j \times h \times y$ components is represented in Fig. 1c. Fig. 1a represents the 3-dimensional data and its 3 slices are presented in Fig. 1b.

3.2. The STriGen algorithm

STriGen is a triclustering algorithm based on a genetic evolutionary heuristic that finds groups of similar behaviour patterns in 3-dimensional streaming datasets.

Data streams are continuous flows of data supplying new information. As data can possibly have an infinite volume, there are some constraints that every streaming algorithm needs to satisfy [39]: single-pass and chronological order, i.e., streams are processed one by one, only once and in the order of arrival. In addition, stream models have to incorporate new information updating themselves dynamically and they have to detect and eliminate outdated data effects (also called concept drift detection). Stream models also have to deal with very important constraints as bounded memory and bounded response time. Therefore, only a summary of specific data can be stored and outputs must be provided as fast as possible and the execution time of the learning model must increase linearly according to the number of instances. The STriGen algorithm deals with data streams and satisfies all mentioned constraints.

Moreover, as streams can be infinite, there are different options to define which time window of data has to be considered [1]. The proposed algorithm uses the w sliding window where the model takes into consideration only the most recent instances. In particular, w is an integer number from 2 to the current number of streams. Thus, 2-dimensional data arrive with all L instances and all K features, as defined in Section 3.1, so one stream is formed by $L \times K$ samples, as shown in

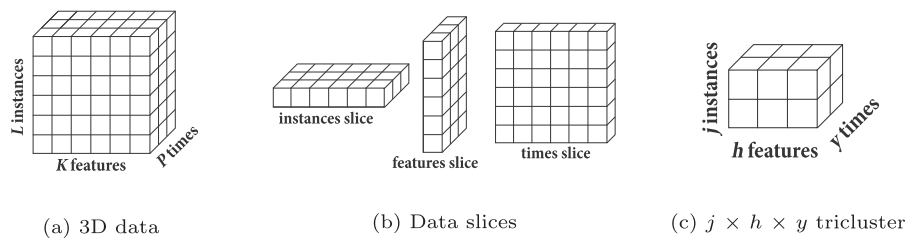


Fig. 1. Representation of triclusters.

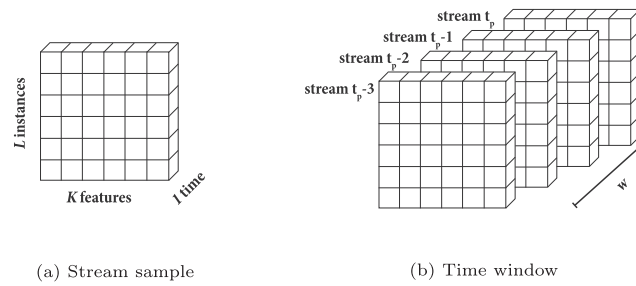


Fig. 2. Representation of streams.

Fig. 2a. Fig. 2b represents a particular case where w is 3 and so, by definition of sliding window, only the 3 most recent streaming samples are considered.

There are two main computational approaches to model data streams [1]: the incremental learning and the offline-online learning. In the first one, incremental learning, the model evolves and adapts incrementally to concept drift in data streams. However, in the offline-online learning, the modeling is divided in an offline phase that creates a summary (or sketch) of the data without execution time restrictions and an online phase to update the synopsis data in real time as fast as possible. The proposed algorithm uses the offline-online learning approach and uses Apache Kafka as streaming platform. Fig. 3 is an overview of the workflow of the proposed algorithm. It starts with the offline phase that creates a summary model containing the components of the triclusters found by the algorithm as explained in Section 3.2.1. Then, the Kafka producer publishes the streaming data in a Kafka topic. A Kafka topic is a container that stores the data streams. The Kafka consumer receives the data in streaming from the Kafka topic and executes the online phase of the algorithm. In this phase online, the initial triclusters resulting from the offline step evolve over time as it receives new data streams to keep the model always updated as explained in Section 3.2.2.

These two phases of the algorithm are described in detail below. The offline phase is described in Section 3.2.1 and the online in the Section 3.2.2.

3.2.1. The offline phase

The offline phase of streaming algorithms processes data in a static or batch mode, i.e., not considering the requirements of the streaming algorithms above-mentioned. However, the output of this phase has to be a summary or sketch of the data that is the base of the online phase.

In particular, the offline phase of the proposed algorithm creates a summary of the components of each tricluster found. STriGen applies evolutionary meta-heuristics of genetic algorithms to find triclusters. The process of natural selection is represented by the selection of the fittest individuals for reproduction in order to produce the offspring of the next generation. Therefore, there are two main actors in the algorithm: population and individuals. One individual is a potential solution of the algorithm, i.e., a tricluster containing instances, features and streams, and a population is composed of several individuals.

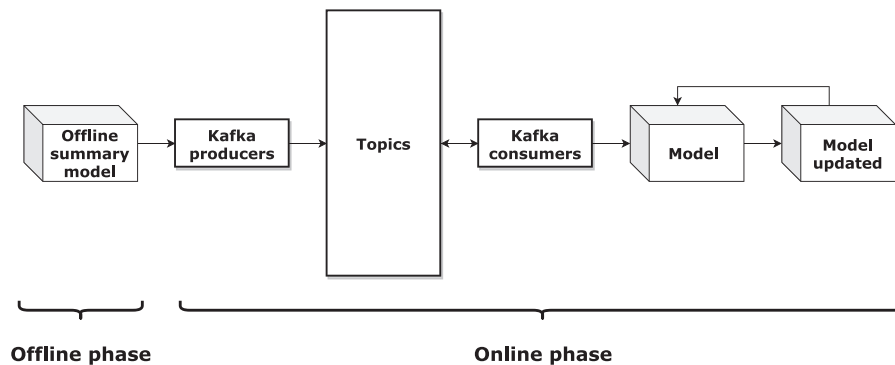


Fig. 3. Offline and online phases of the STriGen.

The algorithm starts creating an initial population with a random selection of instances, features and streams for each individual. Afterwards, four genetic operators are used to make the population of individuals evolves optimizing a fitness function. Once a solution tricluster is found, this process is repeated for all the rest of the desired triclusters but with a different way of creating the initial population. In other words, considering N as the number of triclusters to find, for the rest $N-1$ triclusters, a percentage of individuals are generated randomly and the others are generated considering the non-explored areas of the previous solution triclusters in order to avoid overlapping solutions.

Inspired by [6], the genetic operators, together with the initialization of the population, are crossover to make connections between individuals and share their components, mutation to make specific changes to individuals and explore new possible components, selection to choose which individual stays in the next generation and evaluation to measure the quality of the population. These operations are made considering some triclustering configuration parameters: the number of solution triclusters, the number of generations, the members of a population, the aleatory factor for the initial population, the selection rate and the mutation probability rate. The crossover is implemented by means of a random one-point crossover [40]. Mutation is made by inserting, deleting or changing instances or features of the current tricluster. The tournament selection mechanism is used to select the best individuals and the evaluation is implemented by means of a fitness function.

The fitness function employed by the proposed algorithm is the Multiple Square Lines (MSL), which is based on the similarity among the angles of the slopes formed by the components of the tricluster at each time point. MSL is completely described in [7]. In addition two terms are added to control the size of the triclusters regarding the number of instances, features and times and to control the overlapping of the triclusters, respectively.

Finally, the measure used to evaluate the quality of a solution tricluster TRI is the Graphical Quality (GRQ) measure:

$$GRQ(TRI) = 1 - \frac{MSL(TRI)}{2\pi} \quad (1)$$

A tricluster will have a higher graphical quality as smaller the MSL value is, which is minimized by means of the fitness function.

Finally, the components of the triclusters found in the offline phase are the individuals with the best fitness function value, and the final summary of the offline phase is the initial base of the online phase. A scheme of the main steps of the offline phase is shown in Algorithm 1.

Algorithm 1. Offline phase.

Result: summary model with triclusters components

```

repeat
  initialize population;
  evaluate population;
  repeat
    select individuals;
    crossover individuals;
    mutation individuals;
    evaluate new population;
  until number of generations;
  select the individual that minimizes the fitness function;
  tricluster is made up of the components of the selected individual;
until number of triclusters;
```

3.2.2. The online phase

Once the summary of the offline phase is ready, the online phase starts. In this phase, the algorithm considers all streaming requirements.

Firstly, as the algorithm uses a sliding window, just the most recent w samples are considered. In particular, if the new stream arrives at the instant point z , all triclusters components that include samples up to the $z - w - 1$ instant point are

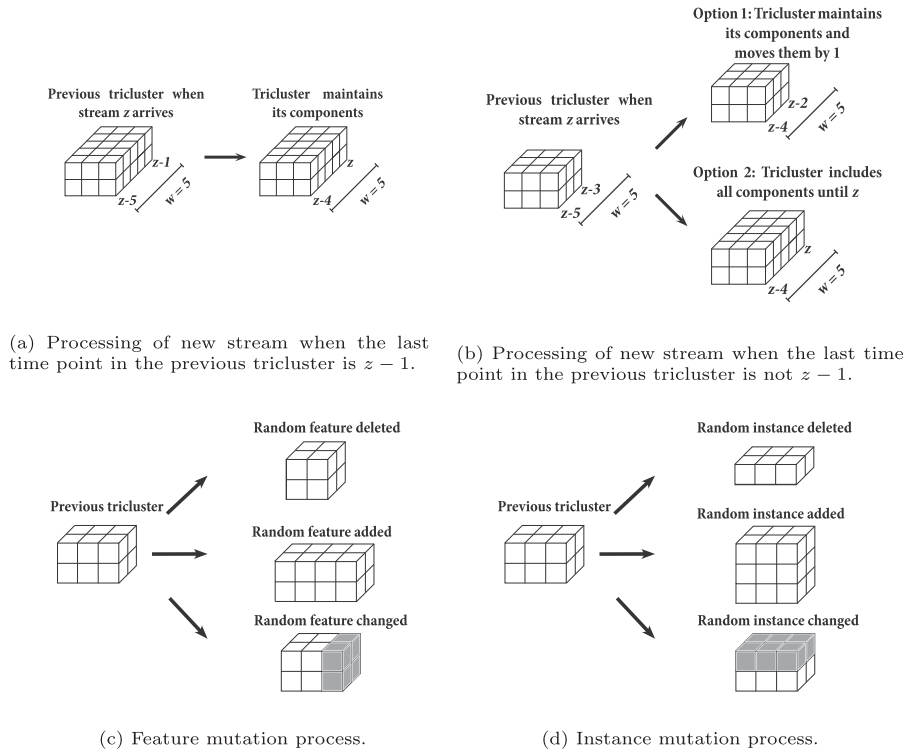


Fig. 4. Examples of the updating operations during the online phase.

deleted, i.e., they are not considered anymore. This is a common procedure in the streaming environment [41]. Thereby, the sliding window model removes the oldest samples every time a new stream sample arrives and so, triclusters contain the most recent data at every moment (see Fig. 2b).

Afterwards, the model has to evolve to include knowledge from the new stream sample. In traditional clustering techniques applied to streaming, as for example K-Means, a common strategy is to add the new instance to its nearest “centroid” if it is within the boundary of the cluster [42]. The STriGen algorithm computes different updating options and selects the one with the highest GRQ value (Eq. 1), that corresponds to the best graphical quality of each tricluster. All updating options try to find the most suitable tricluster’s components from the w streams.

One updating option is to maintain the same components of the existing tricluster from the instant point $z-1$ to $z-w$. In other words, including as components at instant or stream z the same instances and features of the previous tricluster. Fig. 4a shows a graphical example of this updating. In the particular case of triclusters that do not include the instant $z-1$ in their components, two updating options are carried out: either adding just the following instant point of the previous tricluster or all of the instant points until z . Fig. 4b is a graphical description of this updating, where the maximum number of instant points that can be included in a tricluster is five, as $w=5$. Thus, the tricluster contains just three instant points ($z-4$ to $z-2$) for the option 1 and the tricluster contains all five possible instant points ($z-4$ to z) for the option 2.

In addition, as the offline phase is a NP hard problem, it is probable that even with the application of these updating heuristics, the solution triclusters obtained might be a local rather than global optimal. To manage this issue and to allow triclusters to evolve and adapt to new data streams, the mutation operator is included as another updating possibility. This genetic operator deletes, changes or/and adds randomly samples (instances or/and features) in the triclusters resulting from the above-mentioned updates. Firstly, the type of mutation is randomly selected: deleting an existing coordinate, changing an existing coordinate to a new one or adding a new coordinate. Then, the specific instance or feature is also randomly selected. If the deleting option is randomly selected and a specific feature of the tricluster is selected, this feature is removed for all the instances and time points of the existing tricluster. In the case of the adding option, if a feature is randomly

selected from the data that are not included in the tricluster, this feature is added for all the instances and time points of the current tricluster and so, all its expression values. For the changing option, the deleting option is firstly executed and then the adding one. The instance mutations follow the same procedure of the feature mutations but considering the instances. Fig. 4c and Fig. 4d are graphical examples of mutation of features and instances, respectively. Once mutation is carried out, the GRQ is computed for all updated triclusters and the tricluster that maximizes the GRQ is selected. This updating process is represented in the pseudo code of the Algorithm 2.

Algorithm 2. Updating of the triclusters.

Result: Updated tricluster

$z \leftarrow$ current time;

$z-t \leftarrow$ last time of the tricluster TRI ;

if $z - t \neq z-1$ **then**

$newTRI_1 \leftarrow add(z - t + 1, TRI)$;

$newTRI_2 \leftarrow mutate(newTRI_1)$;

$newTRI_3 \leftarrow add([z - t + 1, \dots, z], TRI)$;

$newTRI_4 \leftarrow mutate(newTRI_3)$;

$[bestTRI, bestGRQ] \leftarrow$

$evaluate(newTRI_1, newTRI_2, newTRI_3, newTRI_4)$;

else

$newTRI_1 \leftarrow add(z, TRI)$;

$newTRI_2 \leftarrow mutate(newTRI_1)$;

$[bestTRI, bestGRQ] \leftarrow evaluate(newTRI_1, newTRI_2)$;

end

return $bestTRI$ and $bestGRQ$

Another important aspect when dealing with stream computing is to consider the concept drift problem. Concept drift occurs when an existing concept changes or a new concept appears [43]. There are different types of concept drift: incremental, gradual, abrupt or reoccurring. When any change starts to occur in expression values of the triclusters components, the GRQ decreases abruptly. In such cases, only one updating is not enough. The parameter *minGRQ* let the algorithm entering in a loop of a maximum of *numIt* iterations or until the GRQ is higher than *minGRQ*. Each iteration corresponds to a mutation updating process. In this way, STriGen can adjust rapidly to small changes but also to abrupt changes detecting new components of the triclusters or making the current ones disappear. This additional updating to deal with changes is represented in the pseudo code of the Algorithm 3.

Algorithm 3. Additional updating.

$iter \leftarrow 0$;

while $currentGRQ < minGRQ$ and $iter < numIt$ **do**

$TRI \leftarrow mutate(TRI)$;

$currentGRQ \leftarrow evaluate(TRI)$;

$iter \leftarrow iter + 1$;

end

return TRI

In summary, an overview of the online phase is depicted in the pseudo code of the Algorithm 4.

Algorithm 4. Online phase.

Result: Set of updated triclusters S

$S \leftarrow \{\};$

while new data stream at time z **do**

for each TRI in current model **do**

$TRI \leftarrow \text{remove}(z - w - 1, TRI);$

$[\text{newTRI}, \text{currentGRQ}] = \text{updating}(TRI) \text{ //algorithm 2};$

if $\text{currentGRQ} < \text{minGRQ}$ **then**

$\text{updatedTRI} \leftarrow \text{additionalUpdating}(\text{newTRI}) \text{ //algorithm 3};$

else

$\text{updatedTRI} \leftarrow \text{newTRI};$

end

$S \leftarrow S \cup \text{updatedTRI};$

end

end

return S

3.2.3. Validation of triclusters

In the case of using synthetic data as experimental data, the triclusters' coordinates that STriGen should find are previously known. To evaluate the performance of the STriGen in this situation two metrics, F1 score and accuracy, are computed. Accuracy is the most used measure in the classification field. It represents the ratio of correctly predicted observation with respect to the total number of observations, and it is appropriate if the dataset is symmetric, that is, when values of false positive and false negatives are almost the same. In our case, the coordinates not being part of a tricluster solution are far more abundant than the ones belonging to a tricluster. Therefore, we also include the F1 score, which is a weighted average of the precision and recall and considers both false positives and false negatives.

In the case of using real data as experimental data, F1 score and accuracy can not be computed. Then, we compute the GRQ value to assess the quality of the triclusters found by the STriGen.

4. Results

This Section presents the results obtained by the application of the STriGen algorithm to different datasets, three synthetics in Section 4.1 and one real in Section 4.2. In particular, Section 4.1.1 describes the synthetics datasets, in which STriGen has been tested. Then, a summary of the main parameter settings can be found in Section 4.1.2. Section 4.1.3 discusses the experimental results obtained by STriGen with different configuration settings.

4.1. Synthetic datasets

Synthetic datasets are essential to evaluate the quality of the algorithm, as we know in advance which should be the results, i.e., triclusters that STriGen must find.

4.1.1. Description of the synthetic datasets

The algorithm is applied on three synthetic datasets that follow additive, multiplicative and dynamic additive models respectively. A full description of these models can be found in [44,45]. This way, we can test how our model is capable to adapt to different types of changes over time in data streams. All three datasets are made up of 800 instances, 4 features

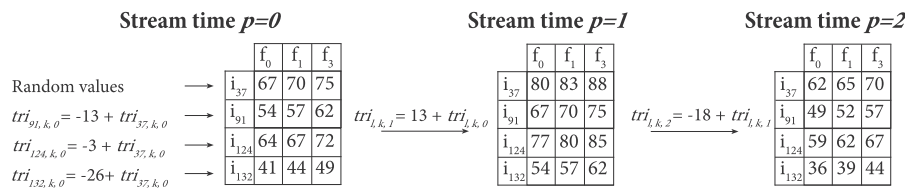


Fig. 5. Example of a tricluster in the additive dataset.

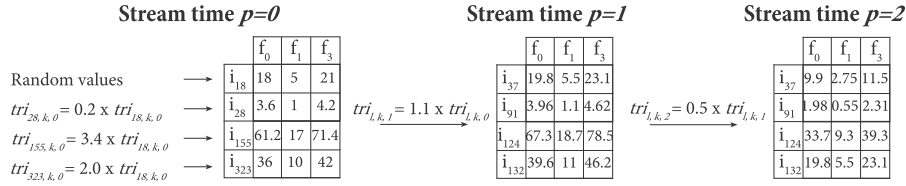


Fig. 6. Example of a tricluster in the multiplicative dataset.

and 500 streams for the two first datasets and 2100 streams for the third dataset. Instances are represented in rows, features in columns and streams represent the depth of each 3D dataset. Each dataset contains three triclusters with a different number of instances (between 25 and 30) and features (between 3 and 4). The components that are not in any tricluster, take random values.

- **Additive dataset:** For the additive synthetic dataset, a value $\gamma_p \in [-30, 30]$ has been added to each expression value in the tricluster at time p in order to create the additive pattern over time. Therefore, a tricluster tri at the stream time p in the additive dataset is defined as:

$$tri_{l,k,p} = \gamma_p + tri_{l,k,p-1} \quad 1 \leq l \leq L \quad 1 \leq k \leq K \quad (2)$$

where the tricluster at the initial time, $tri_{l,k,0}$, is generated with random expression values for each feature of the first instance and the expression values for all features of the remaining instances according to the following equation:

$$tri_{l,k,0} = \alpha_l + tri_{1,k,0} \quad 2 \leq l \leq L \quad 1 \leq k \leq K \quad (3)$$

where α_l is a random number between -30 and 30 . In addition, an instance or a feature is added or deleted from the components of the tricluster at random times in order to represent a small evolution of the tricluster. The initial expression values for each instance or feature added at random times as a new tricluster component are random, and in the next time they will follow the same behaviour of the rest of the components of the tricluster. An example of a tricluster in the additive dataset is shown in Fig. 5.

- **Multiplicative dataset:** For the multiplicative dataset, each expression value in the tricluster at time p has been multiplied by a value $\sigma_p \in [0.1, 5.0]$ in order to create the multiplicative pattern over time. Thus, a tricluster tri at the stream time p in the multiplicative dataset follows the equation:

$$tri_{l,k,p} = \sigma_p \times tri_{l,k,p-1} \quad 1 \leq l \leq L \quad 1 \leq k \leq K \quad (4)$$

where the tricluster at the initial time, $tri_{l,k,0}$, is also generated with random expression values for each feature of the first instance and the expression values for all features of all the other instances according to the following equation:

$$tri_{l,k,0} = \beta_l \times tri_{1,k,0} \quad 2 \leq l \leq L \quad 1 \leq k \leq K \quad (5)$$

where β_l is a random number between 0.1 and 5.0 . The procedure of adding or deleting instances or features of the tricluster at random times follows the same behaviour explained in the generation of the additive dataset. Fig. 6 shows a graphical example of a tricluster in the multiplicative dataset.

- **Dynamic additive dataset:** The third dataset is a dynamic additive dataset, that is, it is generated with different additive datasets. In particular, it is made up of three different additive models: the first one from stream 0 to 399, the second from 400 to 1299 and the third from 1300 to 2100. The goal of this dataset is to test the performance of the proposed algorithm to abrupt changes.

4.1.2. Experimental setting

There are two main groups of parameters in the STriGen algorithm: typical parameters of the evolutionary algorithm in the offline phase and specific parameters of the STriGen. After a parameter tuning process and considering [6], the traditional triclustering parameters have been fixed. In particular, the number of solutions has been set to 3, the generations to 200, the members of the population to 400, the aleatory probability for the generation of the initial population to 0.2, the selection probability to 0.7 and the mutation probability to 0.4. To assess the influence of the specific parameters of the STriGen, five

Table 1
STriGen parameters for synthetic datasets.

Parameter	Run 1	Run 2	Run 3	Run 4	Run 5
<i>minGRQ</i>	0.85	0.975	0.975	0.95	0.975
<i>numIt</i>	50	25	25	35	15
<i>w</i>	15	15	10	15	15

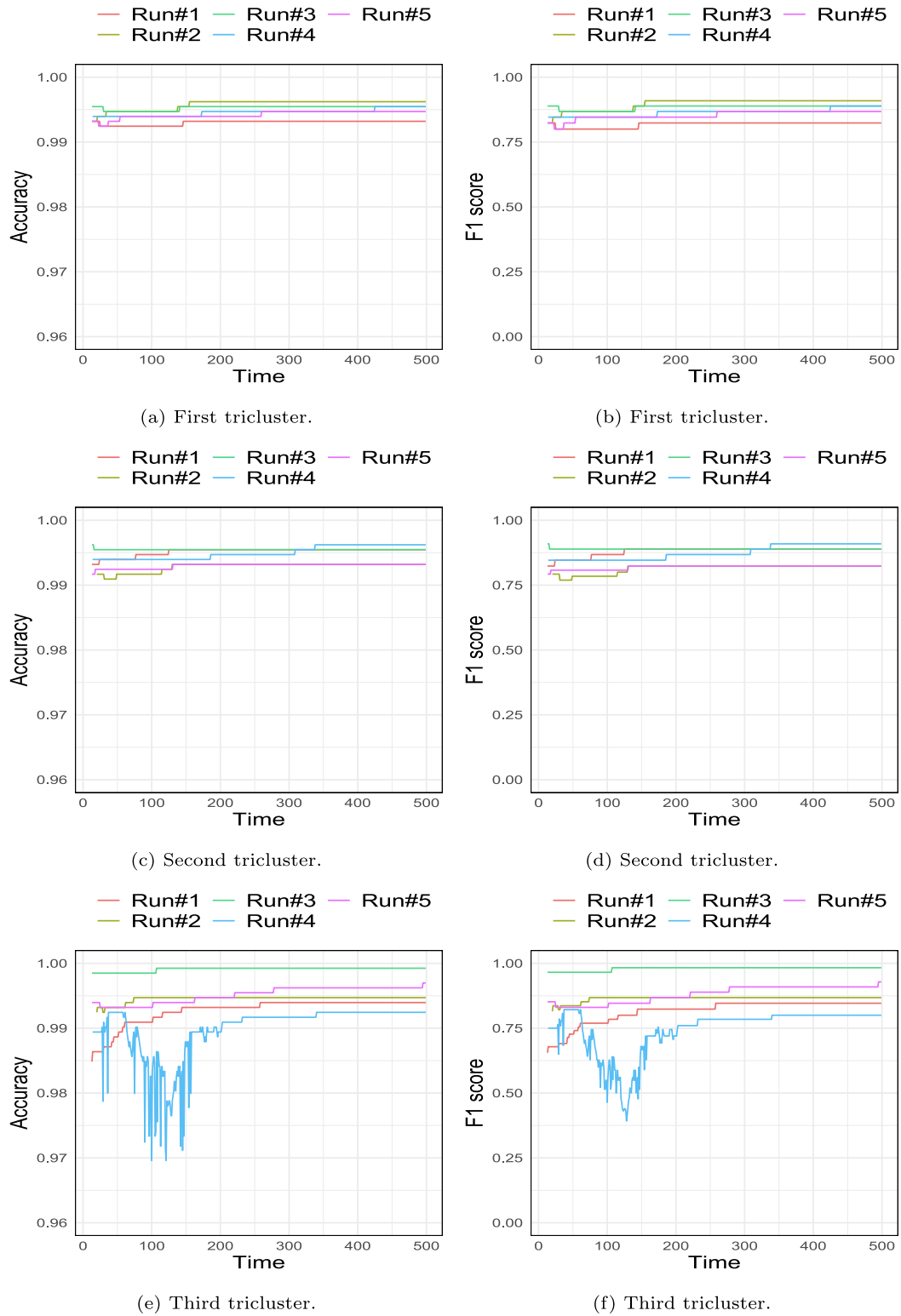


Fig. 7. Accuracy and F1 score of the tricluster solutions in the additive dataset.

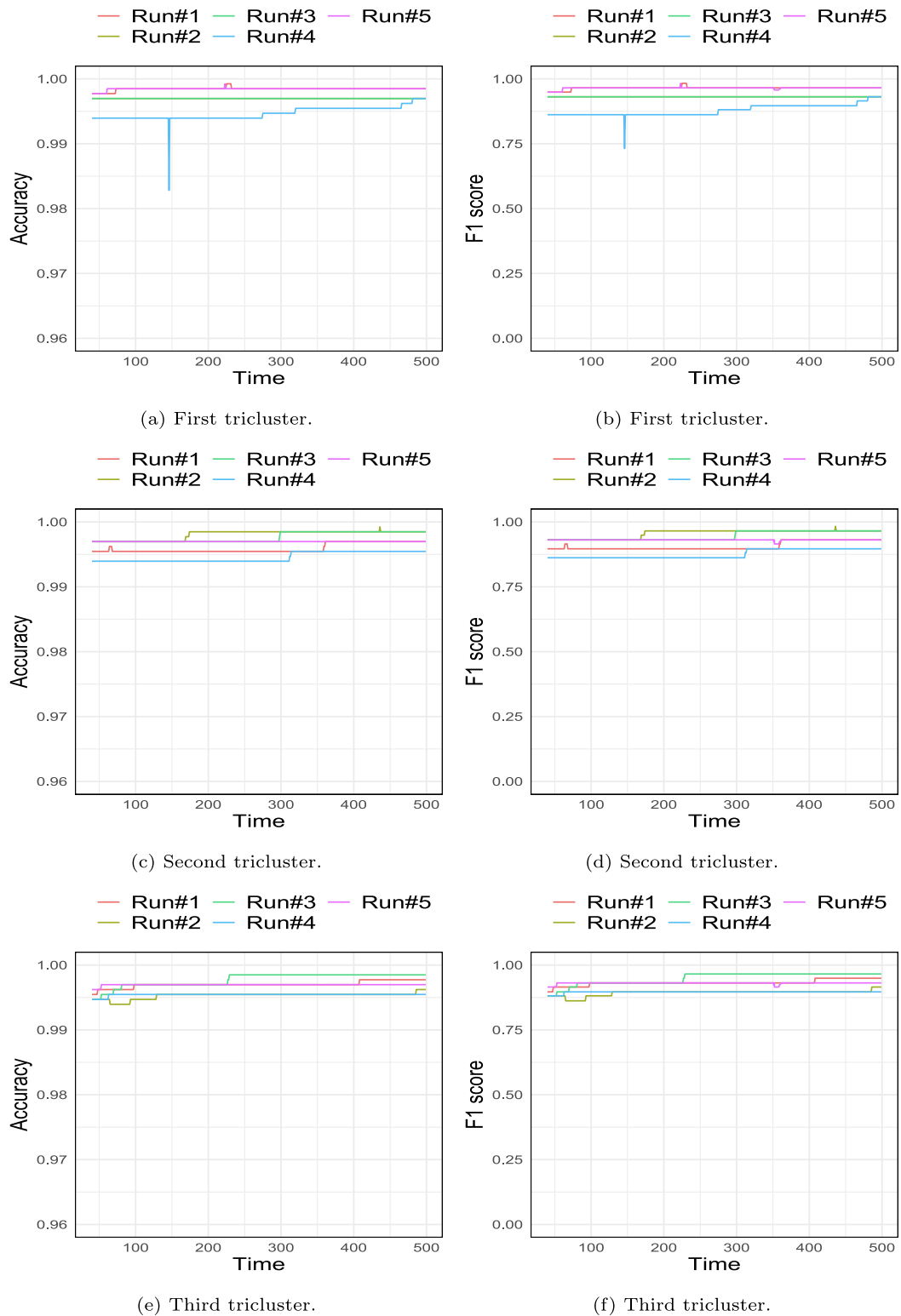


Fig. 8. Accuracy and F1 score of the tricluster solutions in the multiplicative dataset.

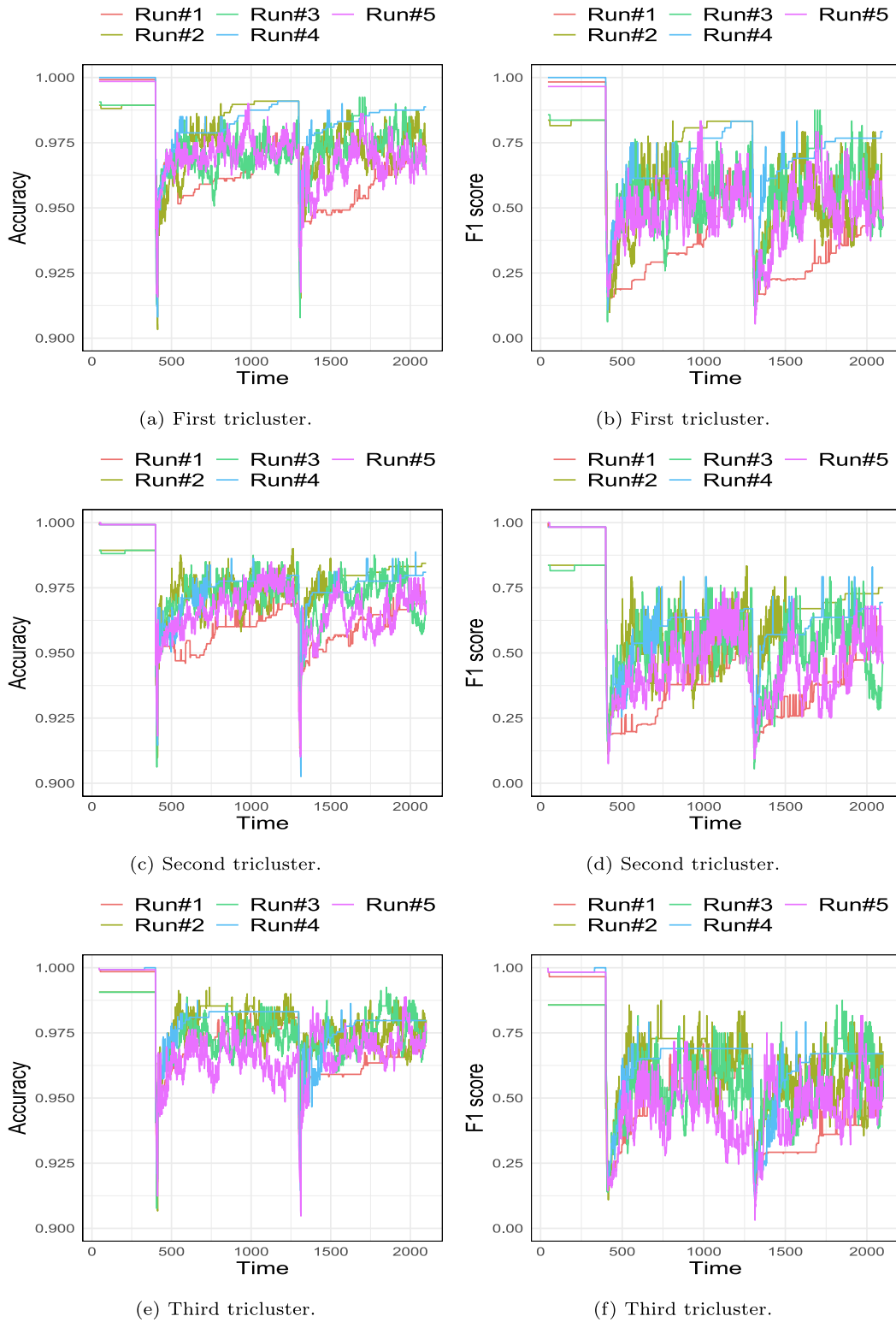


Fig. 9. Accuracy and F1 score of the tricluster solutions in the dynamic additive dataset.

experiments with different parameters values have been carried out. Experiments shall be referred as 'Run' from this point on-wards. Table 1 presents the parameters for each experiment for all three synthetic datasets.

4.1.3. Analysis of results

Figs. 7–9 present the evolution over time of the accuracy and F1 score of the three triclusters obtained by the STriGen algorithm for the additive, multiplicative and dynamic additive datasets, respectively.

- **Additive dataset:** In Fig. 7, it can be seen that both accuracy and F1 score improve over time with final metrics closer to 1 (the maximum value). The mean value of the accuracy for all fifteen solution triclusters (five “runs” for each of the three triclusters) is 0.995. The maximum value of the F1 score is 0.983 and is reached with the parameters of the third run for the third solution tricluster. The smallest F1 score is 0.8 and is reached for the third solution tricluster but with the configuration of the forth run. Considering that the whole dataset has 800 instances, the STriGen algorithm has good results and, furthermore, it is capable of adapting the solutions to the new streams of data arriving. All triclusters found in the additive dataset follow a similar pattern in the procedure of finding the triclusters, excluding the third tricluster found on the forth run, as shown in Fig. 7e and in Fig. 7f. This exception is due to the high *minGRQ* and high *numIt* fixed for the forth run and because the GRQ obtained in the offline phase is 0.945. Therefore, STriGen has to improve the quality of the third tricluster by introducing more mutations from the beginning and not trying to find the evolution of the values as in the other triclusters. At the end of the forth run, the accuracy is 0.992 and the F1 score is 0.8. Regarding the control parameters, the tricluster with the highest accuracy and F1 score is the third tricluster with the parameters of the third run that get a mean GRQ value of 0.976.
- **Dynamic dataset:** The performance of the algorithm is also accurate for the multiplicative dataset as shown in Fig. 8. The mean value of the accuracy for all fifteen solution triclusters is 0.997. The final F1 score values range between 0.896 and 0.965, which are quite high values. All triclusters follow a similar pattern and they find the optimal components over time. All three triclusters are found for all five runs without any incidence. It is important to consider that the spikes are due to the fact that the F1 score increases when finding one new instance or feature on one particular stream and not gradually. In this case, even if all experiments with different parameters provide similar results, the configuration with the highest accuracy and F1 score is also obtained for the setup of the third run for the third tricluster that reaches a mean GRQ value of 0.987.
- **Dynamic additive dataset:** For the dynamic additive dataset, the components of the triclusters change completely at some streams and the difficulty in finding these new components that differ totally from the previous ones is high as can be seen in Fig. 9. In this way, theoretically, with higher *minGRQ* and *numIt* the algorithm is forced to maintain a high quality level and when it is not reached (usually when abrupt changes appear), it mutates instances and attributes components repeatedly to find the new ones and maintain a good tricluster quality. In addition, in this type of dataset, if *w* is very high, the algorithm might not notice that an abrupt change occurs until there are more streams of the new pattern than of the previous one. For the experiments of the setup of the third run with a low *w*, the GRQ value decreases earlier when an abrupt change occurs. The mean value of the accuracy for all the triclusters is 0.974. The evolution pattern of the values is similar each time a new abrupt change happens. When the change occurs, the GRQ, accuracy and F1 score decrease dramatically. Mutations allow to find the new members of the triclusters and so GRQ, accuracy and F1 score increase continuously. The best accuracy and F1 score is 0.9887 and 0.793, respectively, both them are reached for the first solution tricluster for the parameter configuration setup of the fourth run.

Summarizing, despite abrupt changes in the components of the triclusters, the STriGen algorithm performs correctly and obtains good results in terms of adaptation to concept drift, finding the majority of the new components even if they are totally different. Thus, it can be concluded that the STriGen has a huge potential as it is clearly able to detect patterns in data streams even if they change abruptly along time.

4.1.4. Comparison with TriGen benchmarking algorithm

In this Section, the common points and differences between the TriGen and STriGen algorithms are firstly presented. Next, a comparison of the quality of the triclusters by means of F1 and accuracy along with the computational time is carried out for the triclusters found by both algorithms.

Table 2
Similarities and differences between TriGen and STriGen.

Common features	STriGen new features
<ul style="list-style-type: none"> • Evolutionary process. • Genetic operators. • Genetic parameters. 	<ul style="list-style-type: none"> • Consecutive instant points. • Mutations keeps the consecutive instant points. • Streaming execution. • Streaming control parameters. • Streaming input dataset.

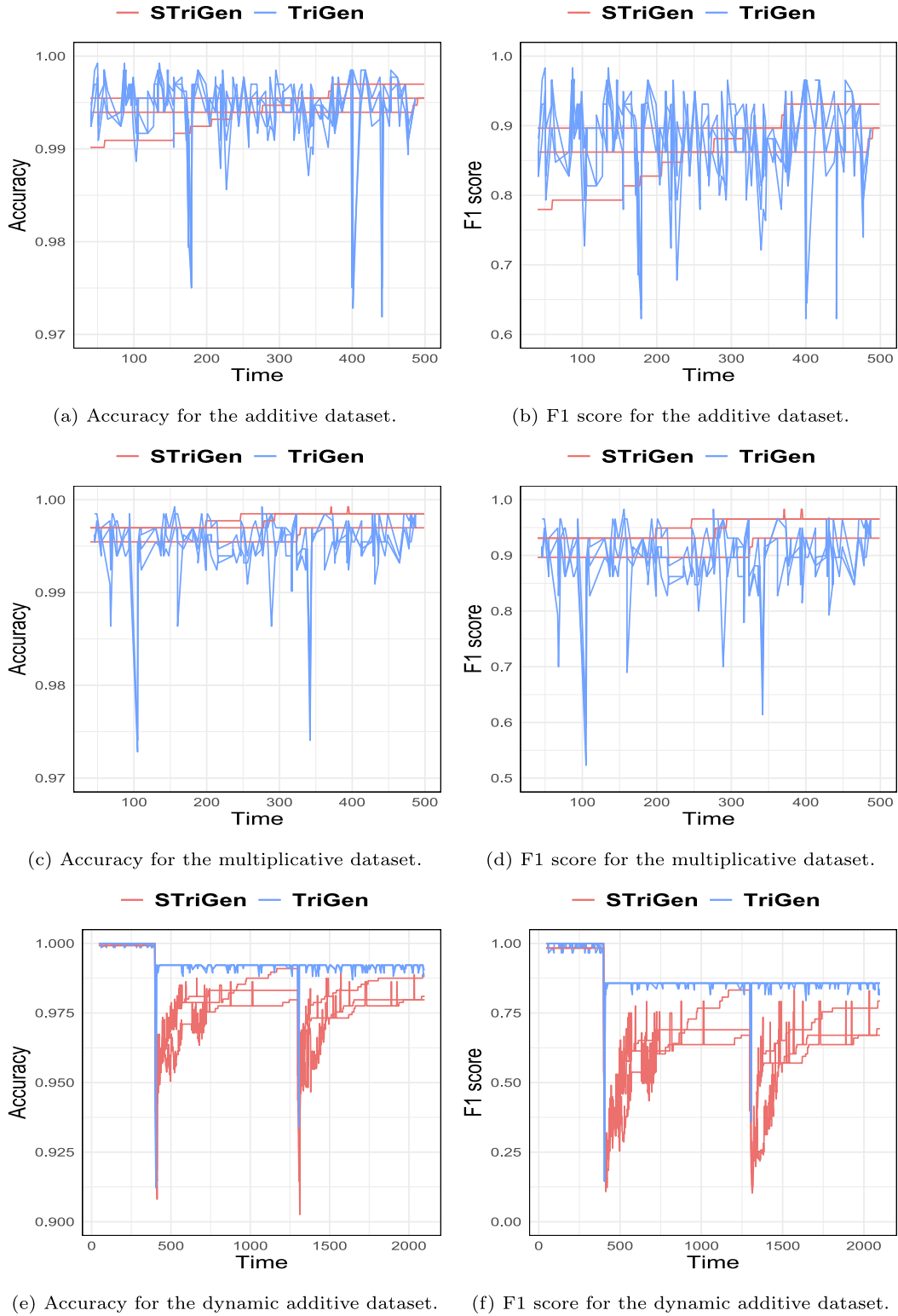


Fig. 10. Comparison of the performance for STriGen and TriGen algorithms.

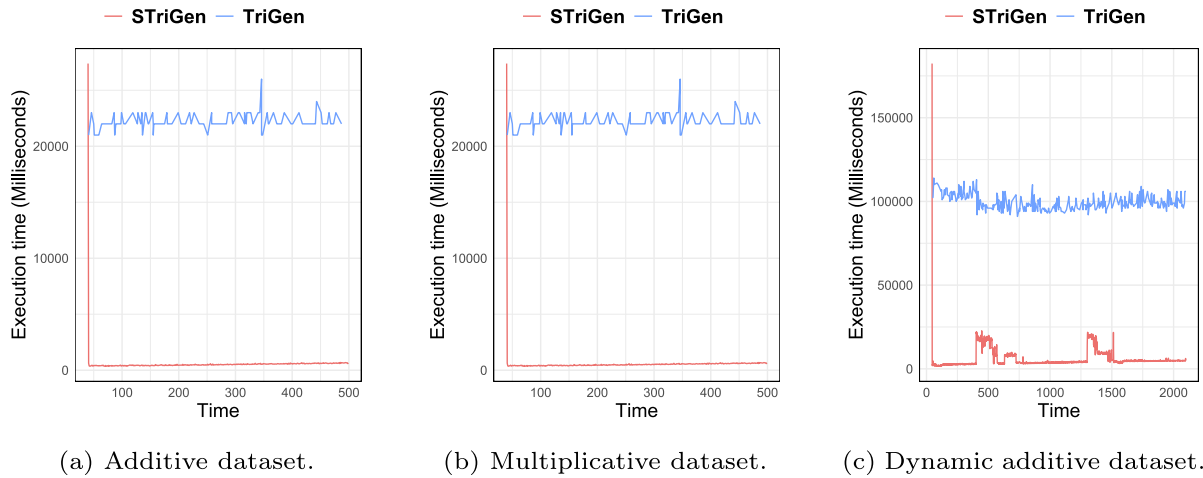


Fig. 11. Comparison of the execution times for STriGen and TriGen algorithms.

A summary of the common characteristics between TriGen and STriGen and the new ones included by STriGen is shown in Table 2.

The common characteristics between both algorithms are mainly reduced to the offline phase of STriGen. TriGen and STriGen implement a triclustering algorithm based on an evolutionary process, which minimizes a fitness function to obtain the triclusters. Thus, both approaches share the fitness function to be minimized, and use the same genetic operators and typical control parameters of an evolutionary process in the phase offline of STriGen.

As for the new features of STriGen, there are two main novelties. Firstly, about the instant points of a tricluster, STriGen considers them as a complete and well-formed time series. That is, as a consecutive sequence of time points in opposition to the TriGen algorithm, where this feature is not taken into account. Furthermore, the mutation operator of STriGen respects the consecutive instant point feature of the tricluster when they are altered. This new characteristic is a key-point in the new STriGen algorithm in order to be adapted to the streaming environment.

Finally, the second novelty is the capability of the STriGen algorithm to analyze streaming data. STriGen is run in a streaming environment where it analyses an in-motion input dataset, increasing whenever the time points move forward. Therefore, STriGen can find the evolution of patterns during the arrival of data streams and adapt the model accordingly, in opposition to TriGen, where both the input data and its models are static. In addition, the streaming feature of STriGen implies the inclusion of new control parameters in its online phase.

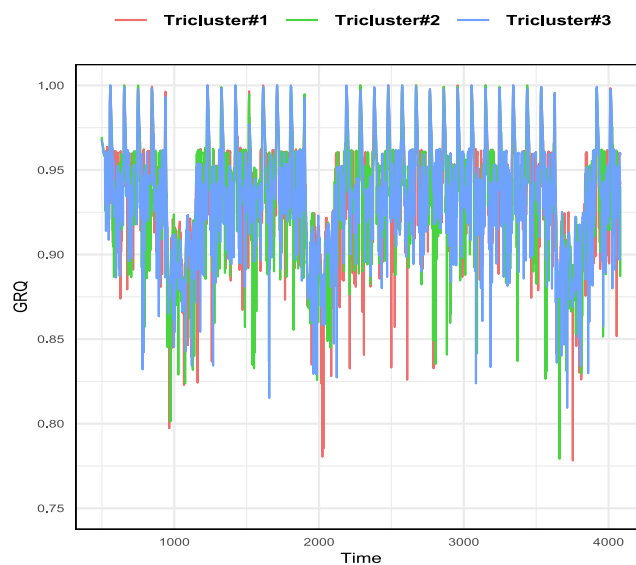


Fig. 12. GRQ values using STriGen for the real sensor dataset.

Next, the results obtained by STriGen have been compared to the TriGen triclustering algorithm [6]. The best STriGen execution for each synthetic dataset between all five “run” is selected to carry out the comparison. The TriGen algorithm is re-run each time a new data stream arrives. At each execution, TriGen considers just the previous w data samples. Fig. 10 shows the mean of the accuracy and F1 score for the three triclusters obtained by the STriGen and TriGen algorithms. It can be observed that the STriGen takes advantage of the sequential behaviour and finds the corresponding components of the triclusters for the additive and multiplicative datasets easily as shown in Fig. 10a, 10b, 10c and 10d. However, as TriGen does not consider the evolution over time of the triclusters components, it does not find the corresponding components as accurately as the STriGen streaming algorithm. In the case of the dynamic additive dataset when abrupt changes occurs, the STriGen algorithm takes some time to find the new components of the triclusters in contrast to TriGen as shown in Fig. 10e and 10f. The main reason is that TriGen considers just the previous w samples and not the evolution over time of the triclusters’ components as STriGen does.

One of the critical aspects in streaming models is the bounded response time. Fig. 11 presents the runtimes of the STriGen and TriGen algorithms. The offline phase of STriGen is quite time consuming (first value of the graphic) and the next executions consume almost imperceptible time compared to TriGen execution times, since TriGen has to be recomputed each time a new data stream arrives.

4.2. Real sensor dataset

4.2.1. Description of the real dataset

The real-world dataset contains data of seven sensors that record environmental data such as atmospheric pressure, precipitation, relative humidity, solar radiation, temperature, wind direction and wind speed. These sensors are placed in 12 different areas of Malaga (Spain). Some ones record data in just one location. However, other ones register data from different locations in its territory, for example Malaga capital city that records in 10 different locations.

4.2.2. Experimental setting

The offline phase is computed with the first 500 streams. Afterwards, each time a new stream arrives, the STriGen algorithm updates in quasi real time the triclusters and provides results. The experiment is carried out for 5000 streams, where w is fixed to 20, $minGRQ$ to 0.975 and $numIt$ to 35 after a process of tuning of the parameters in order to use the most accurate values.

4.3. Analysis of the results

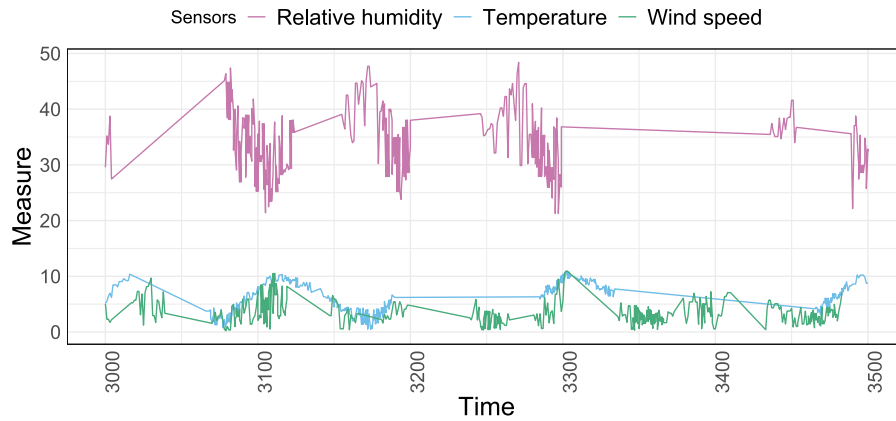
Fig. 12 shows the evolution over time of the values of GRQ for the STriGen algorithm. The average GRQ value for the three triclusters is 0.9468. These GRQ values are between 0.78 and 0.99, so STriGen exhibits a notable improvement.

However, as it is a real dataset and the ground truth is not known, the GRQ quality measure is not sufficient. In order to improve it, a baseline algorithm is used to compare results. The baseline algorithm is a simple triclustering in streaming. In particular, the STriGen algorithm is modified in such a way that the algorithm do not select the best tricluster, i.e. with the best GRQ, as defined in Algorithm 2, but a random tricluster between all the possibilities. This random assignment of the triclusters is the baseline algorithm used to compare the results. The mean and standard deviation of the GRQ values for each tricluster found by the STriGen algorithm and the baseline algorithm are presented in Table 3. The results of the STriGen algorithm are higher than the ones of the baseline. Thus, STriGen is adapting correctly to the evolution of the data streams of the sensors dataset.

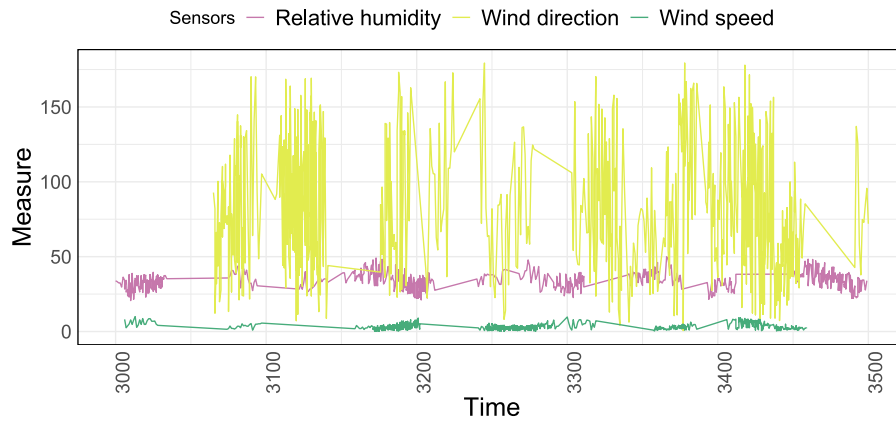
Fig. 13 represents the three triclusters obtained by the STriGen algorithm from time 3000 to 3500 in order to show the usefulness of the patterns found. Note that patterns evolve over time as streaming data arrives, for example at a time z , the components of a tricluster can be totally different from the ones at time $z + 50$. For this reason, this figure represents the evolution of the three triclusters only from time 3000 to 3500 as it is complex to visualize all patterns for each of the 4500 data streams of the dataset. The first tricluster in Fig. 13a corresponds to interior areas, in particular the areas of Antequera and Ronda, and they mainly include components of the relative humidity, temperature and wind speed. The second tricluster in Fig. 13b is made up of the nearest areas to the sea of the capital city of Málaga and they contain mainly components of the relative humidity, wind direction and wind speed sensors. The third tricluster in Fig. 13c contains west coast areas such as the towns of Estepona and Fuengirola, and include mainly instances of the solar radiation, temperature and wind speed sensors. This is just a particular sample of found patterns. In this case, there are three clearly different areas (in-

Table 3
STriGen and baseline comparison.

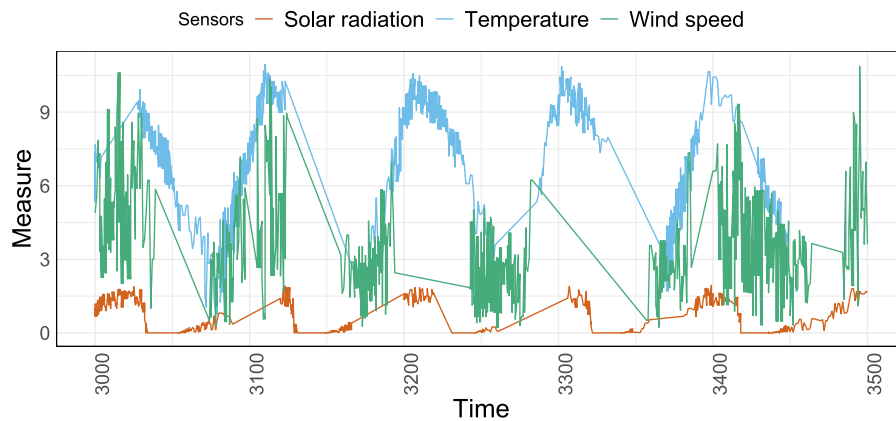
Algorithm	Tricluster 1		Tricluster 2		Tricluster 3	
	Mean	Std	Mean	Std	Mean	Std
STriGen	0.9398	0.0294	0.9390	0.0289	0.9614	0.0299
Baseline	0.7653	0.0314	0.7682	0.0302	0.7665	0.3533



(a) Tricluster#1.



(b) Tricluster#2.



(c) Tricluster#3.

Fig. 13. Triclusters patterns for the real sensor dataset from time 3000 to 3500.

terior, near to the sea in the east-side and near to the sea in the west-side) with different sensors instances. The precipitation and atmospheric pressure sensors are not present in these patterns as the precipitation is zero in all areas and the pressure is very similar in these time intervals. In the same way, Fig. 13 only includes the three more significant sensors for each tricluster, as the other sensors are only present in these triclusters occasionally and are not relevant for the meaning of the patterns.

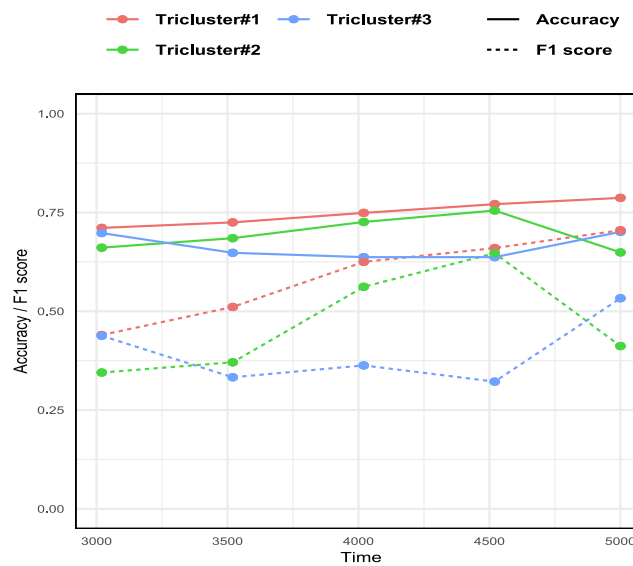


Fig. 14. Comparison of results using TriGen and STriGen for the real sensor dataset.

In addition, the TriGen algorithm has been executed in order to consider its triclusters as the “real” ones and the STriGen triclusters as the “found” ones. Besides the comparison with a baseline algorithm, this is an additional option to deal with real datasets where the ground truth is unknown. This test has been carried out 5 times, in particular for streams from 3000 to 3020, 3500 to 3520, 4000 to 4020, 4500 to 4520 and 4980 to 5000. The TriGen algorithm is executed just in the w last streams, namely 20 streams, and therefore, in this particular case the evolution over the different streams is not considered. Fig. 14 presents the accuracy and F1 score obtained when comparing the TriGen and STriGen results using the real sensor dataset. The highest accuracy is 0.787 and the highest F1 score is 0.705 for the STriGen algorithm. Even if TriGen is not executed in all streams, the F1 score, accuracy and GRQ results of the STriGen ensures its good performance. In addition, the STriGen algorithm keeps improving its results over time.

5. Conclusions

This work has introduced a new triclustering algorithm in the streaming environment. The algorithm consists of two stages: an offline or batch phase and an online phase. The first one creates a sketch or summary model with the triclusters components that optimize the fitness function. This fitness function considers the similarity between the angles of the slopes that represent the values of the tricluster components. In addition, the GRQ quality measure is computed to evaluate the evolution over time of the triclusters. Then, considering the offline summary model, the online phase of STriGen is computed satisfying all the requirements of data streaming, i.e., each sample is processed just once and in the order of its arrival and the model stores a limited amount of data and updates the model with a low computational cost. The STriGen has been proved to be able to discover collections of resembling patterns in 3D stream data. The algorithm has been applied to three synthetic datasets with different characteristics and to one real dataset of environmental sensors. Results have shown that the algorithm detects both little and huge changes of instances and features in triclusters components. The validation of the experiments is carried out comparing the found triclusters to the real triclusters in the case of the synthetic datasets and to the solution found by the TriGen batch triclustering algorithm published in the literature in the case of the real sensor dataset. For both type of datasets, the quality of the triclusters found is similar to the quality of the triclusters obtained by the TriGen with much less execution time. Up to our knowledge, there are not many triclustering streaming algorithms in the literature, so this field of research is noteworthy due to vast amount of streaming data sources available nowadays. Another advantage of the proposed algorithm is the good performance in terms of accuracy and execution times provided for datasets of different types and volumes.

The future work will be focused on addressing when a new stream concept is an outlier pattern and alert about that. In addition, we plan to add different fitness functions and evaluation measures to extend the approach of the STriGen proposed algorithm.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank the Spanish Ministry of Science, Innovation and Universities for the support under the project TIN2017-88209-C2.

References

- [1] P. Larrañaga, D. Atienza, J.D. Roza, A. Ogbechie, C. Puerto-Santana, C. Bielza, *Industrial Applications of Machine Learning*, CRC Press, 2018.
- [2] H. Wang, A. Zubin, Concept drift detection for streaming data, in: *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2015, pp. 1–9.
- [3] J. Gama, A survey on learning from data streams: current and future trends, *Progr. Artif. Intell.* 1 (1) (2012) 45–55.
- [4] C. Rubio-Escudero, F. Martínez-Álvarez, R. Romero-Zaliz, I. Zwir, Classification of gene expression profiles: comparison of k-means and expectation maximization algorithms, in: *Proceedings of the 8th International Conference on Hybrid Intelligent Systems*, 2008, pp. 831–836.
- [5] J.A. Hartigan, Direct clustering of a data matrix, *J. Am. Stat. Assoc.* 67 (337) (1972) 123–129.
- [6] D. Gutiérrez-Avilés, C. Rubio-Escudero, F. Martínez-Álvarez, J. Riquelme, TriGen: a genetic algorithm to mine triclusters in temporal gene expression data, *Neurocomputing* 132 (2014) 42–53.
- [7] D. Gutiérrez-Avilés, C. Rubio-Escudero, MSL: a measure to evaluate three-dimensional patterns in gene expression data, *Evolut. Bioinform.* 11 (2015) 121–135.
- [8] B. Zhou, J. Li, X. Wang, Y. Gu, L. Xu, Y. Hu, L. Zhu, Online Internet traffic monitoring system using spark streaming, *Big Data Mining Anal.* 1 (1) (2018) 47–56.
- [9] L.P. Liu, Y. Jiang, Z.H. Zhou, Least square incremental linear discriminant analysis, in: *Proceedings of the IEEE International Conference on Data Mining*, 2009, pp. 298–306.
- [10] C. Za'in, M. Pratama, E. Pardede, Evolving large-scale data stream analytics based on scalable PANFIS, *Knowl.-Based Syst.* 166 (2019) 186–197.
- [11] B. Krawczyk, Active and adaptive ensemble learning for online activity recognition from data streams, *Knowl.-Based Syst.* 138 (2017) 69–78.
- [12] H. He, S. Chen, K. Li, X. Xu, Incremental learning from stream data, *IEEE Trans. Neural Networks* 22 (12) (2011) 1901–1914.
- [13] A. Bifet, G.F. Morales, Big data stream learning with SAMOA, in: *Proceedings of the IEEE International Conference on Data Mining Workshop*, 2015, pp. 1199–1202.
- [14] S.C. Pallaprolu, R. Sankineni, M. Thevar, G. Karabatis, J. Wang, Zero-day attack identification in streaming data using semantics and Spark, in: *Proceedings of the IEEE International Congress on Big Data*, 2017, pp. 121–128.
- [15] U. Rajeshwari, B.S. Babu, Real-time credit card fraud detection using streaming analytics, in: *Proceedings of the 2nd International Conference on Applied and Theoretical Computing and Communication Technology*, 2016, pp. 439–444.
- [16] S. Papadimitriou, J. Sun, C. Faloutsos, Streaming pattern discovery in multiple time-series, in: *Proceedings of 31st International Conference on Very Large Data Bases*, vol. 2, 2005, pp. 697–708.
- [17] K. Yu, W. Ding, D.A. Simovici, H. Wang, J. Pei, X. Wu, Classification with streaming features: an emerging-pattern mining approach, *ACM Trans. Knowl. Discovery Data* 9 (4) (2015) 1–31.
- [18] Y. Chen, K. Chen, M.A. Nascimento, Effective and efficient shape-based pattern detection over streaming time series, *IEEE Trans. Knowl. Data Eng.* 24 (2) (2012) 265–278.
- [19] C. García, A. Esmin, D. Leite, I. Škrjanc, Evolvable fuzzy systems from data streams with missing values: with application to temporal pattern recognition and cryptocurrency prediction, *Pattern Recogn. Lett.* 128 (2019) 278–282.
- [20] I. Škrjanc, J.A. Iglesias, A. Sanchis, D. Leite, E. Lughofer, F. Gomide, Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey, *Inf. Sci.* 490 (2019) 344–368.
- [21] E. Lughofer, C. Cernuda, S. Kindermann, M. Pratama, Generalized smart evolving fuzzy systems, *Evolving Syst.* 6 (4) (2015) 269–292.
- [22] D. Leite, G. Andonovski, I. Škrjanc, F. Gomide, Optimal rule-based granular systems from data streams, *IEEE Trans. Fuzzy Syst.* 28 (3) (2020) 583–596.
- [23] I. Škrjanc, Cluster-volume-based merging approach for incrementally evolving fuzzy gaussian clustering—egauss+, *IEEE Trans. Fuzzy Syst.* 28 (9) (2020) 2222–2231.
- [24] D. Leite, I. Škrjanc, F. Gomide, An overview on evolving systems and learning from stream data, *Evolving Syst.* 11 (2) (2020) 181–198.
- [25] M. Ackerman, S. Dasgupta, Incremental Clustering: The Case for Extra Clusters, in: *Proceedings of the Neural Information Processing Systems*, 2014, pp. 1–13.
- [26] M. Capó, A. Pérez, J.A. Lozano, An efficient approximation to the K-means clustering for massive data, *Knowl.-Based Syst.* 117 (2017) 56–69.
- [27] U. Kokate, A. Deshpande, P. Mahalle, P. Patil, Data stream clustering techniques, applications, and models: comparative analysis and discussion, *Big Data Cognit. Comput.* 2 (4) (2018) 32.
- [28] R. Henriques, S. Madeira, Triclustering algorithms for three-dimensional data analysis: a comprehensive survey, *ACM Comput. Surv.* 51 (5) (2018) 1–43.
- [29] L. Zhao, M.J. Zaki, triCluster: an effective algorithm for mining coherent clusters in 3D microarray data, in: *Proceedings of the ACM SIGMOD International Conference on Management of data*, 2005, pp. 694–705.
- [30] H. Jiang, S. Zhou, J. Guan, Y. Zheng, gTRICLUSTER: A More General and Effective 3D Clustering Algorithm for Gene-Sample-Time Microarray Data, in: *Proceedings of the Data Mining for Biomedical Applications*, 2006, pp. 48–59.
- [31] J. Liu, Z. Li, X. Hu, Y. Chen, Multi-objective evolutionary algorithm for mining 3D clusters in gene-sample-time microarray data, in: *Proceedings of the IEEE International Conference on Granular Computing*, 2008, pp. 442–447.
- [32] N. Narmadha, R. Rathipriya, Evolutionary correlation triclustering for 3d gene expression data, in: *Innovative Data Communication Technologies and Application*, Springer International Publishing, 2020, pp. 637–646.
- [33] A. Tchagang, S. Phan, F. Famili, H. Shearer, P. Fobert, Y. Huang, J. Zou, D. Huang, A. Cutler, Z. Liu, Y. Pan, Mining biological information from 3D short time-series gene expression data: the oprcluster algorithm, *BMC Bioinform.* 13 (2012) 54.
- [34] F. Martínez-Álvarez, D. Gutiérrez-Avilés, A. Morales-Esteban, J. Reyes, J. Amaro-Mellado, C. Rubio-Escudero, A novel method for seismogenic zoning based on triclustering: application to the iberian peninsula, *Entropy* 17 (12) (2015) 5000–5021.
- [35] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, A. Troncoso, High-content screening images streaming analysis using the strigen methodology, in: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 537–539.
- [36] D. Soares, R. Henriques, M. Gromicho, S. Pinto, M. de Carvalho, S.C. Madeira, Towards triclustering-based classification of three-way clinical data: A case study on predicting non-invasive ventilation in als, in: *Practical Applications of Computational Biology & Bioinformatics*, 14th International Conference (PACBB 2020), Springer International Publishing, 2021, pp. 112–122.
- [37] P. Mahanta, H.A. Ahmed, D.K. Bhattacharyya, J.K. Kalita, Triclustering in gene expression data analysis: a selected survey, in: *2011 2nd National Conference on Emerging Trends and Applications in Computer Science*, 2011, pp. 1–6.
- [38] N. Narmadha, R. Rathipriya, Triclustering: an evolution of clustering, in: *Proceedings of the Online International Conference on Green Engineering and Technologies*, 2016, pp. 1–4.
- [39] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams: theory and practice, *IEEE Trans. Knowl. Data Eng.* 15 (3) (2003) 515–528.
- [40] D.A. Umbarkar, P. Sheth, Crossover operators in genetic algorithms: a review, *ICTACT J. Soft Comput.* 6 (1) (2015) 1083–1092.

- [41] M. Ghesmoune, M. Lebbah, H. Azzag, State-of-the-art on clustering data streams, *Big Data Anal.* 1 (1) (2016) 1–27.
- [42] M.S. Hammoodi, F. Stahl, A. Badii, Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining, *Knowl.-Based Syst.* 161 (2018) 205–239.
- [43] R.H. Moulton, H.L. Viktor, N. Japkowicz, J. Gama, Clustering in the presence of concept drift, in: *Proceedings of the ECML/PKDD Machine Learning and Knowledge Discovery in Databases*, 2018, pp. 339–355.
- [44] B. Pontes, R. Giráldez, J.S. Aguilar-Ruiz, Quality measures for gene expression biclusters, *PLOS ONE* 10 (3) (2015) 1–24.
- [45] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, On biclustering of gene expression data, *Curr. Bioinform.* 5 (2010) 204–216.

5.1.6 | "Generating a seismogenic source zone model for the Pyrenees: a GIS-assisted triclustering approach"

Authors: Amaro-Mellado J. L., Melgar-García L., Rubio-Escudero C., Gutiérrez-Avilés D.

Publication type: Journal article.

Journal: Computers and Geosciences.

Year: 2021.

Volume: 150

Pages: 104736

DOI: 10.1016/j.cageo.2021.104736

IF: 2.991 42/109 Computer Science, Interdisciplinary Applications.

Quartil: Q2.



Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Generating a seismogenic source zone model for the Pyrenees: A GIS-assisted triclustering approach

José L. Amaro-Mellado^{a,b}, Laura Melgar-García^c, Cristina Rubio-Escudero^d, David Gutiérrez-Avilés^{c,*}^a Department of Graphic Engineering, University of Seville, ES-41092, Seville, Spain^b National Geographic Institute of Spain, Andalusia Division, ES-41013 Seville, Spain^c Data Science & Big Data Lab, Pablo de Olavide University, ES-41013 Seville, Spain^d Department of Computer Languages and Systems, University of Seville, ES-41012, Seville, Spain

ARTICLE INFO

Keywords:
Seismic sources
GIS
Data science
Triclustering

ABSTRACT

Seismogenic source zone models, including the delineation and the characterization, still have a role to play in seismic hazard calculations, particularly in regions with moderate or low to moderate seismicity. Seismic source zones establish areas with common tectonic and seismic characteristics, described by a unique magnitude–frequency distribution. Their definition can be addressed from different views. Traditionally, the source zones have been geographically outlined from seismotectonic, geological structures, and earthquake catalogs. Geographic information systems (GIS) can be of great help in their definition, as they deal rigorously and less ambiguously with the available geographical data. Moreover, novel computer science approaches are now being employed in their definition. The Pyrenees mountain range – in southwest Europe – is located in a region characterized by low to moderate seismicity. In this study, a method based purely on seismic catalogs, managed with a GIS and a triclustering algorithm, were used to delineate seismogenic zones in the Pyrenees. Based on an updated, reviewed, declustered, extensive, and homogeneous earthquake catalog (including detailed information about each event such as date and time, hypocentral location, and size), a triclustering algorithm has been applied to generate the seismogenic zones. The method seeks seismicity patterns in a quasi-objective manner following an initial assessment as to the best suited seismic parameters. The eight zones identified as part of this study are represented on maps to be analyzed, being the zone covered by the Arudy–Arette region to Bagnères de Bigorre as the one with the highest seismic hazard potential.

1. Introduction

Seismogenic source zones are necessary for applying the most widely used seismic hazard calculation method proposed by Cornell (1968) and McGuire (1976). Originally, this method contemplated the zones as an alternative to model the probabilistic space of distance-to-source relationships with the area under analysis, particularly when associations between seismicity and faults capable of triggering an earthquake are not apparent (García-Mayordomo, 2015). In addition, the method conceived a fault as a linear element, but it has evolved to the use of fault as zones (even as 3D planes). This second option is much more widely used today. Currently, there is a tendency to make greater use of faults, but through a characterization collected in a database. These records include geological and seismic data, such as the slip rate, the fault length, the depth, recurrence periods, or maximum expected magnitude (García-Mayordomo et al., 2012a).

The main assumptions of the Cornell-McGuire method regarding the occurrence of seismicity are the following (García-Mayordomo, 2015):

1. Earthquakes are independent random events (Poissonian).
2. The probability of earthquakes occurrence is the same throughout the entire area.
3. The size of earthquakes is related to their frequency through the Gutenberg–Richter Law (Gutenberg and Richter, 1944) of each source zone, usually limited to a maximum magnitude value.
4. The seismic activity rate of each zone is constant over time.

These cannot be strictly fulfilled in practice since the mechanism of accumulation and release of energy in earthquakes is a memory process with long-range dependence (Barani et al., 2018), and the rate of seismic activity is variable over time. However, in seismic engineering,

* Corresponding author.

E-mail addresses: jamaro@us.es (J.L. Amaro-Mellado), lmelgar@upo.es (L. Melgar-García), crubioescudero@us.es (C. Rubio-Escudero), dgutavi@upo.es (D. Gutiérrez-Avilés).<https://doi.org/10.1016/j.cageo.2021.104736>

Received 26 August 2020; Received in revised form 5 February 2021; Accepted 18 February 2021

Available online 24 February 2021

0098-3004/© 2021 Elsevier Ltd. All rights reserved.

seismogenic source zones are sufficiently adequate to estimate the probability according to different levels of ground shaking (García-Mayordomo, 2015). In this case, a memory process refers to the fact that shocks do not happen independently at any time. By contrast, they remember the occurrence (time and magnitude) of the last earthquake (Corral, 2006). The main strength of using zones is that they are efficient because their associated parameters are somehow easily obtained, and their calculations are fast. Thus, they have been used for over 50 years so far.

Seismic source delimitation is a crucial task when a Probabilistic Seismic Hazard Analysis (hereinafter, PSHA) is conducted (Morales-Esteban et al., 2014). Once they have been geometrically outlined, their activity is characterized based on the recorded seismicity. Both stages define the seismic potential (geometric shape and size, maximum expected magnitude, seismic activity, and frequency distribution in size). Subsequently, the equations for predicting strong movements should be considered, and, finally, the annual probability of exceeding a certain magnitude earthquake should be obtained.

Despite the importance of the seismogenic zonation to conduct a seismic hazard analysis, the criteria used for its definition are often not uniform. It is a very complex process and depends strongly on the analyst group's criteria (Amaro-Mellado et al., 2017). There are two main groups within these criteria: one based exclusively on the distribution of seismicity and the other on geological domains or structures. The criterion, or its weight, depends on the background of the analyst who defines the zones. On the one hand, much weight is placed on the characterization of sources, it will mainly be based on seismicity. This can lead to inconsistent limits from a geological point of view but valid for the calculation of hazards. On the other hand, if the analyst relies on the location of the geological structure, two opposing situations arise. In the first, areas are based exclusively on the limits of large geological units, even if they have no relation to current tectonic activity. In the second, they only consider areas with active faults, even though the recorded seismicity does not allow for statistically significant characterization.

Computational robust methods are becoming increasingly popular, as they can be applied in different geographical and geological contexts (Morales-Esteban et al., 2014; Martínez-Álvarez et al., 2015; Scitovski, 2018). Besides, when handling and representing georeferenced spatial information such as the distribution of epicenters, the use of a geographic information system (hereinafter, GIS) is powerful. Its ability to manage, represent, and generate geographical information is a useful tool for integrating different geographic data kinds.

This work aims at proposing a seismic source model for the Pyrenees range in southwest Europe. To this end, the use of a GIS and a tricluster algorithm, called TriGen (Gutiérrez-Avilés et al., 2014), has been applied to derive seismicity patterns from a seismic catalog. Later, the source models have been drawn. Finally, seismic parameters have been obtained from these zones, and they have been analyzed. The range of seismicity in the Pyrenees is estimated as low to moderate and diffuse, where earthquakes with a magnitude larger than or equal to 5.0 (M5+) are infrequent. Besides, in these areas, obtaining a correlation between epicenters and faults is difficult (Drouet et al., 2020). Therefore, seismic zonings usage to drive a PSHA is adequate (Martínez-Álvarez et al., 2015; Amaro-Mellado and Tien Bui, 2020).

2. Related work

In this section, research made in both domains presented in this study is described: firstly, the studies in seismogenic zonings and, secondly, the triclustering approach.

2.1. Previous seismic zonations for the Pyrenees

The Pyrenees are located in a border-area between France, Spain, and Andorra, so each of these countries have studied their seismicity. They are also included in some European seismicity projects as SHARE (Woessner et al., 2011; Stucchi et al., 2013), or SIGMA (Pecker et al., 2017).

In the Spanish seismic sources, the Pyrenees are an important region due to their activity, and they are included in different seismic zonations. For example, that conducted by Martín (1984), who established 27 zones for the whole Iberian Peninsula and adjacent area; Mezcua et al. (2011) addressed a PSHA as a combination from the one defined by Molina (1998) and the CODE one (from NCSE-2002 Ministerio de Fomento (Gobierno de España), 2002, which is very similar to Martín, 1984). More recently, in the frame of the Updated Map of PSHA for Spain, both García-Mayordomo et al. (2012b) and Bernal (2011) defined a set of seismic sources. Later, Morales-Esteban et al. (2014) Morales-Esteban et al. undertook a zonation for the whole Peninsula from Mahalanobis distances, based only on an earthquake catalog (including information on the shock such as date and time, hypocentral location, or size) from 1978 to 2012. In Martínez-Álvarez et al. (2015), Martínez-Álvarez et al. used the triclustering methodology to obtain seismic sources for the whole Iberian Peninsula.

French institutions have also defined different seismogenic zonations. In Metropolitan France, the first PSHA map was deployed in 2002 (Martin et al., 2002) and gave the foundations for the French zoning (Drouet et al., 2020). Moreover, in 2004, Marin depicted PSHA maps for France (Marin et al., 2004). Later, Baize et al. (2013) proposed a new seismotectonic zoning for Metropolitan France based on geological, seismological, and tectonic data. Recently, Drouet et al. (2020) constructed a seismotectonic model (GEOTER) for Metropolitan France, considering geological, structural, neotectonic, geophysical, and seismological data. Besides, some seismic zonations have been conducted specifically for the Pyrenees. In Njike-Kassala et al. (1992), the authors established nine zones to calculate the *b*-value of the Gutenberg and Richter relation (Gutenberg and Richter, 1944). Later, Secanell et al. (2008) defined a seismotectonic zonation as a result of the ISARD project, held by France and Spain. Finally, Rigo et al. (2015) proposed eight zones to estimate stress tensors locally.

Finally, the Institut d'Estudis Andorrans holds an internet site (<https://www.iea.ad/sismoweb>, last accessed on November 2020) with information on Andorran seismicity.

2.2. Triclustering

Regarding the second area of study of this research, the state of the art of triclustering techniques is presented. The traditional clustering approach is a data mining technique that works by grouping objects according to a predefined similarity in a one-dimensional space. When dealing with subspace clustering, the problem can be addressed with biclustering (Pontes et al., 2015) or triclustering (Zhao and Zaki, 2005) if the context data is structured over two or three dimensions, respectively.

In the last decade, the tricluster algorithm has evolved, and many different algorithms have been proposed. For example, some triclustering algorithms are based on iterative searches, distribution parameter identification, biclustering algorithms, pattern mining procedures, or evolutionary multiobjective optimization (Henriques and Madeira, 2018).

In particular, in Liu et al. (2008) and Bhar et al. (2015) some triclustering algorithms based on multiobjective techniques can be found. Furthermore, classic bio-inspired meta-heuristics as genetic algorithms (Holland, 1992) or newer, as the COVID-19 propagation model (Martínez-Álvarez et al., 2020) emerge as a proper method to optimize multiobjective functions. In this sense, the TriGen algorithm presented in Gutiérrez-Avilés et al. (2014) is based on a population

of individuals and some genetic operators. Different fitness functions are also proposed in order to assess each research problem most accurately. TriGen is the triclustering algorithm used in this study and is further explained in 4.3. Besides developing triclustering methodologies, another open line of investigation is to measure the quality of the triclustering algorithms' solutions. In this sense, we can find correlation-based measures inspired by Pearson and Filon (1898), and Spearman (Spearman, 1910) correlation indexes. In Gutiérrez-Avilés and Rubio-Escudero (2014) the authors propose a three-dimensional version of the mean squared residue, a classical biclustering evaluation measure (Cheng and Church, 2000). An evaluation measure based on the last squared lines depicted by the discovered patterns in the triclusters was presented in Gutiérrez-Avilés and Rubio-Escudero (2014). Finally, in Gutiérrez-Avilés and Rubio-Escudero (2015), the authors present a new approach to measure the tricluster quality based on the graphical representation of it.

Triclustering has many applications: biological as Li and Tuck (2009) that combines gene regulator information with expression data; medical as Melgar-García et al. (2020) that applies triclustering in the streaming environment to high-content screening images or the analysis of social networks as Gnatyshak et al. (2012). The domain of seismic data has not yet been deeply analyzed with triclustering techniques. To the best of our knowledge, only the work in Martínez-Álvarez et al. (2015) applies triclustering techniques, particularly, the TriGen algorithm, to data of the Iberian Peninsula, in order to discover possible seismogenic zones.

3. Seismicity and geological settings

The Pyrenees mountain range, constituted from the collision between Iberian and Eurasian plates, was formed in Alpine times (Visser and Meijer, 2012); however, current deformation rates are relatively low. Spanning over 450 km in E-W direction and 150 km in N-S (Rigo et al., 2018), the Pyrenees have nearly cylindrical symmetry (Njike-Kassala et al., 1992). It is bordered to the north by the North Pyrenean Fault, which matches a 10–15 km vertical Moho offset, where the crust is thicker on the Iberian plate (Gallart et al., 1981). The North Pyrenean Fault extends along the whole chain in E-W direction, and it is supposed to be the boundary between the two plates (Njike-Kassala et al., 1992), and currently, between the well-known North Pyrenean Zone and the Axial Zone. Upon Visser and Meijer (2012), the Pyrenees are characterized by, from north to south: first, a north-directed thrust-belt including Mesozoic and Tertiary sediments of the Aquitanian Basin; second, the North Pyrenean Zone, formed mainly by Paleozoic basement and Mesozoic sediments; third, the Axial Zone consisting of Paleozoic rocks; and, finally, the southern Pyrenees where the southward thrust Mesozoic and Tertiary sediments prevail, and they part of the Ebro foreland basin system.

Regarding seismicity, it is very complex (Souriau et al., 2014), and it can be considered as low to moderate (Amaro-Mellado and Tien Bui, 2020). In the western part, the seismicity is generally concentrated along the North Pyrenean Zone. In contrast, it is more diffuse in the eastern part and not particularly associated with the known faults (Njike-Kassala et al., 1992). Besides, the identification and parametrization of seismogenic structures have to face some challenges. The most important ones are the lack of surface faulting related to shocks in the last centuries, the low deformation rates, or the fact of being a political-border region (partially solved by the projects above mentioned) (Lacan and Ortuño, 2012).

Over time, as can be deduced from earthquake catalogs, some shocks have produced severe damages, mainly in the historical period, even with estimated magnitudes between 6 and 7 or with MSK intensities between VIII and IX (Lacan and Ortuño, 2012). Among others, the 1373 earthquake ($I_o = VIII - IX$) by Maladetta Massif, in the Central Pyrenees; those happened during 1427 Amer and 1428 Querulbs, in Catalonia (up to $I_o = IX$), in the eastern Pyrenees;

the 1660 Bigorre earthquake ($I_o = IX$), near Lourdes, and the most recent (1750) in Juncalas ($I_o = VIII$), close to the Bigorre one. In the instrumental period, some relevant shocks (magnitude equal to or larger than 5) have occurred, such as 1967 Arette (M5.3), in the western Pyrenees, and 1996 St-Paul Fenouillet (M5.0), in the Agly Massif, in the east (Amaro-Mellado and Tien Bui, 2020).

Some specific studies have been driven to study the seismicity of the whole Pyrenees Njike-Kassala et al. (1992), Secanell et al. (2007), Rigo et al. (2018), Amaro-Mellado and Tien Bui (2020). Besides, the Arudy-Arette region has been analyzed in-depth by Gallart et al. (1980) and Sylvander et al. (2008).

Fig. 1 shows a general view of the major events that have shocked the Pyrenees from the raw data obtained from the earthquake catalog published by the Instituto Geográfico Nacional (IGN)–National Geographic Institute of Spain (NGIS) (Instituto Geográfico Nacional, 2020). The map only represents the events with a magnitude larger than or equal to 3.5, or macroseismic intensity, I_o , larger than or equal to “V” in macroseismic scale. If both data appear, the magnitude one prevails.

4. Methods

In this section, the methodology driven to generate the seismogenic sources for the Pyrenees are described. Thus, the seismic dataset used in this study is presented; later, the data processing undertaken in this work is explained; and finally, a summary of the TriGen algorithm is set out.

4.1. Generation of the earthquake catalog

The dataset employed in this study comes from the NGIS earthquake catalog (Instituto Geográfico Nacional, 2020). In-depth research on the NGIS catalog is presented in González (2017). In particular, that compiled in Amaro-Mellado and Tien Bui (2020) has been chosen in this work. In this catalog, starting from the NIGS earthquake database, different steps were taken to generate an updated, reviewed, homogeneous, declustered, and extensive catalog, including events with a magnitude larger than or equal to 2.0; from 1373 to 2019; and the geographical extension was limited by 2.5° W and 3.5° E meridians, and 41° N and 44° N parallels.

The steps driven to compile the catalog were the following:

1. Data from other sources, such as specific studies or other catalogs, were considered to re-evaluate the magnitude of some shocks, particularly the major ones.
2. To conduct a seismic analysis, the size of the events must be homogeneous. In this sense, the vast majority of the researches use M_w (Hanks and Kanamori, 1979), due to its direct relationship with the released energy in the rupture by scalar seismic moments. Besides, it does not get saturated for the largest events (Das et al., 2019). Over time (from 1373 to 2019), different kinds of sizes have been assigned to the earthquakes (macroseismic intensities and several magnitude types). Herein, to homogenize the size, M_w was assigned to all the events, according to Cabañas et al. (2015). Besides, after estimating a lower magnitude of completeness (1.8), the cut-off magnitude was set as 2.0.
3. As stated in Section 1, the earthquakes must be independent events to fulfill the Poisson distribution. To this end, foreshocks, aftershocks, and seismic swarms must be eliminated. In seismicity, this process is called declustering. In that work (Amaro-Mellado and Tien Bui, 2020), it was based on the definition of temporal and spatial windows following the method proposed by Gardner and Knopoff (1974).

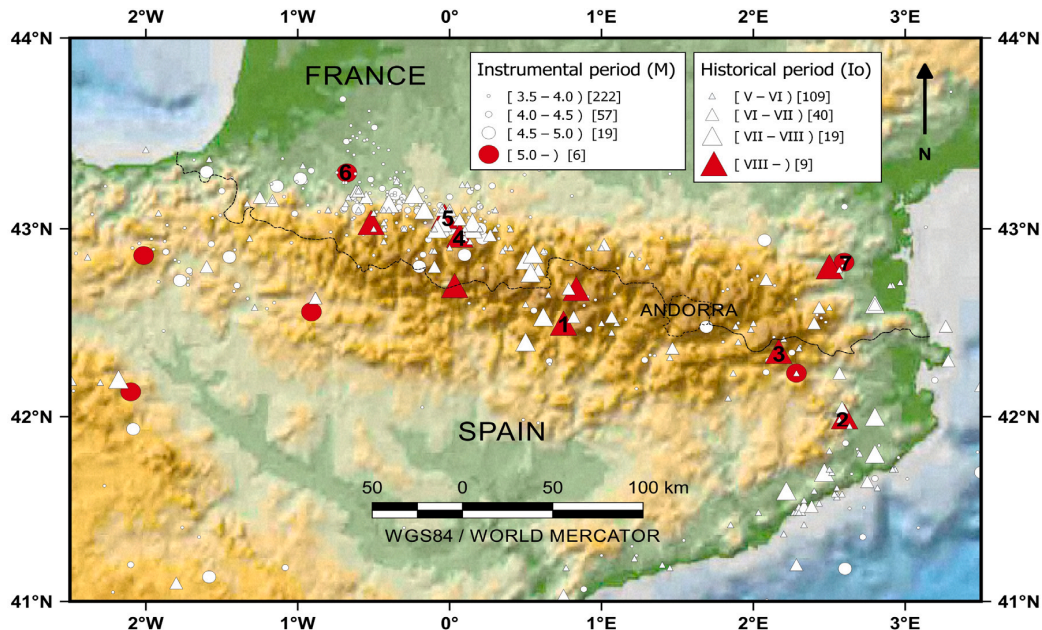


Fig. 1. Seismicity in the Pyrenees from the raw NGIS earthquake catalog. Events instrumentally recorded (Circles); events with only macroseismic intensity (triangles). Above referred earthquakes have been numbered: 1—Maladetta(1373); 2—Amer(1427); 3—Queralbs(1428); 4—Bigorre(1660); 5—Juncalas(1750); 6—Arette(1967); 7—St-Paul Fenouillet(1996).

4. Although it is not a specific task in a catalog generation, the year of completeness related to different levels of magnitude (M_c —year of completeness) were defined: M2—2013; M3—1978; M4—1943; and, M5—1810.
5. The final catalog, after removing the events deeper than 65 km, since they are not important for the seismic hazard of the region (IGN-UPM-WorkingGroup, 2013), consists of 7706 earthquakes.

The present work aims at defining a source zone model, so such a low cut-off value as 2.0 should not be established. Therefore, the cut-off magnitude has been set as 2.5, as pointed out in other works (Njike-Kassala et al., 1992; Talbi et al., 2013). Besides, it has been considered in a recent work for the region (Drouet et al., 2020). This value is judged as a good trade-off between the seismic hazard and the low to moderate Pyrenees seismicity.

The final catalog of this work consists of 3500 earthquakes, with M_w larger than or equal to 2.5, and it is shown in Fig. 2.

4.2. Dataset construction

The seismic data, analyzed in the previous section, must be converted into a 3-D dataset (or cube of data) in order to be interpreted by the TriGen algorithm. Herein, several new attributes must be generated to transform a 2D dataset into a 3D one. Firstly, using a GIS, all data included in the catalog have been sorted into 30×20 cells, each representing an area of approximately $16.7 \text{ km} \times 16.4 \text{ km}$. Afterward, each cell is characterized as a set of features. These features, originally defined by literature (Scitovski and Scitovski, 2013; Reyes and Cárdenas, 2010), have been re-adapted to the Pyrenees seismicity and are:

1. M_{\max} : maximum earthquake magnitude recorded in the cell.
2. D : mean depth of all earthquakes' epicenters recorded in the cell.
3. $M_{2.9}$: number of earthquakes occurring with a magnitude larger than or equal to 2.9 in the cell considered.
4. $M_{3.3}$: number of earthquakes occurring with a magnitude larger than or equal to 3.3 in the cell considered.

5. $M_{3.7}$: number of earthquakes occurring with a magnitude larger than or equal to 3.7 in the cell considered.
6. $M_{4.1}$: number of earthquakes occurring with a magnitude larger than or equal to 4.1 in the cell considered.
7. N : total number of earthquakes occurring.

As can be seen in Fig. 3, the result of applying this data transformation is a data cube (or 3D dataset) where every cell is defined by three coordinates x , y and f . The x coordinate is in $[1, 30]$ representing the relative latitude, the y coordinate is in $[1, 20]$ representing the relative longitude, and, the f coordinate is in $\{M_{\max}, D, M_{2.9}, M_{3.3}, M_{3.7}, M_{4.1}\}$ and represents the particular feature. Therefore, as an example, the cell $C_{1.2, M_{3.7}}$ represents the number of earthquakes occurring with a magnitude larger or equal to 3.7 in the cell with relative latitude 1 and relative longitude 2.

4.3. The TriGen algorithm

The triclustering algorithm used in this research is TriGen (Gutiérrez-Avilés et al., 2014), which stands for Triclustering Genetic based algorithm. As it can be appreciated in Fig. 4(a), the TriGen algorithm receives the 3D input dataset as well as the control parameters to yield a set of tricluster solutions. To do this, TriGen performs an evolutionary heuristic search with a population of individuals that evolves using genetic operators during a specific number of generations with the main goal of optimizing an evaluation function. Considering that an individual is a potential solution tricluster, in this case, each individual represents a zone of the Pyrenees.

Focusing on the algorithm, TriGen repeats for each tricluster solution (N control parameter) the following process. Firstly, it creates an initial population that is evaluated based on a fitness function that has to be optimized. Afterward, four genetics operators are applied to the initial population during a number of generations (G control parameter) in order to select the best individual that is going to be the solution tricluster. The four genetics operators that are repeated are selection, crossover, mutation, and evaluation.

Each operator has specific characteristics as were deeply analyzed in Gutiérrez-Avilés et al. (2014), Gutiérrez-Avilés and Rubio-Escudero (2015):

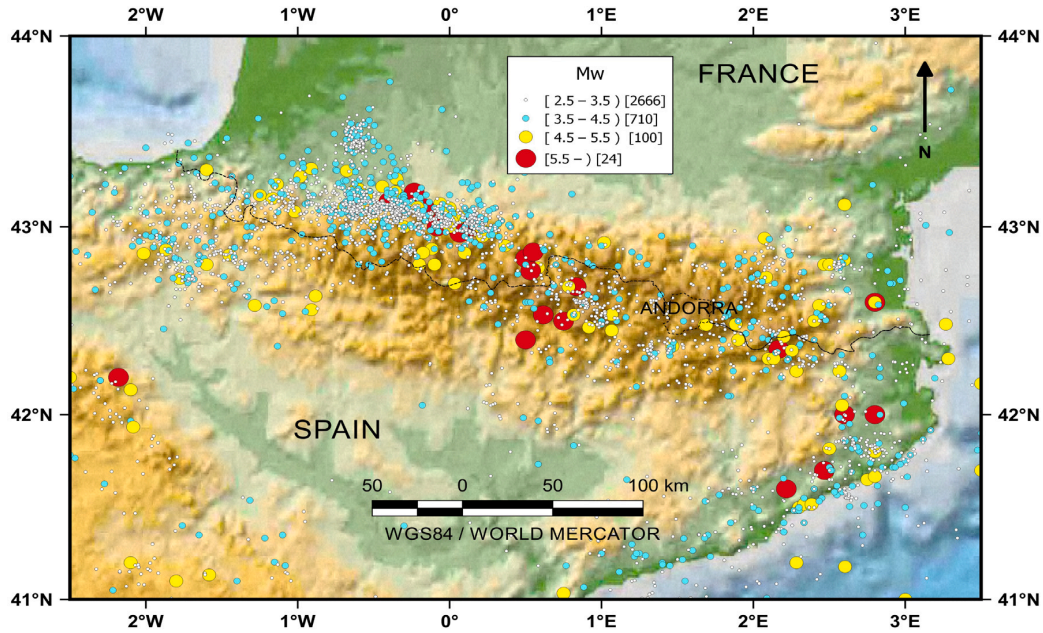


Fig. 2. Catalog of the work. The size of all the events is given as moment magnitude (M_w).

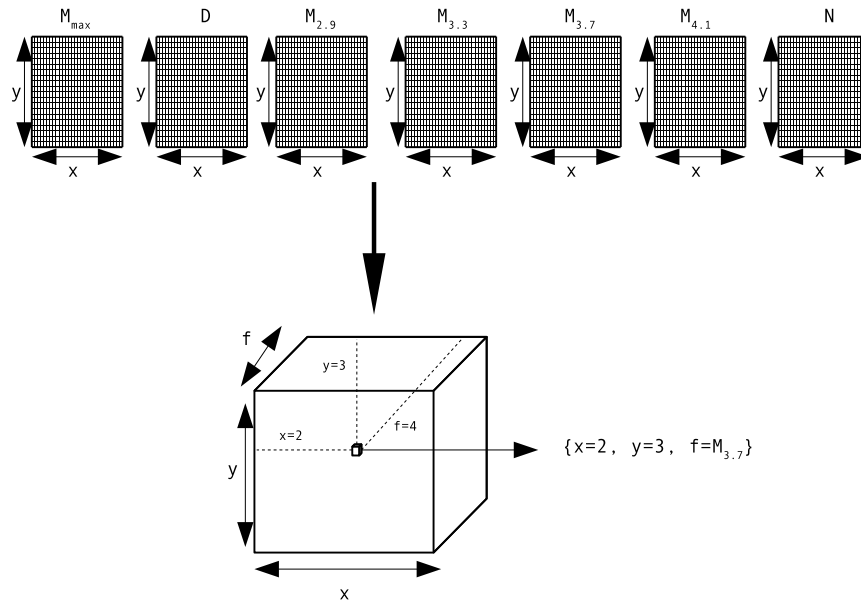


Fig. 3. TriGen input dataset.

- The initial population is created differently to search for the first solution tricluster and all the other $N-1$ solution triclusters searches. The first one is produced with a random subset of all three data coordinates. For the next solution triclusters, some individuals are also randomly generated, and the other ones are created considering not to overlap the previously founded solutions.
- The evaluation operator is very useful in promoting the best individuals to continue in the next generation and selecting the final solution tricluster. TriGen offers three different fitness function: the mean square residue measure (MSR) (Gutiérrez-Avilés and

Rubio-Escudero, 2014), the least square lines (LSL) (Gutiérrez-Avilés and Rubio-Escudero, 2014), and the multi slope measure (MSL) (Gutiérrez-Avilés and Rubio-Escudero, 2015). For this particular case, the MSL fitness function is selected. In general, the MSL function is based on the resemblances of the slope angles formed by the expression values of the tricluster coordinates.

- The selection operator is the first operator computed in the loop of G generations. The tournament selection is implemented, and so the selected individual is promoted to the next generation.

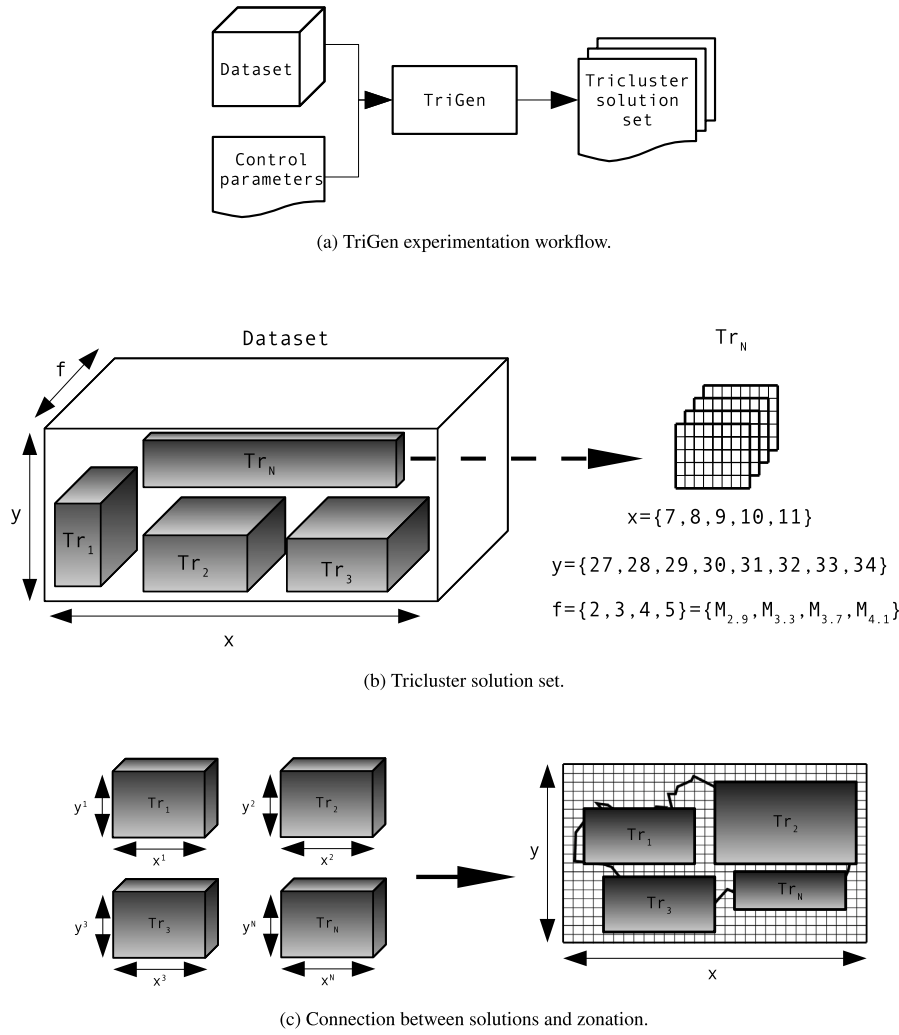


Fig. 4. TriGen algorithm overview.

- The crossover operator applied is the one point crossing technique. In particular, two selected individuals (by a probability of crossover parameter p_c) mix their coordinates randomly and generate two new individuals (their children).
- The mutation operator also considers a probability of mutation parameter p_m that if it is satisfied, applies one of the six possible actions: add or remove one of the three coordinates of data. The goal of the mutation operator is to guarantee variability for the next generations.

The algorithm ends when all the best solution triclusters are found. The produced tricluster solution set represents a partition of the 3D input dataset, whereas each tricluster is a subset of x coordinates, y coordinates, and f coordinates. Therefore, as it is represented in the example in Fig. 4(b), a tricluster Tr_N of the solution set is a dataset subset composed by the x coordinates from 7 to 11, the y coordinates from 27 to 34 and, the $\{2, 3, 4, 5\}$ features, that is, the $M_{2.9}$, $M_{3.3}$, $M_{3.7}$ and $M_{4.1}$ features. To yield the zonation discovered by the algorithm, for each tricluster, its x and y coordinates will be an area of the complete analyzed grid.

5. Results and discussion

In this section, the experimental workflow carried out in this research is explained, as well as an analysis of the obtained results. For this purpose, the experimental setup of the TriGen algorithm, followed to produce the zonation, is described in Section 5.1. Finally, the discovered zone and a critical discussion are addressed in Section 5.2.

5.1. TriGen experimental setup

The TriGen algorithm has been executed four times. As there is a component of randomness in each execution, different executions can lead to different solutions. From each tricluster solution set of each TriGen run, the non-overlapping zones are selected to yield the final zonation presented in the next Section 5.2. As can be seen in Table 1, for each run (from R_1 to R_4), a particular parameter configuration has been used as the input dataset described in Section 4.2, in order to obtain tricluster solutions in the widest range possible.

For all runs, TriGen has been set up to find ten solutions (N) to guide the algorithm to cover the $X - Y$ space. The number of generations (G) and the number of individuals in the first population (I) has been $\{20, 30\}$, and $\{15, 20\}$ respectively. This is considered a proper

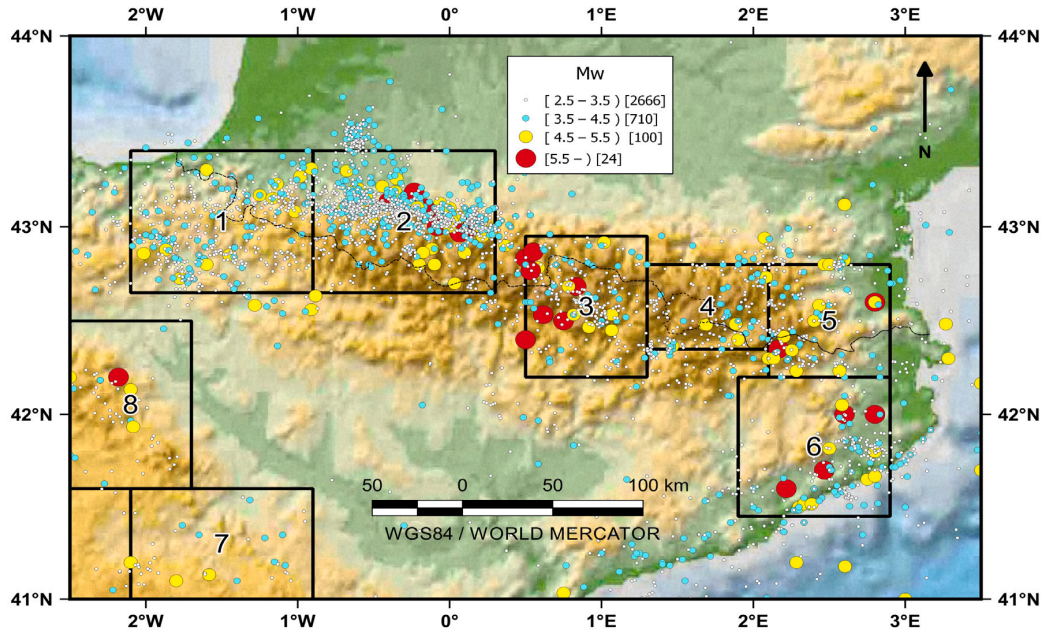


Fig. 5. Zonation proposal (eight zones). The size of all the events is given as moment magnitude (M_w). Source zone #1: Atlantic Pyrenees; #2 Arudy-Arette-Bagnères de Bigorre; #3: Central Pyrenees; #4: Andorra; #5: Easternmost Pyrenees; #6: Eastern Catalonia; #7: Central Iberian System; #8: Northeast Iberian System.

Table 1
TriGen parameters setup.

Parameter	R_1	R_2	R_3	R_4
N	10	10	10	10
G	20	30	20	30
I	15	20	20	15
p_c	0.2	0.2	0.8	0.8
p_m	0.5	0.5	0.1	0.1

balance between intensification and diversification of the evolutionary process. This fact has been taken into account when the crossover and mutation parameters were set. Firstly, p_c varies in $\{0.2, 0.8\}$ to obtain high intensified and low diversified solutions in the $p_c = 0.2$ runs and low intensified and high diversified solutions in the $p_c = 0.8$ runs. In a similar way, with p_m varying in $\{0.1, 0.5\}$, to get solutions with high and low variability.

5.2. Zonation obtained and its representation

With the solutions obtained by the application TriGen, following the procedure described in Section 5.1, the zones with less than 25 events have been ruled out, according to Bender (1983) and Skordas and Kulhánek (1992). The use of a GIS (QGIS, <https://www.qgis.org>, last accessed November 2020) allows a rigorous representation, and the data can be handled for further analysis. The resulting zonation, as well as the epicenters distribution, is shown in Fig. 5.

5.2.1. Seismic parameters of the obtained zones

Once the zones have been delineated according to solutions provided by TriGen, the next step is to seismically characterize each zone, in which some seismic parameters must be uniform, namely, the b -value, or the annual rate, as well as the maximum magnitude. Therefore, they must be estimated for the proposed zones.

Thus, the Gutenberg-Richter law (Gutenberg and Richter, 1944, 1954) holds:

$$\log_{10} N(M) = a - bM \quad (1)$$

where the b -value estimates the relationship between small and large earthquakes, and it is related to the physics of the source. The lower its value is, the more energy can be accumulated. N is the number of events with a magnitude larger than or equal to a cut-off magnitude M , and a refers to the seismic productivity.

The Maximum-Likelihood-Estimate proposed by Bender (1983) and Aki (1965) for binned data (Δ) has been considered, in terms of b -value.

$$b = \frac{1}{\ln(10)(\bar{M} - M_{min} - \frac{\Delta}{2})} \quad (2)$$

where \bar{M} is the average magnitude; M_{min} is the magnitude of completeness, which means all the events of magnitude larger than or equal to this value have been recorded from the reference date;

In this work, M_{min} has been set as 3.0, and the corresponding year of completeness is 1978, as stated in Section 4.1, and the Δ value for this work is 0.1. Besides, it is interesting to give uncertainty parameters in geology (Bárdossy and Fodor, 2001). In this case, the b -value uncertainty (Sigma b) has been calculated by the approach suggested by Kijko and Smit (2012).

Another relevant parameter is the annual rate referred to a unit area (km^2). It means the number of events exceeding a threshold per year and related to km^2 .

$$\lambda(M_{min}) = \frac{N}{t} \quad (3)$$

where t is the time-lapse. In this research, from the beginning of 1978 until the end of 2019, i.e., 42 years.

When this value is normalized with the surface, by km^2 , and multiplied by 10,000, it is called $AR3$, in this paper.

$$AR3 = 10,000 \frac{\lambda(M_{min})}{km^2} \quad (4)$$

Finally, the maximum magnitude M_{max} is calculated for each zone. The resulting values can be found in Table 2.

Table 2
Obtained zones. Seismic parameters.

Zone	Name	Area (km ²)	<i>b</i> -value	<i>AR</i> ₃	<i>M</i> _{max}	<i>Depth</i> _{mean}	Sigma <i>b</i>
1	Atlantic Pyrenees	8157	1.01	5.43	5.2	8.4	0.07
2	Arudy–Arette–Bagnères de Bigorre	8157	0.99	14.54	6.7	6.0	0.04
3	Central Pyrenees	5478	1.20	3.30	6.2	5.7	0.14
4	Andorra	3287	1.32	3.19	5.0	5.7	0.20
5	Easternmost Pyrenees	4388	1.17	3.91	6.5	5.8	0.14
6	Eastern Catalonia	6931	1.34	1.89	5.8	6.3	0.18
7	Central Iberian System	6709	1.13	0.50	5.2	8.6	0.30
8	Northeast Iberian System	6630	1.40	0.61	5.7	8.0	0.34

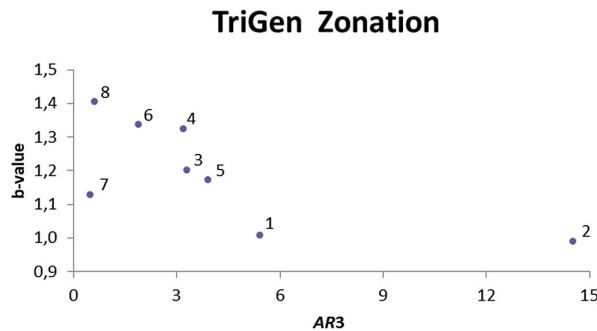


Fig. 6. *AR*₃ vs *b*-value.

5.3. Discussion

After calculating some of the most relevant seismic parameters, the analysis of the results is driven.

First, in order to check that the zones are seismically different from one another, a graphical representation of the *AR*₃ and *b*-value is presented in Fig. 6. In addition, to support this discrimination, different color maps have been deployed. Herein, Fig. 7 shows the *b*-value variation; *M*_{max}; in Fig. 8 the *AR*₃ is depicted; Fig. 9 represents the *M*_{max}; and finally, the mean depth is illustrated in Fig. 10.

Second, the analysis of each zone is undertaken.

Zone 1 is located in the Atlantic Pyrenees and presents one of the lowest *b*-values and the highest annual rate. However, the maximum magnitude is one of the smallest of the proposed zones (5.2), so the seismic hazard can be considered low to moderate in the Pyrenees region.

The most seismically hazardous zone is undoubtedly the number 2, which runs from the Arudy–Arette region to the Bagnères de Bigorre environment. This is due to the combination of the minimum *b*-value, and both maximum annual rate (the highest in the Iberian Peninsula) and *M*_{max}. Therefore, this zone deserves special attention when a PSHA is driven in the Pyrenees.

Zone 3 is situated in the Central Pyrenees, in the Axial Zone. This area, which includes the Maladetta Massif, has been shocked by some of the strongest events in the Pyrenees (M6.2). Besides, it presents medium values in both annual rate and *b*-value. Herein, the seismic hazard can be defined as moderate; it might be even said moderate to high.

The smallest zone, numbered 4, is located in the Andorra environment, and not significant earthquakes have been taken place in it. Its *b*-value is high, and the annual rate is medium.

Like Zone 3, the 5 one, in the easternmost Pyrenees, is characterized by a medium *b*-value and annual rate but has suffered the second most energetic earthquake in the whole belt. Therefore, the seismic hazard can be estimated as moderate to high.

In zone 6, fully contained in eastern Catalonia (Spain), although the *M*_{max} is not relatively high, four events with a magnitude exceeding 5.5 have occurred. Besides, it shows a high *b*-value, and the annual rate is low to moderate so that the seismicity can be rated as moderate. The epicenter distribution may indicate that this zone could be extended along the Catalanian Coastal Range to the southwest or consider a new one.

The following zones, 7 and 8, could not be closely related to the Pyrenees, but in its environment, in the Iberian System. Besides, both zones are in the limit to be seen as a seismic zone, individually. Both zones have the lowest annual rate by far, and the maximum magnitude is not notably high (5.7 and 5.2). Regarding the *b*-value, zone 7 (Central) shows an average value, and zone 8 (Northeast) the highest in the region. Given the few events available for *b*-value calculations, these are not entirely meaningful. The epicenter distribution points out that both zones could correspond to the same seismic zone if a proper rotation were to be carried out.

Although it is a debated topic, the highest seismic potential of source zone #2 (Arudy–Arette–Bagnères de Bigorre) can be related to the presence of several major faults. For example, the 25–30 km-long Herrère right lateral fault (Arudy) or the Lourdes reverse faults (60 km in three segments). Besides, the relevant seismicity of source zone #3 is related to the North Maladetta normal fault (30 km long) or Coronas normal fault (10 km) (Lacan and Ortuño, 2012). It is also remarkable that in zone 5 (Easternmost Pyrenees), the more destructive earthquake could be caused by the Vallfogona or Ribes–Camprodon thrusts or the Amer normal fault. The latter fault generated the major 1428 Queralbs earthquake (in zone 6) (Perea, 2009).

Regarding the method limitations, one of them would be that it only produces “rectangular zones” along parallels and meridians arches, as the original TriGen algorithm works that way. Thus, to overcome this issue, the results could be improved using rotated polygons, which could have been useful to extend zone 6, or join zones 7 and 8. Another method limitation is related to the fact that conceptually, an active fault should not be cut by a source zone boundary (Hamdache and Peláez, 2019), so it could be considered in further research.

6. Conclusions

In this research, a seismic zonation has been proposed for the Pyrenees mountain range. The zoning is purely based on the application of the TriGen algorithm to an earthquake catalog and a grid division. Herein, a GIS has been used to manage and represent both the input and the output data. The method aims to be as objective as possible, seeking patterns for the best considered seismic indicators and minimizing human interaction.

The triclustering methodology has been the primary procedure to yield the zonation and applied to a pre-processed 3D dataset using the TriGen algorithm. It is based on the genetic algorithm model. Thus, from an initial population of potential solutions, by applying genetic operators (crossovers, mutations), it evolves to optimal solutions based on the *MSL* fitness function.

The starting point is an updated, reliable, homogeneous, extensive, and with independent events earthquake catalog, coupled with a geographical grid defined by the GIS. This grid cell size has been set out as a nearly square shape in 16.7 km × 16.4 km, a trade-off between the coverage and the low to moderate Pyrenees seismicity. Then, as a previous step to run the evolutionary algorithm, the suggested best indicators have been adapted to the Pyrenean seismicity. Finally, the algorithm

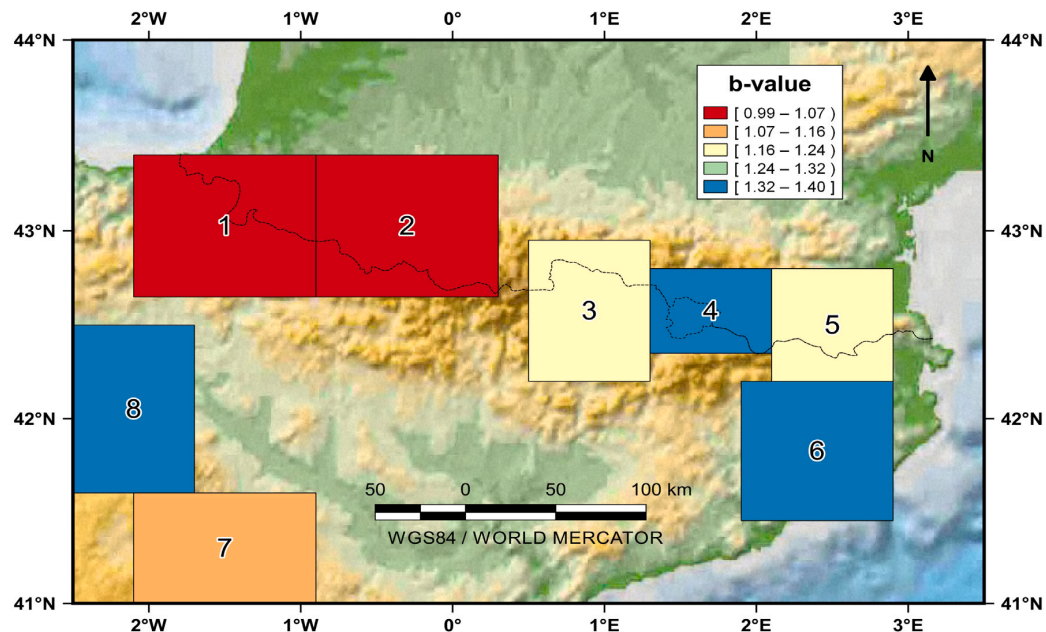


Fig. 7. Color map showing the b -value for the obtained source zones. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

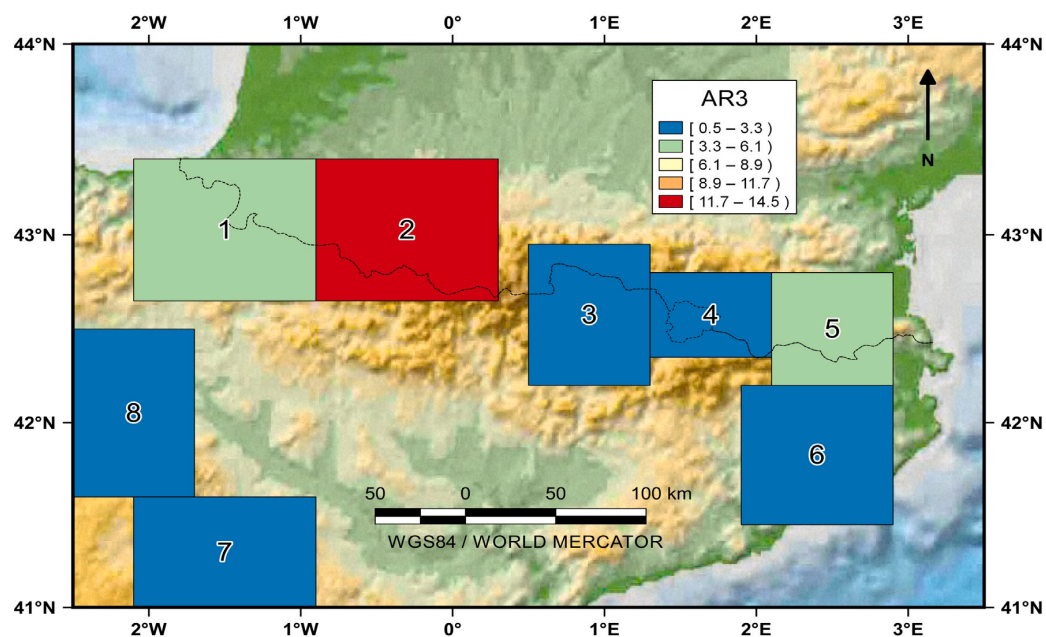


Fig. 8. Color map showing the $AR3$ for the obtained source zones. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

has generated the eight seismogenic zones, and their geometries have been integrated into a GIS.

The following step is characterizing the zones seismically. To this end, the most relevant seismic parameters have been calculated for each source zone by a GIS to further analysis. This analysis reveals that the source zone covered by the Arudy–Arette–Bagnères de Bigorre region is the most relevant seismically by far, due to its minimum b -value, the maximum annual rate, and the highest M_{max} . Therefore,

this zone deserves special attention when a PSHA is driven in the Pyrenees. The seismic hazard is also remarkable in both the Axial Zone, that includes the Maladetta Massif, and in the easternmost Pyrenees, along the Spain–France border, including $M6+$ events over time. In the Catalanian Coastal Range, the zone rated as medium seismicity could be extended through this range to the southwest. In the Atlantic Pyrenees and the Andorra environment, the seismic activity is less pronounced. Finally, in the southwest of the studied region (even

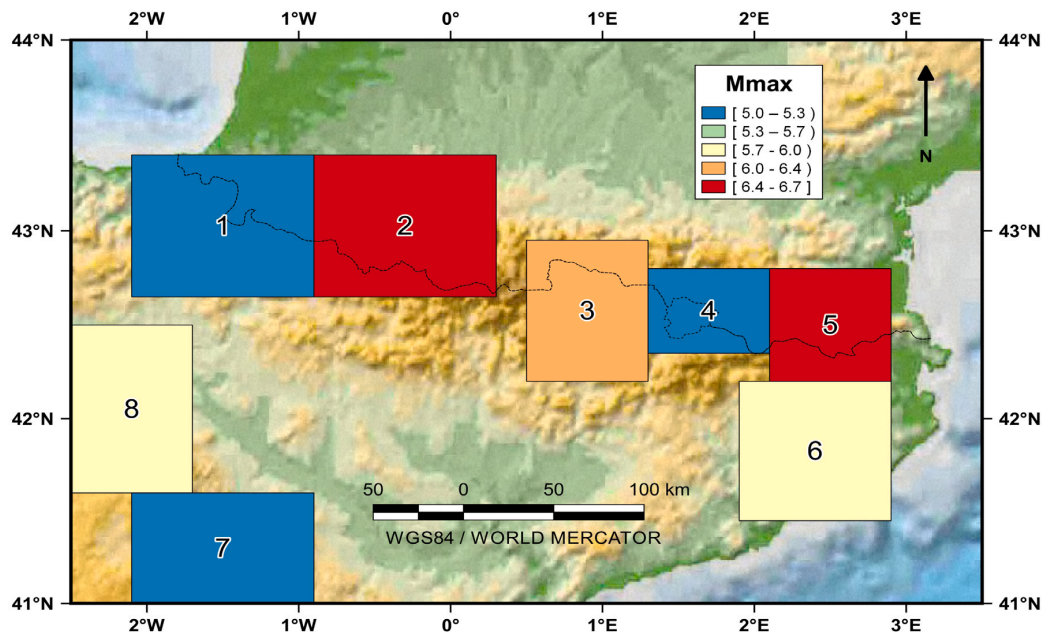


Fig. 9. Color map showing the M_{max} for the obtained source zones. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

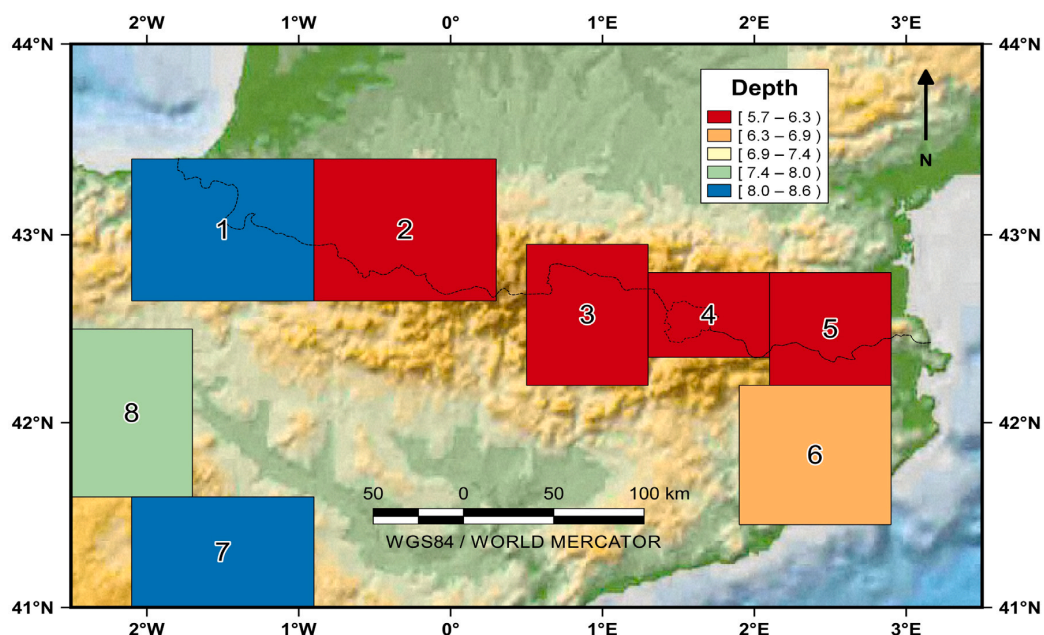


Fig. 10. Color map showing the mean depth for the obtained source zones. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

outside the Pyrenees environment), the seismic activity is significantly lower, and the two obtained zones should be merged into one.

The method has shown to be efficient as the delineated zones are different from one another and cover the vast majority of the region's epicenter. Besides, the most relevant point is that the procedure is almost human-free-action.

CRediT authorship contribution statement

José L. Amaro-Mellado: Conceive and design the experiments, Retrieve and analyze the data, Writing - original draft, Contribution to the writing of the manuscript, Agreement with manuscript results and conclusions. **Laura Melgar-García:** Contribution to the writing

of the manuscript, Agreement with manuscript results and conclusions. **Cristina Rubio-Escudero**: Contribution to the writing of the manuscript, Agreement with manuscript results and conclusions, Development of the structure and arguments of the paper, Revision and approbation of the final manuscript. **David Gutiérrez-Avilés**: Conceive and design the experiments, Retrieve and analyze the data, Writing - original draft, Contribution to the writing of the manuscript, Agreement with manuscript results and conclusions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors want to thank the financial support given by the European Commission 0313-PERSISTAH project, the Spanish Ministry of Economy and Competitiveness project TIN2017-88209-C2, and the Junta de Andalucía US-1263341 project.

Computer code availability

TriGen 3.5, GPL v3 license, <https://github.com/davgutavi/trlab-trigen>.

References

- Aki, K., 1965. Maximum likelihood estimate of b in the formula $\log N = a - bM$ and its confidence limits. *Bull. Earthq. Res. Inst.* 43, 237–239.
- Amaro-Mellado, J.L., Morales-Esteban, A., Asencio-Cortés, G., Martínez-Álvarez, F., 2017. Comparing seismic parameters for different source zone models in the Iberian Peninsula. *Tectonophysics* 717 (July), 449–472.
- Amaro-Mellado, J.L., Tien Bui, D., 2020. GIS-based mapping of seismic parameters for the pyrenees. *ISPRS Int. J. Geo-Inf.* 9 (7), 452.
- Baize, S., Cushing, E.M., Lemeille, F., Jomard, H., 2013. Updated seismotectonic zoning scheme of Metropolitan France, with reference to geologic and seismotectonic data. *Bull. Soc. Geol. France* 184 (3), 225–259.
- Barani, S., Mascandola, C., Riccomagno, E., Spallarossa, D., Albarello, D., Ferretti, G., Scafidì, D., Augliera, P., Massa, M., 2018. Long-range dependence in earthquake-moment release and implications for earthquake occurrence probability. *Sci. Rep.* 8 (1), 1–11.
- Bárdossy, G., Fodor, J., 2001. Traditional and new ways to handle uncertainty in geology. *Nat. Resour. Res.* 10 (3), 179–187.
- Bender, B., 1983. Maximum likelihood estimation of b values for magnitude grouped data. *Bull. Seismol. Soc. Am.* 73 (3), 831–851.
- Bernal, A., 2011. Anexo I del informe técnico IGN-PSE. ZF. P03. IGN-PSE. ZF. P03.
- Bhar, A., Haubrock, M., Mukhopadhyay, A., Wingender, E., 2015. Multiobjective triclustering of time-series transcriptome data reveals key genes of biological processes. *BMC Bioinformatics* 16, 200.
- Cabañas, L., Rivas-Medina, A., Martínez-Solares, J.M., Gaspar-Escribano, J.M., Benito, B., Antón, R., Ruiz-Barajas, S., 2015. Relationships between M_w and other earthquake size parameters in the Spanish IGN seismic catalog. *Pure Appl. Geophys.* 172 (9), 2397–2410.
- Cheng, Y., Church, G.M., 2000. Biclustering of expression data. In: *International Conference on Intelligent Systems for Molecular Biology*. pp. 93–103.
- Cornell, C.A., 1968. Engineering seismic risk analysis. *Bull. Seismol. Soc. Am.* 58, 1583–1606.
- Corral, Á., 2006. Dependence of earthquake recurrence times and independence of magnitudes on seismicity history. *Tectonophysics* 424 (3–4), 177–193.
- Das, R., Sharma, M.L., Wason, H.R., Choudhury, D., Gonzalez, G., 2019. A seismic moment magnitude scale. *Bull. Seismol. Soc. Am.* 109 (4), 1542–1555.
- Drouet, S., Ameri, G., Le Dortz, K., Secanell, R., Senfaute, G., 2020. A probabilistic seismic hazard map for the metropolitan France. *Bull. Earthq. Eng.* 18 (5), 1865–1898.
- Gallart, J., Banda, E., Daignières, M., 1981. Crustal structure of the Paleozoic Axial Zone of the Pyrenees and transition to the North Pyrenean Zone. *Ann. Géophys.* 37 (3), 457–480.
- Gallart, J., Daignières, M., Banda, E., Suriñach, E., Hirn, A., 1980. The eastern Pyrenean domain: lateral variations at crust-mantle level. *Ann. Geophys.* 36 (2), 141–158.
- García-Mayordomo, J., 2015. Creación de un modelo de zonas sísmicas para el cálculo del mapa de peligrosidad sísmica de España. In: *Riesgos Geológicos/Geotecnia n° 5*, Instituto Geológico y Minero de España, p. 125.
- García-Mayordomo, J., Insua-Arévalo, J.M., Martínez-Díaz, J.J., Jiménez-Díaz, A., Martín-Banda, R., Martín-Alfageme, S., Álvarez-Gómez, J.A., Rodríguez-Peces, M., Pérez-López, R., Rodríguez-Pascua, M.A., Masana, E., Perea, H., Martín-González, F., Giner-Robles, J., Nemser, E.S., Cabral, J., 2012a. The quaternary active faults database of Iberia (QAFI v.2.0). *J. Iberian Geol.* 38 (1), 285–302.
- García-Mayordomo, J., Martínez-Díaz, J.J., Capote, R., Martín-Banda, R., Insua-Arévalo, J.M., Álvarez-Gómez, J.A., Perea, H., González, A., Lafuente, P., Martínez-González, F., Pérez-López, R., Rodríguez-Pascua, M.A., Giner-Robles, J., Azahón, J., Masana, E., Moreno, X., Benito, B., Rivas, A., Gaspar-Escribano, J.G., Cabañas, L., Vilanova, S., Fonseca, J., Nemser, E., Baize, S., 2012b. Modelo de zonas sísmicas para el cálculo de la peligrosidad sísmica en España. In: *Actas de la 7 Asamblea Geodesia y Geofísica*. pp. 23–28.
- Gardner, J.K., Knopoff, L., 1974. Is the sequence of earthquakes in Southern California, with aftershocks removed, Poissonian. *Bull. Seismol. Soc. Am.* 64 (5), 1363–1367.
- Gnatyshak, D., Ignatov, D., Semenov, A., Poelmans, J., 2012. Gaining insight in social networks with biclustering and triclustering. In: *Perspectives in Business Informatics Research*. In: *Lecture Notes in Business Information Processing*, vol. 128, pp. 162–171.
- González, Á., 2017. The Spanish National Earthquake Catalogue: Evolution, precision and completeness. *J. Seismol.* 21 (3), 435–471.
- Gutenberg, B., Richter, C.F., 1944. Frequency of earthquakes in California. *Bull. Seismol. Soc. Am.* 34, 185–188.
- Gutenberg, B., Richter, C.F., 1954. *Seismicity of the Earth*. Princeton University.
- Gutiérrez-Avilés, D., Rubio-Escudero, C., 2014. LSL: A new measure to evaluate triclusters. In: *2014 IEEE International Conference on Bioinformatics and Biomedicine*. BIBM, IEEE, pp. 30–37.
- Gutiérrez-Avilés, D., Rubio-Escudero, C., 2014. Mining 3D patterns from gene expression temporal data: A new tricluster evaluation measure. *Sci. World J.* 2014, 1–16.
- Gutiérrez-Avilés, D., Rubio-Escudero, C., 2015. MSL: A measure to evaluate three-dimensional patterns in gene expression data. *Evol. Bioinform.* 11, 121–135.
- Gutiérrez-Avilés, D., Rubio-Escudero, C., Martínez-Álvarez, F., Riquelme, J.C., 2014. TriGen: A genetic algorithm to mine triclusters in temporal gene expression data. *Neurocomputing* 132, 42–53.
- Hamdache, M., Peláez, J.A., 2019. Comment on the paper “seismic hazard analysis of surface level, using topographic condition in the northeast of Algeria” by Mouloud Hamidatou, Mohammedi Yahia, Abdelkrim Yelles-Chaouche, Itharam Thallak, Dietrich Stromeyer, Saad Lebdioui, Fabrice Cotton. *Pure Appl. Geophys.*
- Hanks, T., Kanamori, H., 1979. Moment magnitude scale. *J. Geophys. Res.* 84 (B5), 2348–2350.
- Henriques, R.U.I., Madeira, S.C., 2018. Triclustering algorithms for three-dimensional data analysis: A comprehensive survey. *ACM Comput. Surv.* 51 (5).
- Holland, J.H., 1992. Genetic algorithms. *Sci. Am.* 267 (1), 66–73.
- IGN-UP-WorkingGroup, 2013. Actualización de mapas de peligrosidad sísmica 2012. Instituto Geográfico Nacional, p. 267.
- Instituto Geográfico Nacional, 2020. Catálogo de terremotos. Instituto Geográfico Nacional, URL: <http://www.ign.es/web/ign/portal/sis-catalogo-terremotos>.
- Kijko, A., Smit, A., 2012. Extension of the Aki-Utsu b -value estimator for incomplete catalogs. *Bull. Seismol. Soc. Am.* 102 (3), 1283–1287.
- Lacan, P., Ortuño, M., 2012. Active Tectonics of the Pyrenees: A review. *J. Iberian Geol.* 38 (1), 9–30.
- Li, A., Tuck, D., 2009. An effective tri-clustering algorithm combining expression data with gene regulation information. *Gene Regul. Syst. Biol.* 3, 49–64.
- Liu, J., Li, Z., Hu, X., Chen, Y., 2008. Multi-objective evolutionary algorithm for mining 3D clusters in gene-sample-time microarray data. In: *2008 IEEE International Conference on Granular Computing*. pp. 442–447.
- Marin, S., Avouac, J.P., Nicolas, M., Schlupp, A., 2004. A probabilistic approach to seismic hazard in metropolitan France. *Bull. Seismol. Soc. Am.* 94 (6), 2137–2163.
- Martín, A.J., 1984. Riesgo sísmico en la península Ibérica (Ph.D.). Instituto Geográfico Nacional.
- Martin, C., Secanell, R., Combes, P., Lignon, G., 2002. Preliminary probabilistic seismic hazards assessment of France. In: *12th European Conference in Earthquake Engineering*. London. p. 870.
- Martínez-Álvarez, F., Asencio-Cortés, G., Torres, J.F., Gutiérrez-Avilés, D., Melgar-García, L., Pérez-Chacón, R., Rubio-Escudero, C., Riquelme, J.C., Troncoso, A., 2020. Coronavirus optimization algorithm: A bioinspired metaheuristic based on the COVID-19 propagation model. *Big Data* 8 (4), 308–322.
- Martínez-Álvarez, F., Gutiérrez-Avilés, D., Morales-Esteban, A., Reyes, J., Amaro-Mellado, J., Rubio-Escudero, C., 2015. A novel method for seismogenic zoning based on triclustering: Application to the Iberian peninsula. *Entropy* 17 (12), 5000–5021.
- McGuire, R.K., 1976. FORTRAN Computer Program for Seismic Risk Analysis. Technical Report 76-67, US Geological Survey Open-File Report, p. 90.
- Melgar-García, L., Gutiérrez-Avilés, D., Rubio-Escudero, C., Troncoso, A., 2020. High-content screening images streaming analysis using the STriGen methodology. In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. SAC '20, Association for Computing Machinery, pp. 537–539.
- Mezcua, J., Rueda, J., García Blanco, R.M., 2011. A new probabilistic seismic hazard study of Spain. *Nat. Hazards* 59 (2), 1087–1108.
- Ministerio de Fomento (Gobierno de España), 2002. Norma de la Construcción Sismorresistente Española (NCSE-02). Boletín Oficial del Estado.

- Molina, S., 1998. Sismotectónica y peligrosidad sísmica del área de contacto entre Iberia y África. Universidad de Granada, Spain.
- Morales-Esteban, A., Martínez-Álvarez, F., Scitovski, S., Scitovski, R., 2014. A fast partitioning algorithm using adaptive Mahalanobis clustering with application to seismic zoning. *Comput. Geosci.* 73, 132–141.
- Njike-Kassala, J.D., Souriau, A., Gagnepain-Beyneix, J., Martel, L., Vadell, M., 1992. Frequency-magnitude relationship and Poisson's ratio in the Pyrenees, in relation to earthquake distribution. *Tectonophysics* 215 (3–4), 363–369.
- Pearson, K., Filon, L.N.G., 1898. Mathematical contributions to the theory of evolution. IV. On the probable errors of frequency constants and on the influence of random selection on variation and correlation. *Phil. Trans. R. Soc. Lond. Ser. A* 229–311, Containing Papers of a Mathematical or Physical Character.
- Pecker, A., Faccioli, E., Gurpinar, A., Martin, C., Renault, P.L.A., 2017. An Overview of the SIGMA Research Project: A European Approach to Seismic Hazard Analysis. In: *Geotechnical, Geological and Earthquake Engineering*, vol. 42, Springer, p. 172.
- Perea, H., 2009. The Catalan seismic crisis (1427 and 1428; NE Iberian Peninsula): Geological sources and earthquake triggering. *J. Geodyn.* 47 (5), 259–270.
- Pontes, B., Giráldez, R., Aguilar-Ruiz, J.S., 2015. Biclustering on expression data: A review. *J. Biomed. Inform.* 57, 163–180.
- Reyes, J., Cárdenas, V.H., 2010. A Chilean seismic regionalization through a Kohonen neural network. *Neural Comput. Appl.* 19 (7), 1081–1087.
- Rigo, A., Souriau, A., Sylvander, M., 2018. Spatial variations of b-value and crustal stress in the Pyrenees. *J. Seismol.* 22 (1), 337–352.
- Rigo, A., Vernant, P., Feigl, K.L., Goula, X., Khazaradze, G., Talaya, J., Morel, L., Nicolas, J., Baize, S., Chery, J., Sylvander, M., 2015. Present-day deformation of the Pyrenees revealed by GPS surveying and earthquake focal mechanisms until 2011. *Geophys. J. Int.* 201 (2), 947–964.
- Scitovski, S., 2018. A density-based clustering algorithm for earthquake zoning. *Comput. Geosci.* 110, 90–95.
- Scitovski, R., Scitovski, S., 2013. A fast partitioning algorithm and its application to earthquake investigation. *Comput. Geosci.* 59, 124–131.
- Secanell, R., Bertil, D., Martin, C., Goula, X., Susagna, T., Tapia, M., Dominique, P., Carbon, D., Fleta, J., 2008. Probabilistic seismic hazard assessment of the Pyrenean region. *J. Seismol.* 12 (3), 323–341.
- Secanell, R., Martin, C., Goula, X., Susagna, T., Tapia, M., Bertil, D., 2007. Evaluación probabilista de la peligrosidad sísmica de la región pirenaica. In: 3° Congreso Nacional de Ingeniería Sísmica, no. 1. Asociación Española de Ingeniería Sísmica, pp. 1–17.
- Skordas, E., Kulháněk, O., 1992. Spatial and temporal variations of Fennoscandian seismicity. *Geophys. J. Int.* 111 (3), 577–588.
- Souriau, A., Rigo, A., Sylvander, M., Benahmed, S., Grimaud, F., 2014. Seismicity in central-western Pyrenees (France): A consequence of the subsidence of dense exhumed bodies. *Tectonophysics* 621, 123–131.
- Spearman, C., 1910. Correlation calculated from faulty data. *Br. J. Psychol.*, 1904–1920 3 (3), 271–295.
- Stucchi, M., Rovida, A., Gomez Capera, A.A., Alexandre, P., Camelbeeck, T., Demircioglu, M.B., Gasperini, P., Kouskouna, V., Musson, R.M.W., Radulian, M., Sesetyan, K., Vilanova, S., Baumont, D., Bungum, H., Fäh, D., Lenhardt, W., Makropoulos, K., Martinez Solares, J.M., Scotti, O., Živčić, M., Albin, P., Batlo, J., Papaioannou, C., Tatevossian, R., Locati, M., Meletti, C., Viganò, D., Giardini, D., 2013. The SHARE European earthquake catalogue (SHEEC) 1000-1899. *J. Seismol.* 17 (2), 523–544.
- Sylvander, M., Souriau, A., Rigo, A., Tocheport, A., Toutain, J.P., Ponsolles, C., Benahmed, S., 2008. The 2006 November, M L = 5.0 earthquake near Lourdes (France): new evidence for NS extension across the Pyrenees. *Geophys. J. Int.* 175 (2), 649–664.
- Talbi, A., Nanjo, K., Satake, K., Zhuang, J., Hamdache, M., 2013. Comparison of seismicity declustering methods using a probabilistic measure of clustering. *J. Seismol.* 17 (3), 1041–1061.
- Visser, R.L.M., Meijer, P.T., 2012. Iberian plate kinematics and Alpine collision in the Pyrenees. *Earth-Sci. Rev.* 114 (1–2), 61–83.
- Woessner, J., Danciu, L., Kästli, P., Monelli, D., 2011. Grant agreement no. 226967 seismic hazard harmonization in Europe project acronym: SHARE. pp. 1–23, SHARE, 226967.
- Zhao, L., Zaki, M., 2005. TRICLUSTER: an effective algorithm for mining coherent clusters in 3D microarray data. In: *Proc. of the 2005 ACM SIGMOD International Conference on Management of Data*. pp. 694–705.

5.1.7 | "Nearest neighbors-based forecasting for electricity demand time series in streaming"

Authors: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A.

Publication type: Conference article.

Conference: XIX Conference of the Spanish Association for Artificial Intelligence (CAEPIA 20/21).

Publication: Advances in Artificial Intelligence, Springer International Publishing, Cham.

Year: 2021.

Volume: 12882 LNAI

Pages: 185-195

DOI: 10.1007/978-3-030-85713-4_18



Nearest Neighbors-Based Forecasting for Electricity Demand Time Series in Streaming

L. Melgar-García^{1(✉)}, D. Gutiérrez-Avilés², C. Rubio-Escudero²,
and A. Troncoso¹

¹ Data Science and Big Data Lab, Pablo de Olavide University, 41013 Seville, Spain
{lmelgar, atrolor}@upo.es

² Department of Computer Science, University of Seville, Seville, Spain
{dgutierrez3, crubioescudero}@us.es

Abstract. This paper presents a new forecasting algorithm for time series in streaming named StreamWNN. The methodology has two well-differentiated stages: the algorithm searches for the nearest neighbors to generate an initial prediction model in the batch phase. Then, an online phase is carried out when the time series arrives in streaming. In particular, the nearest neighbor of the streaming data from the training set is computed and the nearest neighbors, previously computed in the batch phase, of this nearest neighbor are used to obtain the predictions. Results using the electricity consumption time series are reported, showing a remarkable performance of the proposed algorithm in terms of forecasting errors when compared to a nearest neighbors-based benchmark algorithm. The running times for the predictions are also remarkable.

Keywords: Forecasting · Nearest neighbors · Streaming time series · Electricity demand

1 Introduction

The explosive increase of global data, based on technology improvements, has led to the gathering of information as an automatic and relatively inexpensive task [16], taking us to the big data era. Data science offers a solution to gain knowledge from these enormous amounts of data, by means of adapting the existing models to the big data paradigm. This adaptation is a challenge for the research community.

There are several fields in which the application of the new big data analysis techniques represent a great improvement in problem solving, such as the energy consumption forecasting [17, 25]. Governments and private companies are focusing on this topic as the improvement in the prediction levels will have both economic and environmental positive consequences [22]. In this sense, some classifiers have already been successfully applied to electricity consumption forecast

186 L. Melgar-García et al.

[24], such as the weighted k-nearest neighbors classifier (WKNN). The WKNN [4] is a generalization of the k-nearest neighbors method (KNN) [2] that assigns weights to the neighbors based on their distance from the element to predict.

The direct application of these methods to the big data domain is not feasible due to the computational needs, in terms of time and memory. Several proposals have adapted nearest neighbor proposals to the big data paradigm using the Apache Spark distributed computation framework [16,21,22].

Data streams are generated in many practical applications as temporally ordered, fast changing and massive flows of data [13]. Mining these data streams is concerned with extracting knowledge structures represented in models and patterns in non-stopping streams of information [6], and the research on this area has gained a high attraction. In this proposal, we go a step further and propose a general purpose forecasting algorithm based on nearest neighbors for big volumes of streams of data, to create a method capable to be integrated in real-world systems in which data are constantly generated as streams, such as the demand prediction in the electricity market.

In this work, we propose the StreamWNN algorithm for streaming time series forecasting based on nearest neighbors. This algorithm consists of two phases: a batch phase to generate an initial model, and an online phase for forecasting in real time by using the model previously created in the batch phase. The proposal has been applied to a dataset of 497,832 samples of electrical energy consumption in Spain.

The rest of the paper is structured as follows. Section 2 describes a review of the state of the art approaches related to data streaming and forecasting analysis in the electricity market. Section 3 presents the methodology applied for time series forecasting in streaming. The experimental setup along with the results obtained using electricity demand time series can be found in Sect. 4. Finally, in Sect. 5, the final considerations extracted from this work are presented.

2 Related Works

A wide range of approaches for data streaming analysis is currently emerging. The primary trend of this research field is the development of machine learning methodologies to the streaming environments. From this perspective, the authors in [26] presented an online version of the support vector machine model to predict air pollutant levels from the monitored air pollutant in Hong Kong. An online version of the linear discriminant analysis algorithm for dimension reduction was presented in [14]. On the other hand, it has carried out research efforts to develop frameworks for adaptation of standard machine learning methods to streaming [10]. Another streaming framework is SAMOA presented in [1], where the authors developed an API to apply machine learning algorithms to streams of data in a big data context. Different algorithms to analyze data streams from the Internet of Things (IoT) networks are also currently being developed. In [5], the authors presented a streaming linear regression method to forecast streams data generated by IoT networks. Finally, several surveys have been published

about the streaming analysis. In this sense, the authors in [20] analyzed the difference between the real-time processing and the stream processing of big data, and by contrast, a survey of the open-source technologies that support big data in a real-time/near real-time environments was introduced in [15].

Concerning researches focused on forecasting for the electricity demand time series data, in addition to the nearest neighbors for a big data environment proposed in [22] and the new big data-based multivariate and multi-output forecasting approach in [21], other approaches have been published. In [7], the authors applied decision gradient boosted trees and random forest ensemble methods to the electricity demand problem. Also, deep learning techniques have been applied to predict energy power consumption in big data environments [23]. A Temporal Convolutional Network has been used in [11] for demand energy forecasting. A complete review of deep learning architectures for time series forecasting was published in [12]. On the other hand, several streaming techniques have been applied to this problem. In [3], the authors presented an incremental pattern characterization algorithm to mine data streams from smart meters of RMIT University for the purpose of applying it to electricity consumption analysis and forecasting. The authors in [8] proposed a complete data streaming analysis system combining an online clustering model and neural-networks to predict in real-time the electricity load demand from sensor networks.

Besides the forecasting, other problems related to energy have been addressed. The authors in [19] presented a methodology to extract electric energy consumption patterns in big data time series based on the application of the distributed version of the k-means algorithm. In [9] the authors presented a big data system to classify fraudulent behaviors of the leading electricity company in Spain. Regarding the streaming environment, an incremental ensemble learning method is developed for the on-line classification of the electricity pricing in Australia in [18]. Furthermore, in [27], the authors presented the DStreamEPK algorithm, a new streaming clustering method applied to electric power data.

3 Methodology

This Section presents the proposed algorithm, named StreamWNN, for streaming time series forecasting based on nearest neighbors.

The time series forecasting problem consists in predicting the next h values from the historical past values. The StreamWNN forecasting algorithm has of two phases: a batch phase to generate an initial model, and an online phase for forecasting in real time by using the model created in the batch phase.

A time series X_t is defined as a set of ordered chronologically values $\{x_1, \dots, x_t\}$ and can be always transformed into N instances formed by features and class as follows:

$$X_t = \{(x^1, y^1), \dots, (x^N, y^N)\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \quad (1)$$

where x^i are the features of the i -th instance, representing the past w values to the class y^i formed by the next h values. For the batch phase, the time series X_t

188 L. Melgar-García et al.

from Eq. (1) is divided into training set and test set. Then, the prediction method based on nearest neighbors searches for the k closest neighbors to a window composed of the past w values to the h values to be predicted. Afterwards, a weight is calculated for each neighbor depending on its distance to the past values window. Thus, the initial model M consists of the pairs of the features of the instances from the test set and a list of the classes corresponding to the neighbors of these features from the training set. That is:

$$M = \langle x^i, \langle y(n_1(x^i)), \dots, y(n_K(x^i)) \rangle \rangle \quad (2)$$

where K is the number of neighbors, x^i are the w features of the i -th instance of the test set, $n_j(x^i)$ is the j -th neighbor of the x^i and $y(n_j(x^i))$ is the class corresponding to the j -th neighbor.

When a time series is received in streaming, a temporal data stream ds_t can be a chunk of the time series of length w , that is, $ds_t = \langle x_t, x_{t+1}, \dots, x_{t+w-1} \rangle$. For the online phase, once the ds_t data stream is received, the nearest neighbor of the ds_t from test set is obtaining by this equation:

$$x^* = \arg \min_{x^i \in Test} d(x^i, ds_t) \quad (3)$$

Then, the prediction is obtained using the K neighbors of x^* and weights already computed in the M model from Eq. (2). In particular, the prediction is made by applying a weighted average of the h samples following those k closest neighbors. Thus, the StreamWNN algorithm predicts by means of the following equation:

$$\hat{y}(ds_t) = \frac{1}{\sum_{j=1}^K w_j^*} \sum_{j=1}^K w_j^* y(n_j(x^*)) \quad (4)$$

where $n_j(x^*)$ is the j -th neighbor of x^* , $y(n_j(x^*))$ is the class corresponding to the j -th neighbor, and w_j^* is the weight associated to the j -th neighbor. This weight depends on the distance, with a greater weight to the closest neighbors and a smaller weight to the farthest neighbors according to a distance d . In this work, the Euclidean distance has been chosen, and the weights are defined by:

$$w_j^* = \frac{1}{d^2(x^*, n_j(x^*))} \quad (5)$$

Consequently, it is possible to obtain forecasts in real time as the prediction consists of making an average with neighbors and weights previously computed in the batch phase using the historical data.

4 Experimental Results

This section specifies the dataset used in the experimentation and reports the results obtained after the application of the proposed streaming algorithm. In particular, Sect. 4.1 describes the dataset and the experiments carried out, specifying in each case the parameters of the algorithm. Finally, in Sect. 4.2 the results of the experimentation are shown and discussed.

4.1 Dataset and Experimental Setup

The experimentation uses a dataset of 497,832 samples of electrical energy consumption in Spain. Each sample has 12 attributes related to electricity. For this work, only two attributes are used: the energy demand in megawatt (MW) and the date and time of the measured value.

In particular, the dataset contains 1 sample for every 10 min during 9 years and 6 months, starting the 1 January 1st 2007 and finishing June 21st 2016. The whole dataset is chronologically divided into 3 sets of data: training, test and streaming sets. The training and test sets are approximately a 70% of the dataset: the training set contains data from January 1st 2007 to August 23rd 2011 and the test set contains data from August 24th 2011 to August 19th 2013. The algorithm predicts almost 3 years, i.e., the streaming set is from August 20th 2013 to June 21st 2016.

In this study, the experiments are carried out with the same parameters and prediction horizons established in [22]. Each of the four experiments has a different horizon: 4, 8, 12 and 24 hours. As the dataset contains 1 sample each 10 min, the prediction horizons are 24, 48, 72 and 144 samples, respectively. The goal is to analyze the behaviour of the algorithm for different prediction horizons considering the optimal parameters of [22].

The parameters for each experiment are listed below, where h is the prediction horizon, w corresponds to the number of past values used for predicting the next h values and K is the number of nearest neighbors of the training set to consider when creating the M model, as defined in Sect. 3:

- For the prediction horizon $h = 24$, optimal parameters are $w = 144$ and $K = 4$.
- For the prediction horizon $h = 48$, optimal parameters are $w = 288$ and $K = 2$.
- For the prediction horizon $h = 72$, optimal parameters are $w = 576$ and $K = 4$.
- For the prediction horizon $h = 144$, optimal parameters are $w = 864$ and $K = 4$.

4.2 Results

The four experiments are run on a cluster located at the Data Science and Big Data Laboratory in Pablo de Olavide University. The cluster is formed by 4 nodes: 3 slaves and 1 master. The whole cluster has 4 Processors Intel(R) Core(TM) i7-5820K CPU with 48 cores, 120 GB of RAM memory. It uses Ubuntu 16.04.1 LTS, Apache Spark 2.3.4, HDFS on Hadoop 2.7.7 and Apache Kafka 2.11.

The metrics used to evaluate the performance of the algorithm are the mean absolute percentage error (MAPE), expressed as a percentage, and the mean absolute error (MAE), expressed in MW [24]. Table 1 presents the above-mentioned metrics of error when forecasting the streaming set of data for the different prediction horizons. Moreover, the maximum, minimum and standard deviation (st. dev.) of the MAPE for the streaming set are depicted. It can be noticed that both MAPE and MAE increase with higher values of the prediction horizon. Considering that in this work the offline summary model is not updated,

190 L. Melgar-García et al.

the standard deviation and values of MAPE and MAE lead to think that the offline summary model represents in an accurate way the streaming data.

Table 1. Metrics of errors for different prediction horizons

h	w	k	Maximum MAPE	Minimum MAPE	St. dev. MAPE	MAPE	MAE
24	144	4	33.0031	0.2464	2.0745	2.4288	670.1298
48	288	2	31.2719	0.4101	2.0842	2.7617	766.8640
72	576	4	34.3861	0.6002	2.8199	3.3535	933.9924
144	864	4	29.3277	0.6548	3.6136	3.8465	1072.8357

Figures 1 and 2 show the worst forecasts (the maximum MAPE) and the best ones (the minimum MAPE) for each prediction horizon, respectively. They both show the real and forecasted electricity demand values in the vertical axis and the hours of the day in the horizontal axis. Each sub-figure includes the day (in format day/month/year) and the horizon of the maximum or minimum MAPE. All worst days correspond to public holidays in Spain: in summer for the prediction horizons 24 and 48 and, in winter for the prediction horizons 72 and 144. For prediction horizons 24, 48 and 72, it can be observed abrupt changes at the last time sample of the horizon as the following forecasted values correspond to the next prediction horizon on the same day. On the other hand,

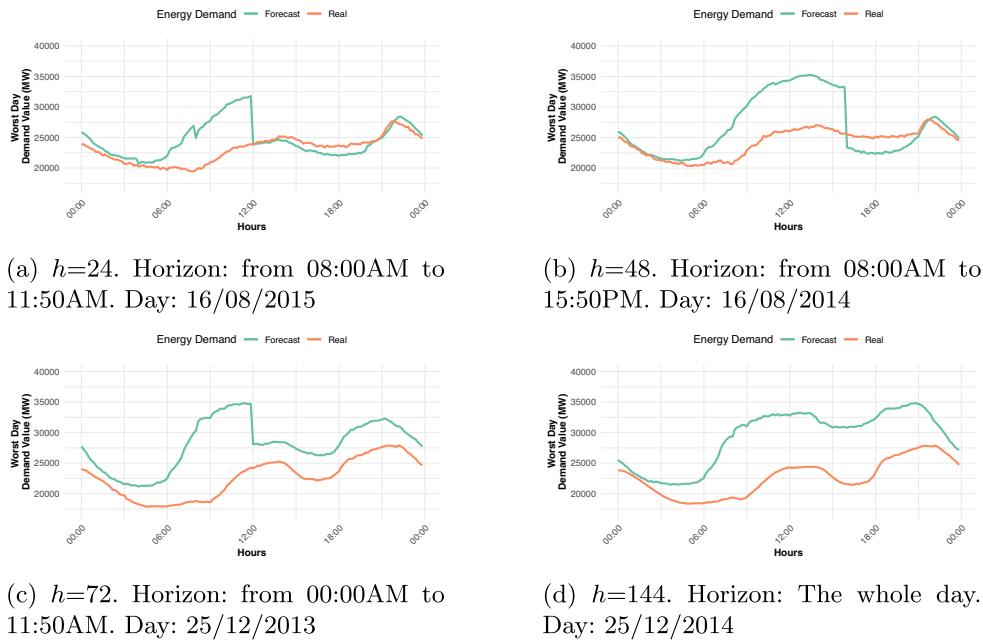


Fig. 1. Days with the worst forecasts for each h horizon

Fig. 2 shows that, in these days, the data used in the offline phase represents well the online data because even without any update of the summary offline model, the forecasted values are quite accurate.

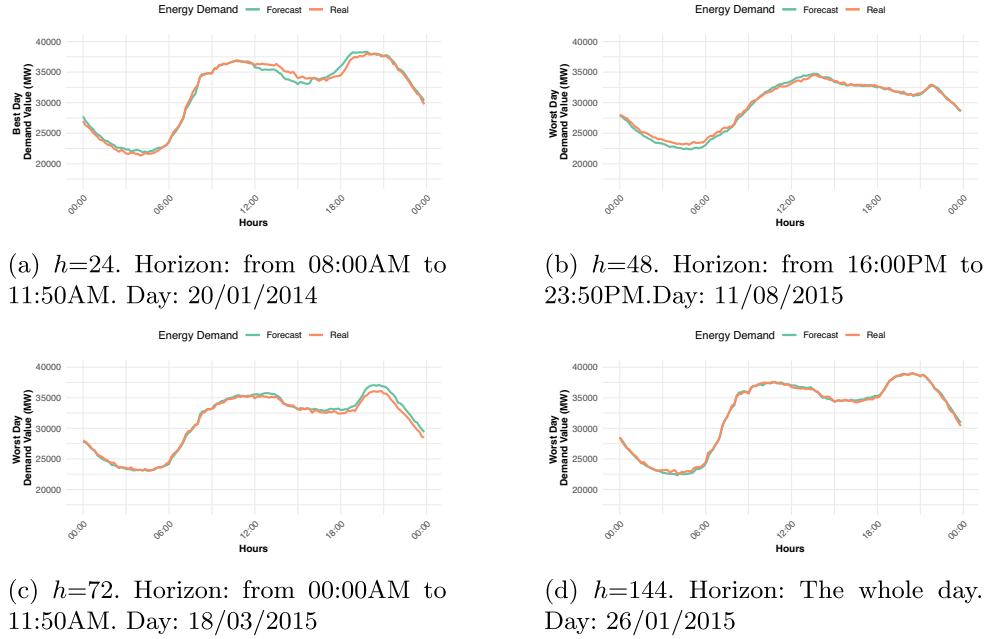


Fig. 2. Days with the best forecasts for each h horizon

Table 2 shows the MAE obtained when applying the algorithm recently published in [22] and the proposed StreamWNN algorithm using the same set of data and the same parameters for comparison purposes. It can be observed that the error of the proposed algorithm is higher just for $h = 24$. However, the MAEs of the StreamWNN are quite smaller than the ones in [22] for all the other prediction horizons.

Table 2. The MAE (in MW) for the StreamWNN and the algorithm in [22].

h	[22]	StreamWNN
24	524.14	670.13
48	920.87	766.86
72	1313.40	933.99
144	1514.92	1072.84

Figure 3 represents the mean values for each hour, both of the forecasted and of the real energy demand values of the $h = 24$ prediction horizon setup. The

192 L. Melgar-García et al.

representation of the other three forecast horizons is very similar. It confirms that the forecast results have behave very similar to the ones of the real data.

Besides the good performance, a streaming algorithm has to provide timely results during the online phase. Even if the offline phase of the streaming algorithm is not limited in execution time, the offline phase of the proposed algorithm is fast considering the huge amount of data, both in training and test sets. The offline phase of the proposed algorithm for $h = 24$ takes 222.09s, 167.16s for $h = 48$, 153.47s for $h = 72$ and 122.50s for $h = 144$.

The online execution time for all four prediction horizons is presented in Fig. 4. This figure shows for every 200 iterations of the algorithm, the time in seconds from the beginning of the online phase. The number of iterations for each experiment is different as w and h changes. In addition, as smaller these values are, less time is taken to compute the iterations (as in the offline phase). It can be observed that the algorithm increases linearly the execution time as more iterations have been previously made, which is very important in streaming algorithms. Considering these results, a forecast of h values is made in an average of 1.4s for $h = 24$, 1.6s for $h = 48$, 1.9s for $h = 72$ and 2.3s for $h = 144$. These results are presented in Table 3.

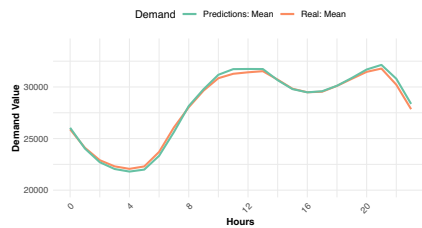


Fig. 3. Hourly average of the actual and forecasted energy demand

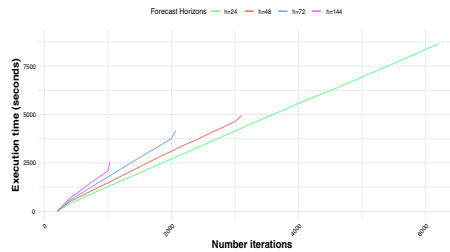


Fig. 4. Execution time of the online phase versus number of iterations

Table 3. Computation times (in seconds) for different prediction horizons

h	Offline phase time	Online prediction time of h values
24	222.09	1.4
48	167.16	1.6
72	153.47	1.9
144	122.50	2.3

5 Conclusions

The StreamWNN algorithm for time series forecasting in the streaming environment has been proposed. The StreamWNN consists of two stages: an offline or

batch phase and an online phase. The first stage creates a summary prediction model with the K nearest neighbors for each window of w samples and their next h samples of the training set. Afterwards, in the second stage, the time series of the streaming set are processed satisfying the streaming requirements. When streams arrive, the model predicts the h next values with a weighted average using the selected K nearest neighbor from the batch prediction model. The algorithm has been applied to an electricity demand time series dataset containing records over nine years. The performance of the algorithm has been evaluated with the MAPE and MAE error metrics for each prediction horizon. A good performance has been shown when comparing these errors with a benchmark algorithm, that used the same dataset and parameters.

The future works will be focused on some characteristics of the algorithm such as updating the summary batch model considering the knowledge of the previous time series streams, detecting novelties and outliers in the streams or studying the process to select the optimal values of the parameters.

Acknowledgements. The authors would like to thank the Spanish Ministry of Science, Innovation and Universities for the support under project TIN2017-88209-C2.

References

1. Bifet, A., Morales, G.F.: Big data stream learning with SAMOA. In: Proceedings of the IEEE International Conference on Data Mining Workshop (ICDM), pp. 1199–1202 (2015)
2. Cover, T., Hart, P.: Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **13**(1), 21–27 (1967)
3. De Silva, D., Yu, X., Alahakoon, D., Holmes, G.: Incremental pattern characterization learning and forecasting for electricity consumption using smart meters. In: Proceedings of the IEEE International Symposium on Industrial Electronics, pp. 807–812 (2011)
4. Dudani, S.A.: The distance-weighted k-nearest-neighbor rule. *IEEE Trans. Syst. Man Cybern.* **6**(4), 325–327 (1976)
5. Fernández, A.M., Gutiérrez-Avilés, D., Troncoso, A., Martínez-Álvarez, F.: Real-time big data analytics in smart cities from LoRa-based IoT networks. In: Martínez Álvarez, F., Troncoso Lora, A., Sáez Muñoz, J.A., Quintián, H., Corchado, E. (eds.) SOCO 2019. AISC, vol. 950, pp. 91–100. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-20055-8_9
6. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. *ACM SIGMOD Rec.* **34**(2), 18–26 (2005)
7. Galicia, A., Talavera-Llames, R., Troncoso, A., Koprinska, I., Martínez-Álvarez, F.: Multi-step forecasting for big data time series based on ensemble learning. *Knowl. Based Syst.* **163**, 830–841 (2019)
8. Gama, J., Rodrigues, P.P.: Stream-based electricity load forecast. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 446–453. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74976-9_45

194 L. Melgar-García et al.

9. Gutiérrez-Avilés, D., et al.: SmartFD: a real big data application for electrical fraud detection. In: de Cos Juez, F., et al. (eds.) HAIS 2018. LNCS, vol. 10870, pp. 120–130. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92639-1_11
10. He, H., Chen, S., Li, K., Xu, X.: Incremental learning from stream data. *IEEE Trans. Neural Networks* **22**(12), 1901–1914 (2011)
11. Lara-Benítez, P., Carranza-García, M., Luna-Romera, J.M., Riquelme, J.C.: Temporal convolutional networks applied to energy-related time series forecasting. *Appl. Sci.* **10**(7), 2322 (2020)
12. Lara-Benítez, P., Carranza-García, M., Riquelme, J.C.: An experimental review on deep learning architectures for time series forecasting. *Int. J. Neural Syst.* **31**(03), 2130001 (2021)
13. Li, Y., Li, D., Wang, S., Zhai, Y.: Incremental entropy-based clustering on categorical data streams with concept drift. *Knowl. Based Syst.* **59**, 33–47 (2014)
14. Liu, L.P., Jiang, Y., Zhou, Z.H.: Least square incremental linear discriminant analysis. In: Proceedings of the IEEE International Conference on Data Mining, pp. 298–306 (2009)
15. Liu, X., Iftikhar, N., Xie, X.: Survey of real-time processing systems for big data. In: Proceedings of the International Database Engineering and Applications Symposium, pp. 356–361 (2014)
16. Maillo, J., Ramírez, S., Triguero, I., Herrera, F.: kNN-IS: an iterative spark-based design of the k-nearest neighbors classifier for big data. *Knowl. Based Syst.* **117**, 3–15 (2017)
17. Martínez-Álvarez, F., Troncoso, A., Riquelme, J.C., Aguilar-Ruiz, J.S.: Energy time series forecasting based on pattern sequence similarity. *IEEE Trans. Knowl. Data Eng.* **23**(8), 1230–1243 (2010)
18. Ng, W.W.Y., Zhang, J., Lai, C.S., Pedrycz, W., Lai, L.L., Wang, X.: Cost-sensitive weighting and imbalance-reversed bagging for streaming imbalanced and concept drifting in electricity pricing classification. *IEEE Trans. Ind. Inform.* **15**(3), 1588–1597 (2019)
19. Pérez-Chacón, R., Luna-Romera, J.M., Troncoso, A., Martínez-Álvarez, F., Riquelme, J.C.: Big data analytics for discovering electricity consumption patterns in smart cities. *Energies* **11**(3), 683 (2018)
20. Shahrivari, S.: Beyond batch processing: towards real-time and streaming big data. *Computers* **3**(4), 117–129 (2014)
21. Talavera-Llames, R., Pérez-Chacón, R., Troncoso, A., Martínez-Álvarez, F.: MV-kWNN: a novel multivariate and multi-output weighted nearest neighbours algorithm for big data time series forecasting. *Neurocomputing* **353**, 56–73 (2019)
22. Talavera-Llames, R., Pérez-Chacón, R., Troncoso, A., Martínez-Álvarez, F.: Big data time series forecasting based on nearest neighbours distributed computing with spark. *Knowl. Based Syst.* **161**, 12–25 (2018)
23. Torres, J.F., Galicia, A., Troncoso, A., Martínez-Álvarez, F.: A scalable approach based on deep learning for big data time series forecasting. *Integr. Comput. Aided Eng.* **25**(4), 335–348 (2018)
24. Troncoso, A., Riquelme-Santos, J.M., Gómez-Expósito, A., Martínez-Ramos, J.L., Riquelme-Santos, J.C.: Electricity market price forecasting based on weighted nearest neighbors techniques. *IEEE Trans. Power Syst.* **22**(3), 1294–1301 (2007)
25. Troncoso, A., Riquelme, J.C., Aguilar-Ruiz, J.S., Riquelme-Santos, J.M.: Evolutionary techniques applied to the optimal short-term scheduling of the electrical energy production. *Eur. J. Oper. Res.* **185**(3), 1114–1127 (2008)

Nearest Neighbors-Based Forecasting for Electricity Time Series in Streaming 195

26. Wang, W., Men, C., Lu, W.: Online prediction model based on support vector machine. *Neurocomputing* **71**(4–6), 550–558 (2008)
27. Zhang, X., Qian, Z., Shen, S., Shi, J., Wang, S.: Streaming massive electric power data analysis based on spark streaming. In: Li, G., Yang, J., Gama, J., Natwichai, J., Tong, Y. (eds.) *DASFAA 2019. LNCS*, vol. 11448, pp. 200–212. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18590-9_14

5.1.8 | "A new big data triclustering approach for extracting three-dimensional patterns in precision agriculture"

Authors: Melgar-García L., Gutiérrez-Avilés D., Godinho M. T., Espada R., Brito I. S., Martínez-Álvarez F., Troncoso A., Rubio-Escudero C.

Publication type: Journal article.

Journal: Neurocomputing.

Year: 2022.

Volume: 500

Pages: 268-278

DOI: 10.1016/j.neucom.2021.06.101

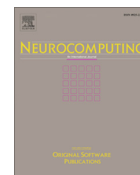
IF: 5.719 30/139 Computer Science, Artificial Intelligence.

Quartil: Q1.



Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A new big data triclustering approach for extracting three-dimensional patterns in precision agriculture



Laura Melgar-García^a, David Gutiérrez-Avilés^b, Maria Teresa Godinho^{c,d}, Rita Espada^e, Isabel Sofia Brito^{f,g}, Francisco Martínez-Álvarez^{a,*}, Alicia Troncoso^a, Cristina Rubio-Escudero^b

^a Data Science & Big Data Lab, Pablo de Olavide University, ES-41013 Seville, Spain

^b Department of Computer Science, University of Seville, Avda. Reina Mercedes s/n, Seville 41012, Spain

^c Department of Mathematical and Physical Sciences, Polytechnic Institute of Beja, Portugal

^d Center for Mathematics, Fundamental Applications and Operations Research, University of Lisboa, Portugal

^e Associação dos Agricultores do Baixo Alentejo, Beja, Portugal

^f Department of Engineering, Polytechnic Institute of Beja, Portugal

^g Instituto de Desenvolvimento de Novas Tecnologias – Centre of Technology and Systems, Lisboa, Portugal

ARTICLE INFO

Article history:

Received 28 February 2021

Revised 27 May 2021

Accepted 12 June 2021

Available online 25 May 2022

Keywords:

Big data triclustering

Precision agriculture

Spatio-temporal patterns

ABSTRACT

Precision agriculture focuses on the development of site-specific harvest considering the variability of each crop area. Vegetation indices allow the study and delineation of different characteristics of each field zone, generally invisible to the naked-eye. This paper introduces a new big data triclustering approach based on evolutionary algorithms. The algorithm shows its capability to discover three-dimensional patterns on the basis of vegetation indices from vine crops. Different vegetation indices have been tested to find different patterns in the crops. The results reported using a vineyard crop located in Portugal depicts four areas with different moisture stress particularities that can lead to changes in the management of the vineyard. Furthermore, scalability studies have been performed, showing that the proposed algorithm is suitable for dealing with big datasets.

© 2022 Published by Elsevier B.V.

1. Introduction

It is a well-established fact that the era of Big Data [1] has changed the way in which data are generated, stored and processed, to the extent that 90% of the data that exist in the world has been generated during the last years [2]. These vast amount of data can be difficult to understand or even to analyze, and therefore the need for techniques to process this information arises. In this sense, new tools have been developed under the title of Data Science [3].

One of the areas that benefits from these developments is Precision Agriculture (PA), that can be defined as the application of technologies and principles to manage spatial and temporal variability associated to all aspects of agricultural production for the purpose of improving crop performance and environmental quality [4]. It is a fact that shortage of natural resources endangers our future. Public awareness of these problems urges local authorities to intervene and impose tight regulations on human activity. In this environment, reconciling economic and environmental objec-

tives in our society it is mandatory. PA has an important role in the pursuit of such aspiration, as the techniques used in PA permit to adjust resource application to the needs of soil and crop as they vary in the field. In this way, specific-site management (that is the management of agricultural crops at a spatial scale smaller than the whole field) is a tool to control and reduce the amount of fertilizers, phytopharmaceuticals and water used on site, with both ecological and economic advantages. Indeed, being able to characterize how crops behave over time, extracting patterns and predicting changes is a requirement of utmost importance for understanding agro-ecosystems dynamics [5].

One of the major concerns associated to the shortage of natural resources is the enormous consumption of water associated to farming activities. Water is a scarce resource worldwide and this problem is particularly acute in the South of Europe, where the Alentejo (Portugal) and Andalusia (Spain) regions are located. Both regions are mainly agriculture-dependent and thus, farmers and local authorities are apprehensive about the future.

In this paper, a new algorithm, hereinafter called bigTriGen, is proposed to delineate management zones by measuring the variability of crop conditions within the field. For this purpose, bigTriGen analyzes time series of geo-referenced vegetation indices,

* Corresponding author.

E-mail address: fmalarv@upo.es (F. Martínez-Álvarez).

obtained from satellite imagery. Thus, the bigTriGen algorithm, based on the evolutionary strategy introduced in the TriGen algorithm [6], is a triclustering method capable to analyze a set of satellite images indexed over time in addition to the ability to analyze vast three-dimensional datasets in a big data environment. It has been applied to a vineyard crop located in Baixo Alentejo, Portugal, with different experimental datasets in order to test its scalability.

The rest of the paper is structured as follows. In Section 2, the recent and related works are reviewed. In Section 3 our proposal is described. In Section 4 the results obtained using the vineyard crop dataset are presented and discussed. Finally, in Section 5, the conclusions of this work and point directions for future works are presented.

2. Related works

Interest in precision agriculture methods applied to viticulture has had a tremendous growth in the last decade: at the research level, the number of papers published has increased from 20 in 2011 [7] to 517 hits in response to googling “vineyard precision agriculture” in Google Scholar, in spite of having restricted the search to the current year. In fact, generally, vineyards meet the three classical conditions that are required in order to site specific management methods to be justified: (1) significant spatial variability within field exists (2) the causes of this variability can be identified and measured, and (3) the information from these measurements can be used to modify crop-management practices to increase profit and quality and decrease environmental impact [8]. These three conditions define themselves three important lines of research that complement each other. This paper addresses the first one, that is, we aim at identifying areas within the field with different behaviors as to grape quality and productivity. This objective involves both gathering data and extracting information from data. In the following, some of the proposed methods to deal with these topics are reviewed.

Vineyards are often planted in irregular/steep terrains resulting in difficult and expensive direct inspection tasks for wine growers. Thus, discrete point sampling, which is the most traditional mean of data collection on soil conditions and/or plant growth and development, is very difficult to implement on this type of crop [8,9]. On the other hand, aerial remote sensors have proven to be very effective means of collecting data as they can provide, at a relatively low cost, a fairly detailed, spatially referenced measure of almost all the same features. Both satellite and airborne imaging systems, namely unmanned aerial vehicles (UAV), with multispectral and hyperspectral cameras have been used for gathering crop related data for a few decades. Several papers have reviewed the use of aerial remote sensors and compared the quality of the information extracted from both means in accessing vineyard variability [10–13]. Satellite imagery is affordable and easily available, but inferior with regard to resolution and more vulnerable to atmospheric interference. Nevertheless, although it is widely accepted that UAV imagery provides a more complete view of the field [10–12], it has also been shown that good correlation exists between Normalized Difference Vegetation Index (NDVI) data from Sentinel 2 and NDVI unfiltered data from UAV [13]. Additionally, [14] shows that by complementing aerial data with information gathered by ground-based sensors, high-resolution management zones can be delineated.

Rather comprehensive reviews on methods for data analysis in precision agriculture are presented in [15,16]. The identification of site-specific management zones is achieved mostly through clustering techniques. Twenty of those techniques are compared in [17]. The comparison was conducted with data obtained between 2010 and 2015 from three commercial agricultural fields cultivated

with soya bean and maize in Brazil. Then, the divisions suggested by the results of a one-way ANOVA performed on the yields were compared to the divisions obtained using the various algorithms. The results showed that 17 out of the 20 produced quite good results, although McQuitty's Method and Fanny were considered to be the best choices. [18] presents a smaller study on four unsupervised methods applied to vineyard canopy segmentation in three different scenarios, with both RGB (Red-Green-Blue) and NRG (Near Infrared-Red-Green) imagery. The k-means algorithm has proven to be the more stable over the identification in the orthomosaic and sub-regions regarding the RGB acquisitions, whereas the HSV-RGN algorithm is the more stable over the identification in the orthomosaic and sub-regions regarding the NRG acquisitions. Many other studies are available, where a given method is proposed to define management zones in vineyards, based in various characteristics of the crop (disease detection, berry composition and sanitary status under humid conditions, among others) but we are not aware of the existence of a wider recent comparison on that matter.

Clustering tools to determinate time space patterns in precision agriculture can also be found in the literature: the evapotranspiration of a Pinot noir commercial vineyard in California was characterize through the unsupervised fuzzy c-means algorithm in [19] and, in [20], NDVI spatio-temporal patterns were obtained for a corn field in the Alentejo, Portugal, by means of a triclustering methodology.

Triclustering methodology has become a very researched area in the last years [21]. Some algorithms are based on genetic operators as [6] that included different evaluation measures [22–25] or [26] which used COVID-19 propagation model to optimize multi-objective functions. The characteristic of mining spatio-temporal patterns can be applied to different study areas as: medical [27], seismic [28] or even in environmental sensors in online learning [29]. Regarding the big data characteristic, [30] introduced a new parallel batch algorithm based on k-means providing speed results. [31] presented a parallel and scalable validation model for simple clusters in big data using Apache Spark. However, there is still much research to conduct in the development of big data triclustering algorithms.

3. Methodology

In this section, the methodology in order to obtain triclusters that enclose patterns from crops images is presented. Firstly, the triclustering that models the problem is described in Section 3.1, and finally, the way in which triclustering is applied, that is, the bigTriGen algorithm is presented in Section 3.2.

3.1. Problem modeling: triclustering

The triclustering techniques emerge as an evolution of clustering techniques applied over three-dimensional (3D) datasets. Triclustering aims at obtaining a set of triclusters (3D clusters) from the input dataset, with the values of each tricluster representing a pattern of behavior.

To formalize the triclustering concepts, firstly a three-dimensional dataset D_{3D} composed of the three sets D^I , D^F and D^{TP} is defined as follows:

$$D_{3D} = \{D^I, D^F, D^{TP}\} \quad (1)$$

where $D^I = \{i_1, i_2, \dots, i_I\}$ represents the I instances, $D^F = \{f_1, f_2, \dots, f_F\}$ the F features and $D^{TP} = \{t_1, t_2, \dots, t_{TP}\}$ the TP time points of the dataset.

Each pair instance-feature of the dataset represents a time series D_{TS} , that is, a sequence of time-indexed values from the time instant t_1 to t_{TP} as shown in the following equation:

$$D_{TS}(i, f) = \{v_{t_1}, v_{t_2}, \dots, v_{t_{TP}}\}, \quad \forall i \in D^I, \quad \forall f \in D^F, \quad \forall t \in D^{TP} \quad (2)$$

In conclusion, D_{3D} is typically arranged as a data cube where the rows are the instances, the columns are the features and the depths are the time points of the time series.

Secondly, a tricluster T is a subset of instances T^I , features T^F and time points T^{TP} of D_{3D} defined by the following equation:

$$T = \{T^I, T^F, T^{TP}\} \text{ with } T^I \subset D^I, \quad T^F \subset D^F, \quad T^{TP} \subset D^{TP} \quad (3)$$

The time points in T^{TP} for a particular instance and feature of the tricluster make up a continuous and ordered sub-sequence of values of the entire sequence of the dataset, that is, a time series T_{TS} from initial time t_s (first tricluster time point) to final time t_{TS} (last tricluster time point) defined as follows:

$$T_{TS}(i, f) = \{v_{t_s}, v_{t_{s+1}}, \dots, v_{t_{TS}}\}, \quad \forall i \in T^I, \quad \forall f \in T^F, \quad \forall t \in T^{TP} \quad (4)$$

Thus, the behavior patterns (BP) depicted by each time series of the tricluster will present similar behavior regarding the values or tendency. To summarize, a tricluster is a subset of instances, features, and time points of a three-dimensional dataset, with time series that depict a similar behavior pattern.

$$BP(T_{TS}(i_A, f_A)) \sim BP(T_{TS}(i_B, f_B)), \quad \forall i_A, i_B \in T^I, \quad \forall f_A, f_B \in T^F \quad (5)$$

Finally, a triclustering model of a three-dimensional dataset, $M_{D_{3D}}$, is a set of N triclusters defined as:

$$M_{D_{3D}} = \{T_1, T_2, \dots, T_N\} \quad (6)$$

3.2. The bigTriGen algorithm

bigTriGen is applied to obtain a triclustering model providing a set of behavior patterns from the input dataset. bigTriGen is based on the paradigm of genetic algorithms. In that sense, a complete evolutionary process is performed for each tricluster to be obtained, i.e., T_1, T_2, \dots, T_N . First, each evolutionary process applies the genetic operators described in Section 3.2.1 over a population of individuals. Then, the process presented in Section 3.2.2 makes the population evolve based on the optimization of a fitness function during a specific number of generations.

bigTriGen receives a three-dimensional dataset, D_{3D} , as an input. Each slice of time represents an image of a crop, where each pixel (x, y) is a space point representing the value of a particular vegetation index collected at a given instant $t_i \in \{t_1, t_2, \dots, t_{TP}\}$. Therefore, the D^I set corresponds to the X coordinates of the image and the D^F set to the Y coordinates. Fig. 1a shows the NDVI index represented on the images. That is, the point (200, 81) at t_1 is the NDVI value of the pixel in the row 81 and column 200 at the time instant t_1 .

An individual of the evolutionary process corresponds to a tricluster, T_i , being a particular area from the whole input space at a given time window. Thus, T_i is a subset of X and Y coordinates and a continuous subset of time points. Fig. 1b shows an individual represented by the subspace limited by the coordinates $Y = \{87, 88, 89, 90, 91, 92, 93, 94\}$ and $X = \{200, 201, 202, 203, 204\}$, and containing the index values for the time series from t_8 to t_{11} .

The output of the bigTriGen algorithm is a set of triclusters that correspond to a triclustering model $M_{D_{3D}}$ of the input dataset D_{3D} . Each tricluster of this model is a sub-area of an original image of the input dataset, as shown in Fig. 1d. Fig. 1c depicts the behavior patterns for each of the time series that make up a tricluster. Each

(x, y) point corresponds to a time series of the specific vegetation index.

Therefore, the aim of the bigTriGen algorithm is to discover a triclustering model, $M_{D_{3D}}$, from a three-dimensional dataset, D_{3D} , where each tricluster determines a sub-area of the original image of the dataset and the time series associated to the tricluster present patterns with similar behavior.

3.2.1. Genetic operators

Several updates have been carried out in bigTriGen with respect to TriGen to deal with satellite imagery related to the precision agriculture. These updates are mainly focused on the genetic operators, which are described below.

Initial population In this phase, the initial individuals of the populations are built. A subset of X and Y coordinates are randomly selected from the input dataset. The (x, y) points resulting from the combination of both subsets is a subspace of the original image of the input dataset. Each new individual's time points are randomly selected from the input dataset, forming a continuous sequence. The number of individuals built in the initial population is determined by the control parameter ln .

Selection A tournament algorithm is chosen for this operator. The individuals of the population are firstly separated into three groups, then they are ordered by fitness. A percentage of the population is selected from these three ordered groups. These selected individuals are directly promoted to the next generation and will be the parents suitable for reproduction by applying the crossover operator. The percentage of selected individuals is defined by the control parameter Sel .

Crossover Two individuals are randomly chosen for the reproduction from the individuals selected by the selection operator. From these two parent individuals P_1 and P_2 , two new children CH_1 and CH_2 will be obtained as shown in Eqs. (7) and (8). The first new individual CH_1 is composed of the X coordinates of the P_1 , the Y coordinates of the P_2 , and the time points of the P_1 . The second one CH_2 is composed of the X coordinates of the P_2 , the Y coordinates of the P_1 , and the time points of P_2 .

$$P_1 = \{P_1^X, P_1^Y, P_1^{TP}\} \quad \text{and} \quad P_2 = \{P_2^X, P_2^Y, P_2^{TP}\} \quad (7)$$

$$CH_1 = \{P_1^X, P_2^Y, P_1^{TP}\} \quad \text{and} \quad CH_2 = \{P_2^X, P_1^Y, P_2^{TP}\} \quad (8)$$

The number of children to obtain is $ln - (ln \times Sel)$, considering Sel as a percentage of selected parents. In order to get them, num_{cross} crossovers are made, where num_{cross} is the quotient plus the remainder of the division by two of the number of children to obtain. As previously mentioned, from one crossover, two children are obtained. Once all crossovers are computed, the specified number of children are selected from all children considering the best fitness function.

Mutation The new individuals obtained by means of the crossover operator are eligible to be altered by mutation. An individual can be altered by removing or adding a random X or Y coordinate or a random time point. The probability of mutation of an individual is set by the control parameter Mut . The operations are controlled by specific parameters referring to maximum and minimum number of coordinates that an individual must have.

3.2.2. Fitness function

As a genetic algorithm, the core of bigTriGen is the fitness function to be optimized. In this work, a fitness function based on the MSL measure [23] has been used. MSL measures the similarity of the behavior patterns contained in a tricluster and it is based on the differences between the angles that every two points of a series form with the X -axis (the slope of a straight line). Thus, this algorithm provides an accurate measure of how similar the behavior

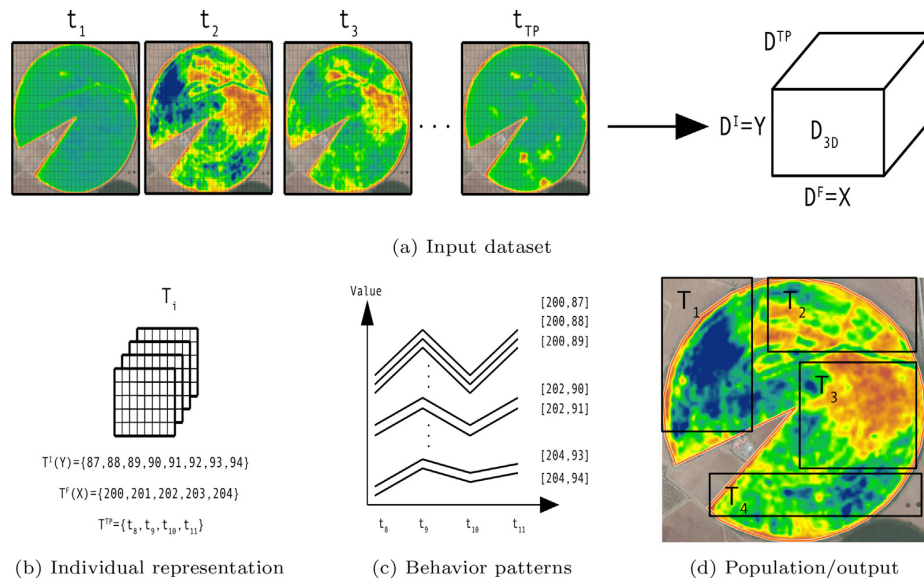


Fig. 1. TriGen overview.

patterns inside a tricluster are. The MSL measure is widely explained and discussed in [23]. Moreover, the fitness function includes a control mechanism to balance the size of the triclusters and the overlapping among them.

The fitness function of a tricluster T is defined by a weighted average as follows:

$$\text{Fitness}(T) = \frac{w_{msl} * \text{MSL}(T) + w_s * S(T^X, T^Y) + w_o * O(T, M_{D_{3D}})}{w_{msl} * w_s * w_o} \quad (9)$$

where $\text{MSL}(T)$ is the MSL index of the tricluster T , $S(T^X, T^Y)$ is the size of the area demarcated by the X and Y coordinates of the tricluster T , $O(T, M_{D_{3D}})$ is the overlapping degree of the tricluster T with the remaining triclusters of the model $M_{D_{3D}}$, and w_{msl} , w_s and w_o are the weights of each component, respectively [23].

3.3. Big data implementation remarks

The bigTriGen algorithm has been developed in a big data environment to provide it with the ability to analyze big three-dimensional datasets. Therefore, a model with bigger triclusters (more X and Y coordinates and time points) will be discovered from datasets with more significant time points and/or more significant areas (X, Y). bigTriGen has been implemented in Scala 2.12 [32] with Apache Spark 2.3.4 [33]. Its implementation is based on the DataFrame object of Apache Spark. The main feature of this data structure is to be distributed through the nodes of the cluster where the application is deployed [34].

For the bigTriGen algorithm, the input dataset D_{3D} is loaded into a DataFrame, where each row represents a point (instance, feature, time) and its associated value.

The population is also implemented using a DataFrame. An example of the structure can be found in Table 1. In this case, each row represents a time series for the particular coordinates (y, x) of a tricluster individual. Therefore, a row will be composed of an individual numerical identifier (IND_{id}), a time series identifier (TS_{id}), the associated Y and X coordinates, the time point list (TPs) and, the time series values (TS). The justification of these implementation decisions is due to two aspects, both related to the application of the Spark DataFrame API actions and transforma-

tions to the population. On the one hand, this structure leads the Spark's actions and transformations to execute the genetic operators in a best-optimized way in a big data environment. On the other hand, this structure boosts the application of the Spark DataFrame API actions and transformations to the population and, therefore, maximizes the distribution of it through the nodes of the Spark cluster where bigTriGen was deployed. In conclusion, with this implementation, the bigTriGen algorithm's scalability, regarding the execution time against the size of the input dataset, is reached. As explained in the above paragraphs, bigTriGen is a novel algorithm with an own design and implementation. A summary of the new features of the bigTriGen is shown in Table 2 where it can be confirmed that bigTriGen differs in the implementation, characteristics and results comparing with TriGen. The original TriGen and the new bigTriGen keep the control parameters of the algorithm, the evolutionary work-flow, and the selection operator in common. In contrast, as discussed above, the bigTriGen allows for the analysis of input datasets and triclusters with sizes impossible to manage on a single machine. Furthermore, it adds the space and time series modeling (presented in Section 3.2) and, therefore, new initial population, crossover and, mutation operators. A detailed description of the original TriGen algorithm can be found in [6,23].

3.4. Validation of the triclusters

In this work, the triclusters of the model $M_{D_{3D}}$ will be validated in three ways. Firstly, the TRIQ quality measure [24] that provides

Table 1
DataFrame example for the population.

IND_{id}	TS_{id}	Y	X	TPs	TS
1	0	2	3	{2, 3, 4, 5}	{0.05, 0.58, 0.23, 0.22}
1	1	2	4	{2, 3, 4, 5}	{0.15, 1.82, 0.38, 0.25}
1	2	3	3	{2, 3, 4, 5}	{0.54, 2.84, 1.25, 0.15}
1	3	3	4	{2, 3, 4, 5}	{0.23, 0.38, 2.23, 1.01}
2	0	20	21	{19, 20, 21, 22, 23}	{0.08, 0.81, 0.09, 0.12, 2.24}
2	1	20	22	{19, 20, 21, 22, 23}	{0.01, 1.12, 0.01, 0.09, 1.25}
2	2	21	21	{19, 20, 21, 22, 23}	{0.02, 1.20, 0.02, 0.14, 3.12}
2	3	21	22	{19, 20, 21, 22, 23}	{0.03, 1.25, 0.25, 0.15, 5.02}

Table 2
Similarities and differences between TriGen and bigTriGen.

Common features	New features of bigTriGen
Control parameters	Bigger input datasets Bigger triclusters
Evolutionary process	(X, Y) space modeling for instances and features Time series modeling (consecutive instant points)
Selection operator	Initial population operator Crossover operator Mutation operator

an index to determine the similarity of the patterns of the triclusters and the correlation level of the time series associated to the tricluster will be used. In particular, TRIQ combines weighted Pearson and Spearman correlation values with a weighted normalization of MSL angle value. It has been shown as a valid measure for representing and summarizing the quality of the triclusters.

Secondly, a visual analysis of the discovered patterns will be carried out. This analysis will determine the coherence of the discovered triclusters in relation to the input dataset. The time series plots will be graphed, and the average of the time series will assess the cohesion of the values of the tricluster. Furthermore, an analysis of the behavior observed will be performed by an expert.

Finally, a global study of the located areas will be also made. The demarcated areas for each tricluster of the model will be also analyzed by an expert. That is necessary to determine any intra-

relation between the selected zones for the different triclusters of the model.

4. Results and discussion

This section reports the analysis of the results obtained by the bigTriGen algorithm when applying to a crop image dataset. In particular, the dataset and the vegetation indices typically used in the crops are described in Section 4.1, the experimentation process is explained in Section 4.2, the discussion of the patterns is presented in Section 4.3 and the scalability analysis to show the ability of the proposed algorithm to deal with big data is in Section 4.4.

4.1. Dataset and vegetation indices

The bigTriGen algorithm is tested in a vineyard crop located in Baixo Alentejo, in Portugal. The study area has 5.15 hectares and its center at the coordinates 37°56'43.62"N 7°52'15.06"W. In particular, the field is monitored during three years (2018, 2019 and 2020) selecting the months that correspond to vineyard season. Data is extracted from Sentinel-2 imagery with high spatial resolution at the defined coordinates using the QGIS software and its Semi-Automatic Classification Plugin. The calculation of the vegetation indices of each image is also made with this software.

Vegetation indices allow the quantitative and qualitative evaluation of different measures of crops, as cover, vigor, growth, type or

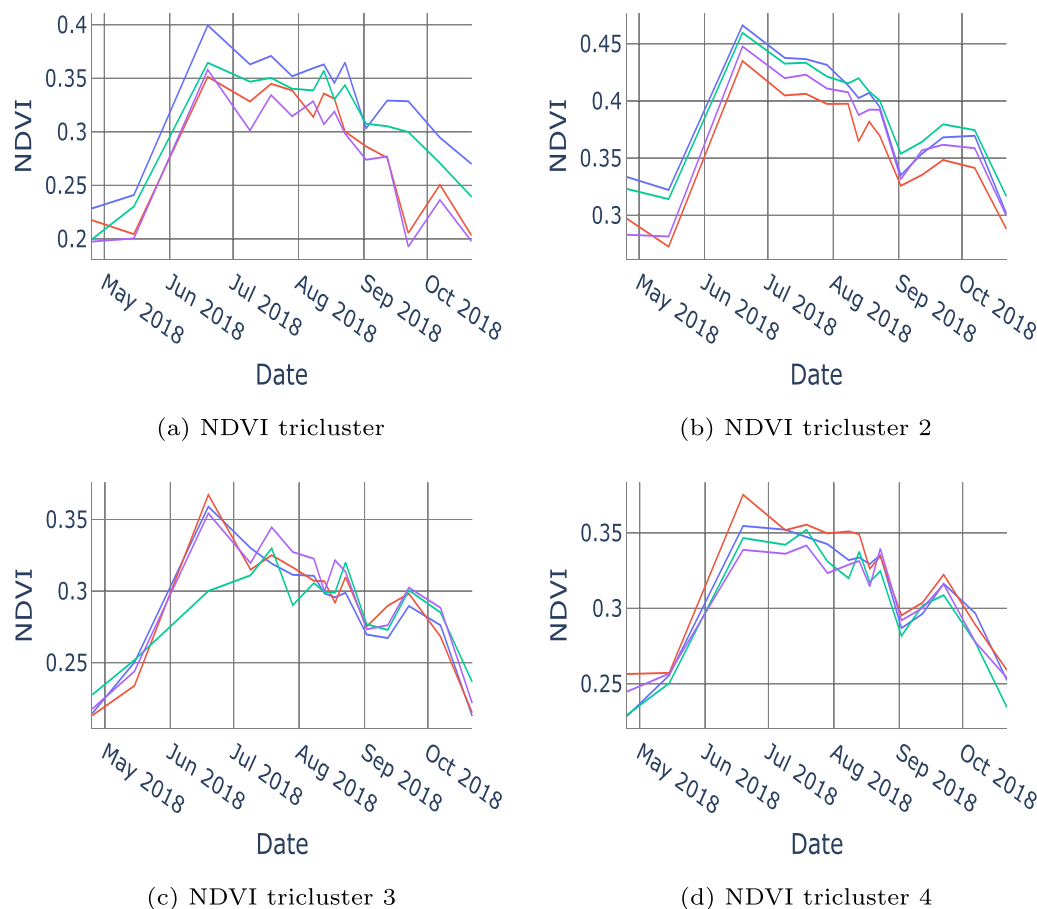


Fig. 2. Triclusters using the NDVI index for the vineyard crop.

quality. They are based on the measured canopy reflectance of different wavelength bands [35]. This canopy reflectance can be detected remotely using satellite imagery as the one provided by Sentinel-2. In this particular study, measuring leads to monitor fruit ripening to develop a site-specific harvesting of each zone of the vineyard crop; it is known as Precision Agriculture or more specifically in this case, Precision Viticulture [36].

One of the most used vegetation indices is the NDVI index. This index is very related to the content of the vegetation and varies from 1.0 to -1.0 , where 1.0 corresponds to the denser and healthier areas. NDVI includes in its calculation the near-infrared band (NIR) and the band for the red (visible) regions (Red). NDVI formula is $NDVI = \frac{NIR - Red}{NIR + Red}$. NDVI is a very useful index, for example, to determine areas of a corn crop that behaves differently [20].

Other indices used in this study that improve NDVI are the Soil-Adjusted Vegetation Index (SAVI) and the Enhanced Vegetation Index (EVI) used to determine grapevine phenology in [37]. The first one introduces L as a correction factor for soil brightness and the second one adds two C_1 and C_2 coefficients to the atmospheric resistance and the Blue band, respectively. SAVI is defined as $SAVI = \frac{NIR - Red}{NIR + Red + L} \times (1 + L)$ and EVI as $EVI = 2.5 \times \frac{NIR - Red}{NIR + C_1 \times Red - C_2 \times Blue + L}$. In this study, L is 0.5, C_1 is 6 and C_2 is 7.5; they are used values for this kind of crop.

The Moisture Stress Index (MSI) and the Green Normalized Difference Vegetation Index (GNDVI) include two different bands:

Green and middle-infrared (MIR), respectively. GNDVI is sensible to the variation of chlorophyll in the crop. On its side, MSI is used to analyze the water stress and it usually varies from 0.4 to 2 where higher values mean higher water stress and so, less soil moisture. Both indices are, as the above-mentioned ones, very studied in vine crops. For example, [38] concludes their vineyard crop study identifying the MSI as the only vegetation index directly related to the content of the vegetation. GNDVI is represented as $GNDVI = \frac{NIR - Green}{NIR + Green}$ and MSI as $MSI = \frac{MIR}{NIR}$.

4.2. Experimental setup

The experiments are run on a cluster located at the Data Science and Big Data Laboratory in Pablo de Olavide University. The cluster is made up of four nodes: one master and three slaves. It has four Processors Intel(R) Core (TM) i7-5820 K CPU with 48 cores, 120 GB of RAM memory. The cluster uses Ubuntu 16.04 LTS, Apache Spark 2.3.4 and HDFS file system on Hadoop 2.7.7. The bigTriGen algorithm is implemented in Scala programming language.

Considering the work published in [20] that discovered three dimensional patterns in a maize plantation area in Baixo Alentejo and after several experimental tests with the bigTriGen algorithm, the selected control parameters for the experimentation are: $N = 4$, $G = 10$, $ln = 200$, $Sel = 0.8$ and $Mut = 0.1$. The fitness function used

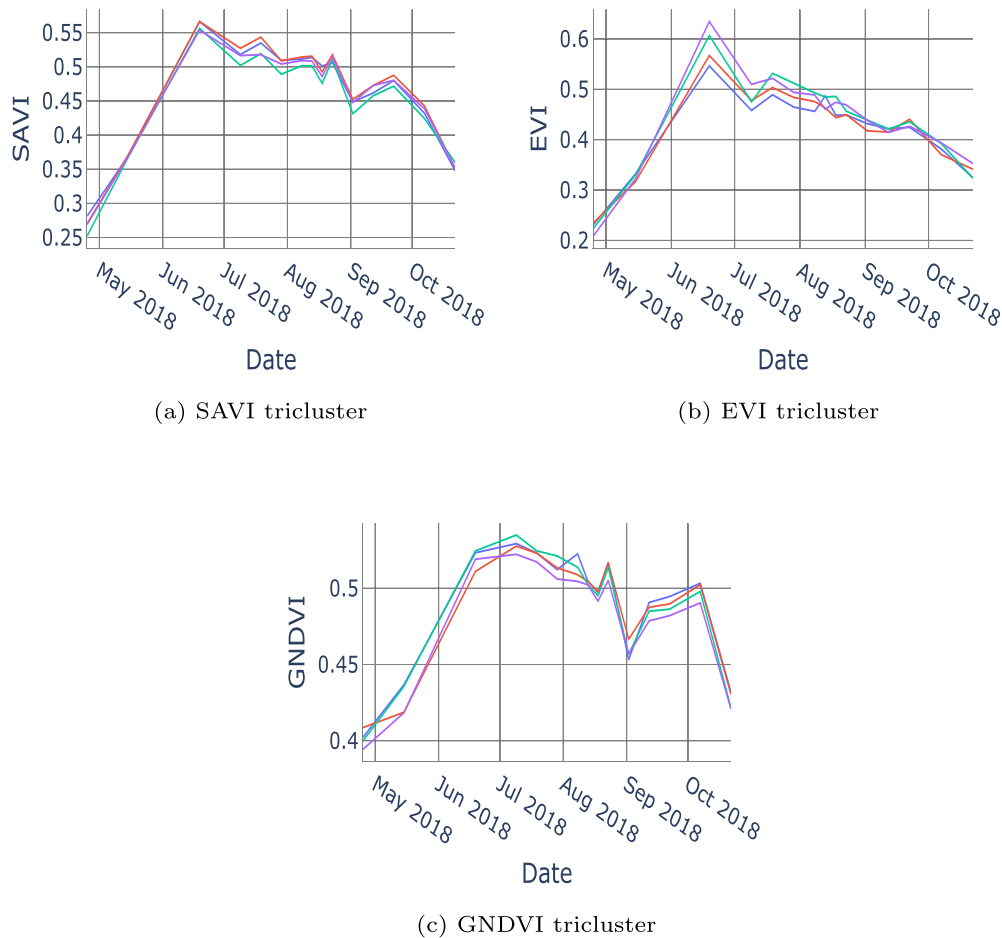


Fig. 3. Triclusters using SAVI, EVI and GNDVI for the vineyard crop.

is described in Section 3.2.2 and the validation of the triclusters is made considering the remarks in Section 3.4.

4.3. Pattern discovery

The process of discovering three-dimensional patterns using the proposed algorithm is performed from two points of view: spatial and temporal.

4.3.1. Spatial patterns

The goal of this analysis is to find behavior patterns that identify spatial zones on the vineyard crop with different characteristics. This analysis is carried out for the 2018 growing season.

Fig. 2 represents the triclusters found by the bigTriGen algorithm using the NDVI index. It shows a great uniformity between all the areas of each sub-figure, as the discovered patterns show very similar behavior curves. In order to confirm this uniformity, more vegetation indices that consider corrections of the NDVI and more environmental factors are used.

Fig. 3 depicts the behavior patterns of the field with SAVI, EVI and GNDVI indices during 2018. The analyses carried out using these indices confirm the assessment made with the NDVI index, i.e., triclusters curves represent a uniform behavior for vegetative growth and development of the crop throughout its extension.

To further study the water stress to which the crop is subjected, an additional analysis is carried out with the MSI index, which introduces *MIR* band in its calculation. Areas of the crop with different trends in the value of water stress are identified, although

this stress does not imply effects on the crop that are perceptible when the rest of indices are used.

Fig. 4 illustrates the different behavior patterns of the four triclusters obtained by the proposed bigTriGen algorithm when using the MSI. Fig. 4a represents, unlike the other three, an area with higher soil moisture during the initial period of the growing season. The trend is similar in the other behavior patterns, tending towards an increase in water stress as the growing season progresses. However, in the final phase, close to harvest time, is where the greatest differences among the different triclusters identified can be seen. While water stress is maintained in the area represented in Fig. 4a, a clear increase in stress is observed in Fig. 4c and d, in contrast to an increase in soil moisture in Fig. 4b, considering that lower MSI values correspond to lower water stress and so, higher soil moisture or water content. The quality of the found triclusters has been measured with the TRIQ measure described in Section 3.4. For values that move in the [0–1] interval, the first tricluster has a TRIQ value of 0.8799, the second of 0.9365, the third of 0.9153 and the fourth of 0.8321, thus ensuring accurate patterns for all cases.

This information may be relevant for the analysis of productivity in each field zone. It is necessary to identify the causes of the different behavior in order to determine whether they are due to productive factors (irrigation inequality, pests, etc.) or to specific factors of the terrain (inclines, type and quality of the terrain, etc.).

Fig. 5 identifies the geographic areas in the field map represented by the found triclusters when using the MSI index.

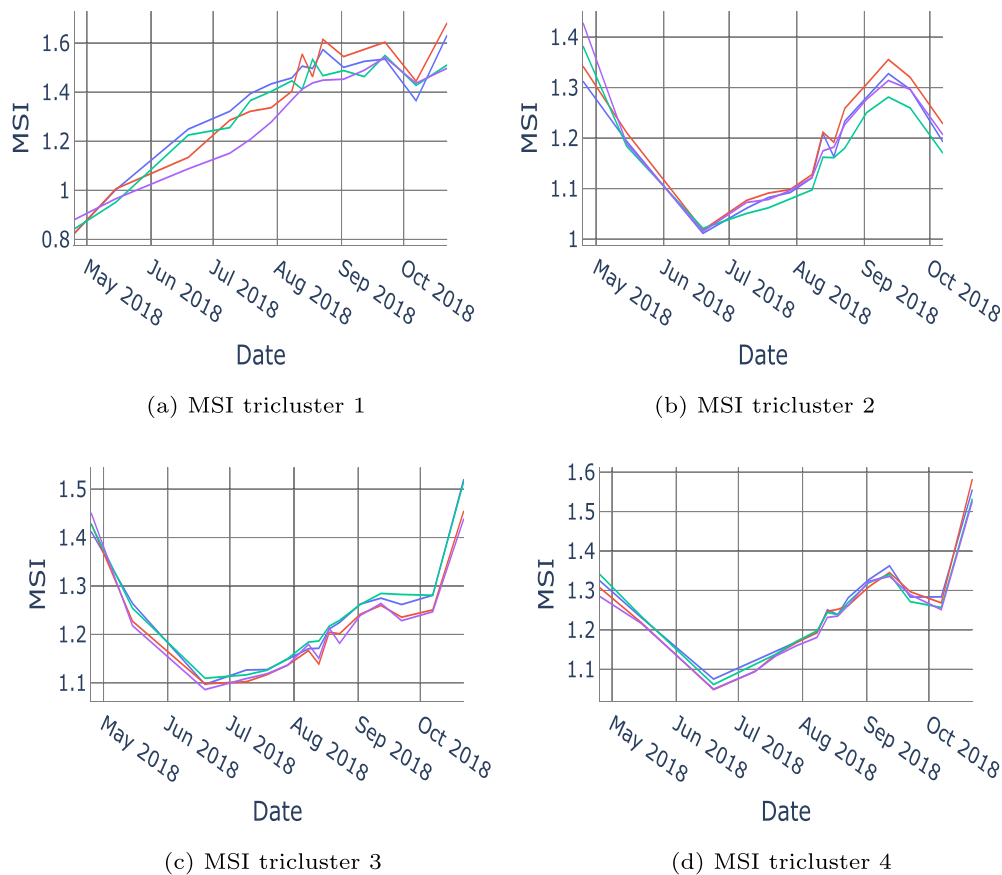


Fig. 4. Triclusters using the MSI for the vineyard crop.



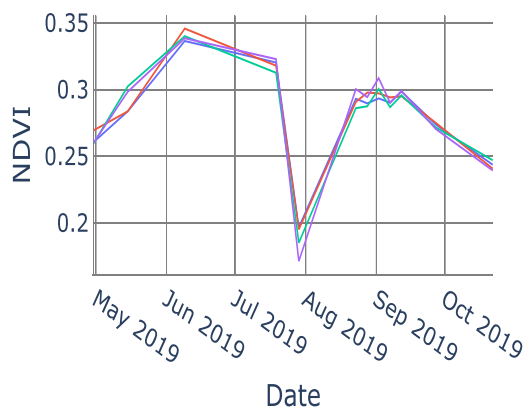
Fig. 5. Geographic location of the triclusters using the MSI index in the vineyard crop.

4.3.2. Temporal patterns

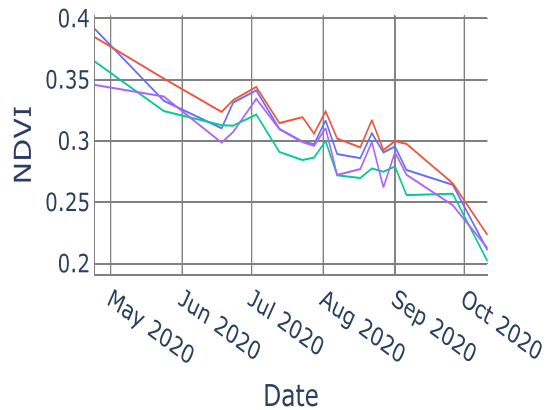
In this Section, the search of different behavior patterns between different growing seasons is considered. Years 2018, 2019 and 2020 have been analyzed, in particular, the analyses were limited to the months between May and November, which correspond to the vine growing season in its different phases.

The results obtained from the determination of triclusters for the NDVI data in the different growing seasons confirm the uniformity of the patterns in terms of crop behavior, i.e., patterns identified each year are very similar to the others found in the same year. However, there is a clear difference between one year and another. The patterns of 2018 are in Fig. 2 and a representation of the ones of 2019 and 2020 in Fig. 6.

The patterns of the last months of the vineyard period for the years 2018 and 2020 are very similar and correspond to the theory of what the NDVI trend should be over the course of a growing season. Nevertheless, the triclusters for 2019 show a different behavior during the month of August. In that period, the crop suffered a drop in NDVI index indicating a loss of quality of the plantation, which has managed to recover in the following months. This incidence coincides with the period of severe forest fires in the area where the field is located. It is very likely that this is the cause of the temporary deterioration of the crop.



(a) NDVI tricluster in 2019



(b) NDVI tricluster in 2020

Fig. 6. Triclusters using the NDVI in the years 2019 and 2020.

The bigTriGen algorithm demonstrates with this analysis that it is suitable for discovering anomalous behavior in a temporal sequence of historical events. Its use with current data can be a good tool to detect indications of anomalies at an early stage, even not perceptible to the naked eye in the crop, allowing corrective measures to be taken as soon as possible to mitigate the effects of these occurrences.

4.4. Scalability analysis

Once the found patterns have been evaluated, the next step is to study the scalability of the proposed bigTriGen algorithm. The evolution of the execution times is analyzed in two parts: in the first one, considering the effect of the number of nodes used and in the second one, considering the influence of the size of the dataset used. These tests are executed with a base dataset with the same characteristics as the one defined in Section 4.1 and the optimal parameters described in Section 4.2.

First, the scalability in terms of resources is analyzed by changing the number of nodes used when executing the bigTriGen algorithm. As explained in Section 4.2, the cluster used is made up of four nodes with twelve cores in each node. This analysis is made with 12 cores, 24 cores, 36 cores and 48 cores.

To analyze the effects of the dataset size, a base dataset of a size of 65 MiB has been used. This characteristic of the scalability analysis is studied by multiplying the length of the base dataset by 1, 2, 4, 6, 16 and 32. It corresponds to six experiments with datasets of 65 MiB, 130 MiB, 260 MiB, 520 MiB, 1040 MiB and 2080 MiB, respectively.

Results of twenty-four scalability experiments are shown in Table 3 and in Fig. 7, where the execution times are presented in

Table 3

Execution times (in minutes) according to number of cores and size of datasets.

Multiplier	12 cores	24 cores	36 cores	48 cores
x 1	19.7759	19.6475	20.3361	20.8268
x 2	25.1146	26.7812	25.9973	26.5552
x 4	42.2082	39.5204	38.5376	40.4623
x 8	76.0349	68.0430	69.9858	64.0017
x 16	141.9072	120.7762	120.9260	125.3108
x 32	278.6696	270.0996	232.3536	239.3660

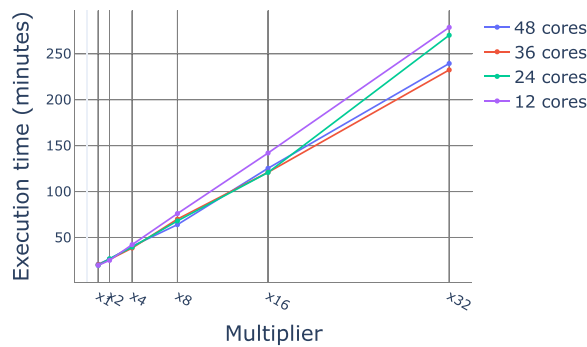


Fig. 7. Scalability analysis.

minutes. The computing time is not very influenced by the number of cores used when the size of the dataset is small, i.e., x1, x2, x4 or x8, but when the size increases, the execution time is smaller for 36 and 48 cores.

The behavior of the bigTriGen leads to express its scalability factor as $Factor_i = \frac{size_i}{size_1}$, where $size$ represents the dataset size and i varies from 2 to 32. This factor is usually smaller than 2 which is better than linear scalability. It is important to considering that the bigTriGen algorithm is influenced by chance, for example by means of the mutation operation, among others. However, in order to get a comparable scalability analysis, these operators have been controlled.

5. Conclusions

In this paper the new bigTriGen triclustering algorithm has been introduced to mine three-dimensional patterns from big datasets. In particular, this algorithm has used specific genetic operators to find triclusters in addition to control the overlapping with the previously found tricluster solutions. The bigTriGen has been applied to a vineyard crop in southern Portugal to find a precision viticulture solution. The accuracy of the algorithm has been shown with respect to two different features: the quality measure of the found patterns and the scalability of the algorithm. On the one hand, different vegetation indices have been calculated using Sentinel-2 images downloaded from QGIS software. The found patterns using these vegetation indices have shown that the index that best fits this field is the MSI. In this way, the algorithm has been able to find four different areas of the vineyard crop that behave differently in terms of their soil moisture. In addition, the algorithm has found different behaviors of the crop during 2018, 2019 and 2020. On the other hand, the scalability of the algorithm has been studied considering the number of nodes used and the size of the dataset. In both cases, the scalability factor of the bigTriGen has been proven to be even better than linear scalability.

The future works will be focused on developing more characteristics of the algorithm such as detecting anomalies in streams, creating methods to select the optimal values of the parameters or using other type of data.

CRedit authorship contribution statement

Laura Melgar-García: Validation. **David Gutiérrez-Avilés:** Conceptualization, Methodology. **Maria Teresa Godinho:** Writing – review & editing. **Rita Espada:** Data curation, Validation. **Isabel Sofia Brito:** Visualization, Investigation. **Francisco Martínez-Álvarez:** Supervision, Investigation. **Alicia Troncoso:**

Supervision. **Cristina Rubio-Escudero:** Conceptualization, Methodology.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors would like to thank the Spanish Ministry of Science and Innovation for the support under the project PID2020-117954RB and the European Regional Development Fund and Junta de Andalucía for projects PY20-00870 and UPO-138516. The authors also thank the Portuguese Agency “Fundação para a Ciência e a Tecnologia” (FCT), in the framework of the project UIDB/00066/2020. This work could not have been done without the support and help of the Farmer's Association of Baixo Alentejo and Francisco Palma during the whole project. Finally, the authors thank António Vieira Lima and Moragri S. A. for giving access to data.

References

- [1] N. Khan, I. Yaqoob, I.A.T. Hashem, Z. Inayat, W.K. Mahmoud Ali, M. Alam, M. Shiraz, A. Gani, Big Data: Survey, Technologies, Opportunities, and Challenges, *Scientific World J.* 2014 (2014) 712826.
- [2] A. Galicia, J.F. Torres, F. Martínez-Álvarez, A. Troncoso, A novel spark-based multi-step forecasting algorithm for big data time series, *Inf. Sci.* 467 (2018) 800–818.
- [3] C.C. Aggarwal, *Data Mining: The Textbook*, Springer Publishing Company, Incorporated, 2015.
- [4] F.J. Pierce, P. Nowak, Aspects of precision agriculture, *Adv. Agron.* 67 (1999) 1–85.
- [5] J. Tan, P. Yang, Z. Liu, W. Wu, L. Zhang, Z. Li, L. You, H. Tang, Z. Li, Spatio-temporal dynamics of maize cropping system in Northeast China between 1980 and 2010 by using spatial production allocation model, *J. Geog. Sci.* 24 (3) (2014) 397–410.
- [6] D. Gutiérrez-Avilés, C. Rubio-Escudero, F. Martínez-Álvarez, J. Riquelme, TriGen: A genetic algorithm to mine triclusters in temporal gene expression data, *Neurocomputing* 132 (2014) 42–53.
- [7] L. Santesteban, S. Guillaume, J. Royo, B. Tisseire, Are precision agriculture tools and methods relevant at the whole-vineyard scale?, *Precision Agric* 14 (2012) 2–17.
- [8] R. Plant, Site-specific management: The application of information technology to crop production, *Comput. Electron. Agric.* 30 (2001) 9–29.
- [9] J. Costa, M. Vaz, J. Escalona, R. Egipto, C. Lopes, H. Medrano, M. Chaves, Modern viticulture in southern Europe: Vulnerabilities and strategies for adaptation to water scarcity, *Agric. Water Manag.* 164 (2016) 5–18.
- [10] A. Khaliq, L. Comba, A. Biglia, D. Ricauda Aimonino, M. Chiaberge, P. Gay, Comparison of Satellite and UAV-Based Multispectral Imagery for Vineyard Variability Assessment, *Remote Sens.* 11(4).
- [11] A. Matese, P. Toscano, S.F. Di Gennaro, L. Genesio, F.P. Vaccari, J. Primicerio, C. Belli, A. Zaldei, R. Bianconi, B. Gioli, Intercomparison of UAV, Aircraft and Satellite Remote Sensing Platforms for Precision Viticulture, *Remote Sens.* 7 (3) (2015) 2971–2990.
- [12] S.F. Di Gennaro, R. Dainelli, A. Palliotti, P. Toscano, A. Matese, Sentinel-2 Validation for Spatial Variability Assessment in Overhead Trellis System Viticulture Versus UAV and Agronomic Data, *Remote Sens.* 11(21).
- [13] L. Pastonchi, S. Di Gennaro, P. Toscano, A. Matese, Comparison between satellite and ground data with uav-based information to analyse vineyard spatio-temporal variability, *XIIIth International Terroir Congress, OENO One* 54 (2020) 919–934.
- [14] C. von Hebel, S. Reynaert, K. Pauly, P. Janssens, I. Piccard, J. Vanderborght, J. Kruk, H. Vereecken, S. Garre, Toward high-resolution agronomic soil information and management zones delineated by ground-based electromagnetic induction and aerial drone data, *Vadose Zone J.*
- [15] P. Janrao, H. Palivela, Management zone delineation in Precision agriculture using data mining: A review, in: 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015, pp. 1–7.
- [16] B.I. Evstatiev, K.G. Gabrovska-Evstatieva, A review on the methods for big data analysis in agriculture, *IOP Conference Series: Materials Science and Engineering* 1032 (2021) 012053.
- [17] A. Gavioli, E.G. de Souza, C.L. Bazzi, L.P.C. Guedes, K. Schenatto, Optimization of management zone delineation by using spatial principal components, *Comput. Electron. Agric.* 127 (2016) 302–310.

L. Melgar-García, D. Gutiérrez-Avilés, Maria Teresa Godinho et al.

- [18] P. Cinat, S.F. Di Gennaro, A. Berton, A. Matese, Comparison of Unsupervised Algorithms for Vineyard Canopy Segmentation from UAV Multispectral Images, *Remote Sens.* 11(9).
- [19] N. Ohana-Levi, K. Knipper, W.P. Kustas, M.C. Anderson, Y. Netzer, F. Gao, M. d. M. Alsina, L.A. Sanchez, A. Karnieli, Using Satellite Thermal-Based Evapotranspiration Time Series for Defining Management Zones and Spatial Association to Local Attributes in a Vineyard, *Remote Sensing* 12 (15).
- [20] L. Melgar-García, M.T. Godinho, R. Espada, D. Gutiérrez-Avilés, I.S. Brito, F. Martínez-Álvarez, A. Troncoso, C. Rubio-Escudero, Discovering spatio-temporal patterns in precision agriculture based on triclustering, in: 15th International Conference on Soft Computing Models in Industrial and Environmental Applications, Springer, Cham, 2021, pp. 226–236.
- [21] R.U.I. Henriques, S.C. Madeira, Triclustering Algorithms for Three-Dimensional Data Analysis: A Comprehensive Survey, *ACM Comput. Surv.* 51 (5) (2018) 43.
- [22] D. Gutiérrez-Avilés, C. Rubio-Escudero, Mining 3D patterns from gene expression temporal data: A new tricluster evaluation measure, *Scientific World J.* 2014 (2014) 1–16.
- [23] D. Gutiérrez-Avilés, C. Rubio-Escudero, MSL: A measure to evaluate three-dimensional patterns in gene expression data, *Evol. Bioinformatics* 11 (2015) 121–135.
- [24] D. Gutiérrez-Avilés, R. Giráldez, F.J. Gil-Cumbreras, C. Rubio-Escudero, TRIQ: a new method to evaluate triclusters, *BioData Mining* 11 (2018) 15.
- [25] D. Gutiérrez-Avilés, C. Rubio-Escudero, LSL: A new measure to evaluate triclusters, in: Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine, 2014, pp. 30–37.
- [26] F. Martínez-Álvarez, G. Asencio-Cortés, J.F. Torres, D. Gutiérrez-Avilés, L. Melgar-García, R. Pérez-Chacón, C. Rubio-Escudero, J.C. Riquelme, A. Troncoso, Coronavirus Optimization Algorithm: A Bioinspired Metaheuristic Based on the COVID-19 Propagation Model, *Big Data* 8 (4) (2020) 308–322.
- [27] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, A. Troncoso, High-content screening images streaming analysis using the strigen methodology, in: Proceedings of the 35th Annual ACM Symposium on Applied Computing, Association for Computing Machinery, 2020, pp. 537–539.
- [28] F. Martínez-Álvarez, D. Gutiérrez-Avilés, A. Morales-Esteban, J. Reyes, J.L. Amaro-Mellado, C. Rubio-Escudero, A novel method for seismogenic zoning based on triclustering: Application to the Iberian Peninsula, *Entropy* 17 (7) (2015) 5000–5021.
- [29] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, A. Troncoso, Discovering three-dimensional patterns in real-time from data streams: An online triclustering approach, *Inf. Sci.* 558 (2021) 174–193.
- [30] R.M. Alguliyev, R.M. Alguliyev, L.V. Sukhostat, Parallel batch k-means for big data clustering, *Comput. Ind. Eng.* 152 (2021) 107023.
- [31] C.-E. Ben Ncir, A. Hamza, W. Bouaguel, Parallel and scalable dunn index for the validation of big data clusters, *Parallel Comput.* 102 (2021) 102751.
- [32] M. Odersky, L. Spoon, B. Venners, Programming in Scala: Updated for Scala 2.12, third ed., Artima Incorporation, Sunnyvale, CA, USA, 2016.
- [33] B. Chambers, M. Zaharia, Spark: The Definitive Guide Big Data Processing Made Simple, first ed., O'Reilly Media Inc, 2018.
- [34] M. Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, I. Stoica, Apache Spark: A Unified Engine for Big Data Processing, *Commun. ACM* 59 (11) (2016) 56–65.
- [35] X. Jinru, B. Su, Significant Remote Sensing Vegetation Indices: A Review of Developments and Applications, *J. Sens.* 2017 (2017) 1–17.
- [36] A. Martínez, V.D. Gomez-Miguel, Vegetation index cartography as a methodology complement to the terroir zoning for its use in precision viticulture, *OENO One* 51 (3) (2017) 289.
- [37] H. Fraga, M. Amraoui, A. Malheiro, J. Moutinho Pereira, J. Eiras-Dias, J. Silvestre, J. Santos, Examining the relationship between the enhanced vegetation index and grapevine phenology, *Eur. J. Remote Sens.* 47 (2014) 753–771.
- [38] E. Laroche-Pinel, M. Albughdadi, S. Duthoit, V. Chéret, J. Rousseau, H. Clenet, Understanding vine hyperspectral signature through different irrigation plans: A first step to monitor vineyard water status, *Remote Sens.* 13 (3) (2021) 31.



Laura Melgar-García is a PhD student in Computer Science at the Pablo de Olavide University with a pre-doctoral researcher grant (FPU) from the Ministry of Science and Innovation of Spain. Before starting her doctoral studies, she earned a Biomedical Engineering Degree in 2017 and a Master in Software Engineering in 2018. Her major fields of research are the modeling and analysis of massive data, focusing on batch and streaming/online processing. The object of her research seeks to obtain both descriptive and predictive models with applications in real problems as precision agriculture, energy consumption or biomedical solutions.



David Gutiérrez-Avilés received the PhD degree in computer engineering from the University of Seville. He has been with the Department of Computer Science at the Pablo de Olavide University since 2016, where he is currently an assistant teacher. His primary areas of interest are bioinformatics, machine learning, data mining, and big data analytics.



Maria Teresa Godinho received the PhD degree in Operations Research from the University of Lisbon. She has been with the Department of Mathematical and Physical Sciences at the Instituto Politecnico de Beja, Portugal, since 2001, where she holds a position as an Adjunct Professor. Her primary areas of interest are integer programming and optimization algorithms.



Rita Isabel Espada has a degree and a master's degree in Agronomy from Escola Superior Agraria de Beja. She has been with AABA- Alentejo Farmers Association in Beja, Portugal, since 2018, where I am a technician in integrated production. Where I am responsible for agricultural advice and support for farmers. She has a special interest in precision farming and new technologies.



Isabel Sofia Brito is a Coordinator Professor at Polytechnic Institute of Beja, Portugal, and a member of the Centre of Technology and Systems (CTS-UNINOVA). Her main research interests are Requirements Engineering and Sustainability Requirements, Model and Data-Driven Development, Multi-Criteria Decision Making and, Big Data where she has published several papers on these topics in journals, international and national conferences, and workshops.



Francisco Martínez-Álvarez received the MSc degree in telecommunications engineering from the University of Seville, and the PhD degree in computer engineering from the Pablo de Olavide University. He has been with the Department of Computer Science at the Pablo de Olavide University since 2007, where he is currently a full professor. His primary areas of interest are time series analysis, data mining, and big data analytics.

L. Melgar-García, D. Gutiérrez-Avilés, Maria Teresa Godinho et al.



Cristina Rubio-Escudero received the Ph.D. degree in Computer Science from the University of Granada, Spain. She has been with the Department of Computer Science at the University of Seville since 2007, where she is currently an associate professor. Her primary areas of interest are bioinformatics, machine learning and big data.



Alicia Troncoso received the Ph.D. degree in Computer Science from the University of Seville, Spain, in 2005. She has been with the Department of Computer Science at the Pablo de Olavide University since 2005, where she is currently a full professor. Her primary areas of interest are time series forecasting, machine learning and big data.

Neurocomputing 500 (2022) 268–278

5.1.9 | "Streaming big time series forecasting based on nearest similar patterns with application to energy consumption"

Authors: Jiménez-Herrera P., Melgar-García L., Asencio-Cortés G., Troncoso A.

Publication type: Journal article.

Journal: Logic Journal of the IGPL.

Year: 2022.

Pages: 1367-0751

DOI: 10.1093/jigpal/jzac017

IF: 0.931 3/21 Logic.

Quartil: Q1.

Streaming big time series forecasting based on nearest similar patterns with application to energy consumption

P. JIMÉNEZ-HERRERA, *Division of Computer Science, Universidad Pablo de Olavide, ES-41013 Seville, Spain.*

L. MELGAR-GARCÍA, *Division of Computer Science, Universidad Pablo de Olavide, ES-41013 Seville, Spain.*

G. ASECIO-CORTÉS, *Division of Computer Science, Universidad Pablo de Olavide, ES-41013 Seville, Spain.*

A. TRONCOSO*, *Division of Computer Science, Universidad Pablo de Olavide, ES-41013 Seville, Spain.*

Abstract

This work presents a novel approach to forecast streaming big time series based on nearest similar patterns. This approach combines a clustering algorithm with a classifier and the nearest neighbours algorithm. It presents two separate stages: offline and online. The offline phase is for training and finding the best models for clustering, classification and the nearest neighbours algorithm. The online phase is to predict big time series in real time. In the offline phase, data are divided into clusters and a forecasting model based on the nearest neighbours is trained for each cluster. In addition, a classifier is trained using the cluster assignments previously generated by the clustering algorithm. In the online phase, the classifier predicts the cluster label of an instance, and the proper nearest neighbours model according to the predicted cluster label is applied to obtain the final prediction using the similar patterns. The algorithm is able to be updated incrementally for online learning from data streams. Results are reported using electricity consumption with a granularity of 10 minutes for 4-hour-ahead forecasting and compared with well-known online benchmark learners, showing a remarkable improvement in prediction accuracy.

Keywords: Time series forecasting, real time, streaming data, energy consumption.

1 Introduction

The current technological context has two main aspects of research and development. On the one hand, an industrial aspect, where the boom in advanced connection of devices or Internet of things (IoT) is changing the means of production and service management systems, leads our society to a new industrial revolution known as Industry 4.0. And on the other hand, the data, as a consequence of the enormous amount of data that is generated daily in our society, come from many different sources, including IoT between them, and lead us to a new technological revolution based on the analysis of large scale data known as big data [31, 32].

Of the three Vs that initially defined big data (volume, velocity and variety), volume is perhaps the characteristic in which researchers have made the greatest effort. Almost all the technology

*E-mail: ali@upo.es
Vol. 00, No. 0, © The Author(s) 2022. Published by Oxford University Press. All rights reserved.
For permissions, please e-mail: journals.permission@oup.com.
This article is published and distributed under the terms of the Oxford University Press, Standard Journals Publication Model (https://academic.oup.com/journals/pages/open_access/funder_policies/chorus/standard_publication_model)
<https://doi.org/10.1093/jigpal/jzac017>

2 Streaming Big Time Series Forecasting

developed in recent years allows to obtain approximate solutions to problems derived from the dimensionality of the data. However, velocity, despite being a feature present in many problems of data analysis, has not had the same impact, or rather, it is beginning to have it at present. Although the analysis of streaming data has been studied in the last decade, in very few cases forecasting techniques have been developed, which allow to have an updated model that is capable of giving a response in real time to obtain forecasts.

In this work, a new forecasting algorithm based on the nearest similar patterns for streaming big time series, named StreamNSP, is proposed. First, the StreamNSP algorithm determines different patterns in historical data. Once the data stream is received, the algorithm predicts the pattern to which the data stream just arrived belongs. Then, the prediction is obtained applying the nearest neighbours to the data with the same pattern to data stream. Despite that a naive use of the k-means clustering, Naive-Bayes classifier and the nearest neighbour to predict time series was presented in [19], the StreamNSP includes a methodology to obtain the optimal configuration of the clustering, classification and nearest neighbours models to be used. The sensitivity of the StreamNSP with respect to models involved to detect patterns, predict the label of the cluster and predict the values of the time series is also analyzed in order to study the robustness of the proposed method. The performance of the StreamNSP has been tested on a real-world dataset related to energy consumption. Finally, the results have been compared to that of the well-known prediction algorithms for streaming data.

The rest of the paper is structured as follows. Section 2 reviews of the existing literature related to the forecasting algorithms for streaming data. In Section 3, the proposed methodology to forecast streaming big time series is explained. Section 4 presents the experimental results corresponding to the prediction of the energy consumption. Finally, Section 5 closes the paper giving some final conclusions.

2 Related work

Time series forecasting is a broad field of study. The proposed methodology focuses on one of the areas that is gaining more importance in recent years, although it has not been widely studied yet: real-time forecasting and its application in electricity solutions.

The intrinsic characteristics of high-speed streams lead to different options for approaching the real-time learning problem. Assuming that basic requirements of this type of data such as single-pass, bounded memory, real-time decision making or concept drift detection are met, there are different ways to process as well as to model them. The data processing can be made by means of feature transformations, feature selections, reduction of the dimensionality or usage of time windows [14], while the model can be based on incremental learning or online-offline learning [21].

In particular, a review of forecasting algorithms for streaming data from year 2000 to 2015 was presented in [39]. Some of the researches carried out in the past years concerning the different ways of modeling time series prediction problems in real time are discussed below.

Regarding the incremental learning, authors in [2] integrated an ARIMA model, which is one of the most known classical methods for time series prediction, with an online information network called OLIN. The OLIN was modelled with incremental streaming learning using sliding windows for short-term forecasting of hourly electricity load. A double incremental learning algorithm for time series prediction was introduced in [23]. In particular, a support vector machine (SVM) was proposed as base learner and the incremental learning was performed by a combination of the existing base models and the ones induced by the new data. Another algorithm using two

components in incremental streaming learning was proposed in [30]. First, an online clustering process was applied and then, an incremental neural network to predict electrical sensor networks. In addition, authors in [4] introduced an adaptive algorithm for forecasting data streams in traffic flow, which was able to identify how much the ongoing and past flows should contribute to the predictions.

However, there are real-time forecasting algorithms that use offline–online learning instead of incremental learning. An algorithm for streaming time series prediction related to solar particle events was introduced in [8]. It is an embedded approach that combines an online phase for monitoring and predicting in real time with an offline phase for the training and establishment of the prediction model using historical data. In [29], an ARIMA algorithm was modeled in streaming mode with different phases. In the first phase, the best configuration of the model is selected. Then, the forecasting is made based on the previously selected model, and finally, the decision of whether to train a new model with recent streaming data or not is made. A new algorithm for forecasting data streams using an offline stage and an online one was proposed in [19]. In particular, data are divided into clusters with k-means in the offline phase, and afterwards, the nearest neighbour and the Naive–Bayes algorithm are applied in online phase.

The development of streaming algorithms for real applications is expanding to different areas. In [38], the real-time forecasting of two datasets, web traffic and temperature streaming flows, was obtained using a deep neural network. The prediction of clinical records in real time is obtained using a hybrid support system in [11]. This system contains a hierarchical temporal predictor and a LSTM classifier. Moreover, streaming is beginning to be present in other types of machine learning algorithms as triclustering applied to the environmental and medical field [27, 28] or streaming classification algorithms applied to language and temperature data [33].

With respect to the forecasting in the electricity environment, there are more studies on the batch mode than on the streaming mode. For example, authors in [25] reviewed the data mining algorithms published in the literature dealing with electricity series time. Electricity consumption in Spain was used to make predictions with several big data methods in [13] and with deep learning in [34, 35]. Another area of interest for leading electricity utilities is the classification of fraudulent behaviours among customers using big data systems as proposed in [15]. However, there are not many studies on the streaming mode for this type of data. Energy forecasting in smart buildings using IoT sensors for near real-time applications was presented in [16]. In addition, electricity load forecasting using online adaptive recurrent neural networks was proposed in [12]. In [22], clustering techniques were applied in order to identify outlier consumers in smart grids using energy streams.

After this review, it can be concluded that there is still a lot of research to do in the prediction of electricity time series in real time using machine learning algorithms.

3 Methodology

This section presents the proposed methodology divided into two phases and the description of the StreamNSP algorithm, which can be applied to time series of any nature. First, a general overview is provided in Section 3.1. Data preparation is described in Section 3.2. Finally, the offline phase of StreamNSP is described in Section 3.3 and its online phase in Section 3.4.

3.1 Overview

The StreamNSP algorithm has been developed for streaming of time series data, and it is based on a combination of a clustering algorithm along with both a classifier and the k-nearest neighbours (KNN) algorithm [1] for regression.

4 Streaming Big Time Series Forecasting

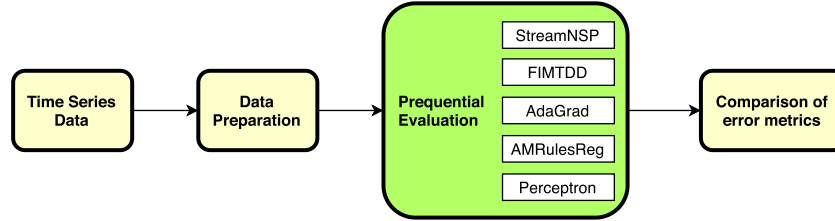


FIGURE 1. Overview of the processes within the proposed methodology.

The general idea behind the proposed forecasting algorithm is to take a training set in an offline phase and firstly divide it in clusters using a clustering algorithm. Then, the KNN algorithm is applied for each cluster providing a list of trained prediction models, one per each cluster. Finally, a classifier is trained for predicting the cluster label of an instance using as training the cluster assignments previously generated by the clustering algorithm.

Once the model of StreamNSP is generated in the offline phase, it is tested and then updated online using data streams. The methodology carried out to train, test and compare StreamNSP with a set of benchmark algorithms is graphically described in Figure 1.

In first place, a previous process of data preparation is performed from the historical time series data. After the preparation of data has been carried out, a model is generated for each forecasting horizon, using both the StreamNSP algorithm and a set of benchmark algorithms. Each model is evaluated using a prequential or interleaved test-then-train evaluation. Finally, a comparison of error metrics was performed. The same evaluation procedure is carried out for both StreamNSP and benchmark algorithms. The benchmark algorithms are described in Section 4.3.

The algorithm StreamNSP was implemented in the Java programming language (Oracle Java 1.8.0_152-b16 SE for 64 bits) and adapted to be compatible for the MOA framework [6]. All experiments for testing and benchmarking of the StreamNSP were automated using the API of the MOA framework (version 19.04).

3.2 Data preparation

The proposed forecasting algorithm is based on attributes (features) and a single numeric class to predict. However, time series data in a streaming context are a sequence of numeric values. For this reason, the following data preparation process was made prior to training, obtaining the best hyperparameters and testing the model. Figure 2 describes visually the procedure carried out to transform the time series data into a set of training, validation and test for each prediction horizon.

Specifically, a set of w lagged variables, a_1, \dots, a_w , was extracted from a time series x_1, \dots, x_n . These values reflect lagged windows of size w from the time series, as it can be seen in Figure 2. These variables will act as input attributes for the model. Along with these attributes, a last column y representing the class was added to the so called propositional tables for each forecasting horizon from 1 to h , where h is the number of future values to predict. These column is a future value, in the time series, with respect to the past values window.

Finally, for each horizon, training, validation and test subsets were taken from propositional tables, maintaining the original temporal order. In the offline phase, the training set will be used to train the model and the validation set will be used to find the best models along with their parameters for the

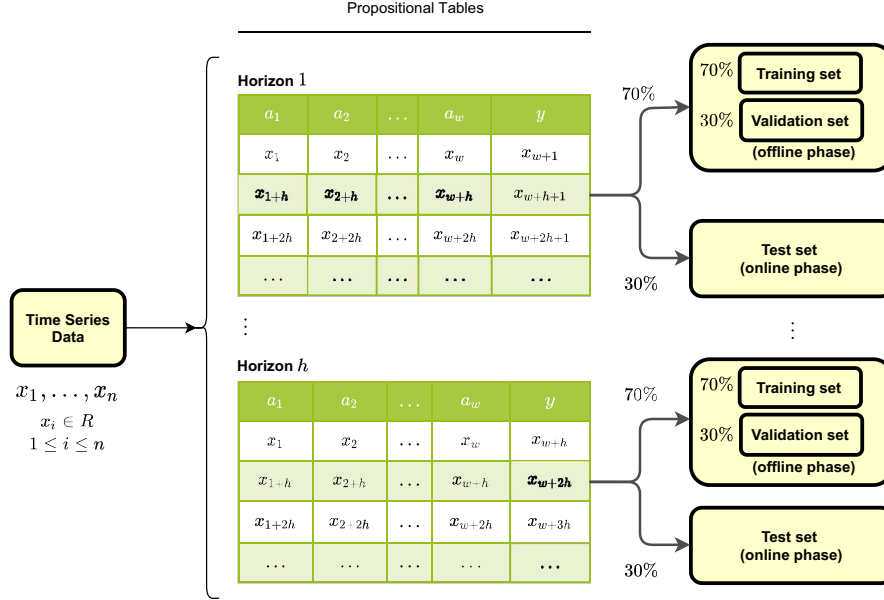


FIGURE 2. Data preparation process.

clustering, classification and the KNN algorithm making up the StreamNSP algorithm. On the other hand, the test set will be used to test and update the model in its online phase.

3.3 Offline phase

The procedure carried out by the StreamNSP algorithm in order to train the models in its offline phase is illustrated in Figure 3. It can be seen that given a training set, each instance composed by the attributes a_1, \dots, a_w and the numeric class y is extracted one by one according to its temporal order. Then, each instance from training set is stored in an internal training buffer.

Once the buffer is completed (i.e. when the number of instances i is equal to the training size $trainSize$), it is the input data for the clustering algorithm. In this work, the k-means [24] and canopy [26] clustering algorithms have been used.

As a result of the clustering, each instance is stored separately according to its assigned cluster. Then, a regression model is trained and stored for each cluster. In this work, KNN regression models [1] are stored.

A table containing the attributes a_1, \dots, a_w of all training instances along with the cluster assignments (the *cluster* label) is also generated and stored (table coloured in orange in Figure 3). Finally, a classification model is trained using such table, with the aim to predict the cluster label of further test instances. In this work, three classifiers have been tested, in particular, the Naive-Bayes classifier [20], a decision tree [36] and an SVM model [37].

Once the models have been trained, a validation set is also used to obtain the best models together with their best parameters for the various clustering, classification and KNN algorithms tested in the offline phase. This entire process is shown in Figure 4. The parameters are the number of clusters for the k-means clustering, the minimal number of instances of a leaf in the decision tree model in order

6 Streaming Big Time Series Forecasting

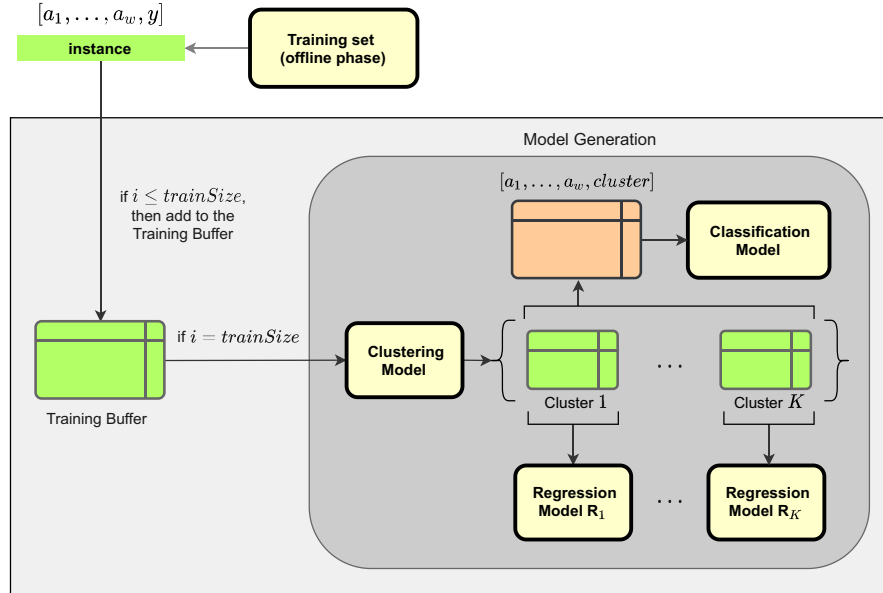


FIGURE 3. Training phase of the StreamNSP algorithm.

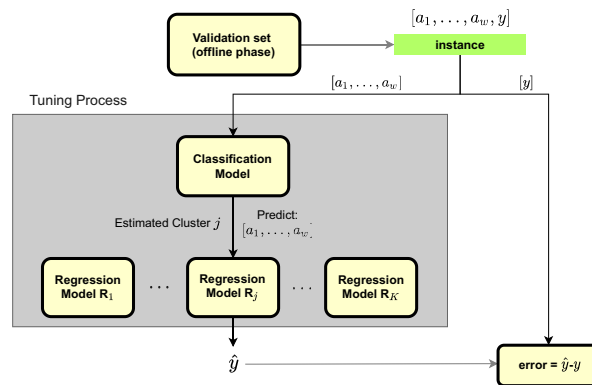


FIGURE 4. Optimal models for the StreamNSP algorithm.

to obtain the best stop criterion for tree building, the kernel for the SVM model and the number of neighbours for the KNN regression model. The mean absolute error is the error to be minimized to find the optimal configuration of both models and parameters for the StreamNSP approach.

Once the optimal models are obtained, these models are again trained using as training set the union of the training and validation sets.

3.4 Online phase

The procedure carried out by the StreamNSP algorithm in its online phase is described in Figure 5. Each instance is extracted one by one in online streaming from a test set. Such instance is composed of the attributes a_1, \dots, a_w and the numeric class y .

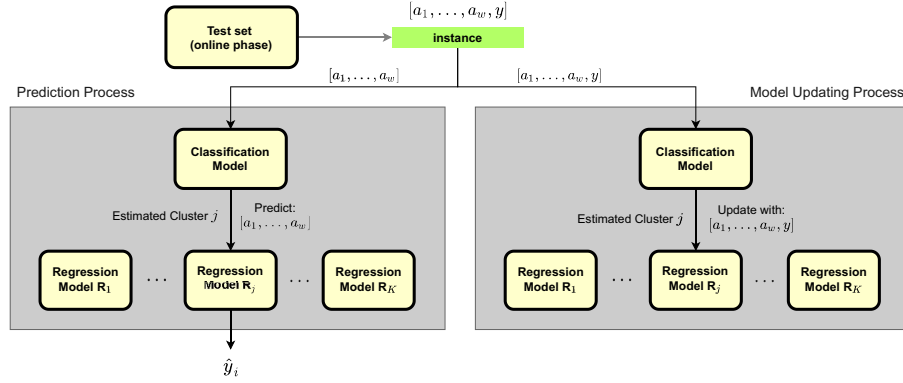


FIGURE 5. Online phase of the StreamNSP algorithm.

Due to the prequential evaluation performed, the online phase is divided in two steps. First is the step of predicting and, after, the step of updating the model.

The prediction step consists in receiving online the attributes a_1, \dots, a_w of the instance and use the optimal classification model previously obtained in the validation phase. This classification model returns the estimated cluster to which the test instance could belong (let be the cluster j). Then, the optimal KNN model for the cluster j (the model j) is used to predict the numeric class \hat{y}_i of the instance.

The model updating step consists in receiving both the attributes a_1, \dots, a_w and the actual class y of the instance. Using the optimal classification model, an estimated cluster is predicted (the same cluster as the prediction step because the instance attributes are the same). Finally, the optimal KNN model for the cluster j is updated using both the attributes and class of the instance.

4 Experimentation and results

This section presents and discusses the experiments carried out to assess the StreamNSP's performance for the dataset described in Section 4.1. The different error metrics used to evaluate the accuracy of the predictions are described in Section 4.2. In Section 4.3, the benchmark algorithms for comparison purposes are presented. Finally, an analysis of the results is carried out in Section 4.4.

4.1 Dataset

The time series considered in this study is related to the electricity consumption in Spain from 1 January 2007 at 00:00 to 21 June 2016 at 23:50. It is a time series of 9 years and 6 months with a high sampling frequency (10 minutes), resulting in 497832 measures in total. This dataset is available at <https://github.com/gualbe/datasets/tree/main/electric-consumption-spain>.

The time series was processed as described in Section 3.2 for a window of 144 lagged values corresponding to one day and a horizon of 24 values corresponding to 4 hours. Thus, the past day is used to predict the next 4 hours. After the preprocessing, we have removed the first 144 rows and the last 24 rows, in order to avoid empty values in the instances.

8 Streaming Big Time Series Forecasting

4.2 Evaluation metrics

The MAE, RMSE and MAPE errors have been used to evaluate the results of the proposed StreamNSP algorithm and the benchmark algorithms.

The MAE is the mean absolute error, which is the average of the absolute differences between actual and predictions. The MAE is defined in Equation (1), where y_i are the actual values, \hat{y}_i are the predicted values and n is the number of predicted samples.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (1)$$

The RMSE is the root mean square error, which represents the square root of the second sample moment of the differences between predicted values and observed values or the quadratic mean of these differences. The RMSE metric is defined in Equation (2).

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2)$$

Finally, the third evaluation metric is the MAPE, which is calculated as the mean of the absolute error in percentage, i.e. the average of actual values minus predicted values divided by actual values as shown in Equation (3).

$$MAPE = \frac{100}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (3)$$

4.3 Benchmark algorithms

In order to compare the forecasting results of the algorithm StreamNSP with other suitable approaches in the literature, four regression algorithms for online learning from data streaming were selected.

The algorithm FIMTDD [18] learns a regression tree and it is able to address time-changing data streams. The algorithm has mechanisms for drift detection and model adaptation, which enable it to maintain accurate and updated regression models at any time. The algorithm observes each example only once at the speed of arrival and maintains at anytime a ready-to-use tree model. The tree leaves contain linear models induced online. The number of instances a leaf should observe between split attempts was set to 200. The threshold below which a split will be forced to break ties was set to 0.05.

The algorithm AdaGrad [10] is an online optimizer for learning linear regression from data streams that incorporates knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning. The adaptation feature of this algorithm derives into strong regret guarantees, which for some natural data distributions achieve high performance. No regularization was used for the linear regression model and the learning rate was set to 0.01.

The algorithm AMRulesReg [3] is an adaptive model that is able to generate decision rules for regression from data streams. In this model, the antecedent of a rule is a conjunction of conditions on the attribute values, and the consequent is a linear combination of attribute values. Each rule uses a Page-Hinkley test [5] to detect changes in the process generating data and react to changes by pruning the rule set.

TABLE 1. Errors obtained for StreamNSP and the benchmark algorithms.

Algorithm	MAE	RMSE	MAPE (%)
AdaGrad	3242.434	3872.137	12.052
FIMTDD	1332.544	1751.836	4.911
Perceptron	612.676	986.839	2.297
AMRulesReg	590.401	944.853	2.211
StreamNSP	461.750	690.500	1.682

TABLE 2. Comparison of StreamNSP with other approaches published previously.

Algorithm	MAPE (%)	Algorithm	MAPE (%)
NDL	1.51	RF	2.22
StreamNSP	1.68	FFNN	2.32
CNN	1.71	EV	3.15
LSTM	1.78	GBM	4.49
ENSEMBLE	1.94	ARIMA	7.63
NN	2.16	DT	8.86

Finally, the algorithm Perceptron [7] is based on the algorithm Hoeffding Trees [17], but instead of using Naive–Bayes models at the leaf nodes it uses perceptron regressors. The perceptron regressors use the sigmoid activation function instead of the threshold activation function and optimize the mean squared error.

4.4 Results

The forecasting results for StreamNSP and the benchmark algorithms using the electricity time series data streams are presented and discussed in this section.

Table 1 shows a comparison of StreamNSP, AdaGrad, FIMTDD, Perceptron and AMRulesReg, in terms of MAE, RMSE and MAPE. These errors have been calculated and averaged for all predicted horizons. As it can be seen, our StreamNSP algorithm achieved better results in MAE, RMSE and MAPE than the other four benchmark learners. The MAE obtained by the StreamNSP has been compared with the best MAE result of the other four benchmark learners, in this case AMRulesReg. The StreamNSP approach provides an error improvement of 128.651 MW. The difference between the RMSE for AMRulesReg and StreamNSP algorithms is also high, in particular 254.353 higher for AMRulesReg. In addition, AMRulesReg obtained a MAPE of 0.529% worse than StreamNSP. Note that AdaGrad algorithm seems to be not suitable for predicting energy consumption data because of its high errors for all the metrics.

Table 2 shows a comparison of the MAPE achieved by StreamNSP with other results published previously [9]. The algorithms were ascending sorted according to its MAPE value. In [9], a new method, named NDL, for time series forecasting based on neuroevolution was proposed for offline learning and tested with the same data as in this work. Specifically, results were reproduced for the same window size (144 values).

As it can be seen in Table 2, StreamNSP achieved better results than the other algorithms except for the NDL method. The reason could be that NDL is designed for offline learning, its model

10 Streaming Big Time Series Forecasting

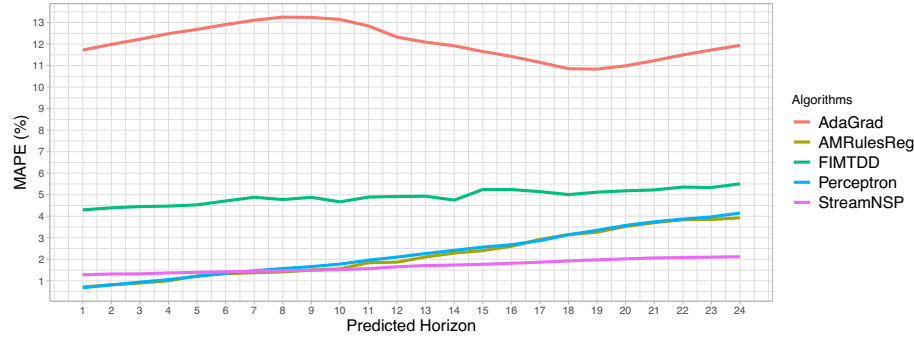


FIGURE 6. MAPE for the StreamNSP and the benchmark algorithms along all prediction horizons.

is more sophisticated and its training implies around five days, as it is explained in [9]. However, StreamNSP was designed for efficiency as it is required for online learning and its initial training (offline phase) took around 15 minutes.

Figure 6 shows the MAPE obtained by all algorithms for each prediction horizon. The errors were averaged on all the days of the test part of the streaming. It can be observed that StreamNSP achieved the most accurate predictions for longer forecast horizons, in particular greater than 10. Both Perceptron and AMRulesReg algorithms perform better for shorter horizons (those lower than 7). StreamNSP has shown the lowest difference of the MAPE along the horizons (0.83%), while Perceptron and AMRulesReg obtained a higher difference of MAPE (3.46% and 3.21%, respectively). The algorithms FIMTDD and AdaGrad achieved the worst prediction performance for all horizons.

Table 3 shows the errors for the StreamNSP algorithm, when using an internal tuning in order to find the best methods composing of the StreamNSP along with the best hyperparameters, and for a naive version of the StreamNSP with a fixed-setting. This setting consists of the k-means with the number of clusters set to 3 as the clustering algorithm, Naive-Bayes as the classifier and the nearest neighbours for the prediction models for each cluster. The errors are presented for each predicted horizon. It can be observed that errors were reduced for all horizons when StreamNSP used the optimal configuration of the methods and hyperparameters instead of using the prefixed setting. However, the differences are not excessively large as shows the average of the errors. This fact shows the robustness of StreamNSP with respect to the chosen clustering, classification or regression methods or its parameters. These results are very promising because frees the user from having to select the optimal parameters of the proposed algorithm.

Figure 7 shows the actual and predicted values obtained by the StreamNSP for each hour of the day. Both values were averaged for each hour of all the test days of the streaming. The x-axis is the hour of the day and the y-axis is the energy consumption in MW (target variable). It can be seen that the predicted values fit very well and they did not show significant errors for any particular hour.

In order to find the best algorithms and hyperparameters to be part of the StreamNSP, a validation process is carried out. For this purpose, k-means and canopy clustering algorithms have been considered, Naive-Bayes, decision tree and SVM as classifiers and KNN as possible predictors for each cluster. The number of clusters from 2 to 10 has been tested for the k-means algorithm, the minimal number of instances of a leaf in the decision tree model has been set to 2 and 4, linear and quadratic polynomial kernel and radial basis function (RBF) kernel have been considered for the

TABLE 3. Errors for the StreamNSP including tuning and a naive StreamNSP with fixed-setting.

Horizon	StreamNSP (fixed setting)			StreamNSP (including tuning)		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
1	421.57	568.97	1.52	353.65	471.65	1.28
2	428.81	578.61	1.55	361.45	480.24	1.31
3	435.30	590.44	1.57	364.45	488.43	1.32
4	443.16	604.72	1.60	375.37	503.87	1.36
5	454.04	621.03	1.64	382.56	514.79	1.37
6	466.11	635.74	1.67	395.42	531.13	1.43
7	470.07	640.12	1.69	400.47	537.66	1.44
8	473.15	644.66	1.70	404.42	541.64	1.46
9	477.32	650.28	1.71	411.61	550.07	1.49
10	483.94	660.73	1.74	418.01	565.64	1.51
11	492.64	682.34	1.78	429.87	586.57	1.56
12	510.16	722.87	1.85	454.12	629.07	1.65
13	527.45	753.94	1.91	470.20	655.74	1.71
14	534.72	776.12	1.94	474.23	673.48	1.73
15	544.48	798.65	1.98	484.65	695.43	1.77
16	552.46	823.93	2.01	496.15	721.57	1.81
17	560.64	854.28	2.05	507.23	749.97	1.86
18	580.30	915.05	2.12	524.46	800.55	1.92
19	593.59	959.36	2.17	539.37	840.07	1.98
20	604.59	996.63	2.21	550.38	871.34	2.02
21	614.68	1031.44	2.25	561.04	901.01	2.06
22	622.03	1051.80	2.27	568.35	917.75	2.08
23	627.83	1063.40	2.29	572.89	922.87	2.10
24	643.61	1129.54	2.34	581.63	971.40	2.12
Average	523.44	781.44	1.90	461.75	671.75	1.68

SVM, and 1 and 3 neighbours have been taken into account for the KNN algorithm (named KNN1 and KNN3, respectively). The algorithm canopy obtains the number of clusters automatically; in this case, 6 clusters were obtained.

Table 4 shows the best methods and hyperparameters obtained by the internal validation of StreamNSP for each horizon. The value of the parameters for each method is shown in brackets. The KNN algorithm with 3 neighbours (KNN3) was selected as the best predictor for all clusters and for all the horizons. It is also noticeable that 2 clusters were automatically selected for short horizons (lower than 12), while 3 clusters were selected for the highest ones (from horizon 12 to 24, except 14). It seems that SVM with RBF as kernel is the best classifier when 3 clusters are selected, and Naive–Bayes when selected 2 clusters for horizons 5 to 11 and 14.

Table 5 shows the MAE obtained by the StreamNSP to find the best methods and to adjust the hyperparameters in the internal validation. These errors were classified depending on the clustering algorithm (by columns) and the classifier used (by rows). The value of the parameters for each method is shown in brackets. Since there is one predictor for each cluster, the values were averaged for all predictors. Excepting the configurations composed of the canopy clustering algorithm and

12 Streaming Big Time Series Forecasting

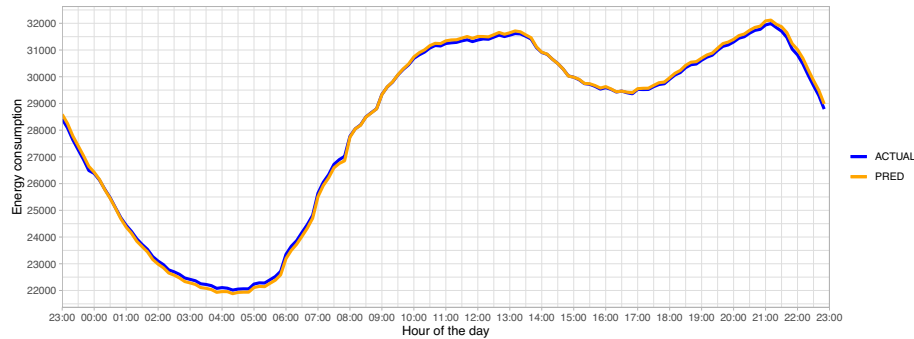


FIGURE 7. Actual and predicted values for each hour of the day.

TABLE 4. Best configurations found by StreamNSP in its internal validation for each horizon.

Horizon	Clustering	Classifier	Regressors
1	k-Means(2)	SVM(RBF)	KNN3,KNN3
2	k-Means(2)	SVM(RBF)	KNN3,KNN3
3	k-Means(2)	SVM(RBF)	KNN3,KNN3
4	k-Means(2)	SVM(RBF)	KNN3,KNN3
5	k-Means(2)	Naive-Bayes	KNN3,KNN3
6	k-Means(2)	Naive-Bayes	KNN3,KNN3
7	k-Means(2)	Naive-Bayes	KNN3,KNN3
8	k-Means(2)	Naive-Bayes	KNN3,KNN3
9	k-Means(2)	Naive-Bayes	KNN3,KNN3
10	k-Means(2)	Naive-Bayes	KNN3,KNN3
11	k-Means(2)	Naive-Bayes	KNN3,KNN3
12	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
13	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
14	k-Means(2)	Naive-Bayes	KNN3,KNN3
15	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
16	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
17	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
18	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
19	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
20	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
21	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
22	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
23	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3
24	k-Means(3)	SVM(RBF)	KNN3,KNN3,KNN3

SVM, which obtain a MAE lower than 551.27, the rest of configurations achieved similar values of MAE with a mean of 660.92 and a standard deviation of 8.77. These results seem to point to the robustness of the StreamNSP algorithm, which can be able to be applied using different clustering, classifier and regressor algorithms with no significant penalization in the error.

TABLE 5. MAE obtained by the StreamNSP in the internal validation depending on the clustering and the classifier.

Classifier	Canopy(6)	k-Means(2)	k-Means(3)	k-Means(4)	k-Means(5)
J48(2)	670.42	657.66	658.23	665.52	672.27
J48(4)	668.08	658.19	658.76	664.32	673.83
Naive-Bayes	692.42	651.66	654.73	659.93	661.67
SVM(Linear)	548.85	651.89	656.82	660.4	657.04
SVM(quadratic)	546.94	651.62	656.96	661.29	658.54
SVM(RBF)	551.27	651.65	653.39	661.41	656.27

TABLE 6. MAE obtained by the StreamNSP in the internal validation depending on the clustering and the predictor.

Cluster	KNN1	KNN3
Canopy(6)	702.59	593.62
k-Means(2)	703.50	604.06
k-Means(3)	707.73	605.24
k-Means(4)	713.70	610.59
k-Means(5)	715.41	611.13

Table 6 shows the MAE obtained by the StreamNSP to adjust the best configuration, classified for the clustering and predictor algorithms used and averaged for all the classification algorithms. As it can be concluded, KNN3 performed notably best in terms of MAE for all the clustering algorithms, and therefore, 3 neighbours are recommended to obtain the final prediction by the StreamNSP.

5 Conclusion

In this work, a new forecasting algorithm named StreamNSP has been proposed for online learning on streaming big time series data. The StreamNSP algorithm is composed of a clustering technique, a classifier and the KNN algorithm. StreamNSP has an offline phase to obtain the prediction model using the historical data and to find the best models and parameters for the clustering, classification and KNN algorithm. This phase consists of splitting the training data into clusters using a clustering algorithm. Then, a KNN algorithm is applied for each cluster providing a list of trained regression models, one per each cluster. In addition to that, a classifier is trained for predicting the cluster label of an instance using as training the cluster assignments previously generated by the clustering algorithm. The algorithm can be updated incrementally for online learning from data streams including new instances into the model corresponding to its estimated cluster. StreamNSP has been tested using the electricity consumption with a granularity of 10 minutes for predicting a horizon of 4 hours. The algorithm widely overcame other four online learners, such as AMRulesReg, Perceptron, FIMTDD and AdaGrad, achieving an average MAPE of 1.68% versus 2.21%, 2.29%, 4.91% and 12.05% obtained by the other algorithms, respectively. Moreover, the StreamNSP has obtained the most accurate predictions for large forecasting horizons (11 or more values ahead). Results achieved by the StreamNSP algorithm varying its clustering, classifier and regressor internal algorithms seem

14 *Streaming Big Time Series Forecasting*

to point to the robustness of the algorithm, which can be able to be applied using different internal algorithms with no significant error penalization.

As future work, other internal algorithms will be tested for the inner regression component of the StreamNSP, such as different architectures of deep neural networks. Furthermore, a sensitivity study of the length of window used in StreamNSP will be performed. Since KNN3 performed better than KNN1 for all the experiments, higher values of the number of neighbours will be tested. Finally, other optimization techniques will be applied to adjust the methods and hyperparameters of the StreamNSP, such as Bayesian optimization or bioinspired metaheuristics.

Funding

The authors would like to thank the Spanish Ministry of Science and Innovation and Junta de Andalucía for the support under the projects PID2020-117954RB-C21, PY20-00870 and UPO-138516, respectively.

References

- [1] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, **6**, 37–66, 1991.
- [2] D. Alberg and M. Last. Short-term load forecasting in smart meters with sliding window-based arima algorithms. *Vietnam Journal of Computer Science*, **5**, 241–249, 06 2018.
- [3] E. Almeida, C. Ferreira and J. Gama. Adaptive model rules from data streams. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases*, pp. 480–492. Springer, Berlin, Heidelberg, 2013.
- [4] M. A. C. Alves and R. L. F. Cordeiro. Effective and unburdensome forecast of highway traffic flow with adaptive computing. *Knowledge-Based Systems*, **212**, 106603, 2021.
- [5] M. Basseville. Detecting changes in signals and systems—a survey. *Automatica*, **24**, 309–326, 1988.
- [6] A. Bifet, G. Holmes, R. Kirkby and B. Pfahringer. MOA: massive online analysis. *Journal of Machine Learning Research*, **11**, 1601–1604, 2010.
- [7] A. Bifet, G. Holmes, B. Pfahringer and E. Frank. Fast perceptron decision tree learning from evolving data streams. In *Proceedings of the Advances in Knowledge Discovery and Data Mining*, pp. 299–310. Springer, Berlin, Heidelberg, 2010.
- [8] J. Chen, T. Lange, M. Andjelkovic, A. Simevski and M. Krstic. Prediction of solar particle events with SRAM-based soft error rate monitor and supervised machine learning. *Microelectronics Reliability*, **114**, 113799, 2020.
- [9] F. Divina, J. F. Torres, M. García-Torres, F. Martínez-Álvarez and A. Troncoso. Hybridizing deep learning and neuroevolution: application to the Spanish short-term electric energy consumption forecasting. *Applied Sciences*, **10**, 1–14, 2020.
- [10] J. Duchi, E. Hazan and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, **12**, 2121–2159, 2011.
- [11] N. O. El-Ganainy, I. Balasingham, P. S. Halvorsen and L. A. Rosseland. A new real time clinical decision support system using machine learning for critical care units. *IEEE Access*, **8**, 185676–185687, 2020.
- [12] M. N. Fekri, H. Patel, K. Grolinger and V. Sharma. Deep learning for load forecasting with smart meter data: online adaptive recurrent neural network. *Applied Energy*, **282**, 116177, 2021.

- [13] A. Galicia, J. F. Torres, F. Martínez-Álvarez and A. Troncoso. A novel spark-based multi-step forecasting algorithm for big data time series. *Information Sciences*, **467**, 800–818, 2018.
- [14] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal and J. Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, **21**, 6–22, 2019.
- [15] D. Gutiérrez-Avilés, J. A. Fábregas, J. Tejedor, F. Martínez-Álvarez, A. Troncoso, A. Arcos and J. C. Riquelme. Smartfd: a real big data application for electrical fraud detection. In *Hybrid Artificial Intelligent Systems*, pp. 120–130. Springer International Publishing, Cham, 2018.
- [16] S. Hadri, Y. Naitmalek, M. Najib, M. Bakhouya, Y. Fakhri and M. Elaroussi. A comparative study of predictive approaches for load forecasting in smart buildings. *Procedia Computer Science*, **160**, 173–180, 2019.
- [17] G. Hulten, L. Spencer and P. Domingos. Mining time-changing data streams. In *Proceedings of the Knowledge Discovery on Databases*, pp. 97–106. Springer, Berlin, Heidelberg, 2001.
- [18] E. Ikonomovska, J. Gama and S. Džeroski. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, **23**, 128–168, 2011.
- [19] P. Jiménez-Herrera, L. Melgar-García, G. Asencio-Cortés and A. Troncoso. A new forecasting algorithm based on neighbors for streaming electricity time series. In *Hybrid Artificial Intelligent Systems*, pp. 522–533. Springer International Publishing, Cham, 2020.
- [20] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pp. 338–345. Morgan Kaufmann Publishers Inc, 1995.
- [21] P. Larrañaga, D. Atienza, J. Diaz Roza, A. Ogbechie, C. Puerto-Santana and C. Bielza. *Industrial Applications of Machine Learning*. CRC Press, 2018.
- [22] P. Laurinec and M. Lucká. Interpretable multiple data streams clustering with clipped streams representation for the improvement of electricity consumption forecasting. *Data Mining and Knowledge Discovery*, **33**, 413–445, 2019.
- [23] J. Li, Q. Dai and R. Ye. A novel double incremental learning algorithm for time series prediction. *Neural Computing and Applications*, **31**, 6055–6077, 10 2019.
- [24] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.
- [25] F. Martínez-Álvarez, A. Troncoso, G. Asencio-Cortés and J. C. Riquelme. A survey on data mining techniques applied to electricity-related time series forecasting. *Energies*, **8**, 13162–13193, 2015.
- [26] A. McCallum, K. Nigam and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 169–178. Association for Computing Machinery, New York, NY, United States, 2000.
- [27] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero and A. Troncoso. Discovering three-dimensional patterns in real-time from data streams: an online triclustering approach. *Information Sciences*, **558**, 174–193, 2021.
- [28] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero and A. Troncoso. High-content screening images streaming analysis using the strigen methodology. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC '20*, pp. 537–539. Association for Computing Machinery, 2020.

16 *Streaming Big Time Series Forecasting*

- [29] Y. NaitMalek, M. Najib, M. Bakhouya and M. Essaïdi. Embedded real-time battery state-of-charge forecasting in micro-grid systems. *Ecological Complexity*, **45**, 100903, 2021.
- [30] P. P. Rodrigues and J. Gama. Online prediction of streaming sensor data. In *Proceedings of the 3rd International Workshop on Knowledge Discovery from Data Streams (IWKDDs 2006)*, in Conjunction with the 23rd International Conference on Machine Learning, p. 12. Association for Computing Machinery, New York, NY, United States, 2006.
- [31] R. L. Talavera-Llames, R. Pérez-Chacón, M. Martínez-Ballesteros, A. Troncoso and F. Martínez-Álvarez. A nearest neighbours-based algorithm for big time series data forecasting. In *Proceedings of the International Conference on Hybrid Artificial Intelligent Systems (HAIS)*, pp. 174–185. Springer, Cham, 2016.
- [32] R. L. Talavera-Llames, R. Pérez-Chacón, A. Troncoso and F. Martínez-Álvarez. Big data time series forecasting based on nearest neighbours distributed computing with spark. *Knowledge-Based Systems*, **161**, 12–25, 2018.
- [33] H. Tavasoli, B. J. Oommen and A. Yazidi. On utilizing weak estimators to achieve the online classification of data streams. *Engineering Applications of Artificial Intelligence*, **86**, 11–31, 2019.
- [34] J. F. Torres, A. Galicia, A. Troncoso and F. Martínez-Álvarez. A scalable approach based on deep learning for big data time series forecasting. *Integrated Computer-Aided Engineering*, **25**, 335–348, 2018.
- [35] J. F. Torres, A. Troncoso, I. Koprinska, Z. Wang and F. Martínez-Álvarez. Deep learning for big data time series forecasting applied to solar power. In *Proceedings of the 13th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO)*, pp. 123–133, 2018.
- [36] L. Vanfretti and V. S. N. Arava. Decision tree-based classification of multiple operating conditions for power system voltage stability assessment. *International Journal of Electrical Power & Energy Systems*, **123**, 106251, 2020.
- [37] P. Vijayaragavan, R. Ponnusamy and M. Aramudhan. An optimal support vector machine based classification model for sentimental analysis of online product reviews. *Future Generation Computer Systems*, **111**, 234–240, 2020.
- [38] S. Wambura, J. Huang and H. Li. Long-range forecasting in feature-evolving data streams. *Knowledge-Based Systems*, **206**, 106405, 2020.
- [39] Z. M. Yaseen, A. El-shafie, O. Jaafar, H. A. Afan and K. N. Sayl. Artificial intelligence based models for stream-flow forecasting: 2000-2015. *Journal of Hydrology*, **530**, 829–844, 2015.

Received 20 February 2021

5.1.10 | "Nearest neighbors with incremental learning for real-time forecasting of electricity demand"

Authors: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A.

Publication type: Conference article.

Conference: IEEE International Conference on Data Mining (ICDM 2022).

Publication: IEEE Computer Society Press.

Year: 2022.

Ranking: GGS class (rating): 1 (A++)

Nearest neighbors with incremental learning for real-time forecasting of electricity demand

Laura Melgar-García
Data Science & Big Data Lab
Pablo de Olavide University
Seville, Spain
lmelgar@upo.es

David Gutiérrez-Avilés
Department of Computer Science
University of Seville
Seville, Spain
dgutierrez3@us.es

Cristina Rubio-Escudero
Department of Computer Science
University of Seville
Seville, Spain
crubioescudero@us.es

Alicia Troncoso
Data Science & Big Data Lab
Pablo de Olavide University
Seville, Spain
atrolor@upo.es

Abstract—Electricity demand forecasting is very useful for the different actors involved in the energy sector to plan the supply chain (generation, storage and distribution of energy). Nowadays energy demand data are streaming data coming from smart meters and has to be processed in real-time for more efficient demand management. In addition, this kind of data can present changes over time such as new patterns, new trends, etc. Therefore, real-time forecasting algorithms have to adapt and adjust to online arriving data in order to provide timely and accurate responses. This work presents a new algorithm for electricity demand forecasting in real-time. The proposed algorithm generates a prediction model based on the K-nearest neighbors algorithm, which is incrementally updated as online data arrives. Both time-frequency and error threshold based model updates have been evaluated. Results using Spanish electricity demand data with a ten-minute sampling frequency rate are reported, reaching 2% error with the best prediction model obtained when the update is daily.

Index Terms—real time forecasting, incremental learning, streaming time series, electricity demand.

I. INTRODUCTION

Nowadays more and more attention is being paid to the topic of time series forecasting, especially because of its interdisciplinary nature. Almost all scientific disciplines consist of data sampled over time, which makes forecasting a task of utmost importance and complexity. Participants in electricity markets (both demand and prices) are particularly interested in forecasting, as obtaining forecasts is critical for many areas in order to increase profits or reduce costs. In addition, climate change is one of the most concerning topics of recent decades. Energy efficiency is the main mitigation factor to slow down the growth of energy consumption and is crucial in sustainable development [1]. In this context, the current progress of Internet of Things (IoT) devices is leading to the possibility of monitoring energy consumption. Furthermore, IoT devices provide extensive amounts of high-dimensional data in streaming [2]. This type of data opens up a huge field

of study in the big data streaming paradigm with the aim of obtaining real time solutions that lead to energetic efficiency.

In this way, big data streaming is becoming one of most widely used trends in big data in recent years. The most important feature for big data streaming is the velocity referred to large continuous flows of data, called streams, and for algorithms dealing with big data streaming is to offer results in real time. For this reason, streams need to be processed and modeled in a special way considering specific requirements [3]. The real-time decision making process contributes to the challenge of developing safe and efficient smart cities with timely responses [4].

In addition to obtaining fast results when working in streaming mode, it is important to develop a model that must be always ready to give responses. However, streaming flows usually variate and suffer transformations which could lead to a non-accurate response if the model does not adapt to them [5]. Thus, prediction models for streaming data must be updated as the data arrives in real time to best match its new behavior. This update must be in real time, so the re-training of the model should be discarded in favor of incremental learning.

In this work, we propose an algorithm to predict in real time electricity demand time series that are received in streaming. This algorithm, named StreamWNN, uses the K-nearest neighbors to compute the final prediction and the neighbors are updated over time through incremental learning. In particular, the StreamWNN algorithm is made up of two phases: a batch phase in which a historical model based on the nearest neighbors is created and an online phase to forecast and update the model. Therefore, the online phase keeps the model always adjusted to the current data. Results using energy consumption data in Spain from 2007 to 2016 are evaluated to show the accuracy of the predictions, the response time of the algorithm and the improvements obtained when the model is updated. The aim of this research paper is to show how the proposed incremental learning for the nearest neighbor method improves

the online predictions.

The rest of the paper is structured as follows. Section II presents a review of forecasting algorithms for energy time series focusing on real-time and data received in streaming. In Section III the methodology of the StreamWNN algorithm is described, including how the updating of the prediction model is performed. Section IV defines the electricity demand dataset used and presents the discussion of the results. The paper ends with some final conclusions and ideas for future approaches in Section V.

II. RELATED WORK

Times series forecasting has been a widely researched field in big data. However, most of the published prediction models for big data time series work in batch mode. There is still a lot of research to do in relation to streaming environments. The challenges of high-dimensional massive data mining in real-time are related to storage, processing and obtaining useful knowledge. Understanding and analyzing data in streaming is necessary to assist in the decision making process [6].

Thanks to the increasing amount of massive data from electronic devices, there are lots of applications for this type of data. In terms of streaming time series forecasting, authors in [7] presented several models to predict the evolution of the COVID-19 pandemic in real-time. In [8] a forecasting model for streaming taxi demand was presented. Other types of algorithms are starting to have more influence in streaming environments. A triclustering algorithm [9] was developed for the real-time processing in [10], [11]. Three-dimensional patterns from environmental sensor and medical streaming data were obtained.

With respect to the electricity demand, the study of its prediction has been made mostly in the batch or traditional mode. In [12] a review of the current methods for electricity price forecasting was presented. Authors in [13] presented a big data approach for electricity consumption in smart cities. Predictions for electricity demand using nearest neighbors from Apache Spark framework for big data were made in [14].

In the real-time environment, a review of forecasting algorithms for streaming data from year 2000 to 2015 was presented in [15]. In [16] a new algorithm for streaming energy demand data forecasting was proposed. It used three different algorithms adapted for streaming data: k-means, nearest neighbors and Naive Bayes. In [17] a randomized version of neural network with a incremental learning using the electric load datasets from Australian energy market was proposed to improve both efficiency and accuracy.

Streaming data can variate and change its behavioral patterns while time passes. It is important to have a streaming model that adapts to the new streams of data.

Many of the approaches in the literature for model updating in streaming environment are based on external algorithms as the Kalman filter. Authors in [18] presented a coupled methodology of the k-nearest neighbor algorithm with a Kalman filter for real-time flood forecasting. The state transition matrix of the Kalman filter was recalculated using the forecasting

method to improve the performance of the model and obtain accurate results. In [19] concepts of ensemble Kalman filter were used to update a rainfall runoff model for forecasting large floods in real-time. Another ensemble Kalman filter was also used in [20]. The ensemble was used as a data assimilation algorithm to update water temperature forecasting considering sensor instances.

Recently, deep neural networks have been used to predict streaming data by taking into account the changes that the data may undergo over time. A wavelet-neural network with an error-updating scheme was proposed in [21] for meteorological predictions in streaming. The need of systematic error-updating for stream flows was proven. Authors in [22] proposed an incremental update method based on support vector machine (SVM) and gate recurrent unit (GRU) considering concept drift for forecasting in real-time. Training models were updated based on the error between batch test results and real values. A combination of a real time autoregression and a deep Long Short-Term Memory (LSTM) recurrent network to predict streaming data from industrial processes was proposed in [23]. The deep LSTM found temporal relationships in data while the autoregression model addressed overlap and transfer between different recurring concepts drifts thus improving the accuracy.

III. METHODOLOGY

The StreamWNN streaming algorithm [24] is based on the K-nearest neighbors [25] which main idea is that nearest data share similar properties or characteristics.

The goal of this time series forecasting problem is to predict the next values considering past ones. The time series used is divided into N instances where the $i - th$ instance consists of x^i representing its w past features and y^i representing its h next classes, i.e., next h values to predict:

$$X_t = \{(x^1, y^1), \dots, (x^N, y^N)\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \quad (1)$$

The dataset is divided into three chronologically ordered sets: $set_{neighbors}$, $set_{patterns}$ and $set_{streaming}$. The StreamWNN algorithm uses the offline-online learning approach to model data streams [26]. The first two data sets are used in the batch or offline phase of the algorithm and the last set in the streaming or online phase. Big data streams requirements have to be fulfilled only in the online phase. These requirements are: only a limited set of past data can be stored; the model has to adapt to concept drift quickly and has to be always ready to make predictions; the model has to work in distributed computational environments so that its computation is fast [27]. In this work, an innovative incremental learning approach during the online phase is presented. In this way, the algorithm is always up to date.

The proposed algorithm is built on Apache Spark 2.3.4 and uses HDFS file system on Hadoop 2.7.7 and the Kappa Architecture on Apache Kafka 2.11 as streaming platform, i.e., a single pipeline is specifically designed for the job.

The methodology of this algorithm is defined in three sections. Section III-A describes the offline phase, Section III-B presents the online phase and how real time forecasting is performed and Section III-C focuses on the incremental learning approach of the StreamWNN algorithm.

A. Offline learning model

The creation of the offline learning model is the first task of the algorithm. It is a very important part as it is the base model for the online phase where streaming data start to arrive. This first phase is based on batch processing but using distributed programming, which allows to get a good model computed in a short time.

The offline phase uses the $set_{neighbors}$ and the $set_{patterns}$ representing approximately 70% and 30% of the chronologically ordered data used for the offline stage. The batch model associates each feature instance of the $set_{patterns}$ with its K closest instances of the $set_{neighbors}$. The offline model is represented as:

$$M = \langle x^i, \langle y(neighbor_1(x^i)), \dots, y(neighbor_K(x^i)) \rangle \rangle \quad (2)$$

where $x^i \in \mathbb{R}^w$ are the features of the i -th instance of the $set_{patterns}$ and $y(neighbor_j(x^i)) \in \mathbb{R}^h$ are the classes of the instance of the $set_{neighbors}$ selected as the j -th closest neighbor of x^i .

In this work, the euclidean distance is the metric selected to measure the proximity between the attributes of two instances, one from the set of patterns $set_{patterns}$ and the other from the set of neighbors $set_{neighbors}$. It can be represented as $d(x^i, x^j)$ where x^i refers to the features of the i -th instance of the set of patterns and x^j to the features of the j -th instance of the set of neighbors.

Once the offline model is generated, predictions of the next h values are made and its performance is tested by calculating an error metric. In the offline phase, predictions for each value l in the prediction horizon of length h are computed by:

$$\hat{y}^l = \frac{1}{\sum_{j=1}^K \alpha_j} \sum_{j=1}^K \alpha_j y(neighbor_j(x^i))^l \quad 1 \leq l \leq h \quad (3)$$

where α_j represents the distance of $d(x^i, x^j)$ defined as follows:

$$\alpha_j = \frac{1}{d(x^i, x^j)^2} \quad (4)$$

Closest data will have a greater α_j distance. Finally, the error of the predictions made in the offline phase is computed. For this work, Mean Absolute Percentage Error ($MAPE$) and Mean Absolute Error (MAE) metrics are used. Thus, in the offline stage, the final errors $MAPE_{offline}$ and $MAE_{offline}$ are the mean of all $MAPE_i$ and MAE_i , respectively, where $MAPE_i$ and MAE_i are the errors of the predictions for each i -th instance of the $set_{patterns}$ and they are defined as:

$$MAPE_i = \frac{1}{h} \sum_{l=1}^h \left| \frac{y^l - \hat{y}^l}{y^l} \right| \times 100 \quad 1 \leq l \leq h \quad (5)$$

$$MAE_i = \frac{1}{h} \sum_{l=1}^h |y^l - \hat{y}^l| \quad 1 \leq l \leq h \quad (6)$$

where y^l corresponds to the l class of the i -th instance and \hat{y}^l is the offline prediction defined by Equation (3).

B. Real-time forecasting

Once the offline model has been computed from the Equation (2) the online phase of the StreamWNN algorithm starts. Data are received in streaming and collected in instances of w features, these data can be called $x^{streaming}$. For each $x^{streaming}$, the euclidean distances between it and all x^i instances of the model M are calculated. Note that the number of distances calculated is the number of instances of the set of patterns $set_{patterns}$. The one with the minimum distance d_{min} is selected as the nearest x^i , called x^{min} from now on.

Once $x^{streaming}$ is associated with its nearest x^{min} , predictions are computed. In this case, online predictions consider not only the K -nearest neighbors of x^{min} but also x^{min} itself as a neighbor for each value l to predict in the prediction horizon of length h . That is:

$$\hat{y}^l = \frac{1}{\alpha} \left(\left(\sum_{j=1}^K \alpha_j y(neighbor_j(x^{min}))^l \right) + \alpha_{min} y(x^{min})^l \right) \quad (7)$$

where α and α_{min} are defined as:

$$\alpha = \left(\sum_{j=1}^K \alpha_j \right) + \alpha_{min} \quad (8)$$

$$\alpha_{min} = \frac{1}{d(x^{min}, x^{streaming})^2} \quad (9)$$

Then, the $MAPE_{streaming}$ and $MAE_{streaming}$ error metrics for each i -th instance of the $set_{streaming}$ can be calculated using Equations (5) and (6) when the real class of stream data $x^{streaming}$ is received. Thus, in the online stage, the final errors $MAPE_{online}$ and MAE_{online} are the mean of all $MAPE_{streaming}$ and $MAE_{streaming}$, respectively.

Therefore, real-time forecast is performed considering the historical data stored in the model M obtained in the offline phase. However, the StreamWNN algorithm includes the possibility of incrementally updating the model during the online phase. This novel update is explained in Section III-C.

C. Incremental learning

The goal of online incremental learning is to keep the model up to date. This is a very important task in data streaming algorithms, as new data patterns may appear and need to be included in the model. Without this incremental update, the model may age and not be suitable for real-time forecasting.

The incremental learning is performed by updating the neighbors of x^{min} , i.e., $neighbor_j(x^{min})$ with $j = 1, \dots, K$.

For this reason, it is said that the model is internally updated, since the dimensions of the model M are maintained but the components are updated considering the new patterns in the streaming data.

The update of neighbors uses a buffer B of possible updates. Let d_{min} be the distance between $x^{streaming}$ and its nearest pattern x^{min} , and let d_K be the distance between x^{min} and its farthest neighbor in the model $neighbor_K(x^{min})$. That is,

$$d_{min} = d(x^{min}, x^{streaming}) \quad (10)$$

$$d_K = d(x^{min}, neighbor_K(x^{min})) \quad (11)$$

Therefore, once the real-time forecast is performed, if d_{min} is less than d_K it means that the actual $x^{streaming}$ is a more accurate neighbor of the x^{min} than its current neighbors in the model M . If it occurs, $x^{streaming}$ is added to the possible update buffer with its corresponding x^{min} as follows:

$$B = \{(x^{min}, x^{streaming}, d_{min})\} \quad (12)$$

The buffer is filled as many times as necessary by adding instances meeting the condition $d_{min} < d_K$ for each instance $x^{streaming}$ in the $set_{streaming}$.

The buffer of possible updates is checked at a specific time and therefore, the model is updated at that specific moment. In this work, this so-called specific time is one of the following five options: there are three temporal possibilities and two possibilities based on threshold errors. The temporal update can be performed: every day, every month or every three months. The other type of update can be performed when the error of the actual forecast is higher than a defined threshold. In this work, two thresholds have been considered: the $MAPE_{offline}$ made when obtaining the offline model or the $MAPE_{offline}$ plus its standard deviation.

When the so-called specific moment occurs, the K nearest instances of x^{min} in the current model M are selected along with all the $x^{streaming}$ associated to x_{min} in the buffer. Then, all neighbors both from the current model and from the buffer are sorted by distance and the K smallest ones are kept. All neighbors selected are the updated neighbors of the x^{min} in the model M . It is possible that all K neighbors of a x^{min} are updated in the model with data streaming instances.

An example of an update of the model when considering three neighbors ($K = 3$) is illustrated in the following equation:

$$M = \langle x^{min}, \langle y(x^{streaming_1}), y(neighbor_1(x^{min})), y(x^{streaming_2}) \rangle \rangle \quad (13)$$

It can be seen that the nearest and furthest neighbors are updated with two streaming instances of the $set_{streaming}$ and the neighbor one of the offline model is still considered a good neighbor, more specifically, it is the second closest neighbor.

Figure 1 represents graphically an example of the incremental updating where n_i is the abbreviation of $neighbor_i$ and x^s_i is the abbreviation of $x^{streaming_i}$.

IV. RESULTS

A. Description of the dataset

The time series used in this experimentation consists of the electrical energy consumption in megawatt (MW) in Spain. The time series has 497832 samples measured every 10 minutes. The historical data contain all samples from January 1st 2007 to June 21st 2016, after a pre-processing step (e.g.: adjustment of time shift days data) [24]. The whole dataset is divided in 3 sets: 70% for the offline part (70% for the $set_{neighbors}$ and 30% for the $set_{patterns}$, respectively) and 30% for the online data ($set_{streaming}$). Thus, the offline set consists of data from January 2007 to mid August 2013 and the streaming set from mid August 2013 to mid June 2016.

B. Experimental setting

The parameters used are the ones considered optimal after a validation process in [24] in order to make a comparison between a non-updating streaming model and an updating streaming model:

- 4 hours prediction horizon ($h=24$): $w=144$ and $K=4$.
- 8 hours prediction horizon ($h=48$): $w=288$ and $K=2$.
- 12 hours prediction horizon ($h=72$): $w=576$ and $K=4$.
- 24 hours prediction horizon ($h=144$): $w=864$ and $K=4$.

The experimentation is made on a machine cluster which is made up of 1 master node and 3 slaves. It has 4 Processor Intel(R) Core(TM) i7-5820K CPU with 48 cores and 120 GB of RAM memory.

C. Discussion of results

Table I shows the MAPE and standard deviation of the MAPE (std. dev. column) in percentage and the MAE in MW obtained when predicting the streaming set for each prediction horizon. Results are provided for each type of online neighbor update, depending on the time or depending on a threshold value. In particular, a daily, monthly and every three months update and an update based on two thresholds are tested. The threshold based updates are performed when the forecasting error is bigger than the MAPE obtained in the offline phase and when the forecasting error is bigger than the offline MAPE plus its standard deviation, abbreviated as $MAPE_{offl}$ and $MAPE_{offl} + \sigma_{MAPE_{offl}}$ respectively in Table I.

It can be observed in Table I that the lowest errors, both MAPE and MAE, are obtained when the daily update is used for all the prediction horizons. The update when using the $MAPE_{offline}$ as threshold provides as well very good results. The forecast errors obtained in the prediction of the streaming set when the model is updated using any type of update are always smaller than the errors obtained when the model is not updated. Regarding the temporal updating, it is clear that the model updates its neighbors in a more accurate way when the update time frequency is smaller. In other words, forecasting improves as more model updates are made from the new data streaming series, i.e., when the update is performed every day. This statement can also be applied when updating the model according to a certain threshold. The model is updated more

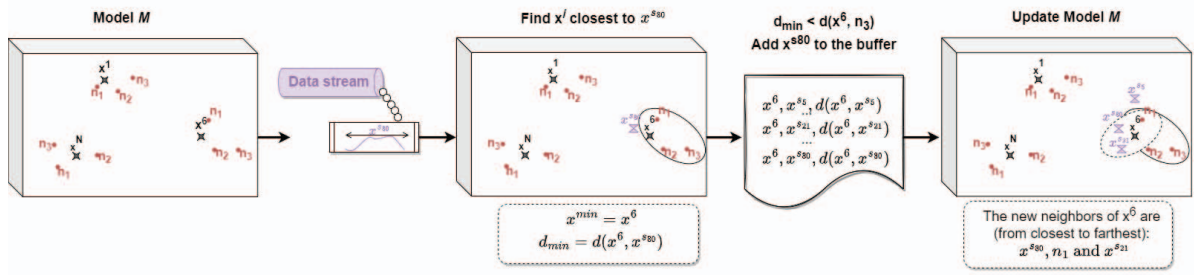


Fig. 1: Graphic representation of the neighbor update process

TABLE I: Metrics of errors depending on the type of updating for all prediction horizons

Updating	MAPE (%)	Std. dev. (%)	MAE (MW)
No update	2.4288	2.0745	670.1298
Day	2.1982	2.0138	605.7047
Month	2.2513	2.0269	620.5407
3 months	2.2710	2.0326	626.0198
$MAPE_{offline}$	2.1996	2.0132	606.1430
$MAPE_{offline} + \sigma_{MAPE_{offline}}$	2.2056	2.0117	607.8910

(a) Metrics of errors for $h=24$

Updating	MAPE (%)	Std. dev. (%)	MAE (MW)
No update	2.7617	2.0842	766.8640
Day	2.5499	2.0878	706.0045
Month	2.5958	2.0938	719.2312
3 months	2.6150	2.0972	724.4448
$MAPE_{offline}$	2.5529	2.0866	706.9218
$MAPE_{offline} + \sigma_{MAPE_{offline}}$	2.5671	2.0860	710.8549

(b) Metrics of errors for $h=48$

Updating	MAPE (%)	Std. dev. (%)	MAE (MW)
No update	3.3535	2.8200	933.9925
Day	3.1350	2.8039	872.1266
Month	3.1769	2.8285	883.6385
3 months	3.1920	2.8299	887.8336
$MAPE_{offline}$	3.1358	2.8036	872.3208
$MAPE_{offline} + \sigma_{MAPE_{offline}}$	3.1511	2.7982	876.3704

(c) Metrics of errors for $h=72$

Updating	MAPE (%)	Std. dev. (%)	MAE (MW)
No update	3.8466	3.6137	1072.8347
Day	3.5741	3.5292	998.0618
Month	3.6257	3.5659	1011.7691
3 months	3.6509	3.5558	1018.8665
$MAPE_{offline}$	3.5872	3.5247	1001.6575
$MAPE_{offline} + \sigma_{MAPE_{offline}}$	3.6236	3.5186	1011.9063

(d) Metrics of errors for $h=144$

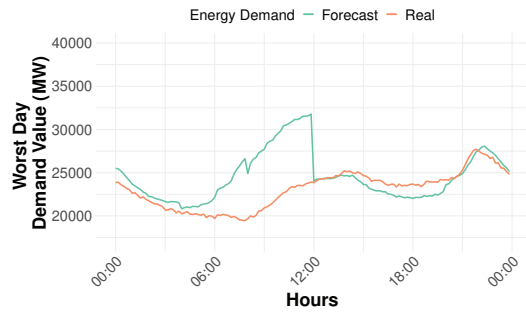
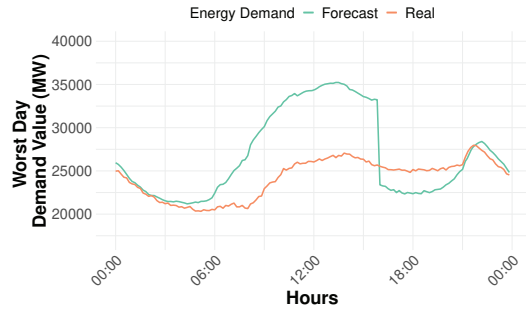
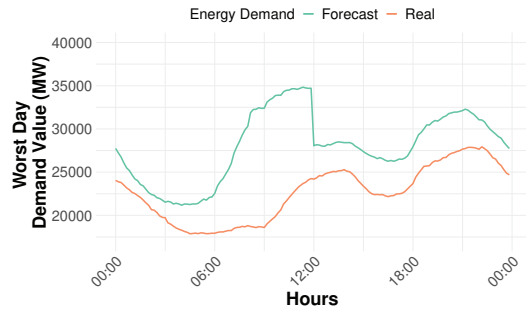
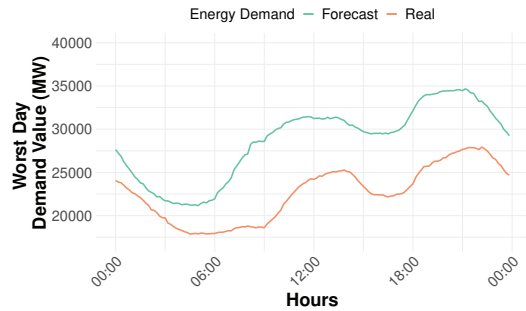
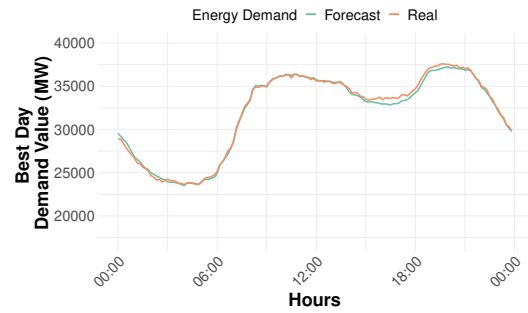
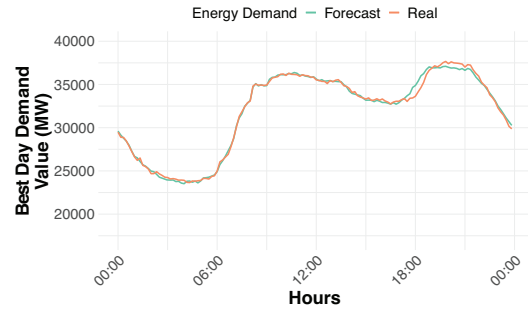
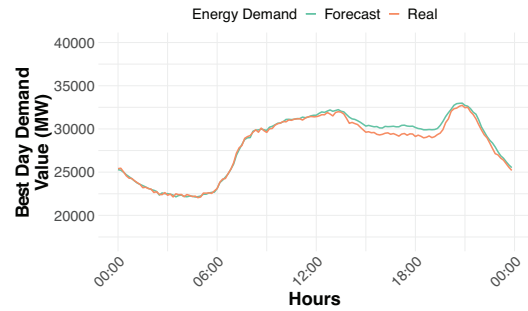
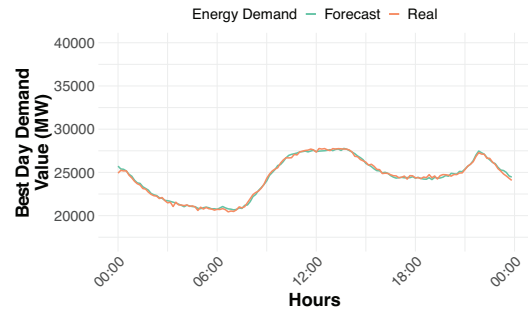
regularly using the $MAPE_{offline}$ as a threshold than using the $MAPE_{offline} + \sigma_{MAPE_{offline}}$, as the latter threshold is more difficult to reach.

Figure 2 shows the worst forecasts for each prediction horizon. They represent the real and forecasted electricity demand values in the vertical axis and the hours of the day in the horizontal axis. Each sub-figure includes the day and the horizon of the maximum MAPE. The worst forecasts do

not improve when the model is updated. In particular, the worst forecasts are obtained for the same date and at the same time and have the same value of MAPE for each prediction horizon whether the model is updated or not, unless for $h=144$. Figure 2a depicts the worst forecasting for $h=24$ with a MAPE of 33.0031%. Figure 2b refers to $h=48$ and has a MAPE of 31.2720%. The worst MAPE for $h=72$ is 24.3860%, as illustrates Figure 2c. The worst day predicted with $h=144$ without updating is in Figure 2d with 29.3278% of MAPE. All the streaming update options for $h=144$ coincide on the day with the worst forecast during December 25th 2013 with a MAPE of 27.0207%. All worst days correspond to public holidays in Spain: in summer for the prediction horizons 24 and 48 and, in winter for the prediction horizons 72 and 144. This is due to the fact that the proposed StreamWNN is a general-purpose approach and the prediction of holidays or special days requires the design of specific methods [28], [29]. For prediction horizons 24, 48 and 72, it can be observed abrupt changes at the last time sample of the horizon as the following forecasted values correspond to the next prediction horizon on the same day.

On the other hand, the MAPE value of the best forecasts for all prediction horizons decreases when updating is performed, comparing to not updating. The MAPE values obtained for the best forecasts when the model is not updated are 0.2464% for $h=24$, 0.4101% for $h=48$, 0.6002% for $h=72$ and 0.6548% for $h=144$. Figure 3 shows the best prediction made for each prediction horizon along with the days corresponding to the lowest MAPE value among all the different updates. In particular, Figure 3a represents the best forecasting for $h=24$, which is achieved only with daily update obtaining a MAPE of 0.2198%. Figure 3b is obtained for $h=48$ using both a daily and monthly update as well as an update with either of the two defined thresholds and getting a MAPE of 0.3208%. Figure 3c shows the best prediction for $h=72$, which is obtained whether the model is updated daily or using either of the two defined thresholds, and corresponds to a MAPE of 0.4493%. Figure 3d depicts the best forecasting for $h=144$, which is achieved only with daily update and has a MAPE of 0.6267%.

One very important aspect in streaming algorithms is to obtain results as fast as possible. Offline-online learning algorithms have to accomplish this requirement only in the online

(a) $h=24$. Horizon: from 08:00AM to 11:50AM. Day: 16/08/2015(b) $h=48$. Horizon: from 08:00AM to 15:50PM. Day: 16/08/2014(c) $h=72$. Horizon: from 00:00AM to 11:50AM. Day: 25/12/2013(d) $h=144$. Horizon: The whole day. Day: 25/12/2014Fig. 2: Days with the worst forecasts in terms of MAPE for each prediction horizon h (a) $h=24$. Horizon: from 08:00AM to 11:50AM. Day: 15/01/2015(b) $h=48$. Horizon: from 08:00AM to 15:50PM. Day: 23/01/2014(c) $h=72$. Horizon: from 00:00AM to 11:50AM. Day: 10/07/2015(d) $h=144$. Horizon: The whole day. Day: 31/05/2013Fig. 3: Days with the best forecasts in terms of MAPE for each prediction horizon h

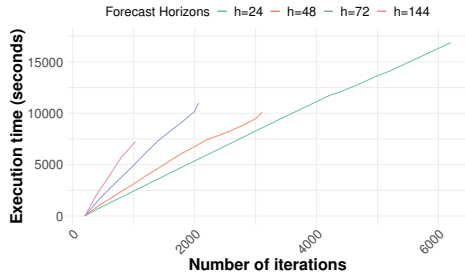


Fig. 4: Execution time versus number of iterations for daily neighbor updating in the online phase.

phase [26]. The historical model generated in the batch phase takes approximately 222 seconds for $h=24$, 167 seconds for $h=48$, 153 seconds for $h=72$ and 122 seconds for $h=144$. Figure 4 presents the execution time spent for the prediction of the streaming set for each forecast horizon when a daily update is carried out. It can be noted that the number of iterations for each prediction horizon is different as w and h values are different. The execution time increases linearly versus iterations for all prediction horizons, but the shorter the prediction horizon, the higher the scalability of the proposed prediction algorithm, as a line with a smaller slope can be seen in the Figure. The other updates have a similar behaviour.

As mentioned in Section III-C, during incremental learning the current model is updated using closer online data, considering closer a smaller distance between features. Therefore, each time the model is incrementally updated, the distance between the instances in the model and their new neighbors decreases, i.e., closer data are found, as Figure 5 illustrates. Figure 5d depicts the average distance of the model at each iteration for the prediction horizon $h=144$. It shows that when the model is not incrementally updated, the average distance remains the same during all the iterations. The monthly and every 3-month incremental update present a decrease in distance in a step-wise manner with a regular frequency, not like the $MAPE_{offline} + \sigma_{MAPE_{offline}}$ updates which can occur at any time. The daily and MAPE options update the model more times and thus the distance decreases more regularly. This is also evidenced by lower errors as also presented in Table I. The behaviour of the rest of the prediction horizons is similar, taking into account that the number of iterations is different for each prediction horizon.

V. CONCLUSIONS

In this paper, a new forecasting algorithm has been proposed to predict electricity demand time series in real time. As previous step to the prediction, the K-nearest neighbors algorithm to obtain an initial forecasting model has been applied using historical electricity consumption data. This model is composed of a set of patterns along with its k corresponding nearest neighbors. Thus, the final prediction has been computed using the K-nearest neighbors of the nearest

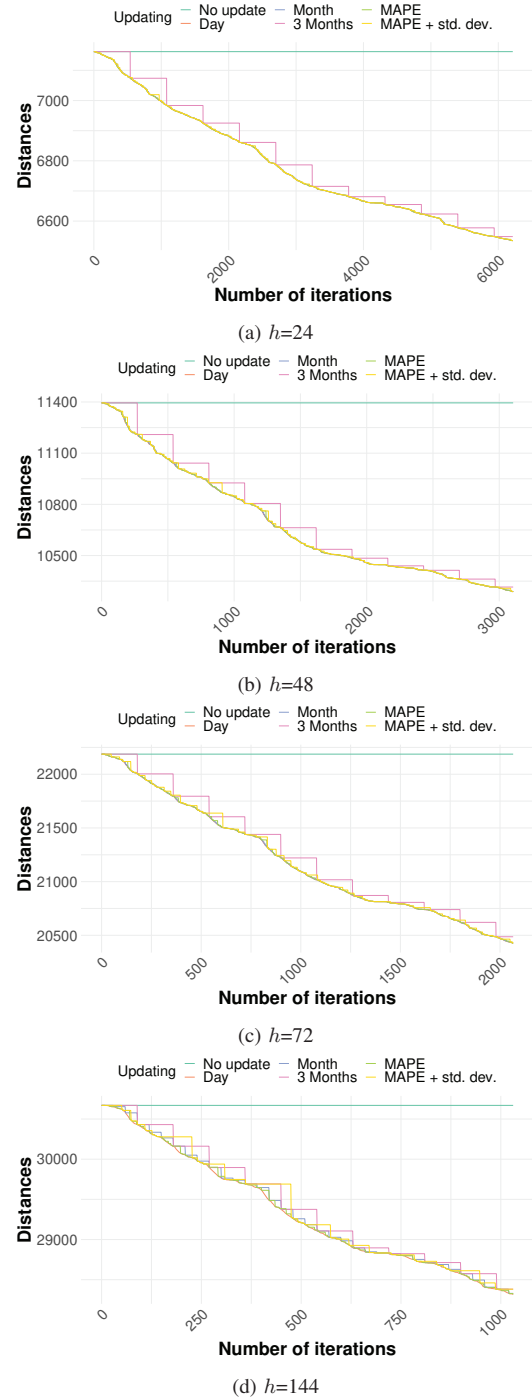


Fig. 5: Evolution of the average distance between neighbors for each update

pattern to the streaming data. In this way, predictions can be obtained in real time because it is not necessary to compute the K-nearest neighbors each time a prediction is performed as these K neighbors are already computed. The algorithm has been successfully applied, obtaining accurate predictions over time by incrementally updating the model. This update consists of updating the neighbors in real time with the new data received in streaming. Different neighbor updates have been tested and the runtimes for calculating the real-time prediction show that the algorithm is efficient as well as scalable with respect to the number of iterations.

The future works will be focused on detecting and differentiating novelties and outliers in the data streams. The novelties will be added to the model and the outliers will raise an alarm.

ACKNOWLEDGEMENTS

The authors would like to thank the Spanish Ministry of Science and Innovation for the support under the project PID2020-117954RB-C21, the European Regional Development Fund and Junta de Andalucía for projects PY20-00870 and UPO-138516 and the US-Spain Fulbright grant.

REFERENCES

- [1] M. González-Torres, L. Pérez-Lombard, J. Coronel, and I. Maestre, "A cross-country review on energy efficiency drivers," *Applied Energy*, vol. 289, p. 116681, 2021.
- [2] N. Shivaraman, S. Saki, Z. Liu, S. Ramanathan, A. Easwaran, and S. Steinhart, "Real-Time Energy Monitoring in IoT-enabled Mobile Devices," in *Proceedings of the 23th Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 991–994.
- [3] R. Sahal, J. G. Breslin, and M. I. Ali, "Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case," *Journal of Manufacturing Systems*, vol. 54, pp. 138–151, 2020.
- [4] F. Yao and Y. Wang, "Towards resilient and smart cities: A real-time urban analytical and geo-visual system for social media streaming data," *Sustainable Cities and Society*, vol. 63, p. 102448, 2020.
- [5] A. Bifet, B. Hammer, and F. Schleif, "Recent trends in streaming data analysis, concept drift and analysis of dynamic data sets," in *27th European Symposium on Artificial Neural Networks (ESANN)*, 2019, pp. 421–430.
- [6] E. Allothali, H. Alashwal, and S. Harous, "Data stream mining techniques: a review," *Telecommunication Computing Electronics and Control*, vol. 17, no. 2, pp. 728–737, 2019.
- [7] K. Roosa, Y. Lee, R. Luo, A. Kirpich, R. Rothenberg, J. Hyman, P. Yan, and G. Chowell, "Real-time forecasts of the COVID-19 epidemic in China from February 5th to February 24th, 2020," *Infectious Disease Modelling*, vol. 5, pp. 256–263, 2020.
- [8] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting Taxi-Passenger Demand Using Streaming Data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 1393–1402, 2013.
- [9] L. Melgar-García, M. T. Godinho, R. Espada, D. Gutiérrez-Avilés, I. S. Brito, F. Martínez-Álvarez, A. Troncoso, and C. Rubio-Escudero, "Discovering Spatio-Temporal Patterns in Precision Agriculture Based on Triclustering," in *Proceedings of the 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO)*. Cham: Springer International Publishing, 2021, pp. 226–236.
- [10] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, and A. Troncoso, "Discovering three-dimensional patterns in real-time from data streams: An online triclustering approach," *Information Sciences*, vol. 558, pp. 174–193, 2021.
- [11] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, and A. Troncoso, "High-Content Screening Images Streaming Analysis Using the STriGen Methodology," in *Proceedings of the 35th Annual Association for Computing Machinery Symposium on Applied Computing (SAC)*, 2020, p. 537–539.
- [12] R. Weron, "Electricity price forecasting: A review of the state-of-the-art with a look into the future," *International Journal of Forecasting*, vol. 30, no. 4, pp. 1030–1081, 2014.
- [13] R. Pérez-Chacón, R. L. Talavera-Llames, F. Martínez-Álvarez, and A. Troncoso, "Finding electric energy consumption patterns in big time series data," in *Proceedings of the 13th International Conference on Distributed Computing and Artificial Intelligence (DCAI)*, 2016, pp. 231–238.
- [14] R. Talavera-Llames, R. Pérez-Chacón, M. Martínez-Ballesteros, A. Troncoso, and F. Martínez-Álvarez, "A nearest neighbours-based algorithm for big time series data forecasting," in *Proceedings of the 11th International Conference on Hybrid Artificial Intelligent Systems (HAIS)*, 2016, pp. 174–185.
- [15] Z. M. Yaseen, A. El-shafie, O. Jaafar, H. A. Afan, and K. N. Sayl, "Artificial intelligence based models for stream-flow forecasting: 2000–2015," *Journal of Hydrology*, vol. 530, pp. 829–844, 2015.
- [16] P. Jiménez-Herrera, L. Melgar-García, G. Asencio-Cortés, and A. Troncoso, "A New Forecasting Algorithm Based on Neighbors for Streaming Electricity Time Series," in *Proceedings of the 15th International Conference on Hybrid Artificial Intelligent Systems (HAIS)*. Cham: Springer International Publishing, 2020, pp. 522–533.
- [17] X. Qiu, P. N. Suganthan, and G. A. Amaratunga, "Ensemble incremental learning random vector functional link network for short-term electric load forecasting," *Knowledge-Based Systems*, vol. 145, pp. 182–196, 2018.
- [18] K. Liu, Z. Li, C. Yao, J. Chen, K. Zhang, and M. Saifullah, "Coupling the k-nearest neighbor procedure with the Kalman filter for real-time updating of the hydraulic model in flood forecasting," *International Journal of Sediment Research*, vol. 31, no. 2, pp. 149–158, 2016.
- [19] J. Komma, G. Blöschl, and C. Reszler, "Soil moisture updating by Ensemble Kalman Filtering in real-time flood forecasting," *Journal of Hydrology*, vol. 357, no. 3, pp. 228–242, 2008.
- [20] R. Q. Thomas, R. J. Figueiredo, V. Daneshmand, B. J. Bookout, L. K. Puckett, and C. C. Carey, "A Near-Term Iterative Forecasting System Successfully Predicts Reservoir Hydrodynamics and Partitions Uncertainty in Real Time," *Water Resources Research*, vol. 56, no. 11, p. 20, 2020.
- [21] T. Nanda, B. Sahoo, and C. Chatterjee, "Enhancing real-time streamflow forecasts with wavelet-neural network based error-updating schemes and ECMWF meteorological predictions in Variable Infiltration Capacity model," *Journal of Hydrology*, vol. 575, pp. 890–910, 2019.
- [22] L. Yan, J. Feng, Y. Wu, and T. Hang, "Data-Driven Fast Real-Time Flood Forecasting Model for Processing Concept Drift," in *Cloud Computing, Smart Grid and Innovative Frontiers in Telecommunications*. Cham: Springer International Publishing, 2020, pp. 363–374.
- [23] L. Sun, Y. Ji, M. Zhu, F. Gu, F. Dai, and K. Li, "A new predictive method supporting streaming data with hybrid recurring concept drifts in process industry," *Computers & Industrial Engineering*, vol. 161, p. 107625, 2021.
- [24] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, and A. Troncoso, "Nearest neighbors-based forecasting for electricity demand time series in streaming," in *Advances in Artificial Intelligence*. Cham: Springer International Publishing, 2021, pp. 185–195.
- [25] J. Jeffers, J. Reinders, and A. Sodani, "Chapter 24 - Machine learning," in *Intel Xeon Phi Processor High Performance Programming (Second Edition)*. Morgan Kaufmann, 2016, pp. 527 – 548.
- [26] P. Larranaga, D. Atienza, J. D. Roza, A. Ogbechie, C. Puerto-Santana, and C. Bielza, *Industrial Applications of Machine Learning*. CRC Press, 2018.
- [27] A. A. Benczúr, L. Kocsis, and R. Pálócs, "Online machine learning algorithms over data streams," in *Encyclopedia of Big Data Technologies*. Springer International Publishing, 2019, pp. 1199–1207.
- [28] O. Trull, J. C. García-Díaz, and A. Troncoso, "Application of discrete-interval moving seasonalities to spanish electricity demand forecasting during easter," *Energies*, vol. 12, no. 6, p. 1083, 2019.
- [29] O. Trull, J. C. García-Díaz, and A. Troncoso, "One-day-ahead electricity demand forecasting in holidays using discrete-interval moving seasonalities," *Energy*, vol. 231, p. 120966, 2021.

5.1.11 | "A novel distributed forecasting method based on information fusion and incremental learning for streaming time series"

Authors: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A.

Publication type: Journal article. Under review.

A novel distributed forecasting method based on information fusion and incremental learning for streaming time series

Laura Melgar-García^{a,*}, David Gutiérrez-Avilés^b, Cristina Rubio-Escudero^b,
Alicia Troncoso^a

^a*Data Science & Big Data Lab, Pablo de Olavide University, ES-41013 Seville, Spain.*

^b*Department of Computer Science, University of Seville, Avda. Reina Mercedes s/n,
Seville, 41012, Spain.*

Abstract

Real-time algorithms have to adapt and adjust to new incoming patterns to provide timely and accurate responses. This paper presents a new distributed forecasting algorithm for streaming time series called StreamWNN. StreamWNN starts with an offline stage in which a forecasting model based on tuples of information fusion is created with historical data. In particular, this model consists of the fusion of patterns composed of past values of the time series with the future values of their k-nearest neighbors. Afterwards, streaming data starts to arrive. The model is incrementally updated in the online stage using a buffer with streaming data that more accurately matches the current model patterns. The model can be updated daily, monthly, quarterly or based on error thresholds. The methodology has been applied to Spanish electricity demand time series providing more accurate results when the model is updated incrementally. The best results are obtained with the daily update of the model, resulting in an error between 2% and 3.5% depending on the prediction horizon. The model provides better results than other algorithms.

Keywords: real-time forecasting, incremental learning, streaming time series, electricity demand.

1. Introduction

Nowadays, researchers and companies are paying more and more attention to time series forecasting, due to the large number of existing applications based on time-indexed data. Moreover, as this data is mostly from electronic devices, the time series are usually very long and are measured in minutes or even seconds,
5 resulting in big data time series. These characteristics open up a huge field of

*Corresponding author

Email address: atrolor@upo.es (Alicia Troncoso)

study in the big data paradigm with the aim of obtaining more accurate and efficient predictive models.

Big data streaming is becoming one of the most widely used big data trends in recent years. Of the three "Vs" that define big data, namely variety, volume and velocity, the most important characteristic in this type of algorithm is velocity. Velocity refers to large continuous flows of data, called streams. Streams need to be processed and modeled in a special way considering specific requirements [1]. The real-time decision making process contributes to the challenge of developing safe and efficient smart cities with timely responses [2].

In addition to the condition of providing fast results when working in streaming mode, it is important to develop a model that must be always ready to give responses. However, streaming flows usually variate and suffer transformations which could lead to a non-accurate response if the model does not adapt to them [3]. For this reason, some streaming models include the option to update themselves taking into account the data arriving online in order to fit as best as possible to their new behavior. It is worth mentioning that even if the streaming model updates, it has to provide, as well, timely results.

In this research, a new forecasting algorithm for time series in streaming named StreamWNN is proposed. The algorithm is made up of two phases: a batch phase and an online phase. In the batch phase, a model from historical data is created, and in the online phase the forecasts are made and the model is updated in real time. Therefore, the online phase fits the model to current streaming data. Both phases are based on the fusion of relevant information for the time series such as patterns and the classes of their k-nearest neighbors. The main innovation of StreamWNN compared to models in the literature is the use of a data structure that merges information of different nature. This information fusion from historical data allows both the real-time prediction and the online incremental learning to adapt the model to new incoming patterns in the streaming data. StreamWNN is applied to electrical energy consumption data in Spain from 2007 to 2016. The results show that the fusion of information related to patterns and their nearest neighbors is adequate to be able to incrementally update the model as well as to make predictions in a computationally efficient way.

The rest of the article is structured as follows. Section 2 presents a review of streaming time series forecasting algorithms. In Section 3 the whole methodology of the StreamWNN is introduced, including the incremental updating of the prediction model based on nearest neighbors. Section 4 defines the time series used and includes the discussion of the forecasting and execution time results. The paper finishes with some final conclusions and ideas for future approaches in Section 5.

2. Related work

Currently, all the topics related to streaming data are being widely investigated. The concept of streaming data is part of the big data environment and therefore, the challenges related to distributed computation, fault-tolerant

computing paradigms, and architectures are still open. These technologies allow researchers to develop streaming data processing and analysis techniques to obtain accurate knowledge [4]. Concretely, understanding streaming events and their behavior is a critical task to accomplish [5].

55 Time series forecasting is crucial for streaming data analytics due to the large amount of existing massive time series data from different sources and its multiple applications. A wide variety of proposals for forecasting of data streams can be found in the literature. In [6], an Apache Spark framework based on the Lambda big data architecture was proposed. The authors implemented a vector
60 auto-regression algorithm using that architecture to achieve better adaptability and efficiency of the model. The authors in [7] proposed a new method for time series forecasting based on a dynamic ensemble selection framework. This framework consisted of a set of baseline regressors trained offline and a set of meta learners that predicted the errors of the first ones. The proposal in
65 [8] described a hybrid model that combined statistical learning methods such as ARIMA with machine learning models as the gradient descent optimization algorithm, obtaining promising results. The authors stated that the statistical and machine learning techniques, correctly combined, provide better results than separately, since the hybrid system will have the adaptability character-
70 istic of machine learning models and the accuracy of the statistical ones. The authors in [9] presented a methodology for time series data modeling and prediction. They proposed a double sliding window as a data-driven model and a combination of gene expression programming and a colony climbing meta-heuristic. The benchmark results obtained better results than the hierarchical
75 temporal memory algorithm [10] used as a baseline. Time series forecasting has also various real-life applications, as in [11] where the authors presented an Android-based mobile application that predicts the physical activity of the user based on the steps taken and calorie consumption. In [12], the authors applied a long-short term memory deep learning architecture using the Coronavirus based
80 optimization algorithm [13] to calculate the optimal hyper-parameters in order to predict the deformation of the Hoa Binh hydropower dam in Vietnam.

Fusion of the data from variables of different sources or the fusion of different methods to improve the prediction is also a state-of-the-art strategy [14]. The authors in [15] presented a hybrid algorithm to predict both univariate and
85 multivariate time series based on a fusion of clustering, classification models and a regressor that makes the final prediction. An adaptation of this proposal to the streaming environment can be found in [16]. In [17] fusion methods are proposed for the aggregation of emergent patterns obtained by an evolutionary fuzzy system applied on each data stream. In [18] a Hoeffding adaptive tree is
90 combined with an ensemble of C4.5 decision trees from massive data streams without increasing the time to obtain the model. In this context, statistical methods are also widely used. A comparative study of several well-known statistical methods such as Vector Autoregressive (VAR), Vector Moving Average (VMA), Vector Autoregressive Moving Average (VARMA), and Theta are compared with recurrent neural network models in [19]. Explainable models are
95 trendy in the machine learning research field. Focusing on time series forecast-

ing, a proposal of a linear regression-based forecasting method that explains the predictions through regression trees can be found in [20].

Pattern recognition is also a relevant topic in the streaming environment
 100 for a direct application such as anomaly and novelty detection and it is also
 the core of a comprehensive collection of streaming time series forecasting algo-
 rithms. In [21] the authors presented an online triclustering algorithm to detect
 three-dimensional behavior patterns from streams. The particularity of this pro-
 105 posal was the high dimensionality of the streams as they were formed by several
 variables indexed over time. This approach has been successfully applied to real
 problems such as precision agriculture [22], and the online analysis of cellular
 high-content screening images [23]. In addition, streaming data can vary and
 change its behavior patterns as time passes. It is crucial to have a streaming
 model that adapts to the new data flows. A wavelet-neural network with an
 110 error-updating scheme was proposed for meteorological predictions in streaming
 in [24]. The need for systematically updating the errors of data streams was
 shown. In order to reduce the computational cost of the optimization algorithm
 used in transductive support vector machine to deal with big data classification
 an incremental learning is proposed in [25]. Authors in [26] proposed an incre-
 115 mental update method based on Support Vector Machines (SVM) and Gated
 Recurrent Unit (GRU), considering concept drift for forecasting in real-time.
 Training models were updated based on the error between batch test results
 and real values. [27] proposed a novel incremental learning approach to identify
 individuals of interest from streaming face data. This approach fed with stream
 120 data, simultaneously predict and update classifiers, combining a deep features
 encoder with ensembles of SVM. Concept drift is also addressed through adap-
 tive decision forest in [28]. The authors used three parallel forests to keep his-
 torical knowledge and the classification is made with the forest which achieves
 the best classification accuracy. In [29] an algorithm to enable incremental class
 125 learning in hierarchical classification was presented. In particular, a structure
 that grows horizontally and vertically in an online way when online data arrives
 is proposed, showing a better performance when comparing with other algo-
 rithms trained using offline data. In [30], the authors developed an incremental
 online multi-label classification method. This method is based on a weighted
 130 clustering model and the weights of the samples decrease to adapt to the change
 of data. Many literature approaches for model updating in the streaming envi-
 ronment are based on external algorithms such as the Kalman filters. Authors
 in [31] presented a coupled methodology of the k-nearest neighbor algorithm
 with a Kalman filter for real-time flood forecasting. The state transition matrix
 135 of the Kalman filter was recalculated using the forecasting method to improve
 the performance of the model and obtain accurate results. Another ensemble
 Kalman filter was used in [32]. The ensemble was used as a data assimilation
 algorithm to update water temperature forecasting considering sensor instances.
 140 The proposed algorithm is applied to electrical energy forecasting. This ap-
 plication field has been studied mainly in the batch or traditional time series
 forecasting mode. On this matter, the authors in [33] presented a big data
 approach for electricity consumption of the sensors in smart cities. Several pro-

posals have adapted nearest neighbor proposals to the big data paradigm using the Apache Spark distributed computation framework [34, 35] for electricity demand forecasting. In the real-time environment, in [36] a new algorithm to predict energy demand time series in streaming was introduced. It used three different algorithms in streaming: k-means, nearest neighbors and naive Bayes. Deep learning models variants have been applied to the electric field, even in a real-time scenario [37]. In [38], the authors developed an innovative deep learning architecture for real-time forecasting of highly volatile and high-frequency electricity price time series. Furthermore, in [38] the authors proposed a multi-scale convolutional and long-short term hybrid neural network for short-term electricity price prediction. In [39] a recurrent neural model to predict the electricity demand in Cambodia was presented. A post-processing consisting of searching for correlated climate variables was carried out to improve the predictions. Finally, in [40] the authors developed a new streaming k-nearest neighbor algorithm for electricity demand in streaming. The algorithm was successfully tested using Spanish electricity demand data. A complete and updated review of the machine learning applications to electrical load forecasting was presented in [41].

3. Material and methods

The StreamWNN methodology is based on a data structure fusing information related to the k-nearest neighbors, which is based on the idea that nearest data share similar properties or characteristics.

The goal of this time series forecasting problem is to predict the next values considering past ones. The time series used are in N instances where the i -th instance consists of x^i representing its w past features and y^i representing its h next classes, i.e., next h values to predict:

$$X_t = \{(x^1, y^1), \dots, (x^N, y^N)\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \quad (1)$$

The dataset is divided into three chronologically ordered sets: $set_{neighbors}$, $set_{patterns}$ and $set_{streaming}$. The StreamWNN algorithm uses the offline-online learning approach to model data streams. In the first phase, the requirements for streaming models must not be met. However, in order to consider the StreamWNN as an algorithm that can work in real-time, all requirements for such algorithms must be fulfilled in the streaming or online phase. These requisites are described in [42]: only a limited set of past data can be stored; the model has to adapt to concept drift quickly and has to be always ready to make predictions; the model has to work in distributed computational environments so that its computation is fast. In this research work, an incremental learning approach during the online phase is presented. In this way, the algorithm is always up to date.

The proposed algorithm is built on Apache Spark 2.3.4 and uses HDFS file system on Hadoop 2.7.7 and the Kappa Architecture on Apache Kafka 2.11 as streaming platform, i.e., a single pipeline is specifically designed for the job.

The methodology of this algorithm is defined in three phases. Section 3.1 describes the offline phase, Section 3.2 the online phase and how forecasting is made in real time and the incremental learning of the algorithm is presented in Section 3.3.

3.1. Offline learning model

The first task of the algorithm is to create an offline learning model in distributed batch processing. This model is the basis for the online phase where streaming data starts to arrive.

The offline historical data is divided chronologically into $set_{neighbors}$ and $set_{patterns}$ with around 70% and 30% of the data respectively. The batch model consists of the fusion of each feature instance of the $set_{patterns}$ with its K closest instances of the $set_{neighbors}$. Thus, the offline model M is represented by tuples of information as follows:

$$M = \langle x^i, \langle y(neighbor_1(x^i)), \dots, y(neighbor_K(x^i)) \rangle \rangle \quad (2)$$

where x^i are the features of the i -th instance of the $set_{patterns}$ and $y(neighbor_j(x^i))$ are the classes of the instance of the $set_{neighbors}$ selected as the j -th closest neighbor of x^i .

The distance used to measure the nearness between the attributes in order to generate the model is the euclidean distance. This distance is represented as $d(x^i, x^j)$ where x^i refers to the features of the i -th instance of the $set_{patterns}$ and x^j to the features of the j -th instance of the $set_{neighbors}$.

After the creation of the offline model, the next h are predicted. In the offline phase, predictions for each value l in the prediction horizon of length h are computed by:

$$\hat{y}^l = \frac{1}{\sum_{j=1}^K \alpha_j} \sum_{j=1}^K \alpha_j y(neighbor_j(x^i))^l \quad 1 \leq l \leq h \quad (3)$$

where α_j is the distance of $d(x^i, x^j)$ defined as:

$$\alpha_j = \frac{1}{d(x^i, x^j)^2} \quad (4)$$

The error between predictions and actual data is computed to evaluate the performance of the offline model. In this research, the selected error metrics are the $MAPE$ Mean Absolute Percentage Error and the MAE Mean Absolute Error.

Errors $MAPE_i$ and MAE_i are calculated for each i -th instance predicted of the set of patterns. Finally, the offline phase gets $MAPE_{offline}$ and $MAE_{offline}$ that are the mean value of each error. These error metrics are defined as:

$$MAPE_i = \frac{1}{h} \sum_{l=1}^h \left| \frac{y^l - \hat{y}^l}{y^l} \right| \times 100 \quad 1 \leq l \leq h \quad (5)$$

$$MAE_i = \frac{1}{h} \sum_{l=1}^h |y^l - \hat{y}^l| \quad 1 \leq l \leq h \quad (6)$$

where y^l is the l class of the i -th instance and \hat{y}^l corresponds to the offline prediction defined by Equation 3.

In brief, the offline model is generated from the Algorithm 1.

Algorithm 1: Offline phase

Result: Offline model M , $MAPE_{offline}$, $MAE_{offline}$
 $w \leftarrow$ Window of past features
 $h \leftarrow$ Prediction horizon or next values to predict
 $K \leftarrow$ Number of neighbors to select
distances $\leftarrow []$
 $M \leftarrow []$
for each x^i in $set_{patterns}$ **do**
 for each x^j in $set_{neighbors}$ **do**
 distances \leftarrow add(distances, $d(x^i, x^j)$);
 end
 $\{neighbors_1(x^i), \dots, neighbors_K(x^i)\} \leftarrow K_smallest(distances, K)$
 $M \leftarrow$ add(M , $< x^i, y(neighbors_j(x^i)) >$)
 $\hat{y}^l \leftarrow$ predict(M, x^i)
 $MAPE_i \leftarrow$ MAPE(y^l, \hat{y}^l)
 $MAE_i \leftarrow$ MAE(y^l, \hat{y}^l)
end
 $MAPE_{offline} \leftarrow mean(MAPE_i)$
 $MAE_{offline} \leftarrow mean(MAE_i)$

220 **3.2. Real-time forecasting**

The second phase of the StreamWNN algorithm is the online phase. This phase starts once the offline model from Equation 2 is computed. This model is the base of the online phase. Streaming or online data is received in real-time and collected in groups of features (of length w), this data can be called $x^{streaming}$. For each $x^{streaming}$, the euclidean distances between it and all x^i features of the model M are calculated. The one with the minimum distance d_{min} is selected as the nearest x^i , called x^{min} from now on.

Real-time forecasts are calculated when $x^{streaming}$ is associated with its nearest x^{min} . In this case, online predictions take into account not only the K nearest neighbours of x^{min} but also x^{min} itself as a neighbor for each value l to

forecast in the prediction horizon of length h :

$$\hat{y}^l = \frac{1}{(\sum_{j=1}^K \alpha_j) + \alpha_{min}} \left(\left(\sum_{j=1}^K \alpha_j y(\text{neighbor}_j(x^{min}))^l \right) + \alpha_{min} y(x^{min})^l \right) \quad (7)$$

where α_{min} is represented as:

$$\alpha_{min} = \frac{1}{d(x^{min}, x^{streaming})^2} \quad (8)$$

Subsequently, the actual class of the online data $x^{streaming}$ arrives and the metric errors are calculated. This metrics use the Equation 5 and 6 to obtain the $MAPE_{streaming}$ and $MAE_{streaming}$ for each i -th instance of the set of streaming. Therefore, at the end of the online stage, the final $MAPE_{online}$ and MAE_{online} errors can be calculated. These errors are the average of all $MAPE_{streaming}$ and $MAE_{streaming}$, respectively.

Up to this stage, the offline or historical data of the model M is the only data that the real-time forecasting takes into account [40]. Section 3.3 presents the incremental learning approach of the StreamWNN. This innovative methodology focuses on obtaining more accurate predictions in real-time by updating the model with new streaming patterns.

3.3. Incremental learning

The goal of online incremental learning is to keep the model up to date to prevent the model from aging and becoming unsuitable for real-time predictions. This step is very important as new streaming patterns may appear and need to be added to the model.

The incremental learning is performed by updating the neighbors of x^{min} , i.e., $\text{neighbor}_j(x^{min})$ with $j = 1, \dots, K$. This is the reason why the model is said to be internally updated. The dimensions of the M model are conserved, however, the components are updated in real-time taking into account new incoming patterns.

The update of neighbors uses a possible update buffer B . Let d_{min} be the distance between $x^{streaming}$ and its nearest pattern x^{min} , and let d_K be the distance between x^{min} and its farthest neighbor in the model $\text{neighbor}_K(x^{min})$. That is,

$$d_{min} = d(x^{min}, x^{streaming}) \quad (9)$$

$$d_K = d(x^{min}, \text{neighbor}_K(x^{min})) \quad (10)$$

Real-time prediction is performed and d_{min} and d_K are compared. In the case where d_{min} is less than d_K , the actual $x^{streaming}$ is considered to be a more accurate neighbor of the x^{min} than its current neighbors in the model M . If it happens, $x^{streaming}$ is added to the possible update buffer with its corresponding x^{min} as follows:

$$B = \{(x^{min}, x^{streaming}, d_{min})\} \quad (11)$$

The buffer is filled as many times as necessary by adding instances meeting the condition $d_{min} < d_K$ for each instance $x^{streaming}$ in the $set_{streaming}$.

265 To perform the model update there are five options to choose from: three based on time and two based on exceeding an error threshold. The update for the time-based options can be conducted on a daily, monthly or very three months basis. The threshold-base options can be performed when the error of the actual forecast is higher than a defined threshold. In this research, the
270 $MAPE_{offline}$ obtained when the offline model is computed is the first threshold-based incremental learning option. The second one is the $MAPE_{offline}$ plus its standard deviation.

When the so-called specific moment occurs, the K nearest instances of x^{min} in the current model M are selected along with all the $x^{streaming}$ associated
275 to x_{min} in the buffer. Then, all neighbors both from the current model and from the buffer are sorted by distance and the K smallest ones are kept. All neighbors selected are the updated neighbors of the x^{min} in the model M . It is possible that all K neighbors of a x^{min} are updated in the model with data streaming instances.

280 The new equation presents an example of updating of the model when the parameter K , used to refer to the number of neighbors, is $K=3$:

$$M = \langle x^{min}, \langle y(x^{streaming_1}), y(neighbor_1(x^{min})), y(x^{streaming_2}) \rangle \rangle \quad (12)$$

In this example the nearest and farthest neighbors are updated with two streaming instances of the set of the streaming. The neighbor one of the offline model is still considered a good neighbor.

285 A graphical representation of the incremental learning approach of the StreamWNN is illustrated in Figure 1, where n_i is the abbreviation of $neighbor_i$ and x^{s_i} is the abbreviation of $x^{streaming_i}$.

The general online phase is represented in Algorithm 2 including the neighbors updating described in Algorithm 3.

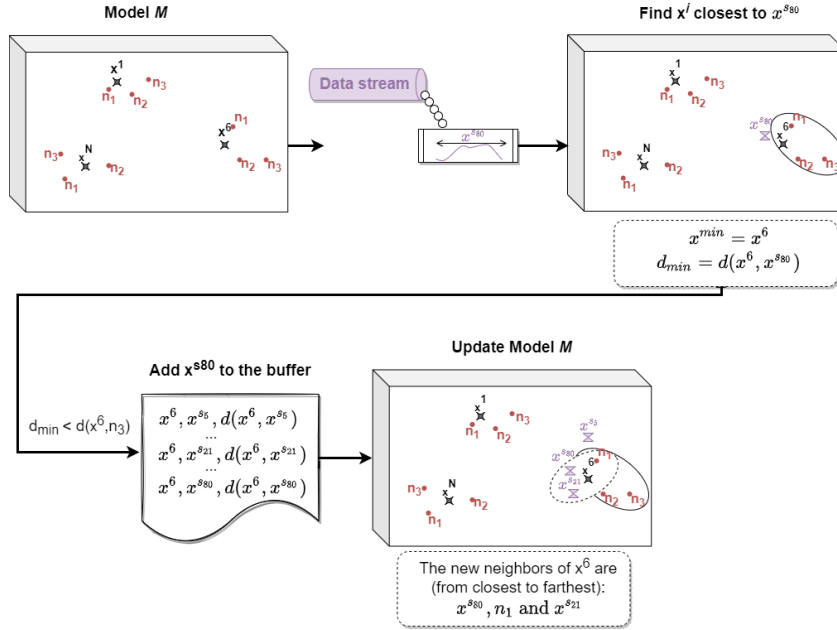


Figure 1: Graphic representation of the neighbor updating process

Algorithm 2: Online phase including neighbor updating

Result: updated M model, forecasts $\hat{y}^{streaming}$ and $MAPE_{online}$ and MAE_{online} errors of the $set_{streaming}$

$M \leftarrow$ Offline model from Algorithm 1

distances $\leftarrow []$

for each $x^{streaming}$ **that arrives do**

for each x^i **in** M **do**

distances \leftarrow add(distances, $d(x^i, x^{streaming})$)

end

$x^{min} \leftarrow$ K-smallest (distances, 1)

$d_{min} \leftarrow d(x^{min}, x^{streaming})$

$\hat{y}^{streaming} \leftarrow$ predict ($M, x^{streaming}$)

$MAPE_{streaming} \leftarrow$ MAPE($y^{streaming}, \hat{y}^{streaming}$)

$MAE_{streaming} \leftarrow$ MAE($y^{streaming}, \hat{y}^{streaming}$)

$M \leftarrow$ Update neighbors from Algorithm 3

end

$MAPE_{online} \leftarrow$ mean($MAPE_{streaming}$)

$MAE_{online} \leftarrow$ mean($MAE_{streaming}$)

Algorithm 3: Neighbor Updating

Result: updated M model

$update_{moment} \leftarrow \text{select}(\text{day, month, three-months, } MAPE_{offline},$
 $MAPE_{offline} + \sigma_{MAPE_{offline}})$

$B \leftarrow []$

$distances_{M \cup B} \leftarrow []$

if $d_{min} < d(x^{min}, neighbor_K(x^{min}))$ **then**

$B \leftarrow \text{add}(B, < x^{min}, x^{streaming}, d_{min} >)$

if $update_{moment}$ and B not empty **then**

$neighbors(x^{min}) \leftarrow \{neighbor_1(x^{min}), \dots, neighbor_K(x^{min})\}$

$distances_{M \cup B} \leftarrow \text{add}(distances_{M \cup B}, d(x^{min}, neighbors(x^{min})))$

$distances_{M \cup B} \leftarrow \text{add}(distances_{M \cup B}, d_{min})$

$new_neighbors(x^{min}) \leftarrow K_smallest(distances_{M \cup B}, K)$

$neighbors(x^{min}) \leftarrow new_neighbors(x^{min})$

$M \leftarrow \text{update}(M, < x^{min}, neighbors(x^{min}) >)$

end

end

4. Results and Discussion

The dataset used in this research work is a time series of the electrical energy consumption in megawatt (MW) in Spain. The time series has 497,832 samples measured every 10 minutes and preprocessed following the considerations in [40]. The time series starts in January 1st 2007 and ends in June 21st 2016. The first 70% of the dataset is used for the offline phase as historical data. The rest of the dataset is used for the online or streaming phase.

The parameters values used in this experimentation are the optimal ones in [40] with the idea of comparing the results of the streaming model without updates and the results obtained from the incremental learning updates developed in this research:

- For prediction horizon of 4 hours ($h=24$): $w=144$ and $K=4$.
- For prediction horizon of 8 hours ($h=48$): $w=288$ and $K=2$.
- For prediction horizon of 12 hours ($h=72$): $w=576$ and $K=4$.
- For prediction horizon of 24 hours ($h=144$): $w=864$ and $K=4$.

The experimentation is performed on the cluster of the Data Science and Big Data Laboratory in Pablo de Olavide University which is made up of 1 master node and 3 slaves. It has 4 Processor Intel(R) Core(TM) i7-5820K CPU with 48 cores and 120 GB of RAM memory.

4.1. Discussion of results

Table 1 presents the errors for each prediction horizon after forecasting the streaming set for each type of update. The errors are reported with the following metrics: MAPE in percent, standard deviation of the MAPE in the column
 315 Std. dev. MAPE and the MAE in MW. The time-dependent update types are the daily, monthly and quarterly updates. The threshold-based updates are performed when the forecast error is larger than the MAPE obtained in the offline phase ($MAPE_{offline}$) and when the forecast error is larger than the offline MAPE plus its the standard deviation ($MAPE_{offline} + \sigma_{MAPE_{offline}}$).

320 Considering the results in Table 1, the lowest MAPE and MAE errors are reached for the daily update at all the prediction horizons. The update using the $MAPE_{offline}$ threshold also achieves very good results. Forecast errors when the model is not updated are larger than those obtained when any type of incremental learning update is performed on the model. The streaming forecasts
 325 become more accurate as more updates are performed on the online model. For example, the monthly update provides better results than the quarterly update, since the first one updates the model more and, therefore, the model adjusts more to the new incoming data. Similarly, results are more accurate for the $MAPE_{offline}$ threshold than the $MAPE_{offline} + \sigma_{MAPE_{offline}}$ threshold, as
 330 the latter threshold is more difficult to reach and thus the model is updated fewer times.

The worst forecasts do not improve when the model is updated. For each prediction horizon, the worst forecasts coincide in date and time and provide the same MAPE value regardless of whether the model is updated or not, except
 335 for $h=144$. The worst prediction for $h=24$ is August 16th 2015 from 8:00AM to 11:50AM with a MAPE of 33.003%. For $h=48$ it is August 16th 2014 from 8:00AM to 15:50PM with a MAPE of 31.272%. For $h=72$ it is December 25th 2013 from 00:00AM to 11:50AM with a MAPE of 24.386%. The worst predicted day with $h=144$ with no update is the whole December 25th 2014 with 29.328%
 340 as MAPE and the December 25th 2013 with a MAPE of 27.021% for all different streaming updates.

The daily MAPE when the model is updated every day for each prediction horizon is shown in Figure 2. The smallest error is achieved when the prediction horizon is 24 (Figure 2a). In this case, the maximum MAPE is 10.161% and
 345 occurs on August 16th 2015, i.e. the day when the worst prediction is obtained for the next 4 hours. For the rest of the prediction horizons, the highest MAPE is also reached on the day with the worst forecast for the next 4 hours.

The MAPE value of the best forecasts decreases when any type of update is performed on the model. The error MAPE metrics for the best predictions of
 350 the model without update are 0.246%, 0.411%, 0.601% and 0.655% for $h=24$, $h=48$, $h=72$ and $h=144$, respectively. The best predictions achieved for each prediction horizon are represented in Figure 3. For $h=24$ the best prediction is obtained with the daily update providing a MAPE of 0.219%. This evolution throughout the day is in Figure 3a. For $h=48$ the best MAPE is reached for the
 355 daily, monthly and the two threshold-dependent types of update. The MAPE

Table 1: Error metrics for each update and prediction horizon

Update type	MAPE (%)	Std. dev. MAPE	MAE (MW)
No update	2.428	2.074	670.129
Daily	2.198	2.014	605.705
Monthly	2.251	2.027	620.541
Quarterly	2.271	2.033	626.019
$MAPE_{offline}$	2.199	2.013	606.143
$MAPE_{offline} + \sigma_{MAPE_{offline}}$	2.206	2.012	607.891

(a) Errors for $h=24$

Update type	MAPE (%)	Std. dev. MAPE	MAE (MW)
No update	2.762	2.084	766.864
Daily	2.550	2.088	706.004
Monthly	2.596	2.094	719.231
Quarterly	2.615	2.097	724.445
$MAPE_{offline}$	2.553	2.087	706.922
$MAPE_{offline} + \sigma_{MAPE_{offline}}$	2.567	2.086	710.855

(b) Errors for $h=48$

Update type	MAPE (%)	Std. dev. MAPE	MAE (MW)
No update	3.354	2.820	933.992
Daily	3.135	2.804	872.127
Monthly	3.177	2.828	883.638
Quarterly	3.192	2.830	887.834
$MAPE_{offline}$	3.136	2.804	872.321
$MAPE_{offline} + \sigma_{MAPE_{offline}}$	3.151	2.798	876.370

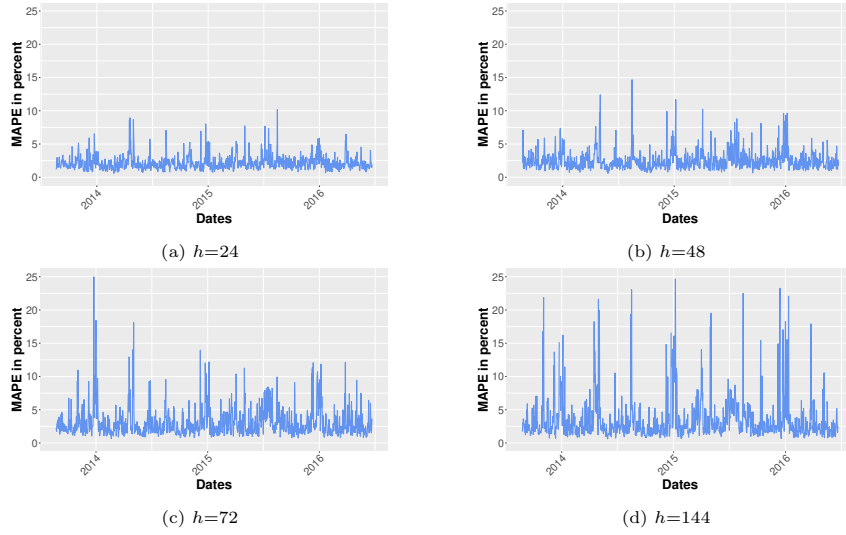
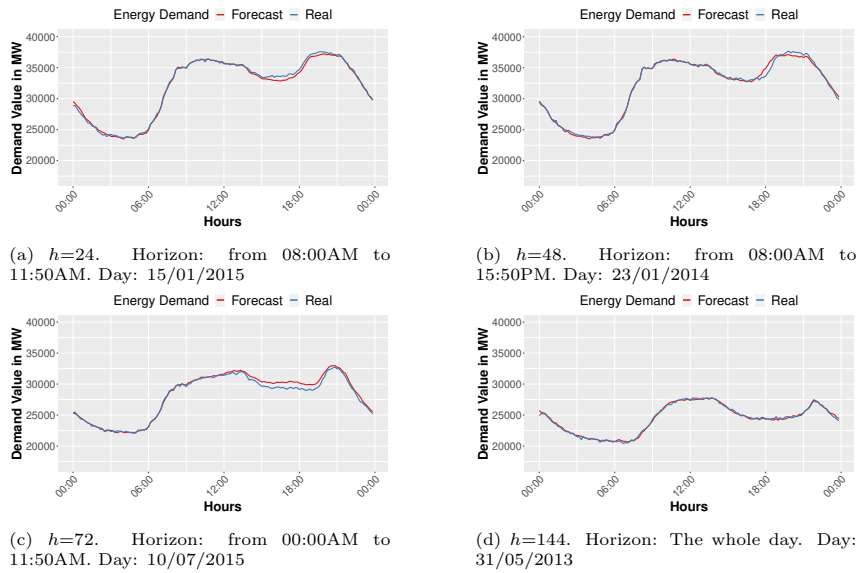
(c) Errors for $h=72$

Update type	MAPE (%)	Std. dev. MAPE	MAE (MW)
No update	3.847	3.614	1072.835
Daily	3.574	3.529	998.062
Monthly	3.626	3.566	1011.769
Quarterly	3.651	3.556	1018.866
$MAPE_{offline}$	3.587	3.528	1001.657
$MAPE_{offline} + \sigma_{MAPE_{offline}}$	3.624	3.519	1011.906

(d) Errors for $h=144$

is 0.321% and it is illustrated in Figure 3b. For $h=72$, the best prediction has a MAPE of 0.449% and is achieved for the daily and for both of the two threshold updates. It is depicted in Figure 3c. The best forecast for $h=144$ is in Figure 3d. It is obtained only with the daily update and has a MAPE of 0.627%.

The incremental learning approach implemented in the StreamWNN offers

Figure 2: Daily MAPE for each h prediction horizonFigure 3: Days with the best forecasts for each h prediction horizon

the possibility to add new patterns from streaming data to the model. This innovative k-nearest neighbor based technique improves online predictions. Figure 4 illustrates the evolution of the average euclidean distance between model neighbors as iterations pass for $h=144$. When this learning technique is not performed, i.e., no model update is conducted, this distance remains the same for all the iterations. However, when the incremental learning update is performed this distance decreases. The updates that achieves the lowest average distance is the daily update. This assumption is also confirmed with the type of update that reaches the lowest errors. Time-dependent updates reduce this average distance in a stepwise fashion. The representations of the mean distance between neighbors and the number of iterations follow the same distribution for the remaining prediction horizons.

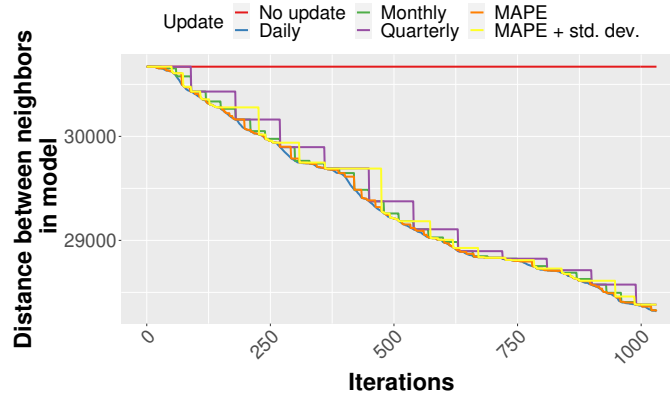


Figure 4: Evolution of mean euclidean distance between neighbors in the model as iterations pass for $h=144$

Streaming algorithms must always be ready to make predictions. These predictions have to be computed as fast as possible in order to provide near real-time results during the online phase. Figure 5 illustrates the evolution of the execution time needed to predict the whole streaming set using the daily update. This evolution is presented for each prediction horizon. The number of iterations each horizon has to perform to predict the online set is different since the values of w and h are different for each of them. The execution time increases linearly with iterations for all prediction horizons, demonstrating the scalability of the algorithm. The shorter the prediction horizon, the higher the scalability, i.e., the represented line has a smaller slope. The evolution of the execution time versus iterations is similar for the rest of the update types.

4.2. Comparison with other methods

In this Section, the best result obtained by the proposed algorithm is compared with other studies that used the same dataset [15, 43, 36]. The works

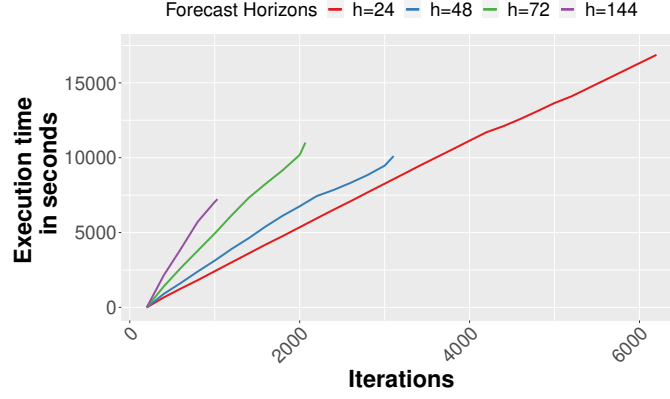


Figure 5: Execution time versus number of iterations for daily neighbor updating in the online phase.

in [15, 43] are not data streaming algorithms, but traditional machine learning algorithms. However, they are used in order to obtain a comparison of the accuracy of the forecasts obtained by the proposed method.

390 All these benchmark algorithms were trained using the data that the streaming method proposed in this work used in the offline stage. The algorithms were evaluated using the streaming set of this work as the test set.

In [15], the authors presented a hybrid model for time series forecasting based on a combination of clustering, classification and forecasting techniques. 395 The k-means algorithm with the optimal number number of clusters ($k=7$) and different prediction and classification methods were used. In particular, classifiers such as a basic classifier based on the distance to the nearest centroid, SVM and Naive Bayes were tested, and the prediction methods evaluated were different ARIMA models, different Holt-Winter models and a deep Long Short-Term Memory (LSTM) recurrent neural network. 400

Table 2 presents the MAPE obtained for each prediction and classification method used in the hybrid model published in [15] and for the StreamWNN proposed algorithm updating the model on a daily basis when predicting the streaming set for the 24-value forecast horizon, i.e., 4-hour forecast ($h = 4$). It 405 can be observed that the StreamWNN algorithm obtains a much more accurate error than that of the other methods.

In summary, our StreamWNN method provides better results comparing with traditional machine learning algorithms. Although it works with additional requirements as forecasts must be made as soon as new data arrives and in the shortest possible time. None of the cited articles included information on 410 execution time.

The benchmark study introduced in [43] included different prediction methods applied to the same time series dataset as this work. In particular, NDL, FFNN, DT and GBM methods were evaluated. The NDL is a deep learning

Table 2: MAPE obtained by the proposed method using daily updating and the methods published in [15]

Algorithm	MAPE (%)				
	Classifier				Update
	Basic	Naive Bayes	SVM	Random	Daily
ARIMA(3,0,1)	4.6	4.6	4.5	5.9	-
ARIMA(5,0,1)	4.3	4.1	4.0	5.8	-
ARIMA(7,0,1)	4.7	4.8	4.75	5.8	-
ARIMA combinations	4.1	4.2	4.0	6.0	-
LSTM	3.0	3.0	3.2	4.5	-
StreamWNN	-	-	-	-	2.19

neural network which used a genetic algorithm to find the sub-optimal hyper-parameters of the network, FFNN is a feed-forward neural network, DT a decision tree algorithm and GBM an algorithm based on gradient boosting.

Table 3 presents the MAPE obtained by the NDL, FFNN, DT, GBM methods and the StreamWNN proposed forecasting method to predict next 4 hours from the previous 24 hours, i.e. 144 values ($w = 144$). It can be seen that the StreamWNN method is more accurate in terms of MAPE error compared to most of the approaches. However, it is not better than the NDL model. Authors state that the computational time required is high while the aim of StreamWNN is to forecast in real-time.

Table 3: MAPE obtained by the proposed method using daily updating and the methods published in [43]

Algorithm	MAPE (%)
NDL	1.51
FFNN	2.32
DT	8.86
GMB	4.49
StreamWNN + daily updating	2.19

A forecasting algorithm for streaming time series, named StreamNSP was presented and compared with benchmark forecasting algorithms for data streams such as the FIMTDD, AdaGrad and AMRulesReg in [36]. StreamNSP combines clustering with a classifier and the nearest neighbor algorithm. The FIMTDD algorithm, which stands for Fast Incremental Model Trees with Drift Detection, models data streams using regression tree. The algorithm detects drift changes in incoming data and adapts the model incrementally. The AdaGrad performs an online gradient-based learning. The model adapts dynamically by adding geometry information from the previous data. The AMRules, i.e., Adaptive Model Rules, is an incremental model for learning rules in streaming for regression problems. Each rule contains a linear model and is trained with incremental gradient descent. The update is performed using a delta rule. It detects online

changes using a Page-Hinkley test.

Table 4 shows the MAPE obtained by the FIMTDD, AdaGrad, AMRulesReg, StreamNSP and the StreamWNN when predicting the streaming set for a prediction horizon of 24 values. It is important to consider that MAPE errors are the average of the results for a prediction horizon of 1 value up to a prediction horizon of 24 values. The result for the StreamNSP is the one corresponding to the 24-value prediction horizon. Authors do not mention execution times even though it is a real-time algorithm.

Table 4: MAPE obtained by the proposed method using daily updating and the methods published in [36]

Algorithm	MAPE (%)
AdaGrad	12.05
FIMTDD	4.91
AMRulesReg	2.21
StreamNSP	2.34
StreamWNN + daily updating	2.19

5. Conclusions

In this work a new streaming forecasting model has been proposed. In particular, the StreamWNN is a forecasting algorithm based on the offline-online learning approach for streaming time series. Firstly, in the offline stage, the StreamWNN creates an initial prediction model based on the fusion of patterns and their k-nearest neighbors. Afterwards, in the online phase, streaming data flows start arriving and online predictions are obtained considering the nearest neighbors of the most similar pattern in the current model generated in the offline phase. The method to update the model is an internal updating, as the model changes internal neighbor instances of a pattern by streaming data but does not add any additional new pattern to the model. The model is updated when a specific moment is achieved: every day, every month, every three months or each time the MAPE of the streaming predictions are higher than a threshold. In this work, the error threshold is the MAPE obtained for the prediction of the test set in the offline phase or the MAPE of the offline phase plus its standard deviation. The MAPE and MAE obtained with the neighbor update are lower than those obtained when the offline model update is not performed. The best results have been obtained using a daily updating of the prediction model. The online execution time requirement has been accomplished getting timely results.

The results show that the fusion of information related to patterns and their nearest neighbors is adequate to be able to incrementally update the model as well as to make predictions in a computationally efficient way. This incremental learning approach demonstrates its effectiveness by comparing the results with some benchmark algorithms. These are the main contributions that differentiate

the algorithm from those that can be found in the literature that do real-time prediction.

The future works will be focused on detecting and differentiating novelties and outliers in the streams. The novelties will be added to the model and the outliers will raise an alarm.

Acknowledgements

The authors would like to thank the Spanish Ministry of Science and Innovation for the support under the projects PID2020-117954RB-C21 and TED2021-131311B-C22 and the European Regional Development Fund and Junta de Andalucía for projects PY20-00870 and UPO-138516.

References

- [1] R. Sahal, J. G. Breslin, M. I. Ali, Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case, *Journal of Manufacturing Systems* 54 (2020) 138–151.
- [2] F. Yao, Y. Wang, Towards resilient and smart cities: A real-time urban analytical and geo-visual system for social media streaming data, *Sustainable Cities and Society* 63 (2020) 102448.
- [3] A. Bifet, B. Hammer, F. Schleif, Recent trends in streaming data analysis, concept drift and analysis of dynamic data sets, in: *Proceedings of the 27th European Symposium on Artificial Neural Networks (ESANN)*, 2019, pp. 421–430.
- [4] T. Dubuc, F. Stahl, E. B. Roesch, Mapping the big data landscape: Technologies, platforms and paradigms for real-time analytics of data streams, *IEEE Access* 9 (2021) 15351–15374.
- [5] I. Souiden, M. N. Omri, Z. Brahmi, A survey of outlier detection in high dimensional data streams, *Computer Science Review* 44 (2022) 100463.
- [6] A. Pandya, O. Odunsi, C. Liu, A. Cuzzocrea, J. Wang, Adaptive and efficient streaming time series forecasting with lambda architecture and spark, in: *Proceedings of the IEEE International Conference on Big Data*, 2020, pp. 5182–5190.
- [7] D. Boulegane, A. Bifet, H. Elghazel, G. Madhusudan, Streaming time series forecasting using multi-target regression with dynamic ensemble selection, in: *Proceedings of the IEEE International Conference on Big Data*, 2020, pp. 2170–2179.
- [8] M. A. Mochinski, J. P. Barddal, F. Enembreck, Improving multiple time series forecasting with data stream mining algorithms, in: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 1060–1067.

- [9] X. Ma, G. Ma, Research on modeling and forecasting driven by time series stream data, in: Proceedings of the 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2019, pp. 413–417.
- 510 [10] J. Wu, W. Zeng, Z. Chen, X.-F. Tang, Hierarchical temporal memory method for time-series-based anomaly detection, in: Proceedings of the IEEE 16th International Conference on Data Mining Workshops (ICDMW), 2016, pp. 1167–1172.
- [11] E. Çiçek, S. Gören, G. Memik, Physical activity forecasting with time series data using android smartphone, *Pervasive and Mobile Computing* 82 (2022) 101567.
- 515 [12] K. T. Bui, J. F. Torres, D. Gutiérrez-Avilés, V. Nhu, D. T. Bui, F. Martínez-Álvarez, Deformation forecasting of a hydropower dam by hybridizing a long short-term memory deep learning network with the coronavirus optimization algorithm, *Computer-Aided Civil and Infrastructure Engineering* (2022).
- 520 [13] F. Martínez-Álvarez, G. Asencio-Cortés, J. F. Torres, D. Gutiérrez-Avilés, L. Melgar-García, R. Pérez-Chacón, C. Rubio-Escudero, J. C. Riquelme, A. Troncoso, Coronavirus optimization algorithm: A bioinspired meta-heuristic based on the covid-19 propagation model, *Big Data* 8 (2020) 308–322.
- 525 [14] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, F. Herrera, Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with mapreduce, *Information Fusion* 42 (2018) 51–61.
- 530 [15] M. Castán-Lascorz, P. Jiménez-Herrera, A. Troncoso, G. Asencio-Cortés, A new hybrid method for predicting univariate and multivariate time series based on pattern forecasting, *Information Sciences* 586 (2022) 611–627.
- [16] P. Jiménez-Herrera, L. Melgar-García, G. Asencio-Cortés, A. Troncoso, Streaming big time series forecasting based on nearest similar patterns with application to energy consumption, *Logic Journal of the IGPL* (2022).
- 535 [17] A. M. García-Vico, C. J. Carmona, P. González, M. J. del Jesus, A distributed evolutionary fuzzy system-based method for the fusion of descriptive emerging patterns in data streams, *Information Fusion* 91 (2023) 412–423.
- 540 [18] A. I. Weinberg, M. Last, Enhat — synergy of a tree-based ensemble with hoeffding adaptive tree for dynamic data streams mining, *Information Fusion* 89 (2023) 397–404.
- [19] V. Tesson, M. Amoretti, Advanced statistical and machine learning methods for multi-step multivariate time series forecasting in predictive maintenance, *Procedia Computer Science* 200 (2022) 748–757.
- 545

- [20] I. Ilic, B. Görgülü, M. Cevik, M. G. Baydoğan, Explainable boosted linear regression for time series forecasting, *Pattern Recognition* 120 (2021) 108144.
- [21] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, A. Troncoso, 550 Discovering three-dimensional patterns in real-time from data streams: An online triclustering approach, *Information Sciences* 558 (2021) 174–193.
- [22] L. Melgar-García, M. T. Godinho, R. Espada, D. Gutiérrez-Avilés, I. S. Brito, F. Martínez-Álvarez, A. Troncoso, C. Rubio-Escudero, Discovering 555 Spatio-Temporal Patterns in Precision Agriculture Based on Triclustering, in: *Proceedings of the 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO)*, 2021, pp. 226–236.
- [23] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, A. Troncoso, 560 High-Content Screening Images Streaming Analysis Using the STriGen Methodology, in: *Proceedings of the 35th Annual Association for Computing Machinery Symposium on Applied Computing (SAC)*, 2020, p. 537–539.
- [24] T. Nanda, B. Sahoo, C. Chatterjee, Enhancing real-time streamflow forecasts with wavelet-neural network based error-updating schemes and ECMWF meteorological predictions in Variable Infiltration Capacity 565 model, *Journal of Hydrology* 575 (2019) 890–910.
- [25] H. Chen, Y. Yu, Y. Jia, B. Gu, Incremental learning for transductive support vector machine, *Pattern Recognition* 133 (2023) 108982.
- [26] L. Yan, J. Feng, Y. Wu, T. Hang, Data-Driven Fast Real-Time Flood Forecasting Model for Processing Concept Drift, in: *Proceedings of the Cloud Computing, Smart Grid and Innovative Frontiers in Telecommunications*, 570 2020, pp. 363–374.
- [27] E. Lopez-Lopez, X. M. Pardo, C. V. Regueiro, Incremental learning from low-lab elle d stream data in open-set video face recognition, *Pattern Recognition* 131 (2022) 108885.
- [28] M. G. Rahman, M. Z. Islam, Adaptive decision forest: An incremental 575 machine learning framework, *Pattern Recognition* 122 (2022) 108345.
- [29] J.-Y. Park, J.-H. Kim, Online incremental hierarchical classification resonance network, *Pattern Recognition* 111 (2021) 107672.
- [30] T. T. Nguyen, M. T. Dang, A. V. Luong, A. W.-C. Liew, T. Liang, J. McCall, 580 Multi-label classification via incremental clustering on an evolving data stream, *Pattern Recognition* 95 (2019) 96–113.
- [31] K. Liu, Z. Li, C. Yao, J. Chen, K. Zhang, M. Saifullah, Coupling the k-nearest neighbor procedure with the Kalman filter for real-time updating of the hydraulic model in flood forecasting, *International Journal of Sediment Research* 31 (2) (2016) 149–158. 585

- [32] R. Q. Thomas, R. J. Figueiredo, V. Daneshmand, B. J. Bookout, L. K. Puckett, C. C. Carey, A Near-Term Iterative Forecasting System Successfully Predicts Reservoir Hydrodynamics and Partitions Uncertainty in Real Time, *Water Resources Research* 56 (11) (2020) 20.
- 590 [33] R. Pérez-Chacón, R. L. Talavera-Llames, F. Martínez-Álvarez, A. Troncoso, Finding electric energy consumption patterns in big time series data, in: *Proceeding of the 13th International Conference on Distributed Computing and Artificial Intelligence*, 2016, pp. 231–238.
- 595 [34] R. Talavera-Llames, R. Pérez-Chacón, A. Troncoso, F. Martínez-Álvarez, MV-kWNN: A novel multivariate and multi-output weighted nearest neighbours algorithm for big data time series forecasting, *Neurocomputing* 353 (2019) 56–73.
- 600 [35] R. L. Talavera-Llames, R. Pérez-Chacón, M. Martínez-Ballesteros, A. Troncoso, F. Martínez-Álvarez, A nearest neighbours-based algorithm for big time series data forecasting, in: *Proceedings of the 11th Hybrid Artificial Intelligent Systems (HAIS)*, 2016, pp. 174–185.
- 605 [36] P. Jiménez-Herrera, L. Melgar-García, G. Asencio-Cortés, A. Troncoso, A New Forecasting Algorithm Based on Neighbors for Streaming Electricity Time Series, in: *Proceedings of the 15th Hybrid Artificial Intelligent Systems (HAIS)*, 2020, pp. 522–533.
- [37] J. F. Torres, A. Troncoso, I. Koprinska, Z. Wang, F. Martínez-Álvarez, Deep learning for big data time series forecasting applied to solar power, in: *International Joint Conference SOCO'18-CISIS'18-ICEUTE'18*, 2019, pp. 123–133.
- 610 [38] H. Yang, K. R. Schell, GHTnet: Tri-branch deep learning network for real-time electricity price forecasting, *Energy* 238 (2022) 122052.
- [39] K. Chreng, H. S. Lee, S. Tuy, Electricity demand prediction for sustainable development in cambodia using recurrent neural networks with era5 reanalysis climate variables, *Energy Reports* 8 (2022) 76–81.
- 615 [40] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, A. Troncoso, Nearest neighbors-based forecasting for electricity demand time series in streaming, in: *Advances in Artificial Intelligence*, Springer International Publishing, Cham, 2021, pp. 185–195.
- 620 [41] A. Azeem, I. Ismail, S. M. Jameel, V. R. Harindran, Electrical load forecasting models for different generation modalities: A review, *IEEE Access* 9 (2021) 142239–142263.
- [42] A. A. Benczúr, L. Kocsis, R. Pálovics, *Online Machine Learning in Big Data Streams: Overview*, Springer International Publishing, Cham, 2019, Ch. 1, pp. 1207–1218.

- ⁶²⁵ [43] F. Divina, J. F. Torres Maldonado, M. García-Torres, F. Martínez-Álvarez, A. Troncoso, Hybridizing deep learning and neuroevolution: application to the spanish short-term electric energy consumption forecasting, *Applied Sciences* 10 (16) (2020) 5487.

5.1.12 | "Identifying novelties and anomalies for incremental learning in streaming time series forecasting"

Authors: Melgar-García L., Gutiérrez-Avilés D., Rubio-Escudero C., Troncoso A.

Publication type: Journal article. Under review.

Identifying novelties and anomalies for incremental learning in streaming time series forecasting

Laura Melgar-García^a, David Gutiérrez-Avilés^b, Cristina Rubio-Escudero^b,
Alicia Troncoso^{a,*}

^a*Data Science & Big Data Lab, Pablo de Olavide University, ES-41013 Seville, Spain.*

^b*Department of Computer Science, University of Seville, Avda. Reina Mercedes s/n, Seville, 41012, Spain.*

Abstract

Performing real-time forecasting offers the possibility to consider new types of patterns in the incoming streaming data, which is not possible when working with historical or batch data. This paper presents a new approach to detect novelties and anomalies in real-time using a nearest-neighbors based forecasting algorithm. The algorithm works with an offline base model that is updated as stream data arrives following an incremental learning approach. It detects unknown patterns called novelties and anomalies. Novelties are included in the model in an online way and anomalies trigger an alarm as they present unexpected behaviors that need to be specifically analyzed. The algorithm has been tested with Spanish electricity demand data. Results show that the model adjusts in real-time to the new patterns of data providing accurate and fast results.

Keywords: real-time forecasting, online incremental learning, novelties and anomalies, streaming time series, electricity demand.

1. Introduction

Nowadays, most of the data generated from fields as diverse as medicine, industry or renewable energies are streaming time series. Time series data received in streaming have its own particularities as streams need to be processed and modelled in a particular way considering specific requirements [25]. One of the most important characteristics for this type of data is the need to obtain timely responses and therefore, obtaining real-time predictions is crucial in order to predict behaviour in the near future for decision making such as cost-saving or optimization of system performance [29].

A streaming time series is a time series that arrives continuously at high speed and has a data distribution that may change over time. In this scenario,

*Corresponding author

Email address: atrolor@upo.es (Alicia Troncoso)

new patterns in the data may appear and known patterns may disappear or evolve into new patterns. These new patterns can be classified into novelties, which indicate emerging patterns that need to be incorporated as new normal patterns in the prediction model or anomalies, which are undesired patterns [12]. Thus, a novelty will be a new recurring pattern over time while an anomaly is a new pattern that does not repeat over time because it is due to an exception.

In addition to prediction models always being ready to provide real-time predictions, they must also adjust to new patterns that appear in the data when dealing with streaming time series, otherwise the prediction results would not be accurate [4]. Because of this, forecasting algorithms must incorporate incremental learning mechanisms so that they can adapt to take into account the data coming online, as it is not feasible to provide answers in time if the algorithm has to be re-trained every time new patterns appear.

This paper proposes a methodology based on a real-time prediction algorithm to detect both novelties and anomalies in the data. This prediction algorithm, called StreamWNNNov, is based on the nearest neighbors and learns incrementally as the temporal data is streamed. This incremental learning is carried out in two distinct ways: internal and external learning. One is an internal incremental learning in which the prediction model updates the nearest neighbors of known patterns with streaming patterns, and an external learning in which new patterns such as novelties are incorporated into the model as normal patterns. Results using electric power data in Spain are reported to assess the proposed algorithm performance, showing remarkable results when compared to other algorithms recently published in the literature. The results show that the incorporation of the novelties improves the accuracy of the prediction algorithm and furthermore the prediction algorithm is computationally efficient, which makes it suitable to obtain real-time predictions.

The rest of the paper is structured as follows. Section 2 presents a review of novelties detection algorithms and forecasting approaches for streaming time series. In Section 3 the methodology to detect novelties and anomalies is introduced along with the StreamWNNNov algorithm and how its incremental learning is carried out. Section 4 defines the electricity dataset used and includes the discussion of the forecasting results for different type of updating of the prediction model and execution time results for different prediction horizons. The paper finishes with some final conclusions and ideas for future approaches in Section 5.

2. Related work

The research field of streaming time series forecasting has made significant progress in the last ten years. The efforts can be divided into two aspects: developing efficient algorithms to forecast data streams and methodologies for novelty and anomaly detection from these data streams.

Regarding forecasting algorithms for data streaming, finding patterns in the stream and matching them with the newly arriving data emerge as an effective strategy. The authors in [16] proposed a methodology for streaming time series

forecasting based on finding similar patterns along the data streams. First, they combined a k-means clustering model with a Naive-Bayes classifier in an offline phase to determine the different patterns from historical data. Then, the forecast is obtained by applying a nearest similar patterns-based methodology in the online phase. In order to evaluate the performance of the framework a comparative study was applied to electricity demand consumption in a big data scenario. To deal with a time variable, the pattern recognition can be extended to the third dimension as made in [20], where the authors developed the STriGen triclustering algorithm to discover 3D patterns from data streams in real time. The proposal applies an ad-hoc genetic algorithm to build an initial model in an offline phase. When streaming data arrives, the model is updated by applying genetic operators systematically. The algorithm is exhaustively evaluated using synthetic and real-world datasets from environmental sensors and compared to a batch triclustering algorithm. Also, in [23], specific structures to find patterns were developed, presenting an incremental methodology to label forecasting in real time. It is based on a data structure, called online incremental hierarchical classification resonance network (OIHCRN), which scales online according to the classes, thus allowing real-time classification. This structure was tested in several problems, such as text and image categorization, protein function prediction, and a multimedia recommendation system. Results were compared to another online classification algorithm, showing the efficiency of the proposed structure to find patterns.

Ensemble models show a well-known outstanding performance in the time series forecasting domain. Focusing on the data streaming scenario, an effort to adapt and improve the dynamic ensemble selection approach for data stream time series forecasting can be found in [6]. The authors presented a new system, called MTR-DES, to enhance the selection of the ensemble model as the data stream arrives. MTR-DES takes into account the dependencies of the model pool and applies an incremental multi-target regression approach to make the selections. The proposal was tested using several real scenarios, such as hospital energy loads, solar radiation, or river flow streaming data. Its efficiency was compared to a collection of other dynamic ensemble selection solutions in terms of prediction accuracy and computational costs.

Streaming forecasting algorithms are also closely linked to the big data environment. In [22], the authors claim the improvement of Vector Autoregression (VAR) for time series forecasting, making it online and concept-drift sensible through a big data Lambda architecture. They used the Apache Spark framework to develop a batch-stream combining system to manage data streams and outperform the VAR algorithm. The batch-stream framework performance was compared separately to the batch and streaming base architectures that Apache Spark provides. Experiments were conducted in four different streaming scenarios: without, gradual, abrupt, and mixed concept drifts data, showing the proposal the best performance.

Not only strict machine learning approaches can be found, but statistical and meta-heuristics-based solutions also provide good performance. The authors in [21] stated that combining statistical methods with machine learning ones out-

performs the isolated solutions in the streaming time series forecasting scenario. They showed that fact by conducting several experiments. In particular, they applied their hybrid proposal to data streams from the International Institute of Forecasters, combining the ARIMA statistical method with well-known machine learning ones, such as AdaGrad and AmRules Regressor. Furthermore, in [18], a new streaming time series data modeling and forecasting method was presented. The data modeling is based on dividing the streams in a double-sliding window and, in a second step, a gene expression algorithm was applied to mine the model from the data composing of these windows of variable length. Furthermore, the colony climbing algorithm was used to improve the adaptation of the model to concept-drift. The proposal was compared to the hierarchical temporal memory algorithm using synthetic data.

For novelty and anomaly detection, there are a plethora of approaches to accomplish this problem in many application domains [12]. Focusing on both machine learning approaches and data streaming scenarios, several strategies emerge such as neighbor-based approaches, ensemble, and neural network models.

Neighbor-based strategies model the novelty detection problem as a k-nearest neighbor problem using different similarity measures to detect a new pattern from the stream. In that sense, the authors in [1] developed an anomaly detection algorithm using a k-nearest neighbor graph. The performance of the method was evaluated using synthetic and real-world datasets, outperforming the methods to which it was compared. In [8], the authors proposed a sophisticated novelty detection algorithm for streaming data based on a mixture of Gaussian distributions. In order to update the model with the newly discovered patterns, they implemented an Expectation Maximization algorithm along with a meta-regression model based on random forest and a k-nearest neighbor algorithm. A comparative experimental study using real-world datasets showed outstanding results.

The ensemble approach also stands out as a competitive strategy to deal with novelties and anomalies. On this matter, the ensemble solution presented in [7] is remarkable, which proposed an architecture that maintains a fixed-size pool of updated classifiers. The classifier pool is combined with the ADWIN drift detector [5] to adapt the system to incoming novelties and update the models if these novelties mean a change in the data stream. The ensemble solution proposed in [14] adds the distributed characteristic to the implementation of the ensemble. The authors combined Random Forest, Logistic Regression, and Support Vector Machines as a stacking ensemble learner. To deal with anomaly detection, a k-means clustering algorithm was applied. The system was implemented using the Apache Spark framework and was tested in different big data streaming scenarios, being its application domain the network intrusion detection.

Neural networks have been used as a promising approach due to their adaptation capability to the problem domain. Thus, the authors in [17] presented a solution for a specific situation with novelties, the malicious injection of false novelties in the data stream. A system based on a restricted Boltzmann machine

combined with online gradient calculation and extended energy function was developed to accomplish this problem, providing robustness against adversarial novelties. The proposal was tested using a data stream benchmark, showing its effectiveness in terms of novelty detection and poisoning data filtering.

Finally, an exhaustive updated literature review of novelty detection can be found in [26]. The authors point out the particular characteristics of the data streaming environment: concept drift, limited time and space requirements, and the curse of dimensionality in high dimensional space, combining these four concepts to present a state-of-the-art analysis. Furthermore, in [10], a discussion of different algorithms for novelty management was presented. The authors provided a detailed overview of evaluation measures and datasets used for evaluating these algorithms.

3. Methodology

Let be a time series' dataset consisting of N instances that can be represented as:

$$X_t = \{(x^1, y^1), \dots, (x^N, y^N)\} \quad x^i \in \mathbb{R}^w \quad y^i \in \mathbb{R}^h \quad (1)$$

where x^i are the features of the i -th instance corresponding to w past values of the time series and y^i are the classes of the i -th instance corresponding to h following values of the time series. Considering (x^i, y^i) , the goal is to forecast the next h classes (y^i) of the w streaming incoming features (x^i) in real-time. The proposed methodology for streaming data is composed of two phases. In a first offline phase, an initial prediction model is built and in a second online phase, this model is updated incrementally in real time as predictions are made in real time. The proposed algorithm StreamWNNov predicts streaming data using K weighted nearest neighbors and updates the model based on novelty detection. In particular, the incremental updating consists of both updating the nearest neighbors and detecting novelties and adding them to the model. The algorithm includes the parameter K that refers to the number of neighbors, or time series, associated with each instance in the model. Neighbors contribute both in the offline and online forecasting process. One of the main problems of the traditional K-nearest neighbors is the required time to build the model. However, in StreamWNNov the model is created only once, during its first phase or offline phase, and in a distributed and efficient way. The second phase or online phase forecasts in real-time, keeps the model updated with incremental learning considering streaming data, identifies anomalies and adds novelties to the model.

Figure 1 presents an overview of the methodology. First, the model M is created in an offline way from historical data and from the stream x^{st} the forecasting \hat{y}^{st} along with its error $MAPE^{st}$ is obtained in real time. This prediction \hat{y}^{st} is based on the nearest neighbors of the closest pattern to x^{st} called x^{min} . Depending on the magnitude of the error, the stream is considered normal or unknown pattern, concretely novelty or anomaly. If the stream x^{st} is normal but is a closer neighbor to x^{min} than the nearest neighbors of x^{min}

stored in the model M , i. e. if $d(x^{st}, x^{min}) < d(x^{min}, n_K)$, the stream is stored in a buffer B in order to update the nearest neighbors. If the stream is a novelty, it is incorporated as a new pattern into the model, and if the stream is an anomaly an alarm is raised. As shown in the Figure, the algorithm continues to run the online phase until there is no more incoming streaming data. Figure 2 illustrates a general outline of the online phase only.

These steps of the methodology are described in detail in the following sections. The process for computing the offline model and the online forecast is in Section 3.1. Section 3.2 explains the methodology developed to detect streaming time series novelties and to identify streaming anomalies in real-time. In addition, this Section presents the procedure to update the model in real-time using incremental learning based on neighbors and novelties.

3.1. StreamWNNov algorithm

This Section presents the StreamWNNov algorithm for streaming time series forecasting. The offline phase to build an initial prediction model is described in Section 3.1.1 and how real-time forecasting is performed in Section 3.1.2.

3.1.1. Offline stage

Historical offline data is divided into two sets: neighbors and patterns, denoted set_n and set_p from now on. These sets correspond to 70% and 30% of the chronologically ordered offline data respectively. Both sets of time series follow the above definition (see Equation 1). For each feature instance of the set_p pattern set, the algorithm searches for the K closest instances from the set_n neighbors' set. In this study, the distance metric used is the euclidean distance between features. The offline model with t instances of the set_p is represented as:

$$M = \langle \langle x^1, \langle y(n_1(x^1)), \dots, y(n_K(x^1)) \rangle, \dots, \langle x^t, \langle y(n_1(x^t)), \dots, y(n_K(x^t)) \rangle \rangle \rangle \quad (2)$$

where x^i corresponds to the features of the i -th instance of the set_p and $y(n_j(x^i))$ is the class of the j -th closest neighbor of x^i from the set_n .

The last step of the offline phase consists of calculating the error obtained in the learning process of the model M known as training error. For this purpose, the error obtained by the model M when predicting the set_p is computed.

For each i -th instance x^i in the model M and for each l -th value in the prediction horizon h , the prediction based on the K nearest neighbors [27, 28] is defined by:

$$\hat{y}^i(l) = \frac{1}{\sum_{j=1}^K \alpha_j} \times \sum_{j=1}^K \alpha_j y(n_j(x^i))(l) \quad 1 \leq l \leq h \quad (3)$$

where α_j is the inverse squared euclidean distance d_j^2 between the x^t pattern feature and the $n_j(x^t)$ neighbor feature, i. e. $\alpha_j = \frac{1}{d_j^2}$.

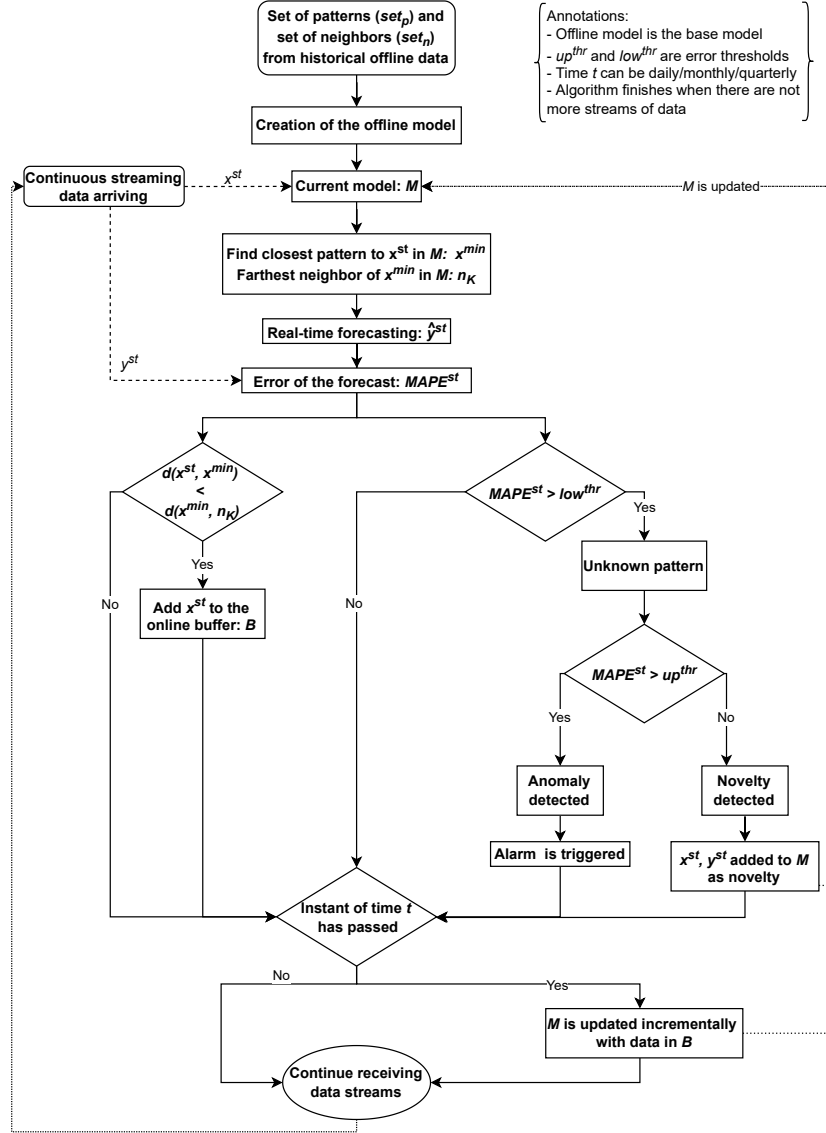


Figure 1: Overview of StreamWNNNov

Thus, the mean absolute percentage error for each $i - th$ instance in the model M , $MAPE^i$, considering the forecast $\hat{y}^i(l)$ and the real $y^i(l)$ classes is

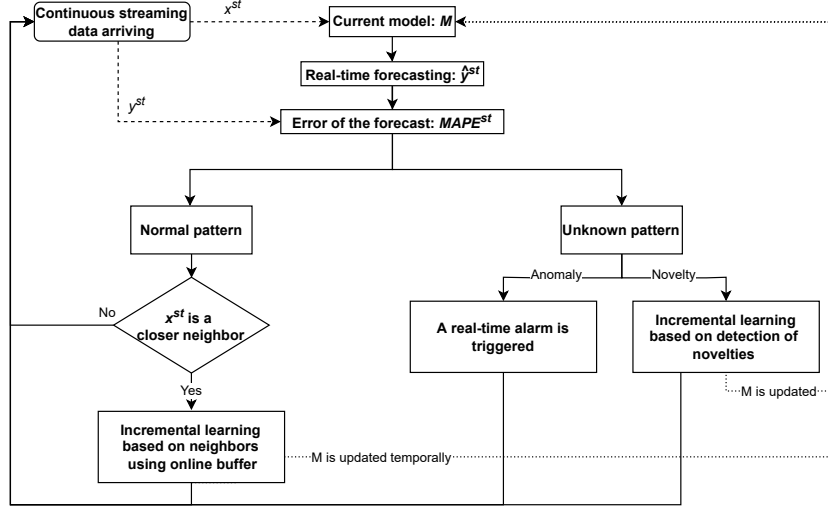


Figure 2: Summary of the online stage of StreamWNNNov

computed by the equation:

$$MAPE^i = 100 \times \frac{1}{h} \sum_{l=1}^h \left| \frac{y^i(l) - \hat{y}^i(l)}{y^i(l)} \right| \quad (4)$$

and the training error, $MAPE_{offline}$, is obtained by averaging all the $MAPE^i$ as follows:

$$MAPE_{offline} = \frac{1}{t} \sum_{i=1}^t MAPE^i \quad (5)$$

Figure 3 illustrates a graphical representation of the offline phase considering that there are t instances in the set_p and K is two, i.e., each pattern of the set_p is associated with its two nearest neighbors from the set_n in the model M . Forecasts \hat{y}^i and the $MAPE^i$ metric for each i -th streaming instance x^i in the set_p are computed following the Equations (3) and (4), respectively, and the MAPE of the offline phase is calculated from the Equation (5).

3.1.2. Real-time forecasting

Once the offline base model is computed and the MAPE value of the offline stage is obtained, StreamWNNNov is ready to start receiving real-time data. In the online phase all streaming requirements are fulfilled [3].

The general procedure consists of associating each streaming data x^{st} to the closest pattern feature instance x^i in the model M , named x^{min} . In other words, it can be said that the nearest neighbor of x^{st} is the instance x^{min} . Then,

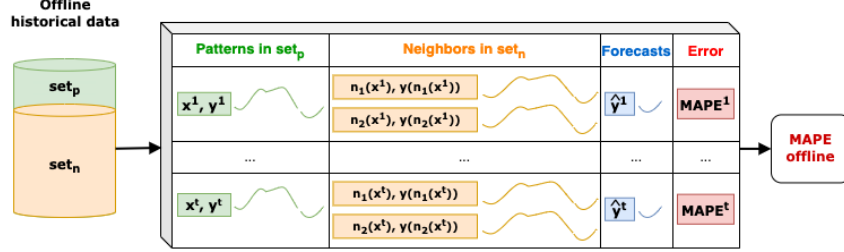


Figure 3: Representation of the offline phase where the set_p is made up of t instances and K is two.

the K nearest neighbors of the x^{min} are recovered from the model M and are considered as the nearest neighbors of the streaming x^{st} . Thus, the online forecast of the classes of the x^{st} , y^{st} , for each $l - th$ value in the prediction horizon h can be computed as:

$$\hat{y}^{st}(l) = \frac{1}{\sum_{j=1}^K \alpha_j + \alpha_{min}} \times \left(\sum_{j=1}^K \alpha_j y(n_j(x^{min}))(l) + \alpha_{min} y(x^{min})(l) \right) \quad (6)$$

where $\alpha_{min} = \frac{1}{d_{min}^2}$ and d_{min} is the euclidean distance between x^{st} and x^{min} .

The strategy of using the K nearest neighbors of my nearest neighbor in the model M as nearest neighbors ensures that the prediction can be computed in real time as neighbors are already stored in the model and do not need to be calculated. This online forecast \hat{y}^{st} takes into consideration the nearest neighbor x^{min} of the x^{st} and the K nearest neighbors of x^{min} in the current model. In particular, the classes of x^{min} are $y(x^{min})$ and the classes of the neighbors of x^{min} are $y(n_j(x^{min}))$ in Equation 6.

Finally, the MAPE metric for \hat{y}^{st} , called $MAPE^{st}$, is calculated each time the real values y^{st} are received in the stream, i.e., a real-time error value is obtained for each h predicted data.

3.2. Detection of novelties and anomalies in real-time

StreamWNNov adds the possibility to incorporate novelties into the model and identify anomalies in the streaming data during the online phase of the algorithm. The detection of novelties and anomalies is explained in Section 3.2.1. The process for updating the model adding novelties to the model as well as updating the K closest neighbors in the model M is described in Section 3.2.2.

3.2.1. Identifying and differentiating unknown streaming patterns

Novelties and anomalies are highly correlated terms that identify a different pattern from the known concepts of the model. However, each of them implies

its own meaning and treatment. On the one hand, novelties in streaming flow of data represent new and emergent behaviors. Novelties must be incorporated into the model for the algorithm to produce results considering these new patterns. On the other hand, anomalies do not conform to expected behaviors and therefore do not need to be added to the model. Anomalies usually require a particular analysis [12]. This identification in online data presents many challenges considering, in addition, that it has to provide timely responses.

In StreamWNNov a x^{st} is considered an unknown pattern if its $MAPE^{st}$, computed after receiving the real y^{st} class, is higher than a defined threshold. In particular, the process of distinguishing unknown patterns between novelties and anomalies is based on two thresholds, an upper threshold (up^{thr}) and a lower threshold (low^{thr}). These two thresholds must be defined after an analysis of the errors obtained in the offline stage.

In [9] authors reviewed some anomaly detection techniques used in nearest neighbors-based models. They concluded with the assumption that anomalies occur far away from their closest neighbors. This idea is the basis of the StreamWNNov algorithm for anomaly detection. Thus, if the $MAPE^{st}$ obtained when predicting the streaming class y^{st} is greater than the up^{thr} threshold, the specific incoming feature x^{st} is considered as an anomaly. When an anomaly is detected, the StreamWNNov triggers a real-time alert.

On the other hand, when the error $MAPE^{st}$ is lower than the up^{thr} threshold but higher than the low^{thr} threshold, online feature data x^{st} are considered as novelties. In this case, the model M is updated by incorporating the new streaming pattern identified as novelty with its corresponding K closest neighbors from historical data. Therefore, it can be said that the model is updated externally. Section 3.2.2 describes in detail the whole process of adding novelties to the model.

For this research work, if the prediction error when predicting y^{st} is greater than the training error, x^{st} will be considered as an unknown pattern. Thus, the lower threshold is the MAPE of the offline phase, i. e. $MAPE_{offline}$. On the other hand, the upper threshold is defined as the $MAPE_{offline}$ plus three times its standard deviation. The pseudocode for identifying anomalies and novelties in real-time is described in Algorithm 1.

3.2.2. Incremental learning

This Section describes the procedure for updating the model with data streams using incremental learning, which consists of both updating the neighbors of the patterns that form the model and incorporating new patterns.

The incremental learning based on neighbors consists of replacing nearest neighbors of the pattern x^i in the model M by streaming patterns x^{st} when these streaming patterns are closer to x^i than their neighbors. In this way, the distances of the nearest neighbors to the pattern x^i will decrease as the incremental learning is carried out and therefore the nearest neighbors will be closer and closer to the pattern x^i [20].

Right after finding the nearest pattern to x^{st} in the model M , x^{min} , if the distance between x^{min} and x^{st} is lower than the distance between x^{min} and its

Algorithm 1: Unknown patterns identification**Result:** updated M model and identification of unknown patterns

```

M ← Current model
lowthr ← MAPEoffline
upthr ← MAPEoffline + 3·std(MAPEoffline)
for each  $x^{st}$  that arrives do
     $\hat{y}^{st} \leftarrow \text{predict}(M, x^{st})$  (see Equation 6)
    MAPEst ← MAPE( $y^{st}, \hat{y}^{st}$ )
    if MAPEst > lowthr then
         $x^{st}$  is an unknown pattern
        if upthr > MAPEst then
             $x^{st}$  is a novelty
            M ← Add novelty to the model (see Equation 7)
        else
            Anomaly identification
            Alert created by the model
        end
    end
end

```

current farthest neighbor $n_K(x^{min})$ in the current model, then x^{st} is added as a new potential nearest neighbor of x^{min} in an online buffer. In order to comply all data streaming requirements, only a few real-time instances can be kept in a buffer.

The online buffer is checked temporarily, in particular daily, monthly or quarterly. Each pattern feature instance x^i in the model can be updated by changing its K nearest neighbors with online incoming data x^{st} from the online buffer. In such a case, the K nearest neighbors are selected from the nearest neighbors of x^i from the current model and the streaming patterns whose nearest neighbour is x^i from the buffer.

Figure 4 represents an example of this online update of the model where n_j are the j – th nearest neighbors of x^i from historical offline data. The online buffer contains the closest pattern instance x^{min} of each x^{st} and the distance between them $d(x^{min}, x^{st})$. All the instances in this buffer have to meet the requirement that $d(x^{min}, x^{st})$ is less than the distance between x^{min} and its farthest neighbor in the model n_K . In Figure 4, after the update process, the nearest neighbors of x^6 are the two data streams instances $x^{st_{s0}}$ and $x^{st_{t1}}$ and the neighbor instance n_1 .

Incremental learning based on the incorporation of new patterns consists of the detection of novelties. The model follows an incremental learning procedure to discover new patterns in the incoming real-time data. These new patterns have to be added to the model to keep it always up to date. This is a very important task in online algorithms [4].

During the online phase, the proposed methodology identifies online feature

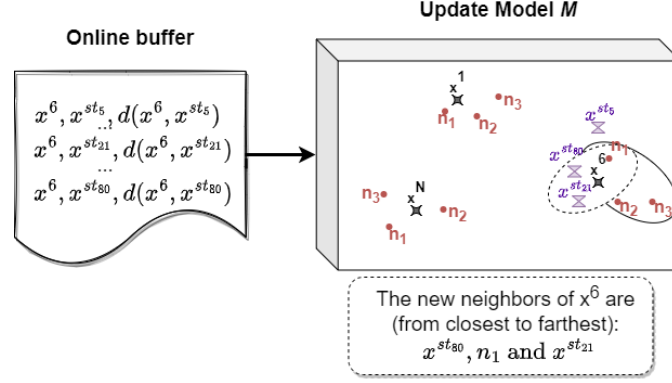


Figure 4: Representation of an example of the incremental learning based on neighbors

data x^{st} as novelty when the $MAPE^{st}$ metric is greater than the defined low^{thr} threshold but lower than the up^{thr} threshold. In particular, for each online data identified as novelty x^{st} , the K nearest neighbors are searched from the offline historical data (both set_p and set_n) and added to the model M . In this way, the online model increases in external dimension each time a streaming instance is identified as novelty. The idea behind this is that new streaming patterns may include neighbors that were not selected as K nearest neighbors for any of the patterns from set_p in the offline phase.

A representation of the update process with the x^{st} streaming feature as a novelty is as follows:

$$\begin{aligned}
 M = & \langle \langle x^1, \langle y(n_1(x^1)), \dots, y(n_K(x^1)) \rangle \rangle, \\
 & \dots, \langle x^t, \langle y(n_1(x^t)), \dots, y(n_K(x^t)) \rangle \rangle, \\
 & \langle x^{st}, \langle y(n_1(x^{st})), \dots, y(n_K(x^{st})) \rangle \rangle \rangle
 \end{aligned} \tag{7}$$

where $y(n_j(x^{st}))$ corresponds to the class of the j -th closest feature to x^{st} from the offline historical data. The Algorithm 2 represents the updating of the model M with the feature x^{st} identified as a novelty.

Figure 5 represents a graphical example of the online update of the model. The example shows a case where the streaming instance x^{st8} has to be included into the current model because its error metric $MAPE^{st8}$ is lower than the defined upper threshold up^{thr} but larger than the defined lower threshold low^{thr} . The current model before the novelty update is made up of t patterns instances. The first nearest neighbor of the t -th instance x^t of the current model is updated by the streaming data x^{st2} and the novelty is added including a new pattern x^{st8} into the model. This new pattern includes as neighbors the two closest instances from the whole offline historical data. Next, the forecast \hat{y}^{st8} and its $MAPE^{st8}$ can be computed.

Algorithm 2: Model update with novelties

Result: updated M model with novelty
 $M \leftarrow$ Current model
distances $\leftarrow []$
 $set_{offline} \leftarrow set_n \cup set_p$
 $x^{st} \leftarrow$ Novelty
for each x^i in $set_{offline}$ **do**
| distances \leftarrow add(distances, $d(x^{st}, x^i)$)
end
 $\{n_1(x^{st}), \dots, n_K(x^{st})\} \leftarrow K_smallest(distances, K)$
 $M \leftarrow add(M, < x^{st}, < y(n_1(x^{st})), \dots, y(n_K(x^{st})) >>$

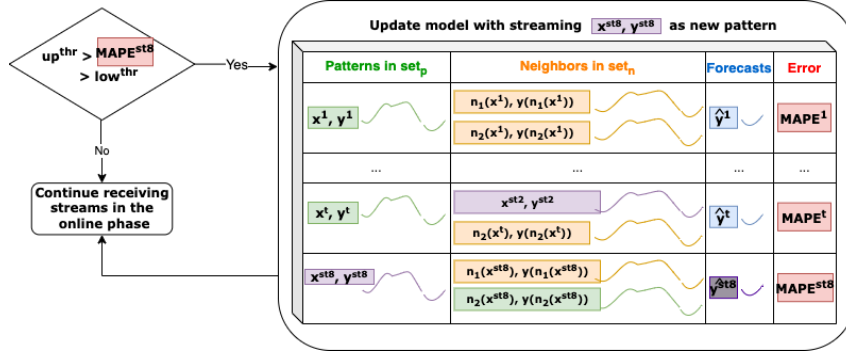


Figure 5: Representation of an example of the incremental learning based on novelties

4. Results

This Section reports the forecasts obtained by StreamWNNNov for different prediction horizons using an electricity consumption time series as well as a comparative analysis between several forecasting algorithms. In particular, Section 4.1 describes the division of the data and the parameters selected for the experimentation. The results are presented and discussed in Section 4.2. The analysis of the anomalies and novelties identified can be found in Section 4.3. Finally, the execution times of the online phase is presented in Section 4.4.

4.1. Dataset and experimental setting

The proposed methodology is tested using a dataset of electrical energy demand (in megawatt) of Spain from 2007 to 2016. The data is recorded every 10 minutes and also contains the date and time. This dataset is split into two sets, offline and stream or online data, which correspond to approximately 70% and 30% of the whole data. As explained in Section 3, the offline data is also separated in 70% and 30% of the data resulting in the above-mentioned set_n and set_p , respectively.

In this work, four prediction horizons have been analyzed, in particular $h=24$, $h=48$, $h=72$ and $h=144$. The optimal size of the w window and the optimal number of nearest neighbors for each horizon used are exactly the same as in [19, 20] with the objective of making a comparison between the results. The four groups of parameters are: (1) $h=24$, $w=144$ and $K=4$, (2) $h=48$, $w=288$ and $K=2$, (3) $h=72$, $w=576$ and $K=4$ and (4) $h=144$, $w=864$ and $K=4$.

The experiments are carried out on a cluster with 1 master node and 3 slaves, with 4 Processor Intel(R) Core(TM) i7-5820K CPU with 48 cores and 120 GB of RAM memory.

4.2. Discussion of the results

The experimentation is performed with the four different sets of parameters presented in Section 4.1. Table 1 presents the MAPE error metric results in percent of the forecasts obtained by the StreamWNNnov when predicting the stream data during the online phase. The four types of updates of the model refer to the following:

- No update: Forecasting algorithm without online updates or identification of novelties [19].
- Online daily: When only an incremental learning based on neighbors is applied, being the buffer checked every day. Worse results are obtained if the buffer is checked monthly or quarterly [20].
- Novelties: When only an incremental learning based on novelties is applied.
- Online daily + novelties: The above two updates together, depicted in Figure 1.

The metrics in the columns of Table 1 correspond to the mean value, the standard deviation and the best and worst errors achieved when predicting the stream data during the online phase. As shown in Table 1, the best forecasting metrics are achieved, for all h prediction horizons, with the updating of the model using incremental learning that includes both the daily updating of neighbors and the detection and incorporation of novelties. This means that the model is able to find different types of patterns in the online incoming data and forecasts are improved by considering all of them. The most accurate results are achieved with a horizon of 24 values, i.e. 4 hours, which suggests that the more times the model is updated, the best results are achieved.

In addition, results in Table 1 show that the online daily update of neighbors is more accurate than just updating the model by incorporating the novelties. The explanation behind this is that the goal of the first-mentioned type of update is to keep the model up to date, i.e., to change the model's own patterns. As showed in [20], with this online daily update of neighbors, the distance between the patterns and their neighbors decreases each time the model is updated. In other words, the patterns found in the online incoming data represent the model

better than the patterns in the offline phase. The goal of the novelties detection and incorporation is to increase the accuracy of the forecasting model and to include in it patterns that did not exist before. When both updates are used together, the StreamWNNNov algorithm takes advantage of both, achieving the best results.

Table 1: MAPE error metric (in percentage) obtained by the StreamWNNNov for each type of update

Type of update	Mean	Std	Best	Worst
No update	2.4288	2.0745	0.2464	33.0031
Online daily	2.1982	2.0138	0.2198	33.0031
Novelties	2.3139	2.1403	0.2464	21.5966
Online daily + novelties	2.0703	1.9974	0.2463	22.8013

(a) Mean metrics of errors for $h=24$

Type of update	Mean	Std	Best	Worst
No update	2.7617	2.0842	0.4101	31.2720
Online daily	2.5499	2.0878	0.3207	31.2720
Novelties	2.6486	2.0339	0.4101	31.2720
Online daily + novelties	2.4296	2.0886	0.2685	31.2720

(b) Mean metrics of errors for $h=48$

Type of update	Mean	Std	Best	Worst
No update	3.3535	2.8200	0.6002	33.3860
Online daily	3.1350	2.8039	0.4493	33.3860
Novelties	3.2692	2.7467	0.6002	33.3860
Online daily + novelties	2.9914	2.7813	0.4493	33.3860

(c) Mean metrics of errors for $h=72$

Type of update	Mean	Std	Best	Worst
No update	3.8466	3.6137	0.6548	29.3278
Online daily	3.5741	3.5292	0.6267	27.0206
Novelties	3.7585	3.5271	0.6548	29.3278
Online daily + novelties	3.4099	3.4238	0.6302	29.3278

(d) Mean metrics of errors for $h=144$

Finally, StreamWNNNov provides the best results when comparing to other streaming techniques using the same distribution of the Spanish electricity demand data [15]. Table 2 presents the results of the different models for the prediction horizon of 4 hours or 24-values. AdaGrad is an adaptative gradient algorithm that adjust dynamically the current model considering the previous data [11]. FIMTDD uses regression trees to identify drift changes in the streaming data [13]. AMRulesReg approaches the streaming forecast with learning rules

based on linear models [2]. StreamNSP clusters the data in different groups and forecasts the demand value using an algorithm based on the traditional nearest neighbor [15].

Table 2: Comparison with other streaming models published in [15]

Algorithm	MAPE in percent
AdaGrad	12.05
FIMTDD	4.91
AMRulesReg	2.21
StreamNSP	2.34
StreamWNNNov (online daily)	2.19
StreamWNNNov (online daily + novelties)	2.07

4.3. Analysis of anomalies and novelties found

As explained in Section 3.2, the main objective of identifying new patterns in the incoming data that are not in the model, i.e., novelties, is to adjust to the evolution of patterns in the incoming data. The main objective of the anomaly detection carried out by the StreamWNNNov is to trigger a real-time alarm to adjust the energy demand in the next few seconds. Anomalies are expected to be identified right at the really alarming moments.

This anomaly and novelty identification process follows an unsupervised approach, since the dataset does not include the ground-truth about the type of each incoming data, which could be identified as normal, anomaly or novelty. To evaluate the accuracy of this process, some particular cases are discussed in this Section.

4.3.1. Novelties in July 2015

Figure 6 depicts the average electrical energy demand per month along with forecasts by the StreamWNNNov algorithm using only online daily update of neighbors, forecasts using only online novelties update and forecasts with the two previous ones together for the whole dataset used in the online phase. Figure 6 corresponds to a prediction horizon of one day ($h=144$) and a past data window of one week ($w=864$). Results for the other parameters follow the same pattern and therefore the same conclusions can be drawn. In the entire dataset used for the streaming forecast, the month with the highest number of novelties is July 2015. Figure 6 shows how the average demand value in July 2014 differs tremendously compared to the average demand value in July 2015. Specifically, the average real demand were 28375.6622 MW and 31544.5781 MW, respectively.

Electricity demand increases with temperature in urban areas [24]. The Meteorological Space Agency (AEMET) recorded 26.5 degree Celsius as the average temperature for the month of July 2015 in Spain, 2.5 degree Celsius higher than the mean temperature for this month for the reference period from 1981 to 2010. The AEMET classified July 2015 as the warmest July in the

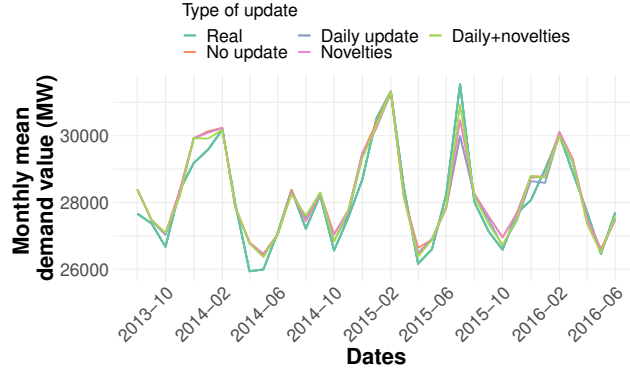


Figure 6: Monthly value of electricity demand for the prediction horizon $h=144$

historical series. This characteristic means that the model does not have many similar patterns in the historical data. Therefore, the update of the model with the novelties carried out by the StreamWNNov is key to get accurate results. Figure 6 shows that the update of the prediction model that achieves a demand value closest to the actual in July 2015 is the daily + novelties update. Table 3 represents the average of the daily MAPE metric for July 2015 for each update type of the model. The best results are always achieved for the model update including both neighbors and novelties performed by the StreamWNNov.

Table 3: Mean of the daily MAPE (in %) for July 2015

Updating	$h=24$	$h=48$	$h=72$	$h=144$
No update	3.2451	4.0598	5.2845	5.5693
Online daily	3.0544	3.9487	5.2256	5.5740
Novelties	3.1273	3.5282	4.0773	4.3776
Online daily + novelties	2.5284	3.1622	3.5613	3.1257

Figure 7 depicts the two days in July 2015, July 20th and July 21st, with the worst forecast when no model update is applied. On both days, the forecast for the model without update and with online daily update of neighbors is the same. In the case of July 20th 2015, the worst prediction coincides for the option without update of the model, for daily update of the neighbors in the model and for update based on novelties only, as shown in Figure 7a. The average MAPE error achieved is above 8.3%. However, the MAPE error value of the online daily + novelties update is 4.0017% and 2.9% for July 20th and 21st respectively, which corresponds to a very good improvement of the error and much more accurate forecast values.

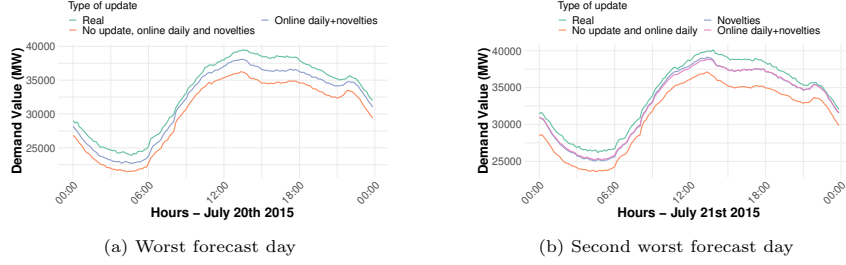


Figure 7: Worst day forecasts in July 2015 for $h=144$ with no model update

4.3.2. Anomalies

By definition of the anomaly detection procedure, anomalies are the worst forecast of the StreamWNNov algorithm. They occur occasionally. In this case, the goal of finding anomalies in real-time is to manually correct the energy demand in the next seconds. Figure 8 shows four anomalies detected when using $w=288$ and $h=48$, i.e., a window composed of the two past days values and eight hours to predict in real-time, respectively. The rest of experiments with different parameters have similar behavior. In general, weekdays present higher energy demand than weekends.

The incoming actual data for Tuesday June 24th 2014 from 8:00 am to 3:50 pm has an energy demand lower than expected as can be seen in Figure 8b. The discussion on the correct classification of this incoming pattern as an anomaly is similar to that in Figure 8a since this same day is not classified as an anomaly neither for 2015 nor for 2016.

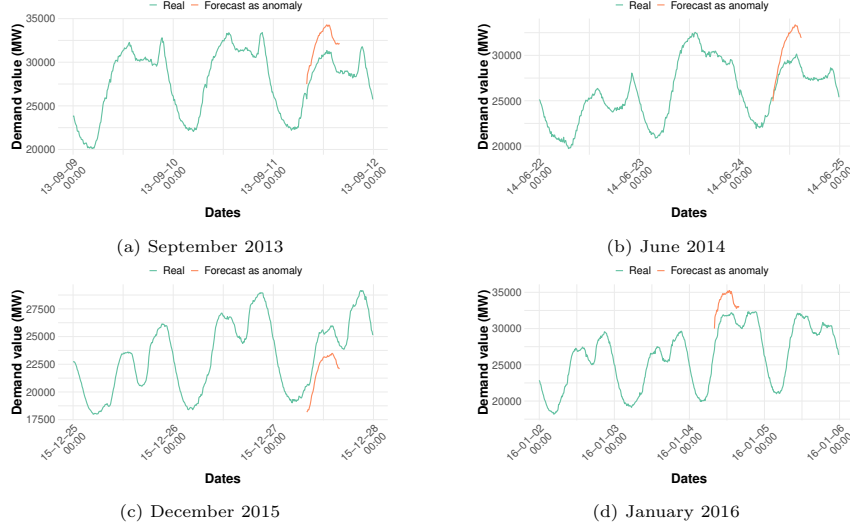
Figure 8c depicts an anomaly on December 27th from 8:00 am to 3:50 pm. Both the actual and anomaly demand follow a similar behaviour but the anomaly detected has a lower value. The main reason could be that it was a Sunday and three days after Christmas holidays.

The behaviour of the actual demand on January 4th, 2016 between 10:00 am and 2:00 pm and between 6:30 pm and 9:30 pm maintain an almost constant value. This performance is not very usual during the working hours of a Monday. The anomaly detected between 8:00 am and 3:50 pm would have helped adjust the demand to a nearly stable value for the remaining business hours of the day and for the following day.

4.4. Discussion of execution time

The execution times of the online phase are very important since the StreamWNNov is a streaming algorithm and therefore one of its main goals is to provide accurate results in near real-time.

Figure 9 shows the scalability of the StreamWNNov when applying incremental learning based on the improvement of the nearest neighbors and the incorporation of novelties. It can be seen that StreamWNNov is scalable as as

Figure 8: Sample of anomalies for the prediction horizon $h=48$

the execution time increases linearly with iterations for all prediction horizons. The number of iterations to compute for each prediction horizon is different because both the window and horizon (w and h) are different. This also influences the time difference between $h=72$ and 144 compared to $h=48$ and 24, since when the model is updated a larger set of nearest neighbors or novelties has to be modified or added to the model, respectively. The average time for an iteration to be performed is approximately 6,66 seconds for $h=24$ and $h=48$. For the other two horizons it is approximately 10 seconds. It is important to consider that h values are being predicted for each iteration taking into account w values that have to arrive in real-time. For example, for $h=72$, the average time to compute the prediction of one of these 72 values is 0.138 seconds.

5. Conclusions

In this research paper, a streaming time-series forecasting model based on the nearest-neighbors algorithm has been introduced. The algorithm begins with an offline stage to create a base model, followed by an online stage where streams start arriving and forecasting is performed. During the online phase, the model is updated in real-time by modifying the neighbors of the model itself following an incremental learning approach. In addition, the model can identify novelties and anomalies in the new incoming data. The distinction between novelties and anomalies in the online data has been addressed by considering two thresholds of the error metric. In this way, each time the actual data stream arrives, the error metric between it and its forecast is calculated. Afterwards, if

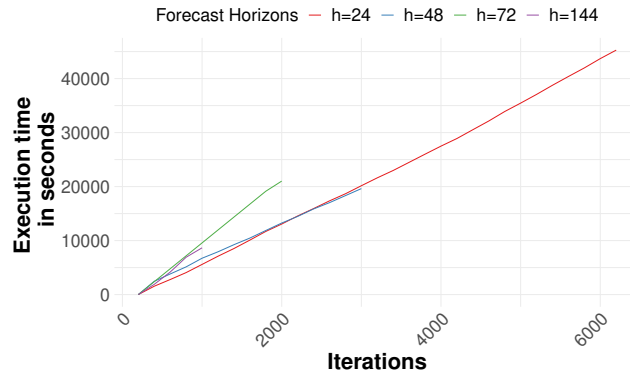


Figure 9: Execution time versus number of iterations for each horizon

this error metric is higher than the upper threshold, it is identified as an anomaly that should be analyzed. On the other hand, if the error metric is between the lower and the upper threshold, the algorithm considers it as a stream novelty. The novelties found are new patterns different from those already in the model. Novelties are added to the model with historical data as neighbors. Results have shown that the model performs better when the model uses incremental learning updating the nearest neighbors and including novelties compared to the case when the model is not updated with novelties during the online phase. In addition, the comparison of the results obtained with StreamWNNNov have been more accurate than that of other benchmark algorithms.

Acknowledgements

The authors would like to thank the Spanish Ministry of Science and Innovation for the support under the projects PID2020-117954RB-C21/C22 and TED2021-131311B-C22 and the European Regional Development Fund and Junta de Andalucía for projects PY20-00870 and P18-RT-2778 and UPO-138516.

References

- [1] Al-Falouji, G., Gruhl, C., Neumann, T., Tomforde, S.: A heuristic for an online applicability of anomaly detection techniques. pp. 107–112. IEEE (9 2022)
- [2] Almeida, E., Ferreira, C., Gama, J.: Adaptive model rules from data streams. In: Proceedings of the Machine Learning and Knowledge Discovery in Databases. pp. 480–492 (2013)
- [3] Benczúr, A.A., Kocsis, L., Pálovics, R.: Online Machine Learning Algorithms over Data Streams, pp. 1199–1207. Springer International Publishing, Cham (2019)

- [4] Bifet, A., Hammer, B., Schleif, F.: Recent trends in streaming data analysis, concept drift and analysis of dynamic data sets. In: Proceedings of the 27th European Symposium on Artificial Neural Networks (ESANN). pp. 421–430 (2019)
- [5] Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM International Conference on Data Mining (SDM). pp. 443–448 (2007)
- [6] Boulegane, D., Bifet, A., Elghazel, H., Madhusudan, G.: Streaming time series forecasting using multi-target regression with dynamic ensemble selection. In: Proceedings of the IEEE International Conference on Big Data. pp. 2170–2179 (12 2020)
- [7] Cano, A., Krawczyk, B.: Rose: robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams. *Machine Learning* **111**, 2561–2599 (7 2022)
- [8] Carreno, A., Inza, I., Lozano, J.A.: Sndprob: A probabilistic approach for streaming novelty detection. *IEEE Transactions on Knowledge and Data Engineering* pp. 1–1 (2022)
- [9] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. *ACM Computing Surveys* **41**(3) (2009)
- [10] Din, S.U., Shao, J., Kumar, J., Mawuli, C.B., Mahmud, S.M.H., Zhang, W., Yang, Q.: Data stream classification with novel class detection: a review, comparison and challenges. *Knowledge and Information Systems* **63**, 2231–2276 (9 2021)
- [11] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for on-line learning and stochastic optimization. *Journal of Machine Learning Research* **12**, 2121–2159 (2011)
- [12] Faria, E.R., Gonçalves, I.J.C.R., de Carvalho, A.C.P.L.F., Gama, J.: Novelty detection in data streams. *Artificial Intelligence Review* **45**, 235–269 (2 2016)
- [13] Ikononovska, E., Gama, J., Džeroski, S.: Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery* **23**(1), 128–168 (2011)
- [14] Jain, M., Kaur, G.: Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data. *Cluster Computing* **24**, 2099–2114 (9 2021). <https://doi.org/10.1007/s10586-021-03249-9>
- [15] Jiménez-Herrera, P., Melgar-García, L., Asencio-Cortés, G., Troncoso, A.: A New Forecasting Algorithm Based on Neighbors for Streaming Electricity Time Series. In: *Hybrid Artificial Intelligent Systems*. pp. 522–533. Springer International Publishing, Cham (2020)

- [16] Jiménez-Herrera, P., Melgar-García, L., Asencio-Cortés, G., Troncoso, A.: Streaming big time series forecasting based on nearest similar patterns with application to energy consumption. *Logic Journal of the IGPL* (2 2022)
- [17] Łukasz Korycki, Krawczyk, B.: Adversarial concept drift detection under poisoning attacks for robust data stream mining. *Machine Learning* (6 2022)
- [18] Ma, X., Ma, G.: Research on modeling and forecasting driven by time series stream data. In: *Proceedings of the 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. pp. 413–417 (6 2019)
- [19] Melgar-García, L., Gutiérrez-Avilés, D., Rubio-Escudero, C., Troncoso, A.: Nearest neighbors-based forecasting for electricity demand time series in streaming. In: *Advances in Artificial Intelligence*. pp. 185–195. Springer International Publishing, Cham (2021)
- [20] Melgar-García, L., Gutiérrez-Avilés, D., Rubio-Escudero, C., Troncoso, A.: Discovering three-dimensional patterns in real-time from data streams: An online triclustering approach. *Information Sciences* **558**, 174–193 (5 2021)
- [21] Mochinski, M.A., Barddal, J.P., Enembreck, F.: Improving multiple time series forecasting with data stream mining algorithms. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. pp. 1060–1067 (10 2020)
- [22] Pandya, A., Odunsi, O., Liu, C., Cuzzocrea, A., Wang, J.: Adaptive and efficient streaming time series forecasting with lambda architecture and spark. In: *Proceedings of the IEEE International Conference on Big Data*. pp. 5182–5190 (12 2020)
- [23] Park, J.Y., Kim, J.H.: Online incremental hierarchical classification resonance network. *Pattern Recognition* **111**, 107672 (2021)
- [24] Romitti, Y., Wing, I.S.: Heterogeneous climate change impacts on electricity demand in world cities circa mid-century. *Scientific Reports* **12**(1) (3 2022)
- [25] Sahal, R., Breslin, J.G., Ali, M.I.: Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case. *Journal of Manufacturing Systems* **54**, 138–151 (2020)
- [26] Souiden, I., Omri, M.N., Brahmi, Z.: A survey of outlier detection in high dimensional data streams. *Computer Science Review* **44**, 100463 (5 2022)
- [27] Talavera-Llames, R., Pérez-Chacón, R., Troncoso, A., Martínez-Álvarez, F.: MV-kWNN: A novel multivariate and multi-output weighted nearest neighbors algorithm for big data time series forecasting. *Neurocomputing* **353**, 56–73 (2019)

- [28] Talavera-Llames, R.L., Pérez-Chacón, R., Martínez-Ballesteros, M., Troncoso, A., Martínez-Álvarez, F.: A nearest neighbours-based algorithm for big time series data forecasting. In: Hybrid Artificial Intelligent Systems. pp. 174–185 (2016)
- [29] Wang, X., Yao, Z., Papaefthymiou, M.: A real-time electrical load forecasting and unsupervised anomaly detection framework. *Applied Energy* **330**, 120279 (2023)

Part IV

Final remarks

6 | Conclusions and future developments

THE last part of this dissertation is addressed in this Chapter. The conclusions obtained during the research are presented in Section 6.1 in English and in Section 6.2 in Spanish in order to be eligible for the International Doctorate Mention. In addition, new directions to be developed in future research are discussed in Section 6.3 in English and in Section 6.4 in Spanish.

6.1 | Conclusions

As general conclusions, significant advances have been achieved by providing new big data and big data streaming algorithms. To the best of our knowledge, these algorithms have covered techniques that have been poorly investigated so far. Moreover, these techniques are easy to apply and have proven their usefulness on real Smart City and medicine data with which they have been tested.

The specific conclusions that can be drawn from each of the solutions implemented from scratch during the PhD Thesis are addressed in this Section.

Triclustering in big data . The most relevant contributions in the field of three-dimensional data clustering or triclustering in batch or offline mode using historical datasets are detailed below:

- The new *bigTriGen* algorithm that mines patterns in three dimensional datasets has been introduced. This algorithm has been developed for large datasets, proving to be a big data algorithm with no algorithmic scalability limitation. It has been demonstrated that the algorithm finds areas in the data with different patterns and that its scalability factor is even better than linear scalability.
- The *bigTriGen* method has been applied to agricultural fields and seismic data. In both real applications, patterns identifying areas with special characteristics have been discovered using vegetation indices and an earthquake catalog, respectively.
- The triclusters discovered in these applications have obtained high quality measurements demonstrating the good performance of the model. Due to the fact that triclustering techniques address learning in an unsupervised mode and therefore the ground-truth is not known, agricultural and seismic experts have validated the triclusters to provide an extra quality measure.
- The *bigTriGen* algorithm works with static or historical data. The following discussed algorithm, known as *STriGen*, discovers patterns in three-dimensional data in real-time.

Triclustering in big data streams . The following conclusions have been obtained with regard to triclustering in real-time:

- The new *STriGen* algorithm that finds three-dimensional data clusters in real-time has been introduced. This algorithm has been developed using an adapted version

of the *bigTriGen* as a base model. Incremental learning has been implemented for the *STriGen* allowing the triclusters to be updated in real-time incorporating, if any, concept drift. In this manner, it has been shown that areas with similar patterns can be found providing accurate and real-time responses. The execution times of each iteration are in the order of seconds.

- The *STriGen* method has been applied to real data from environmental sensors and medical imaging. In both cases, areas with similar patterns have been discovered in real-time.
- The quality of the triclusters found in real-time has been analyzed with a measure specifically developed for the algorithm called *GRQ*. In addition, the performance of *STriGen* has been compared with a baseline algorithm. The results of *STriGen* have been superior to those of the baseline algorithm in all runs, thus demonstrating its good performance.
- The *STriGen* has been able to find concept drift and adjust the model in real-time. It has also detected minor component changes. However, it has not been applied for the moment to detect outliers or anomalies in real-time considering the current triclusters.

Forecasting in big data streams . With respect to real-time prediction, the following conclusions can be drawn:

- The new *StreamWNN* algorithm that performs real-time predictions has been introduced. The algorithm has been implemented to cope with continuous data flow or streams in two phases. First, a model based on the traditional k nearest neighbors algorithm is created in batch or offline mode. Then, the online phase starts and all the streaming requirements are satisfied using Apache Kafka. In the online phase the model has been developed to be updated incrementally taking into account also the novelties and anomalies that may be received in real-time. In addition, it has been demonstrated that the model behaves in a scalable way which allows it to be identified in the big data streaming paradigm.
- The *StreamWNN* has been applied to the electricity energy demand of Spain and has demonstrated a good performance of the model both in terms of error metrics and in execution time. Results improve when the model is updated online considering an incremental learning approach, including novelties and identifying anomalies.
- A full comparison of the results with some benchmark algorithms for the same dataset have shown that the best results are achieved with *StreamWNN*.

- In this application the algorithm has been tested with univariable data. The algorithm can be easily applied to multivariable datasets. Another interesting remark is that the model does not take into account any calendar information. It is a pattern-based algorithm, which has proven to provide accurate results.
- The *StreamWNN* can be easily applied to any time series. For example, it has been applied to gas and electricity demand datasets from a Spanish Hospital. This project was part of a collaboration with the company "Sevilla Futura". The model demonstrated a very accurate performance also with this type of data. If the time series is not stationary, it would be interesting to perform a time series decomposition. This procedure was followed for the application of the algorithm to the New York City taxi demand in real-time using the *StreamWNN*.

6.2 | Conclusiones

Como conclusiones generales, se han logrado avances significativos al proporcionar nuevos algoritmos de big data y big data *streaming*. Estos algoritmos se han enfocado en técnicas poco investigadas hasta la fecha. Además, estos métodos son fáciles de aplicar y han demostrado su gran utilidad en datos reales de ciudades inteligentes o *Smart Cities* y medicina con los que se han probado.

En esta Sección se presentan las conclusiones específicas que pueden extraerse de cada una de las soluciones desarrolladas durante la Tesis Doctoral.

Triclustering en big data . A continuación, se detallan las contribuciones más relevantes en el campo de la agrupación de datos tridimensionales o triclustering en modo *batch* u offline utilizando conjuntos de datos históricos:

- Se ha propuesto el nuevo algoritmo *bigTriGen* que extrae patrones en conjuntos de datos tridimensionales. Este algoritmo se ha desarrollado para grandes conjuntos de datos, probándose que es un algoritmo de big data sin limitación de escalabilidad algorítmica. Se ha demostrado que el algoritmo encuentra áreas con patrones diferentes en los datos y que su factor de escalabilidad es incluso mejor que la escalabilidad lineal.
- El método *bigTriGen* se ha aplicado a campos de cultivo y a datos sísmicos. En ambas aplicaciones reales, se han descubierto patrones que identifican zonas con características especiales utilizando índices de vegetación y un catálogo de terremotos, respectivamente.
- Los triclusters descubiertos en estas aplicaciones han conseguido medidas de alta calidad que demuestran el buen rendimiento del modelo. Debido a que las técnicas de triclustering abordan el aprendizaje de un modo no supervisado y, por tanto, no se conoce la verdad fundamental o *ground truth*, expertos agrícolas y sísmicos han validado los triclusters para proporcionar una medida de calidad adicional.
- El *bigTriGen* funciona con datos en estático o históricos. El siguiente algoritmo comentado, conocido como *STriGen*, descubre patrones en datos tridimensionales en tiempo real.

Triclustering en big data streams . Se han obtenido las siguientes conclusiones con respecto al triclustering en tiempo real:

- Se ha presentado el nuevo algoritmo *STriGen* que encuentra clusters de datos tridimensionales en tiempo real. Este algoritmo se ha desarrollado utilizando como modelo base una versión adaptada del *bigTriGen*. Se ha implementado un aprendizaje incremental para el *STriGen* que permite actualizar los triclusters en tiempo real incorporando, si existe, el cambio de deriva o *concept drift*. De esta forma, se ha demostrado que se pueden encontrar áreas con patrones similares proporcionando respuestas precisas y en tiempo real. Los tiempos de ejecución de cada iteración son del orden de segundos.
- El método *STriGen* se ha aplicado a datos reales procedentes de sensores medioambientales y de imágenes médicas. En ambos casos, se han descubierto áreas con patrones similares en tiempo real.
- La calidad de los triclusters encontrados en tiempo real se ha analizado con una medida desarrollada específicamente para el algoritmo denominada *GRQ*. Además, se ha comparado el rendimiento de *STriGen* con un algoritmo de referencia. Los resultados de *STriGen* han sido superiores a los del algoritmo de referencia en todas las ejecuciones, demostrando así su buen rendimiento.
- El *STriGen* ha sido capaz de detectar los *concept drift* y ajustar el modelo en tiempo real. También ha identificado cambios menores en los componentes. Sin embargo, no se ha aplicado por el momento para detectar valores atípicos o anomalías en tiempo real teniendo en cuenta los triclusters actuales.

Predicción en big data streams . En cuanto a la predicción en tiempo real, se extraen las siguientes conclusiones:

- Se ha presentado el nuevo algoritmo *StreamWNN* que realiza predicciones en tiempo real. El algoritmo trata los flujos de datos continuos en dos fases. En primer lugar, se crea un modelo basado en el algoritmo tradicional *k*-vecinos cercanos o *KNN* en modo *batch* u offline. Seguidamente, se inicia la fase online donde se satisfacen todos los requisitos del paradigma *streaming* utilizando Apache Kafka. Durante la fase online el modelo se actualiza de forma incremental teniendo en cuenta también las novedades y anomalías que se puedan recibir en el flujo continuo o *stream*. Además, se ha demostrado que el modelo se comporta de forma escalable lo que permite identificarlo dentro del paradigma de big data *streaming*.
- El *StreamWNN* se ha aplicado a la demanda de energía eléctrica en España y se ha demostrado un buen rendimiento del modelo tanto en términos de métricas de error como en tiempos de ejecución. Los resultados mejoran cuando el modelo

se actualiza de forma *online* considerando un enfoque de aprendizaje incremental, incluyendo novedades e identificando anomalías.

- Una comparación completa de los resultados con algunos algoritmos de referencia para el mismo conjunto de datos ha demostrado que los mejores resultados se obtienen con *StreamWNN*.
- En la aplicación presentada en la Tesis Doctoral, se ha probado el algoritmo con datos univariados. El algoritmo puede aplicarse fácilmente a conjuntos de datos multivariados. Otra particularidad interesante es que el modelo no tiene en cuenta ninguna información relacionada con el calendario. Se trata de un algoritmo basado en patrones que ha demostrado que se obtienen resultados precisos.
- El *StreamWNN* puede aplicarse fácilmente a cualquier serie temporal. Por ejemplo, se ha aplicado a conjuntos de datos de demanda de gas y electricidad de un hospital español. Este proyecto forma parte de una colaboración con la empresa "Sevilla Futura". El modelo demostró un rendimiento preciso también para estos datos. Si la serie temporal que se quiere usar para el *StreamWNN* no es estacionaria, sería interesante realizar una descomposición de dicha serie. Este procedimiento se siguió para la aplicación del algoritmo a la demanda de taxis de la ciudad de Nueva York en tiempo real utilizando el *StreamWNN*.

6.3 | Future works

Based on the analysis of the results obtained in this dissertation, the following future work studies are suggested:

- Anomaly or outlier detection with real-time microtriclusters by implementing a new version of *STriGen*.

To the best of our knowledge, there is very little research on the detection of anomalous three-dimensional patterns in real-time. We propose to use as a basis the modeling idea of the *StreamWNN* for anomaly detection, taking into account that the type of learning for triclusters is unsupervised.

- Real-time parallel prediction models for spatio-temporal datasets by implementing a new version of *StreamWNN* where the *bigTriGen* triclusters are used in advance.

Starting with the triclusters found from an execution of the *bigTriGen* with historical spatio-temporal data, it is proposed to run the *StreamWNN* in parallel for each of those triclusters. In this way a problem will be divided into sub-models thus providing a more specific modeling for each of the zones of each tricluster. This approach has been started to be developed with taxo demand data in New York during the predoctoral stay at NYU.

- Review of the state of the art of models that work with continuous data flows or data streams.

During the development of the PhD program, different approaches have been found to address the issue of data streaming. However, to the best of our knowledge, there is no current comprehensive review of the subject. We propose to address the detection of patterns, concept drift, anomalies and novelties in online mode or real-time. As well as to establish a conceptual foundation of data streaming after what has been learned during the dissertation.

- Research on current lines of traditional machine learning for data streaming.

To the best of our knowledge, there are not many contributions of algorithms working in real-time or online mode for batch or traditional machine learning approaches that are currently in vogue. For example, meta-learning or transfer learning.

Overall, during this dissertation it has been found that the number of contributions

and research for real-time or streaming data is limited. However, the current trend in companies is to work with and get answers from real-time or streaming data. Therefore, the future of research is oriented in this paradigm in which it is expected to contribute with scientific improvements.

6.4 | Trabajos futuros

A partir del análisis de los resultados obtenidos en esta Tesis Doctoral, se sugieren los siguientes trabajos futuros:

- Detección de anomalías o *outliers* con microtriclusters en tiempo real mediante la implementación de una nueva versión del algoritmo *STriGen*.

En la actualidad existe muy poca investigación sobre la detección de patrones tridimensionales anómalos en tiempo real. Se propone utilizar como base la idea de modelado del *StreamWNN* para la detección de anomalías, teniendo en cuenta que el tipo de aprendizaje para los triclusters es no supervisado.

- Modelos de predicción paralela en tiempo real para conjuntos de datos espacio-temporales mediante la implementación de una nueva versión de *StreamWNN* en la que se utilicen los triclusters generados previamente por *bigTriGen*.

Partiendo de los triclusters encontrados a partir de una ejecución del *bigTriGen* con datos espacio-temporales históricos, se propone ejecutar el *StreamWNN* en paralelo para cada uno de esos triclusters. De esta forma, el problema se dividirá en submodelos, proporcionando así un modelado más específico para cada una de las zonas enmarcadas en los triclusters. Este enfoque se ha empezado a desarrollar con datos de demanda de taxis en Nueva York durante la estancia predoctoral en la NYU.

- Revisión del estado del arte de los modelos que trabajan con flujos continuos de datos o *data streams*.

Durante el desarrollo del Programa de Doctorado, se han encontrado diferentes enfoques para abordar el tema del flujo continuo de datos. Sin embargo, hasta donde sabemos, no existe en la actualidad una revisión exhaustiva del tema. Se propone abordar la detección de patrones, deriva de conceptos o *concept drift*, anomalías y novedades en modo *online* o en tiempo real. Así como establecer una base conceptual del *streaming* de datos tras lo aprendido durante la Tesis Doctoral.

- Investigación sobre las líneas actuales del aprendizaje automático tradicional para el flujo de datos continuo o *data streaming*.

A día de hoy, no hay muchas contribuciones de algoritmos que trabajen en tiempo real o en modo *online* para los enfoques de *machine learning* en *batch* o tradicionales

que están actualmente más en tendencia. Por ejemplo, el meta-aprendizaje (*meta-learning*) o el aprendizaje por transferencia (*transfer learning*).

En general, a lo largo de esta Tesis Doctoral se ha constatado que el número de contribuciones e investigaciones para datos en tiempo real o *streaming* es limitado. Sin embargo, la tendencia actual en las empresas es trabajar y obtener respuestas a partir de este tipo de datos continuos en tiempo real. Por lo tanto, el futuro de la investigación se orienta en este paradigma en el que se espera contribuir con mejoras científicas.

Bibliography

- [1] S. Agrahari and A. K. Singh. Concept drift detection in data stream mining : A literature review. *Journal of King Saud University - Computer and Information Sciences*, 2021. ISSN 1319-1578.
- [2] J. L. Amaro-Mellado, L. Melgar-García, C. Rubio-Escudero, and D. Gutiérrez-Avilés. Generating a seismogenic source zone model for the pyrenees: A gis-assisted triclustering approach. *Computers and Geosciences*, 150:104736, 2021.
- [3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), 2009.
- [4] Databricks. Project lightspeed: Faster and simpler stream processing with apache spark. <https://www.databricks.com/blog/2022/06/28/project-lightspeed-faster-and-simpler-stream-processing-with-apache-spark.html>, 2022. Accessed: 2023-13-01.
- [5] D. Gutiérrez-Avilés and C. Rubio-Escudero. Msl: A measure to evaluate three-dimensional patterns in gene expression data. *Evolutionary Bioinformatics*, 11: 121—135, 2015.
- [6] D. Gutiérrez-Avilés, R. Giráldez, F. J. Gil-Cumbreras, and C. Rubio-Escudero. TRIQ: a new method to evaluate triclusters. *BioData Mining*, 11(1):15, 2018.
- [7] IBM. Fast data. <https://www.ibm.com/uk-en/analytics/fast-data>, 2022. Accessed: 2023-13-01.
- [8] P. Jiménez-Herrera, L. Melgar-García, G. Asencio-Cortés, and A. Troncoso. A new forecasting algorithm based on neighbors for streaming electricity time series. In E. A. de la Cal, J. R. Villar Flecha, H. Quintián, and E. Corchado, editors, *Hybrid Artificial Intelligent Systems*, volume 12344, pages 522–533, Lecture Notes in Computer Science, 2020. Springer International Publishing, Cham.
- [9] P. Jiménez-Herrera, L. Melgar-García, G. Asencio-Cortés, and A. Troncoso.

- Streaming big time series forecasting based on nearest similar patterns with application to energy consumption. *Logic Journal of the IGPL*, 02 2022. jzac017.
- [10] A. Kafka. Apache kafka introduction. <https://kafka.apache.org/intro>, 2022. Accessed: 2023-13-01.
- [11] P. Larranaga, D. Atienza, J. D. Rozo, A. Ogbechie, C. Puerto-Santana, and C. Bielza. *Industrial Applications of Machine Learning*. CRC Press, 2018.
- [12] F. Martínez-Álvarez, G. Asencio-Cortés, J. F. Torres, D. Gutiérrez-Avilés, L. Melgar-García, R. Pérez-Chacón, C. Rubio-Escudero, J. C. Riquelme, and A. Troncoso. Coronavirus optimization algorithm: A bioinspired metaheuristic based on the covid-19 propagation model. *Big Data*, 8(4):308–322, 2020. doi: 10.1089/big.2020.0051.
- [13] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, and A. Troncoso. High-content screening images streaming analysis using the strigen methodology. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC '20*, page 537–539, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368667.
- [14] L. Melgar-García, M. T. Godinho, R. Espada, D. Gutiérrez-Avilés, I. S. Brito, F. Martínez-Álvarez, A. Troncoso, and C. Rubio-Escudero. Discovering spatio-temporal patterns in precision agriculture based on triclustering. In Á. Herrero, C. Cambra, D. Urda, J. Sedano, H. Quintián, and E. Corchado, editors, *15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020)*, pages 226–236, Cham, 2021. Springer International Publishing.
- [15] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, and A. Troncoso. Nearest neighbors-based forecasting for electricity demand time series in streaming. In E. Alba, G. Luque, F. Chicano, C. Cotta, D. Camacho, M. Ojeda-Aciego, S. Montes, A. Troncoso, J. Riquelme, and R. Gil-Merino, editors, *Advances in Artificial Intelligence*, pages 185–195, Cham, 2021. Springer International Publishing.
- [16] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, and A. Troncoso. Discovering three-dimensional patterns in real-time from data streams: An online triclustering approach. *Information Sciences*, 558:174–193, 2021.
- [17] L. Melgar-García, D. Gutiérrez-Avilés, M. T. Godinho, R. Espada, I. S. Brito, F. Martínez-Álvarez, A. Troncoso, and C. Rubio-Escudero. A new big data triclustering approach for extracting three-dimensional patterns in precision agriculture. *Neurocomputing*, 500:268–278, 2022.

- [18] L. Melgar-García, D. Gutiérrez-Avilés, C. Rubio-Escudero, and A. Troncoso. Nearest neighbors with incremental learning for real-time forecasting of electricity demand. *IEEE International Conference on Data Mining (ICDM 2022)*, 2023. In press.
- [19] T. G.-G.-S. G. C. Rating. Explore the gii-grin-scie (ggs) conference rating, 2023. Accessed: 2023-13-01.
- [20] H. y. N. P. Sociedad Española de Gastroenterología. Seghnp, libros de trabajos 2020. https://www.seghnp.org/sites/default/files/2020-09/trabajosSEGHNP2020_0.pdf, 2020. Accessed: 2023-13-01.

