UNIVERSIDAD NACIONAL DE COLOMBIA

A kernel-based embedding framework for high-dimensional data analysis

Sergio Garcia Vega

Universidad Nacional de Colombia
Faculty of Engineering and Architecture
School of Electrical, Electronic, and Computing Engineering
Manizales, Colombia
2019

A kernel-based embedding framework for high-dimensional data analysis

Sergio Garcia Vega

A dissertation submitted in part fulfillment of the requirements for the degree of:
Doctor in Engineering – Automatics

Supervisor:
Ph.D. César Germán Castellanos Domínguez

Research Group:
Signal Processing and Recognition Group

Universidad Nacional de Colombia
Faculty of Engineering and Architecture
School of Electrical, Electronic, and Computing Engineering
Manizales, Colombia
2019

Un marco de representación basado en kernels para el análisis de datos de alta dimensión

Sergio Garcia Vega

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de:
Doctor en Ingeniería – Ingeniería Automática

Director:
Ph.D. César Germán Castellanos Domínguez

Jurados:

Ph.D. Luis Gonzalo Sánchez Giraldo
University of Miami

Ph.D. Ignacio Santamaria
Universidad de Cantabria

Ph.D. Mauricio Orozco Alzate
Universidad Nacional de Colombia – Sede Manizales

Grupo de Investigacón:
Grupo de Control y Procesamiento Digital de Señales

Universidad Nacional de Colombia
Facultad de Ingeniería y Arquitectura
Departamento de Ingeniería Eléctrica, Electrónica y Computación
Manizales, Colombia
2019

To my family

Acknowledgment

I would like to express my gratitude to my supervisor Prof. Germán Castellanos for his support, knowledge, wisdom, and valuable orientation during this research.

To my family, thank you for all your love and for always been there.

To my heavenly father, "*For the Lord gives wisdom; from his mouth come knowledge and understanding*" Proverbs 2:6.

This research was funded by "Programa Doctoral de Becas Colciencias – Formación para la ciencia, la tecnología y la innovación – Convocatoria 617".

Abstract

The world is essentially multidimensional, e.g., neurons, computer networks, Internet traffic, and financial markets. The challenge is to discover and extract information that lies hidden in these high-dimensional datasets to support classification, regression, clustering, and visualization tasks. As a result, dimensionality reduction aims to provide a faithful representation of data in a low-dimensional space. This removes noise and redundant features, which is useful to understand and visualize the structure of complex datasets. The focus of this work is the analysis of high-dimensional data to support regression tasks and exploratory data analysis in real-world scenarios. Firstly, we propose an online framework to predict long-term future behavior of time-series. Secondly, we propose a new dimensionality reduction method to preserve the significant structure of high-dimensional data in a low-dimensional space. Lastly, we propose an *sparsification* strategy based on dimensionality reduction to avoid overfitting and reduce computational complexity in online applications.

Keywords: Dimensionality reduction; High-dimensional data; Kernel adaptive filtering; Embedding; Gradient descent; Online sequential learning; Sparsification.

Resumen

El mundo es esencialmente multidimensional, por ejemplo, neuronas, redes computationales, tráfico de internet y los mercados financieros. El dasafío es descubrir y extraer información que permanece oculta en estos conjuntos de datos de alta dimensión para apoyar tareas de clasificación, regresión, agrupamiento y visualización. Como resultado de ello, los métodos de reducción de dimensión pretenden sumunistrar una fiel representación de los datos en un espacio de baja dimensión. Esto permite eliminar ruido y características redundantes, lo que es útil para entender y visualizar la estructura de conjuntos de datos complejos. Este trabajo se enfoca en el análisis de datos de alta dimensión para apoyar tareas de regresión y el análisis exploratorio de datos en escenarios del mundo real. En primer lugar, proponemos un marco para la predicción del comportamiento a largo plazo de series de tiempo. En segundo lugar, se propone un nuevo método de reducción de dimensión para preservar la estructura significativa de datos de alta dimensión en un espacio de baja dimensión. Finalmente, proponemos una estrategia de *esparsificación* que utiliza reducción de dimensionalidad para evitar sobre ajuste y reducir la complejidad computacional de aplicaciones en línea.

Palabras clave: Reducción de dimensionalidad; Datos de alta dimensión, Filtrado adaptativo kernel; Incrustación; Gradiente descendente; Aprendizaje secuencial en línea; esparsificación.

Notation

The following table summarizes the notation used throughout this thesis.

Notation	**Description**	**Examples**
Scalars	Small *italic* letters	y, p, q
Vectors	Small **bold** letters	$\mathbf{w}$, $\boldsymbol{u}$, $\boldsymbol{\omega}$
Matrices	Capital **BOLD** letters	$\boldsymbol{U}$, $\boldsymbol{V}$
Time or iteration	Subscript indices	y_t, $\boldsymbol{u}_{i,t}$
Scalar constants	Capital *ITALIC* letters	N, M, T

Symbols

This is a list of the main symbols used throughout this thesis for ease of reference.

$\boldsymbol{u}_t$	filter input at time or iteration t
$\tilde{\boldsymbol{u}}_t$	filter input at time or iteration t
y_t	desired output at time or iteration t
h_t	filter output at time or iteration t
$\mathbb{U}$	input domain
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^M$	M-dimensional real Euclidean space
$\mathbb{R}^m$	m-dimensional real Euclidean space
J_t	correntropy cost function at time or iteration t
e_t	output estimation error at time or iteration t
$\mathbf{w}_t$	weight estimate at time or iteration t (a vector in an Euclidean space)
η	step-size parameter
$\varphi(\cdot)$	a mapping induced by a reproducing kernel
$\mathbb{F}$	feature space induced by the kernel mapping
$\kappa(\cdot,\cdot)$	kernel function
$(\cdot)^\top$	vector or matrix transposition
$\boldsymbol{\omega}$	weight estimate at time or iteration t (a vector in a feature space)
$\boldsymbol{C}$	dictionary or center set
$\lVert\cdot\rVert$	Euclidean norm
$\lvert\cdot\rvert$	absolute value
ε	quantization-size parameter
σ	kernel-size parameter
$\boldsymbol{\alpha}$	dictionary weights
N	number of samples
$\boldsymbol{U}$	high-dimensional finite set
M	number of features in a high-dimensional space
$\boldsymbol{V}$	low-dimensional representation
m	number of features in a low-dimensional space
$\boldsymbol{P}$	kernel matrix that contains high-dimensional similarities
$\boldsymbol{Q}$	kernel matrix that contains low-dimensional similarities
p	high-dimensional similarity
q	low-dimensional similarity
$\boldsymbol{v}$	low-dimensional vector
T	number of iterations

Abbreviations

Here is a list of the abbreviations used in this document.

LMS	least-mean-square algorithm
KAF	kernel adaptive filters
RKHS	reproducing kernel Hilbert spaces
KLMS	kernel least-mean-square algorithm
NNs	neural networks
WTI	west Texas intermediate
SMAPE	symmetric mean absolute percentage error
FNN	feedforward neural network
EMD	empirical mode decomposition
SBM	slope-based methods
PCA	principal component analysis
LLE	locally linear embedding
LEM	laplacian eigenmaps
MDS	multidimensional scaling
ISOMAP	isometric feature mapping
SNE	stochastic neighbor embedding
t-SNE	stochastic neighbor embedding with Student's t-distribution
MSE	mean squared error

Contents

List of Figures

List of Tables

List of Algorithms

1. Preliminaries

This chapter is intended to work as an introduction to the research problem and the upcoming chapters. Firstly, we introduce the research problem motivation. Secondly, we describe the issues under consideration. Thirdly, we present previous proposed solutions and their limitations. Lastly, we list the objectives and contributions of this thesis.

1.1. Motivation

Machine learning is at the cross-road of disciplines like artificial neural networks, statistics, applied mathematics, and computer science. It aims to automate, from an engineering point of view, the construction of an analytical model (Salaken et al., 2017). The main tasks of machine learning are (Bishop, 2006): (1) classification; (2) regression; (3) clustering; (4) adaptive filtering; (5) visualization. These tasks are used to discover and extract information that lies hidden in large amounts of data. Finding this information is not a trivial task as datasets are usually high-dimensional (Popov and Heß, 2016). In this sense, dimensionality reduction aims to preserve, as much as possible, the significant structure of high-dimensional data in a low-dimensional space. This preservation removes noise and redundant features, which is useful in the five machine learning tasks mentioned above (Hu et al., 2017; Liu et al., 2018). In practice, dimensionality reduction methods provide a way to understand and visualize the structure of complex datasets.

In a demographic study, for example, it may be easy to find a correlation between the gross annual income of people and their level of education. Note that, in this example, there are two features, i.e., gross annual income and level of education. However, in real-life scenarios, there are usually more than two features, e.g., age, gender, political affiliation, etc. Then, it will be natural to ask questions like: (1) are all these features important?; (2) is there any correlation between these features?; (3) is there any way to "summarize" all these features? Thus, with more variables, it becomes more difficult to answer the previous questions, i.e., the higher the number of features, the harder it gets to visualize the data and then work on it. In contrast, when data is reduced to two or three dimensions, it is possible to plot, visualize, analyze, and observe patterns more clearly (Lee and Verleysen, 2007).

The use of high-dimensional data to solve real-life problems is quite extensive. Here, we list some of them: (1) Netflix prize [1], where the task was to predict whether someone will enjoy a movie based on how much they liked or disliked other movies; (2) Passenger screening challenge [2], which was a competition launched by the U.S. Department of Homeland Security to identify threats at airports; (3) Santander product recommendation [3], which was a challenge launched by Santander bank to predict which products their existing customers will use in the next month based on their past behavior and that of similar customers; (4) Cdiscount challenge [4], launched by the company Cdiscount to build a model that automatically classifies the products based on their images; (5) Trading challenge [5], it was a forecasting competition to predict the stock market's short-term response following large trades.

In a local context, the research group *Signal Processing and Recognition* of the Universidad Nacional de Colombia has been working on: (1) biomedical data to support diagnostic assistance (Alvarez-Meza et al., 2012; Martinez-Vargas et al., 2012; Giraldo-Suarez et al., 2016); (2) human activity recognition (Valencia-Aguirre et al., 2012); (3) video analysis (Álvarez-Meza et al., 2013).

1.2. Problem Statement

The main challenge, when working with high-dimensional data, is to preserve its significant structure in a low-dimensional space (Wang et al., 2017a). This is useful in regression tasks and exploratory data analysis (Hu et al., 2017; Liu et al., 2018), which is the focus of this thesis. In particular, the analysis of high-dimensional data allows to: (1) have information about the long-term future behavior in time series, which is helpful in financial markets, weather forecasting, and other prediction tasks (Pietrzak et al., 2017; Yang et al., 2017; Brock, 2018); (2) remove noise and redundant features, which is useful to understand and visualize the structure of complex datasets (Lee and Verleysen, 2007); (3) avoid overfitting [6] and reduce computational complexity in online applications (Ashiquzzaman et al., 2018).

Firstly, a long-term prediction algorithm, needs to consider complex dependencies between observations, which are separated by long-time intervals (Ben Taieb et al., 2014). These algorithms have two challenges (Taieb et al., 2010): (1) prediction strategy; (2) model selection. On the one hand, there are three prediction strategies (Bontempi, 2008), i.e., *iterated*, *direct* and *mimo*. However, the main drawback of these strategies is the accumulation of errors through iterations, which means poor performance in online applications (Chevillon,

[1] https://www.netflixprize.com/index.html
[2] https://www.kaggle.com/c/passenger-screening-algorithm-challenge
[3] https://www.kaggle.com/c/santander-product-recommendation
[4] https://www.kaggle.com/c/cdiscount-image-classification-challenge
[5] https://www.kaggle.com/c/AlgorithmicTradingChallenge
[6] Overfitting happens when a model learns a finite number of data so well, that it negatively impacts the performance of the model on new data.

2007; Zhang et al., 2014). On the other hand, the models are generally linear and nonlinear, e.g., autoregressive moving average (Tiao and Xu, 1993) and neural networks (Vapnik and Izmailov, 2017). However, their application in prediction tasks is highly conditioned to the assumption that test data has similar samples as the training data (Taieb et al., 2012).

Secondly, dimensionality reduction methods have proven useful to remove noise and redundant features (Ye et al., 2016; Hu et al., 2017; Liu et al., 2018). There are two main challenges in these methods (Lee and Verleysen, 2007): (1) manifold [7] structure; (2) embedding [8] preservation. The main problem is how to measure or characterize the structure of a manifold to preserve its embedding. On the one hand, the manifold structure is usually measured using variance preservation (Jolliffe, 1986), dot product preservation (Borg and Groenen, 2005), affinity matrices (Nadler et al., 2006), and similarity preservation (Van Der Maaten, 2014). Although these methods have proven useful in a wide variety of real-world problems (Chen et al., 2016b; Sun and Wen, 2017), their success is subject to (Peluffo Ordoñez et al., 2014): (1) data points densely sampled; (2) tune several parameters for each dataset. On the other hand, the embedding preservation is calculated using Mercer kernels (Scholkopf and Smola, 2001), Pearson's chi-square test (Slakter, 1965), Kullback-Leibler (Joyce, 2011) or Jensen-Shannon divergences (Fuglede and Topsoe, 2004). However, the selection of one method or another is still an open issue (Álvarez-Meza et al., 2017).

Lastly, the main bottleneck of online prediction algorithms is that complexity increases with the number of samples (Chen et al., 2012b). Thus, the challenge is to select only an important subset of data to train the model (Honeine, 2015), which is also known as *sparsification*. There are two main *sparsification* approaches (Liu et al., 2011): (1) *elimination*, e.g., support vector machines (Vapnik, 2013), regularization networks (Evgeniou et al., 2000), relevance vector machines (Tipping, 2001), and least-squares support vector machines (Suykens et al., 2000); (2) *construction*, e.g., quantized kernel least-mean-square (Chen et al., 2012b) and variants (Zheng et al., 2016). The *sparsification* criterion determines whether or not a new sample is "important" to train the model. On the one hand, if this criterion is too restrictive, then the model may not be trained correctly. On the other hand, if the criterion is relaxed, then *sparsification* no longer makes sense (Liu et al., 2011).

Consequently, this thesis aims to address the following research problems: *i)* How to improve prediction tasks to infer the future behavior of time-series under noisy and non-stationary conditions?; *ii)* How to preserve the significant structure of high-dimensional data in a low-dimensional space to enhance the performance of a pattern recognition system?; *iii)* How to use low-dimensional representations in online prediction tasks to avoid overfitting, reduce computational complexity, and provide stable solutions in real-life scenarios?

[7]In practice, a manifold is the underlying support of data distribution.
[8]Representation of a topological object, like a manifold, in a certain space.

1.3. Previous Work

This section describes the scientific work related to the research problems (see Section 1.2) of this thesis. Note that there is a subsection for each research problem.

1.3.1. Research Problem 1: long-term future behavior

The prediction of time series is of great interest because it can guide decisions in many economic or industrial fields (Coussement et al., 2017; Brock, 2018). Predictions are usually obtained from a sequence of past observations, also known as time-series (Weigend, 2018). The further in the future you can predict, the more useful the predictions will be (Xiong et al., 2013). This is what we call long-term future behavior or prediction of several steps ahead. However, the design of a prediction algorithm for several steps ahead is not a trivial task, i.e., complex dependencies between observations, accumulative errors, and loss of accuracy (Liu, 2009). Additionally, these algorithms have two challenges (Ben Taieb et al., 2014): (1) how to select the prediction strategy; (2) how to choose the model to perform the predictions.

Firstly, there are three commonly used prediction strategies (Taieb et al., 2010): (1) *Iterated*, here a one-step-ahead predictor is iterated many times to obtain several predictions in the future. That is, the future series value is estimated and then this value is fed back as an input to the following prediction. Thus, the predictor takes estimated values as inputs, which means negative consequences in terms of error propagation, e.g., recurrent neural networks and variants (Zaremba et al., 2014; Hardy and Buonomano, 2018); (2) *Direct*, this strategy estimates a set of prediction models, each returning a direct prediction, which means higher functional complexity than iterated ones, e.g., the combination of k-nearest neighbors (Cunningham and Delany, 2007), mutual information (Peng et al., 2005), and nonparametric noise estimation methods (Sorjamaa et al., 2007); (3) *Mimo*, unlike the previous strategies, it returns a vector of future values in a single step, e.g., locally constant method for multi-output regression (Bontempi and Taieb, 2011). This strategy avoids the error accumulation of iterated and the conditional independence assumption of direct (Taieb et al., 2010).

Secondly, the model to perform long-term predictions is usually: (1) linear, e.g., autoregressive moving average (Erdem and Shi, 2011; Isufi et al., 2017), simplified autoregressive moving average (Oliveira et al., 2018), and autoregressive integrated moving average (Zou et al., 2017). However, these models may not be adequate to characterize the stochastic nature of real-world data (Chen and Yu, 2014); (2) nonlinear, e.g., long short-term memory (Hochreiter and Schmidhuber, 1997), extreme learning machine (Huang et al., 2006), feed forward neural networks (Ketkar, 2017; Ojha et al., 2017), and echo state networks (Jaeger, 2007; Chitsazan et al., 2017; Gallicchio and Micheli, 2017). Although nonlinear models based on neural networks have shown high tolerance to noisy conditions (Chen et al., 2013), their application in prediction tasks is subject to the assumption that test data has similar samples as the training data (Taieb et al., 2012).

Kernel-based adaptive filters, like neural networks, are nonlinear approximators. They combine the universal approximation property of neural networks and the convex optimization of linear adaptive filters. In contrast to neural networks, kernel-based adaptive filters are non-parametric, have convex optimization, and moderate computational complexity (Liu et al., 2010). These algorithms, also known as kernel adaptive filters (KAF), operate in a very special Hilbert space of functions called a reproducing kernel Hilbert space (RKHS). The operation with such functions become much easier in RKHS if the computation is restricted to inner products, which is the idea behind kernel methods. The following KAF algorithms have proven to be an alternative in time series prediction: (1) kernel least-mean-square (Liu et al., 2008), which is the simplest among the family of KAF; (2) kernel affine projection (Liu and Príncipe, 2008), which provides a unifying model for several neural network techniques, including kernel least-mean-square algorithms, kernel adaline, and regularization networks; (3) kernel recursive least-squares (Engel et al., 2004), this algorithm performs linear regression in a high-dimensional feature space induced by a Mercer kernel (Scholkopf and Smola, 2001); (4) extended kernel recursive least-squares (Liu et al., 2009), which implements a general linear state model in RKHS; (5) quantized kernel least-mean-square (Chen et al., 2012b) and variants (Chen et al., 2012a; Zhao et al., 2013; Zheng et al., 2016; Luo et al., 2017), which aims to reduce the computational complexity of KAF algorithms. However, as far as we know, there have not been previous work studying how to integrate KAF algorithms with *Mimo* strategies to predict long-term future behavior of time-series.

1.3.2. Research Problem 2: dimensionality reduction

Dimensionality reduction aims to provide a faithful representation of data in a low-dimensional space (Paul and Chalup, 2017). These methods have found extensive application in diverse areas, such as information retrieval (Zhang et al., 2018a; Zhuo et al., 2014), computer vision (Cassisi et al., 2012; Simão et al., 2017), compressive sensing (Gao et al., 2012), and streaming data (Na and Lee, 2014). There are two main challenges in dimensionality reduction methods (Lee and Verleysen, 2007): (1) how to measure the underlying support of data distribution, which is also known as manifold; (2) how to preserve the representation of a topological object, like a manifold, in a certain space.

The first challenge is to find relationships in the high-dimensional and low-dimensional spaces, which are usually calculated using: (1) variance preservation, e.g., principal component analysis (PCA) (Jolliffe, 1986); (2) dot product preservation, e.g., classical multidimensional scaling (MDS) (Borg and Groenen, 2005); (3) weighted distance preservation, e.g., nonlinear variants of MDS (Sammon, 1969); (4) affinity matrices, e.g., Laplacian eigenmaps (Belkin and Niyogi, 2002), locally linear embedding (Roweis and Saul, 2000), diffusion maps (Nadler et al., 2006), and isometric feature mapping (Isomap) (Lee et al., 2004); (5) similarity preservation, e.g., stochastic neighbor embedding (SNE) (Hinton and Roweis, 2003) and variants (Maaten and Hinton, 2008; Van Der Maaten, 2014). Although these methods have proven useful in a wide variety of real-world problems (Sadatnejad and Ghidary, 2016; Chen et al., 2016b; Sun and Wen, 2017), their success is subject to (Tsai, 2010; Peluffo Ordoñez et al., 2014): (1) a reliable local neighborhood; (2) data points densely sampled; (3) tune several free parameters for each data set.

The second challenge, in dimensionality reduction methods, is how to measure the mismatch between the high-dimensional and low-dimensional representations, i.e., embedding preservation. This mismatch is usually calculated using Mercer kernels (Scholkopf and Smola, 2001), Pearson's chi-square test (Slakter, 1965), Kullback-Leibler (Joyce, 2011) or Jensen-Shannon divergences (Fuglede and Topsoe, 2004). For example, a spectral dimensionality reduction method proposes to preserve the Renyi entropy in the input space using a kernel-based estimator (Shi et al., 2015), i.e., the Renyi entropy is expressed in terms of projections onto principal axes in kernel feature space. A recent study proposes a new dimensionality reduction method (Álvarez-Meza et al., 2017), where Mercer kernels are used to measure the manifold structures and the SNE method is combined with a kernel-based entropy criterion to preserve the embedding. On the one hand, due to the Mercer kernels, kernel-based methods creates symmetric and positive-definite matrices. On the other hand, due to the Kullback-Leibler divergence, SNE-based techniques provide asymmetric similarities. In other words, the manifold structure is measured using a symmetric strategy, while the embedding preservation is computed with an asymmetric technique. Consequently, the universal approximating capability and numeric stability, provided by some Mercer kernels (Liu et al., 2011), may be affected by the asymmetry of SNE-like techniques. This means that poor performance, in the preservation of global data structures, can be expected.

1.3.3. Research Problem 3: sparsification

The baseline solutions to perform prediction tasks are the statistical methods, mostly employing some improved versions of regressive models (Yang et al., 2018). However, their imposed analytic models frequently face numerous restrictions when dealing with non-stationarities and nonlinearities of data (Chen and Yu, 2014). To overcome nonlinearities, data-driven approaches are widely used like Neural Networks (NN), employing one or more layers of non-linear units to predict outputs. Nonetheless, NN algorithms tend to demand long training time and may get stuck in local minima (Ren et al., 2015).

Other data-driven approaches that have proven useful in prediction tasks are the kernel methods, which embed data into a potentially infinite dimensional feature space. In contrast to NNs, kernel-based adaptive filters have convex optimization and moderate computational complexity. Even that the choice of a kernel is non-trivial and does usually depend on the specific application, the kernel methods pose three main open issues (Chen et al., 2012b): *i*) selection of an appropriate kernel bandwidth; *ii*) step-size parameter tuning; *iii*) selection of samples to train the model.

Having significant influence on the learning performance, the kernel bandwidth controls the mapping smoothness and can be set manually or estimated in advance by Silverman's rule (Sheather, 2004), penalizing functions (Härdle, 1990), and cross validation (An et al., 2007). For determining an optimal bandwidth in kernel-based adaptive filters, however, an approximation in a joint space must be performed, which is different from density estimation. Thus, a stochastic gradient-based bandwidth optimization is developed in (Chen et al., 2016a), showing that the variable bandwidth fosters the kernel-based adaptive filters to converge faster and achieve better accuracy. Yet, joint optimization of bandwidth and learning-rate parameters remains an open matter. Another parameter to optimize is the step-size that reflects a tradeoff between misadjustment and speed of adaptation. If this parameter is too large, the risk of overfitting increases, while a small step-size decreases misadjustment, but also gives a longer convergence time. A variety of adaptive step-size methods have been proposed to improve performance of the standard least-mean-square algorithm (Li and Hamamura, 2015; Niu and Chen, 2018). However, these methods may not work properly on kernel-based adaptive filters, as they originate from a different problem (Chen et al., 2016a). Thus, this parameter is usually calculated off-line, meaning that it remains unchanged through iterations (Zheng et al., 2016). Lastly, the main bottleneck of kernel-based adaptive filters is that computation scales with the number of samples (Chen et al., 2012b). These algorithms train the model using a sequence of input vectors with their target predictions. Thus, the aim is to select only an important subset of the training data, also known as dictionary, to train the model (Honeine, 2015). Samples stored in the dictionary should cover, as much as possible, the area where new samples are likely to appear. In practice, this means to store "sparse" samples, which is why this technique is also known as *sparsification* (Zhang et al., 2018b).

There are two main *sparsification* approaches (Liu et al., 2011): (1) *elimination*, where all samples are stored in the dictionary and then some of them are eliminated by solving an optimization problem, e.g., support vector machines (Vapnik, 2013), regularization networks (Evgeniou et al., 2000), relevance vector machines (Tipping, 2001), and least-squares support vector machines (Suykens et al., 2000); (2) *construction*, where the algorithm starts with an empty dictionary and gradually adds new samples, according to some criterion, during the learning process, e.g., quantized kernel least-mean-square (Chen et al., 2012b) and variants (Zheng et al., 2016). The challenge is to design a suitable *sparsification* criterion for online prediction. However, if this technique is too restrictive, the model may not be trained correctly and poor performance can be expected, while if the technique is relaxed, then sparsification no longer makes sense (Liu et al., 2011).

1.4. Objectives

1.4.1. General Objective

To develop a kernel-based framework to analyze high-dimensional data for the support of regression tasks within continuous adaptation scenarios. The framework must identify complex dependencies between observations to infer long-term future behavior of noisy and non-stationary time-series. In addition, the significant structure of high-dimensional data must be preserved, as much as possible, in a low-dimensional space to enhance the performance of pattern recognition systems within the context of real-life scenarios.

1.4.2. Specific Objectives

- To develop a kernel-based approach to infer the long-term future behavior of noisy and non-stationary time-series. The proposed approach must highlight the inherent complex dynamics of continuous adaptation scenarios and provide stable solutions within the context of real-life applications.

- To propose a new kernel-based dimensionality reduction approach to preserve the significant structure of high-dimensional data in a low-dimensional space. The proposed approach must remove noise and redundant features from high-dimensional data to support regression tasks.

- To build a kernel-based approach for online prediction tasks using low-dimensional representations from high-dimensional data to reduce computational complexity and provide stable solutions in real-life scenarios.

1.5. Contributions

Considering the results of the proposed models, we highlight the following contributions of this thesis:

- **A kernel-based framework to predict long-term future behavior in time-series is proposed.** The method is able to capture complex dynamics within continuous adaptation scenarios. The proposed approach is validated on two real-world datasets, showing its ability to provide stable solutions in short, medium, and long-term prediction tasks (See Chapter 3).

- **A new dimensionality reduction method that uses a Mercer kernel to measure the manifold structures.** Additionally, a kernel-based cost function is proposed to quantify the embedding preservation. In contrast to other dimensionality reduction approaches, the proposed method is easy to implement in different datasets due to its small number of hyper-parameters. The method is validated on both synthetic and real-world datasets. Simulation results show that our proposal preserves the global structures in a variety of datasets (See Chapter 4).

- **A framework that addresses three well-known problems of kernel-based adaptive filters.** In contrast to similar methods, the proposed framework sequentially optimize the kernel bandwidth and step-size parameters using stochastic gradient algorithms that maximize the correntropy function (See Chapter 5).

- **A *sparsification* approach based on dimensionality reduction is proposed to remove redundant samples from high-dimensional data within continuous adaptation scenarios.** The framework is validated on both synthetic and real-world datasets. Simulation results show that our proposal reduce computational complexity and provide stable solutions in real-world applications (See Chapter 5).

2. Background

This chapter briefly describes the technical background used throughout this thesis. Firstly, we provide the mathematical background on the use of kernel functions for machine learning applications. Secondly, we introduce the simplest and most commonly used form of an adaptive filtering algorithm. Lastly, we formally describe some of the most-well known dimensionality reduction techniques. The content of this chapter is based on the works of (Scholkopf and Smola, 2001; Lee and Verleysen, 2007; Liu et al., 2010).

2.1. Reproducing kernel Hilbert spaces

Mercer kernels are continuous, symmetric, and positive-definite functions (Scholkopf and Smola, 2001). The most commonly known are the Gaussian (Equation (2-1a)) and the polynomial kernels (Equation (2-1b))

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(-\|\boldsymbol{u} - \boldsymbol{u}'\|^2 / 2\sigma^2\right) \tag{2-1a}$$

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \left(\boldsymbol{u}^\top \boldsymbol{u}' + 1\right)^p \tag{2-1b}$$

where $\kappa : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ denotes the Mercer kernel, $\mathbb{U}$ is the input domain, $\|\cdot\|$ stands for ℓ_2 norm, $\sigma \in \mathbb{R}^+$ is the bandwidth that controls the mapping smoothness, p is the polynomial degree, while $\boldsymbol{u}, \boldsymbol{u}' \in \mathbb{U}$ are input vectors. More formally (Liu et al., 2011), let $\mathbb{H}$ be any vector space of all real-valued functions of $\boldsymbol{u}$ that are generated by the kernel $\kappa(\boldsymbol{u}, \cdot)$. In addition, suppose that the following two functions are picked from the space $\mathbb{H}$

$$h = \sum_{i=1}^{l} = a_i \kappa(\boldsymbol{c_i}, \cdot) \tag{2-2a}$$

$$g = \sum_{i=1}^{m} = b_j \kappa(\tilde{\boldsymbol{c}}_{\boldsymbol{i}}, \cdot) \tag{2-2b}$$

Thus, from Equations (2-2a) and (2-2b), the following *bilinear form* is obtained

$$\langle h, g \rangle = \sum_{i=1}^{l} \sum_{j=1}^{m} a_i \kappa(\boldsymbol{c_i}, \tilde{\boldsymbol{c}}_i) b_j \tag{2-3}$$

Note, when $g(\cdot) = \kappa(\boldsymbol{u}, \cdot)$, the above expression leads to the *reproducing property* (Szafraniec, 2015) shown in Equation (2-4)

$$\langle h, \kappa(\boldsymbol{u}, \cdot)\rangle = \sum_{i=1}^{l} a_i \kappa(\boldsymbol{c}_i, \boldsymbol{u}) = h(\boldsymbol{u}) \tag{2-4}$$

The kernel $\kappa(\boldsymbol{u}, \boldsymbol{u}')$, which represents a function of the two vectors, is called a reproducing kernel of the vector space $\mathbb{H}$ if the following two conditions are met (Wu et al., 2015): *i)* for every $\boldsymbol{u} \in \mathbb{U}$, $\kappa(\boldsymbol{u}, \boldsymbol{u}')$ as a function of the vector $\boldsymbol{u}'$ belongs to $\mathbb{H}$; *ii)* it satisfies the *reproducing property*. The previous conditions are satisfied by the Mercer kernel, thereby endowing it with the designation "reproducing kernel". Thus, if the inner product space $\mathbb{H}$, in which the reproducing kernel space is defined, is also complete, then it is called a reproducing kernel Hilbert space (RKHS). In this sense, the Mercer theorem (Burges, 1998) states that any reproducing kernel $\kappa(\boldsymbol{u}, \boldsymbol{u}')$ can be expanded as follows:

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \sum_{i=1}^{\infty} \zeta_i \phi_i(\boldsymbol{u}) \phi_i(\boldsymbol{u}') \tag{2-5}$$

where ζ_i and ϕ_i are the eigenvalues and the eigenfunctions, respectively. The eigenvalues are non-negative. Therefore, a mapping φ can be constructed as

$$\varphi : \mathbb{U} \to \mathbb{F} \tag{2-6}$$

$$\varphi(\boldsymbol{u}) = \left[\sqrt{\zeta_1}\phi_1(\boldsymbol{u}), \sqrt{\zeta_2}\phi_2(\boldsymbol{u}), \ldots\right] \tag{2-7}$$

Note, in the machine learning literature, φ is usually treated as the feature mapping and $\varphi(\boldsymbol{u})$ is the transformed feature vector lying in the feature space $\mathbb{F}$, which is an inner product space (see Figure 2-1).

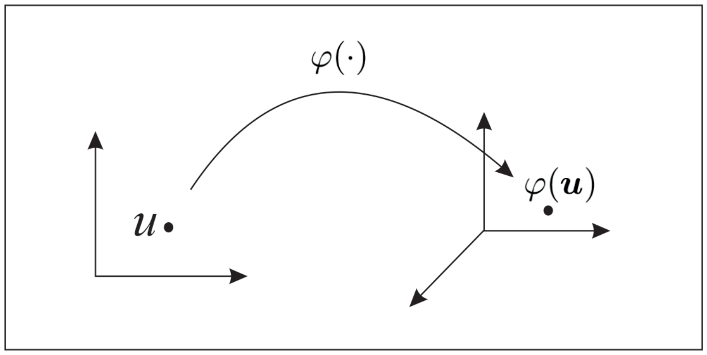


Figure 2-1.: Nonlinear mapping $\varphi(\cdot)$ from the input space to the feature space.

That is, any Mercer kernel $\kappa(\boldsymbol{u}, \boldsymbol{u}')$, induces a mapping φ such that the following relationship, the kernel trick (Scholkopf and Smola, 2001), holds:

$$\varphi(\boldsymbol{u})^\top \varphi(\boldsymbol{u}') = \kappa(\boldsymbol{u}, \boldsymbol{u}') \tag{2-8}$$

The feature space $\mathbb{F}$ is essentially the same as the RKHS induced by the kernel by identifying $\varphi(\boldsymbol{u}) = \kappa(\boldsymbol{u}, \cdot)$, which are the bases of the two spaces, respectively. Thus, as suggested in (Liu et al., 2011), we do not distinguish $\mathbb{F}$ and $\mathbb{H}$ if no confusion is involved.

2.2. Linear Adaptive Filters

The simplest and most commonly used form of an adaptive filtering algorithm is the so-called *least-mean-square* (LMS) algorithm. Let us assume that the goal is to learn a continuous input-output mapping $f : \mathbb{U} \to \mathbb{R}$ based on a sequence of input-output samples $\{\boldsymbol{u}_1, y_1\}$, $\{\boldsymbol{u}_2, y_2\}$, ..., $\{\boldsymbol{u}_t, y_t\}$, where $\boldsymbol{u}_t$ is an M-dimensional input vector that belongs to the input domain $\mathbb{U} \subset \mathbb{R}^M$, and $y_t \in \mathbb{R}$ is the output time series over the time domain. The LMS algorithm operates by minimizing the instantaneous cost function $J_t = \frac{1}{2}e_t^2$, where e_t is defined as follows:

$$e_t = y_t - \mathbf{w}_{t-1}^\top \boldsymbol{u}_t \tag{2-9}$$

The optimal weight $\mathbf{w}_t$ can be computed using the instantaneous version of the gradient descent, that is,

$$\begin{aligned}
\mathbf{w}_t &= \mathbf{w}_{t-1} - \eta \left[\frac{\partial}{\partial \mathbf{w}_{t-1}} J_t\right] \\
\mathbf{w}_t &= \mathbf{w}_{t-1} - \eta \left[-e_t \boldsymbol{u}_t\right] \\
\mathbf{w}_t &= \mathbf{w}_{t-1} + \eta e_t \boldsymbol{u}_t
\end{aligned}$$

where $\eta \in \mathbb{R}^+$ is the step-size parameter. Consequently, the LMS algorithm assumes a linear model and uses the following procedure (Haykin et al., 2003):

$$\begin{cases}
\mathbf{w}_0 = 0 \\
e_t = y_t - \mathbf{w}_{t-1}^\top \boldsymbol{u}_t \\
\mathbf{w}_t = \mathbf{w}_{t-1} + \eta e_t \boldsymbol{u}_t
\end{cases} \tag{2-10}$$

However, if the mapping between y and $\boldsymbol{u}$ is highly nonlinear, then poor performance can be expected from LMS.

2.3. Kernel Adaptive Filters

Kernel adaptive filters (KAF) provide a generalization of linear adaptive filters as the later become a special case of the former when expressed in the dual space. The learning rule is a combination of the *error-correction* and *memory-based learning.* These algorithms reproduce kernel Hilbert spaces (RKHS), which uses a linear adaptive structure to obtain nonlinear filters in the input space.

The kernel *least-mean-square* (KLMS) algorithm is the simplest among the family of KAF. To overcome the limitation of linearity in LMS, KLMS employs a kernel-induced mapping $\varphi : \mathbb{U} \to \mathbb{F}$ to transform the input $\boldsymbol{u}_t$ into a high-dimensional feature space $\mathbb{F}$, which is an inner product space, as $\varphi(\boldsymbol{u}_t)$.

When the LMS algorithm is applied to the sample sequence $\{\varphi(\boldsymbol{u}_t), y_t\}$, the following sequential rule is obtained:

$$\begin{aligned}\boldsymbol{\omega}_0 &= 0\\ e_t &= y_t - \boldsymbol{\omega}_{t-1}^\top \varphi(\boldsymbol{u}_t)\\ \boldsymbol{\omega}_t &= \boldsymbol{\omega}_{t-1} + \eta e_t \varphi(\boldsymbol{u}_t)\end{aligned}$$

where $\boldsymbol{\omega}_t = \eta \sum_{j=1}^{t} e_j \varphi(\boldsymbol{u}_j)$ is the weight vector in $\mathbb{F}$ at iteration t. Thus, when $\boldsymbol{u}'$ arrives to the filter, the output can be computed as follows:

$$\begin{aligned}\boldsymbol{\omega}_t^\top \varphi(\boldsymbol{u}') &= \left[\eta \sum_{j=1}^{t} e_j \varphi(\boldsymbol{u}_j)^\top\right] \varphi(\boldsymbol{u}')\\ \boldsymbol{\omega}_t^\top \varphi(\boldsymbol{u}') &= \eta \sum_{j=1}^{t} e_j \left[\varphi(\boldsymbol{u}_j)^\top \varphi(\boldsymbol{u}')\right]\end{aligned}$$

Note that, $\varphi(\boldsymbol{u}_j)^\top \varphi(\boldsymbol{u}')$ is actually the kernel trick (Equation (2-8)). Thus, the filter output is computed in the input space by kernel evaluations as:

$$\boldsymbol{\omega}_t^\top \varphi(\boldsymbol{u}') = \eta \sum_{j=1}^{t} e_j \kappa(\boldsymbol{u}_j, \boldsymbol{u}')$$

Then, if f_t is denoted as the estimate of the input-output nonlinear mapping at time t, the following sequential rule in the original space for KLMS is obtained (Liu et al., 2010):

$$\begin{cases} f_0 = 0\\ e_t = y_t - f_{t-1}(\boldsymbol{u}_t)\\ f_t = f_{t-1} + \eta e_t \kappa(\boldsymbol{u}_t, \cdot)\end{cases} \tag{2-11}$$

This sequential rule produces a growing radial-basis-function network by allocating a new kernel unit for every new sample with $\boldsymbol{u}_t$ as the center and ηe_t as its coefficient.

2.4. Dimensionality Reduction

The aim of dimensionality reduction approaches is to find a low-dimensional representation from a high-dimensional space so that the performance of a pattern recognition system can be improved (Lee and Verleysen, 2007). Thus, several techniques such as Locally Linear Embedding (LLE) (Roweis and Saul, 2000), Laplacian Eigenmaps (LEM) (Belkin and Niyogi, 2002), and Isometric Feature Mapping (Isomap) (Lee et al., 2004) have been developed. In the following sections, some of the most well-known dimensionality reduction techniques are described.

2.4.1. Locally Linear Embedding – LLE

This technique aims to compute a low-dimensional embedding using simple geometric intuitions. That is, nearby points in the high-dimensional space should remain nearby in the low-dimensional representation (Roweis and Saul, 2000). Let $\boldsymbol{U}{=}\{\boldsymbol{u}_i{\in}\mathbb{R}^M{:}i{\in}[1,N]\}$ be a high-dimensional finite set, containing M features extracted at N samples, for which a low-dimensional representation, $\boldsymbol{V}{=}\{\boldsymbol{v}_i{\in}\mathbb{R}^m{:}i{\in}[1,N]\}$, must be obtained, so that it holds that $m \leq M$. The LLE procedure has the following stages:

1. The k-nearest neighbors per point are searched and measured by Euclidean distance;

2. Compute the weight matrix $\mathbf{W}{\in}\mathbb{R}^{N\times N}$ that minimize the reconstruction error

$$\varepsilon\left(\mathbf{W}\right) = \sum_{i=1}^{N} \left\| \boldsymbol{u}_i - \sum_{j=1}^{N} \omega_{ij}\boldsymbol{u}_j \right\|^2 ; \tag{2-12}$$

3. The low-dimensional representation $\boldsymbol{V}$ is found by minimizing

$$\Phi\left(\boldsymbol{V}\right) = \sum_{i=1}^{N} \left\| \boldsymbol{v}_i - \sum_{j=1}^{N} \omega_{ij}\boldsymbol{v}_j \right\|^2 , \tag{2-13}$$

   subject to: *i)* $\sum_{i=1}^{N} \boldsymbol{v}_i = \mathbf{0}$; *ii)* $\sum_{i=1}^{N} \boldsymbol{v}_i\boldsymbol{v}_i^\top/N = \boldsymbol{I}_{m\times m}$. Note, $\boldsymbol{I}$ is an identity matrix and $\boldsymbol{v}_i{\in}\mathbb{R}^m$ is a row output vector of $\boldsymbol{V}$.

2.4.2. Laplacian Eigenmaps – LEM

This nonlinear dimensionality reduction technique aims to preserve the intrinsic geometric structure of the manifold (Belkin and Niyogi, 2002). Let $\boldsymbol{U}{=}\{\boldsymbol{u}_i{\in}\mathbb{R}^M{:}i{\in}[1,N]\}$ be a high-dimensional finite set, containing M features extracted at N samples. The goal is to provide a low-dimensional representation $\boldsymbol{V}{=}\{\boldsymbol{v}_i{\in}\mathbb{R}^m{:}i{\in}[1,N]\}$, being $m \ll M$. The LEM algorithm has the following main steps:

1. Compute an undirected weighted graph;

2. Construct a weight matrix $\mathbf{W}{\in}\mathbb{R}^{N\times N}$. For this purpose, if nodes i and j are connected, then $W_{ij} = \kappa\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right)$, where $\kappa\left(\cdot,\cdot\right)$ is a kernel function (see Section 2.1), otherwise $W_{ij} = 0$. Given $\mathbf{W}$, the graph Laplacian $\mathbf{L}{\in}\mathbb{R}^{N\times N}$ is defined as in Equation (2-14)

$$\mathbf{L} = \mathbf{D} - \mathbf{W}, \tag{2-14}$$

   where $\mathbf{D}{\in}\mathbb{R}^{N\times N}$ is a diagonal matrix with elements $D_{ii} = \sum_j W_{ji}$. Note, each D_{ii} is usually called the degree of the vertex $\boldsymbol{u}_i$, which can be interpreted as a measure of the empirical density points around each sample;

3. The following expression should be minimized

$$\sum_{ij} \left(\boldsymbol{v}_i - \boldsymbol{v}_j\right)^2 W_{ij}, \tag{2-15}$$

Equation (2-15) incurs a penalty if points $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ are mapped far apart.

2.4.3. Isometric Feature Mapping – Isomap

The Isometric Feature Mapping (Isomap) aims to preserve the intrinsic geometry of the data using the geodesic distance between all pairs of data points (Lee et al., 2004). The Isomap algorithm has the following stages:

1. Construct a discrete representation of the manifold in the form of a topology preserving network;
2. Compute the shortest-path distance between any two points in the network;
3. Construct a global geometry preserving map of the observations $\boldsymbol{V}$ in a low-dimensional Euclidean space

3. Proposed framework to predict long-term future behavior in time-series

In real-life scenarios, sometimes, it is useful to know not only the next value but also to have information about the long-term future behavior in time-series (Xiong et al., 2013). For example, in financial markets, the investors could hedge their assets or take appropriate actions given their investment objectives and risk tolerance (Han et al., 2013). In this chapter [1], a framework to predict long-term future behavior in time-series is introduced. The framework uses a *mimo* strategy and a non-linear model based on Kernel Adaptive Filters (KAF). In practice, for every new sample, the approach proceeds in two stages: *i)* training stage, where both past and future observations are known; *ii)* testing stage, where the prediction for the desired horizon is computed. The framework is validated in the following datasets: *i)* West Texas Intermediate (WTI) crude oil prices; *ii) NN3* competition time-series. Results show that the framework is robust to noisy and non-stationary conditions. In addition, when compared to similar methods, relatively low values of Symmetric Mean Absolute Percentage Error (SMAPE) are obtained, improving accuracy in short, medium and long-term. The main contribution of this chapter is a KAF-based framework to predict long-term future behavior in time-series.

The remainder of the chapter has the following structure: Section 3.1 introduces the proposed method; Sections 3.2 and 3.3 describe the simulation results; Section 3.4 presents a brief summary of the chapter.

[1]The outcomes of this chapter were published in the *IEEE International Joint Conference on Neural Networks – IJCNN 2016* (Garcia-Vega et al., 2016).

3.1. A mimo strategy based on kernel adaptive filters

Given the univariate time series $y_{t-M}, \ldots, y_t$, the goal is to predict the next M observations $y_{t+1}, \ldots, y_{t+M}$. The proposed framework comprises two stages: *i)* a training stage operating on a known vector $\boldsymbol{u}_t = [y_{t+1}, \ldots, y_{t+M}]$ that contains the next true observations of the univariate time series; *ii)* a testing stage operating on the vector $\tilde{\boldsymbol{u}}_t = [\tilde{y}_{t+1}, \ldots, \tilde{y}_{t+M}]$, which is an approximation of $\boldsymbol{u}_t$.

3.1.1. Training stage

Let $\boldsymbol{u}_t \in \mathbb{R}^M$ be a vector containing, at time t, the next true observations of a time series. The goal is to use $\boldsymbol{u}_t$ and the available dictionary[2] to find an accurate approximation $\hat{\boldsymbol{u}}_t$ of $\boldsymbol{u}_t$, where the dictionary and its weights are provided by the KAF method (see Section 2.3). The proposed algorithm starts when the first sample of the dictionary $\boldsymbol{u}_1 \in \mathbb{R}^M$ and its weight ηy_1 are created (Lines 3-4, Alg. 1). Then, the output of the adaptive filter is calculated[3] using $\boldsymbol{u}_t$ and the dictionary $\boldsymbol{C}$ (Line 9, Alg. 1). The *mimo* strategy is applied to get the approximation $\hat{\boldsymbol{u}}_t$ (Lines 11-13, Alg. 1). Finally, when $\hat{\boldsymbol{u}}_t$ is computed, the network is updated (Lines 15-19, Alg. 1) as in the traditional quantized KLMS algorithm (Chen et al., 2012b). The training stops when the average error of the current epoch is greater than the previous one [4].

Algorithm 1: Training stage to predict long-term future behavior in time-series.

```
1  Initialization:
2  $\sigma \in \mathbb{R}^+$– kernel bandwidth ; $\eta \in \mathbb{R}^+$– step-size parameter; $\varepsilon \geq 0$– quantization-size;
3  $\boldsymbol{C} = \{\boldsymbol{u}_1\}$– initial dictionary
4  $\boldsymbol{\alpha} = [\eta y_1]$– initial weight
5  Computation:
6  while $\boldsymbol{u}_t$ available do
7      Compute adaptive filter output:
8      for $n$=1:$M$ do
9          $h_n = \sum_{j=1}^{size(\boldsymbol{C})} \alpha_{j,n} \kappa_\sigma (c_{j,n}, y_{t+n})$
10     Mimo strategy:
11     for $n$=1:$M$ do
12         $\hat{y}_{t+n} = \sum_{r=n}^{M} h_r$
13     $\hat{\boldsymbol{u}}_t = [\hat{y}_{t+1}, \ldots, \hat{y}_{t+M}]$
14     Compute error: $e_t = \frac{\|\boldsymbol{u}_t - \hat{\boldsymbol{u}}_t\|^2}{\|\boldsymbol{u}_t\|^2}$
15     Select the closest sample to $\boldsymbol{u}_t$: $j^* = \arg \max_{1 \leq j \leq i^*} \kappa_\sigma (\boldsymbol{c}_j, \boldsymbol{u}_t)$
16     if $\kappa_\sigma (\boldsymbol{c}_{j^*}, \boldsymbol{u}_t) > \varepsilon$ then
17         Update the weight of the closest sample $j^*$: $\alpha_{j^*} = \alpha_{j^*} + \eta e_t$
18     else
19         Create a new sample in the dictionary: $\boldsymbol{C} = \{\boldsymbol{C}, \boldsymbol{u}_t\}$; $\boldsymbol{\alpha} = [\boldsymbol{\alpha}, \eta e_t]$
```

[2]The dictionary is a selected set of input samples used to estimate the nonlinear model (Saide et al., 2015).

[3]Note: (1) $c_{j,n}$ denotes the *n-th* value of $\boldsymbol{c}_j$; (2) The Mercer kernel κ_σ is assumed to be the Gaussian kernel due to its universal approximating capability and numeric stability (Liu et al., 2010).

[4]An epoch is a full training cycle on the training set.

3.1.2. Testing stage

The input vector $\boldsymbol{u}_t$ that contains the next true observations is not available during testing stage. Thus, to overcome this limitation, we propose to compute the approximation $\tilde{\boldsymbol{u}}_t$ as follows:

1. **if** $\frac{1}{M-1}\sum_{i=1}^{M-1}\left(y_{t-i}-y_{t-(i+1)}\right)>0$ **and** $\frac{1}{i^*M}\sum_{j=1}^{i^*}\sum_{n=1}^{M}c_{j,n}<y_t$:

$$\tilde{\boldsymbol{u}}_t=\left[y_{t-M}+\frac{1}{M-1}\sum_{i=1}^{M-1}\left(y_{t-i}-y_{t-(i+1)}\right),\dots,y_{t-1}+\frac{1}{M-1}\sum_{i=1}^{M-1}\left(y_{t-i}-y_{t-(i+1)}\right)\right] \tag{3-1}$$

2. **if** $\frac{1}{M-1}\sum_{i=1}^{M-1}\left(y_{t-i}-y_{t-(i+1)}\right)<0$ **and** $\frac{1}{i^*M}\sum_{j=1}^{i^*}\sum_{n=1}^{M}c_{j,n}>y_t$:

$$\tilde{\boldsymbol{u}}_t=\left[y_{t-M}-\frac{1}{M-1}\sum_{i=1}^{M-1}\left(y_{t-i}-y_{t-(i+1)}\right),\dots,y_{t-1}+\frac{1}{M-1}\sum_{i=1}^{M-1}\left(y_{t-i}-y_{t-(i+1)}\right)\right] \tag{3-2}$$

3. **otherwise**:

$$\tilde{\boldsymbol{u}}_t=\left[\max_{1\le j\le i^*}P\left(c_{j,1}|y_t\right),\quad \max_{1\le j\le i^*}P\left(c_{j,2}|y_t\right),\quad \dots,\quad \max_{1\le j\le i^*}P\left(c_{j,N}|y_t\right)\right] \tag{3-3}$$

Then, it is possible to build the sequence $\tilde{\boldsymbol{u}}_t$ during testing. Note that the algorithm could lead to poor performance when tries to predict a dynamic that was not present in the training set. Thus, the current trend of the time series is used to improve prediction (see Equations (3-1) and (3-2)). Lastly, once the sequence $\tilde{\boldsymbol{u}}_t$ is estimated, the algorithm predicts the next M values and the network is updated (see Alg. 2).

Algorithm 2: Testing stage to predict long-term future behavior in time-series

```
1  Initialization:
2  $\boldsymbol{C}$: provided by Alg. 1; $\boldsymbol{\alpha}$: provided by Alg. 1
3  Computation:
4  while $\tilde{\boldsymbol{u}}_t$ available do
5      $i^* = size(\boldsymbol{C})$
6      for $n$=1:$M$ do
7          $h_n = \sum_{j=1}^{i^*}\alpha_{j,n}\kappa_\sigma\left(c_{j,n},\tilde{y}_{t+n}\right)$
8      for $n$=1:$M$ do
9          $\hat{y}_{t+n} = \sum_{r=n}^{N} h_r$
10     $\hat{\boldsymbol{y}}_t = [\hat{y}_{t+1},\dots,\hat{y}_{t+M}]$
11     $e_t = \frac{\|\tilde{\boldsymbol{u}}_t-\hat{\boldsymbol{u}}_t\|^2}{\|\tilde{\boldsymbol{u}}_t\|^2}$
12     $j^* = \arg\max_{1\le j\le i^*}\kappa_\sigma\left(\boldsymbol{c}_j,\tilde{\boldsymbol{u}}_t\right)$
13     if $\kappa_\sigma\left(\boldsymbol{c}_{j^*},\tilde{\boldsymbol{u}}_t\right)>\varepsilon$ then
14         $\alpha_{j^*} = \alpha_{j^*} + \eta e_t$
15     else
16         $\boldsymbol{C} = \{\boldsymbol{C},\tilde{\boldsymbol{u}}_t\}$
17         $\boldsymbol{\alpha} = [\boldsymbol{\alpha},\eta e_t]$
```

3.2. Experimental Design

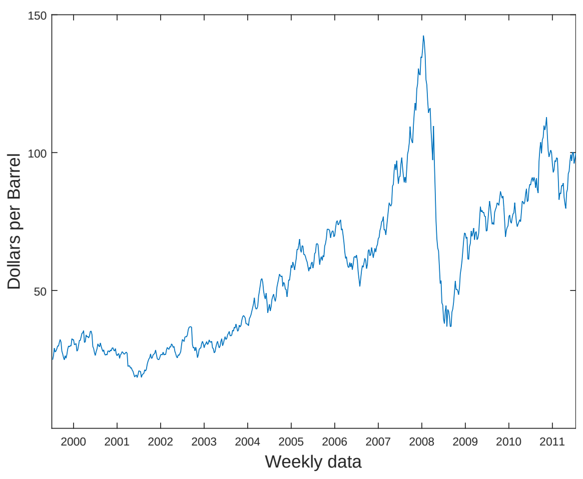
We validate the proposed framework for long-term prediction using two performance measures: *i)* SMAPE (Makridakis and Hibon, 2000); *ii)* Directional Symmetry (Yu et al., 2008).

3.2.1. Datasets

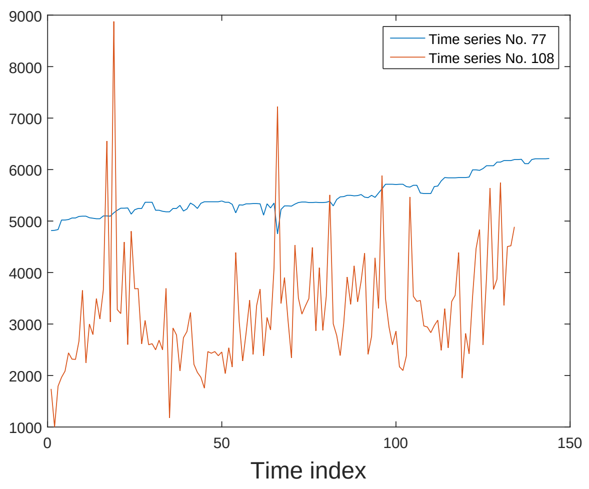
Testing is carried out on the following two publicly available datasets:

- ***WTI crude oil prices*** [5]*:* This dataset shows crude oil prices, in dollars per barrel, in a weekly resolution, covering January 7, 2000, to December 30, 2011 (see Figure 3.1(a)). Here, the task is to predict the next 4, 8, 12, 16, 20, and 24 values using the previous consecutive samples. The data are normalized for the computation convenience. The training set covers January 7, 2000, to January 10, 2008, while the test set covers January 11, 2008, to December 30, 2011.

- ***NN3 competition*** [6]*:* The dataset is composed by 111 monthly time series drawn from homogeneous population of empirical business (see Figure 3.1(b)). Here, the task is to predict the last 18 observations of each time series. The data are normalized for the computation convenience. The last 18 consecutive samples of each time series are used as test set, while the rest are used as the training set.



(a) WTI crude oil prices



(b) NN3 time-series

Figure 3-1.: Datasets considered in the experiments.

[5]This dataset publicly available at https://www.eia.gov

[6]This dataset publicly available at http://www.neural-forecasting-competition.com/NN3/datasets.htm

3.2.2. Reference methods

WTI crude oil prices

For comparison purposes, the proposed framework is contrasted with the following methods: *i)* Feedforward Neural Network (FNN) (Eldan and Shamir, 2016); *ii)* Empirical Mode Decomposition (EMD) (Ali et al., 2015); *iii)* Slope Based Method (SBM) (Xiong et al., 2013). In addition, the previous methods are tested using three commonly multi-step-ahead prediction strategies proposed in the literature, including *Iterated*, *Direct*, and *Mimo*. The considered feedforward neural network, as suggested in (Xiong et al., 2013), has one hidden layer with 15 neurons.

NN3 competition

The following methods are used for comparison purposes (Bao et al., 2014): *i)* Naïve; *ii)* Seasonal Naïve; *iii)* Support Vector Regression (Iterated); *iv)* Support Vector Regression (Direct); *v)* Support Vector Regression (Mimo).

3.2.3. Parameter settings

WTI crude oil prices

– ***Proposal***: The kernel bandwidth σ is calculated using the method proposed by (Cardenas et al., 2014); the step-size is set at $\eta = 0{,}09$; the quantization-size is set at $\varepsilon = 0{,}85$. The parameters were heuristically adjusted to provide the best possible accuracy.

– ***Reference methods***: For comparison purposes, we include the accuracy results estimated by (Xiong et al., 2013).

NN3 competition

– ***Proposal***: The set-up is the same as in *WTI crude oil* dataset.

– ***Reference methods***: For comparison purposes, we include the accuracy results estimated by (Bao et al., 2014).

3.3. Results

3.3.1. WTI Crude oil prices

Table **3-1** shows the SMAPE values in the test set [7]. Overall, the best performance is reached with a prediction horizon of 4, i.e., the longer the prediction horizon the higher the SMAPE values. The *iterated* strategy shows to be less accurate than the *direct* and *mimo* strategies. This indicates that, in the *iterated* strategy, the predictions may be affected by the accumulation of errors. In addition, we see that our proposal outperforms other methods, converging to smaller values of SMAPE. In our proposal, unlike the considered methods, the dictionary and its weights are also updated during the test stage (see Alg. 2). This allows the framework to adjust to abrupt changes in the system.

Method	**Prediction Horizon**					
	4	*8*	*12*	*16*	*20*	*24*
FNN (*Iterated*)	5.851	6.124	6.621	8.514	10.349	10.854
FNN (*Direct*)	4.242	4.413	4.851	6.048	8.594	9.524
FNN (*Mimo*)	4.275	4.628	5.128	6.241	8.196	8.818
EMD-FNN (*Iterated*)	4.123	4.284	5.685	6.219	7.928	8.548
EMD-FNN (*Direct*)	3.874	4.051	4.385	5.618	6.415	6.954
EMD-FNN (*Mimo*)	3.365	3.518	4.428	5.916	6.382	7.248
EMD-SBM-FNN (*Iterated*)	3.951	4.281	5.518	6.824	7.294	8.158
EMD-SBM-FNN (*Direct*)	**2.282**	**3.048**	4.351	**5.149**	6.161	6.948
EMD-SBM-FNN (*Mimo*)	3.214	3.531	**4.019**	5.348	**5.812**	**6.507**
Proposal	**0.621***	**1.351***	**2.029***	**2.436***	**3.033***	**3.591***

Table 3-1.: SMAPE values during testing in WTI crude oil prices.

Table **3-2** shows the directional symmetry values in the test set [8]. Once again, our proposal outperforms FNN, EMD, and SBM. In particular, the best performance is achieved for long prediction horizons, i.e., 16, 20, and 24 weeks ahead.

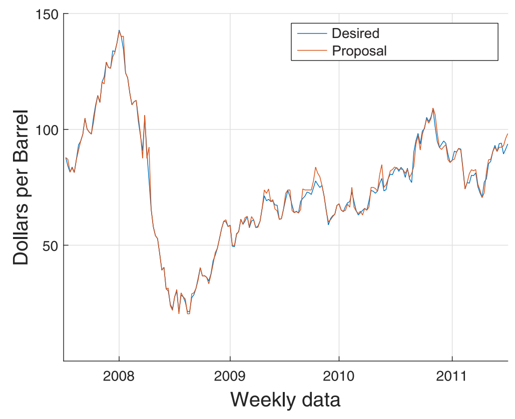
[7] Feedforward neural network (FNN); Empirical mode decomposition (EMD); slope-based methods (SBM). The words *Iterated*, *Direct*, and *Mimo* indicate the prediction strategy used by each method. For each column, the bold notation and marked with an asterisk indicates the method which is better than all the other, and the entry with the second-best value is highlighted in bold. For the sake of comparison, we include accuracies estimated in (Xiong et al., 2013).

[8] Feedforward neural network (FNN); Empirical mode decomposition (EMD); Slope-based methods (SBM). The words *iterated*, *direct*, and *mimo* indicate the prediction strategy used by each method. For each column, the bold notation and marked with an asterisk indicates the method which is better than all the other, and the entry with the second-best value is highlighted in bold. For comparison purposes, we include the accuracies estimated in (Xiong et al., 2013).

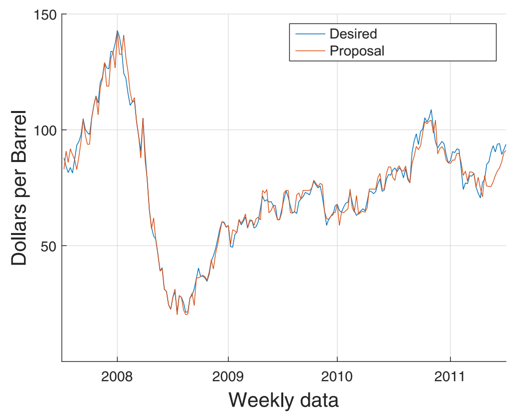
Method	**Prediction Horizon**					
	4	*8*	*12*	*16*	*20*	*24*
FNN (*Iterated*)	0.752	0.654	0.624	0.516	0.533	0.462
FNN (*Direct*)	0.797	0.765	0.733	0.651	0.652	0.524
FNN (*Mimo*)	0.801	0.744	0.723	0.692	0.707	0.598
EMD-FNN (*Iterated*)	0.786	0.704	0.642	0.676	0.562	0.514
EMD-FNN (*Direct*)	0.857	**0.832***	**0.824***	0.731	0.704	0.636
EMD-FNN (*Mimo*)	0.865	0.783	0.772	0.746	0.682	0.615
EMD-SBM-FNN (*Iterated*)	0.821	0.717	0.655	0.593	0.597	0.546
EMD-SBM-FNN (*Direct*)	**0.878**	0.815	0.803	0.752	0.685	0.597
EMD-SBM-FNN (*Mimo*)	0.810	0.823	0.812	**0.782**	**0.725**	**0.661**
Proposal	**0.903***	**0.826**	**0.821**	**0.787***	**0.734***	**0.748***

Table 3-2.: Directional symmetry values during testing in WTI crude oil prices.

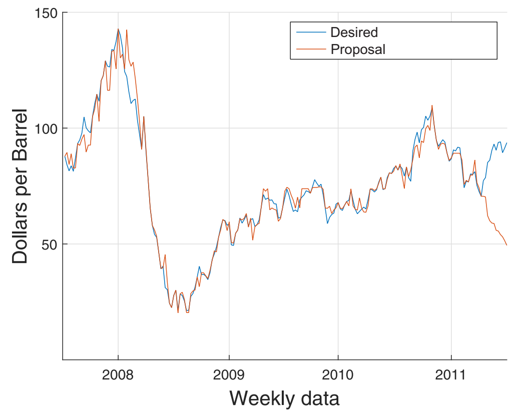
Figure **3-2** shows the predictions of our proposal in the test set. The case of short-term prediction performs the best accuracy (see Figures 3.2(a) and 3.2(b)), reaching SMAPE values of 0,621 and 1,351 (see Table **3-1**), respectively. For the medium-term prediction (see Figures 3.2(c) and 3.2(d)), although the performance slightly decreases, it is still acceptable (see Table **3-1**). The long-term prediction gives the worst performance (see Figures 3.2(e) and 3.2(f)) but achieves the best accuracy among the compared methods (see Tables **3-1** and **3-2**). Note, a big challenge during the testing stage is to predict observations that were not available in the training set. For example, the upward trend from January 11, 2008, to July 18, 2008, is only available in the test set. Thus, this upward trend is completely unknown by Alg. 2, which could lead to poor performance. To address this problem, we consider the most recent trend of the time series (see Equations (3-1) and (3-2)), which is provided by the last consecutive samples. Simulation results show that this strategy works well for prediction tasks in both the short and medium term (see Figure **3-2**).



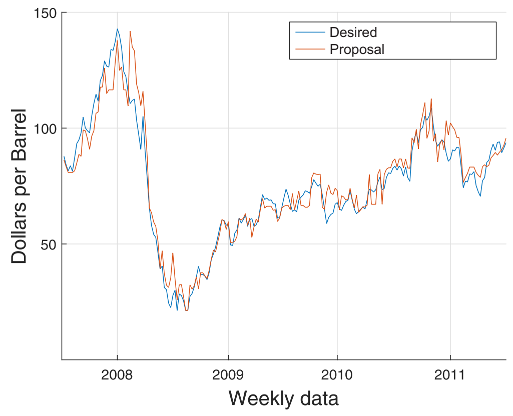
(a) Prediction Horizon = 4



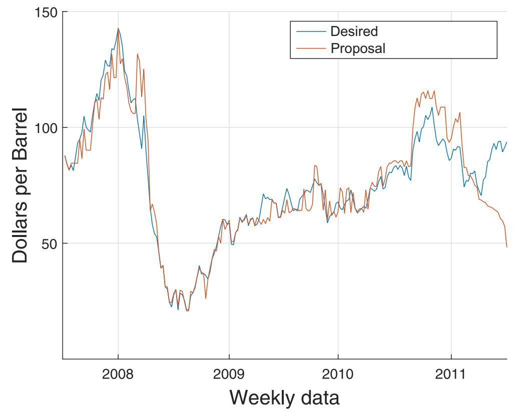
(b) Prediction Horizon = 8



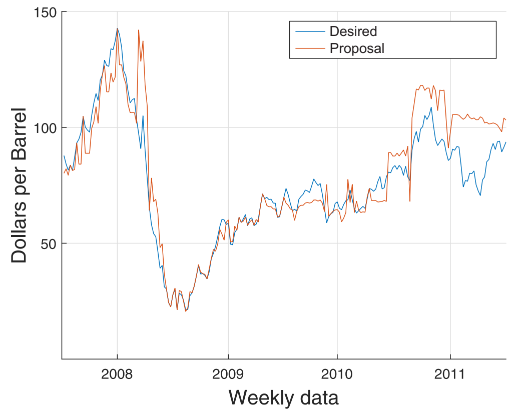
(c) Prediction Horizon = 12



(d) Prediction Horizon = 16



(e) Prediction Horizon = 20



(f) Prediction Horizon = 24

Figure 3-2.: Crude oil prices prediction in the test set.

3.3.2. NN3 time-series

Table **3-3** shows the SMAPE values in the test set[9]. The column labeled as *Average* shows the average SMAPE over the prediction horizon 1 to 18. Once again, our proposal achieves the best performance over all the prediction horizons. This behavior can be explained by the following reason: the proposed approach identifies the most recent trend of the time series to improve the prediction. This trend depends on the prediction horizon, i.e, the longer the horizon, the better the estimate of the trend. Thus, the proposed framework works better in long-term prediction tasks.

Method	**Testing Samples**				
	Prediction Horizon				**Average**
	2	*3*	*6*	*18*	*1 to 18*
Naïve	19.439	22.812	23.014	25.886	22.554
Seasonal Naïve	17.238	20.805	**16.629**	22.277	18.512
Support Vector Regression (*iterated*)	**8.824***	**10.916***	19.879	22.409	18.493
Support Vector Regression (*direct*)	**8.909**	**11.031**	20.724	21.781	17.193
Support Vector Regression (*mimo*)	9.077	11.074	17.370	**21.640**	**16.659**
Proposal	13.123	13.984	**15.942***	**16.139***	**14.753***

Table 3-3.: SMAPE values during testing in NN3 data set.

Figure **3-3** shows the performance of the proposed framework on each prediction horizon along all the time-series of NN3 dataset, i.e., each box is representing the prediction performance along the 111 time-series in terms of SMAPE. Note, there is one box per prediction horizon. In each box: the central mark is the median; the edges of the box are 25th and 75th percentiles; the whiskers extend to the most extreme data points, disregarding outliers; and outliers are plotted individually. As seen from this figure, the performance is stable on each prediction horizon. However, it is easy to identify some outliers, which are represented by the symbol "+". Note that each outlier represents the time-series with the worst performance in the prediction task. In this sense, we found that the proposed approach achieves the worst performance when tries to predict the time-series number 25 (this happens for all the prediction horizons), while the prediction of the time-series number 77 shows the best performance. Thus, according to these results, the proposed method fails when tries to predict highly non-stationary signals, e.g., time-series number 25.

[9]The words *iterated*, *direct*, and *mimo* indicate the prediction strategy used by each method. For each column, the bold notation and marked with an asterisk indicates the method which is significantly better than all the other, and the entry with the second-best value is highlighted in bold. For the sake of comparison, we include the accuracies estimated in (Bao et al., 2014).

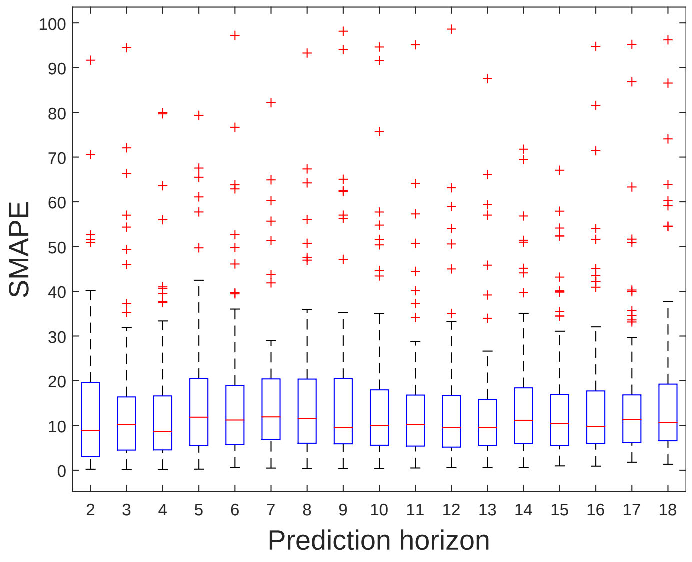


Figure 3-3.: Performance of the proposed framework on the NN3 dataset.

3.4. Summary

In this study, we proposed a *mimo* strategy using a kernel-based adaptive filter to predict long-term future behavior in time-series. The proposed approach employs multiple inputs from the previously stored samples to produce multiple predictions in the future through an adaptive filter based on kernels. This is done in two stages, i.e., training and testing. In the training stage, the network is trained according to the desired response. In the testing stage, the future values are predicted using the information provided by the training set. Additionally, the network continues learning from the time series, even during the testing stage. Simulation results show that the proposed approach outperforms the compared reference methods in all the considered prediction horizons.

4. Proposed nonlinear dimensionality reduction within kernel-based framework

Dimensionality reduction aims to preserve, as much as possible, the significant structure of high-dimensional data in the low-dimensional space. This allows removing noise and redundant features, which is useful in exploratory data analysis, classification, and regression tasks. There are two main challenges in dimensionality reduction: (1) how to measure the manifold structures; (2) how to quantify the embedding preservation. On the one hand, previous approaches try to measure the manifold structure using variance, dot product, distance, and similarity preservation. On the other hand, the embedding quality is usually quantified with divergence-based measures such as Kullback-Leibler and Jensen-Shannon. In this chapter [1], we propose a dimensionality reduction method that minimizes the mismatch between high and low-dimensional spaces. Unlike traditional dimensionality reduction formulations, the proposed approach uses a kernel-based cost function to quantify the embedding quality. Our approach is validated on both synthetic and real-world datasets. In terms of visual inspection and quantitative evaluation of neighborhood preservation, results show that our proposal preserves global data structures in the low-dimensional representation.

The remainder of the chapter has the following structure: Section 4.1 introduces the proposed method; Sections 4.2 and 4.3 describe the simulation results; Section 4.4 presents a brief summary of the chapter.

[1]The outcomes of this chapter were published in the peer-reviewed scientific journal *Pattern Recognition Letters* (Garcia-Vega and Castellanos-Dominguez, 2019).

4.1. Nonlinear dimensionality reduction within kernel-based framework

For reducing high-dimensional data, the Mercer kernel has been proposed before to measure the similarity between the high and low dimensional spaces, enabling the formulation of a kernel-based cost function that, in this work, is optimized through the gradient descent method.

Let $\boldsymbol{U}=\{\boldsymbol{u}_i\in\mathbb{R}^M:i\in[1,N]\}$ be a high-dimensional finite set, containing M features extracted at N samples, for which a low-dimensional representation, $\boldsymbol{V}=\{\boldsymbol{v}_i\in\mathbb{R}^m:i\in[1,N]\}$, must be obtained, so that it holds that $m<M$. With the aim of encoding all non-linear data relationships within the spaces, a couple of kernel matrices are introduced: i) input kernel, $\boldsymbol{P}\in\mathbb{R}^{N\times N}$ that holds elements $p_{ij}=\kappa_U(\boldsymbol{u}_i,\boldsymbol{u}_j)$, with $\kappa_U:\mathbb{R}^M\times\mathbb{R}^M\to\mathbb{R}^+$; ii) output kernel, $\boldsymbol{Q}\in\boldsymbol{R}^{N\times N}$ with elements $q_{ij}=\kappa_V(\boldsymbol{v}_i,\boldsymbol{v}_j)$, $\kappa_V:\mathbb{R}^m\times\mathbb{R}^m\to\mathbb{R}^+$. Both real-valued kernels, which are positive-definite (or Mercer) and infinitely divisible, are assumed to be the Gaussian due to their universal approximating capability, desirable smoothness, and numeric stability (Liu et al., 2011). The computation of the similarity matrices scales with the number of samples, meaning that it will take several hours to find a good embedding for large datasets. Hence, the similarity measures of high and low dimensional spaces are, respectively, given as follows:

$$p_{ij}=\exp\left(-\|\boldsymbol{u}_i-\boldsymbol{u}_j\|^2/2\sigma^2\right) \tag{4-1a}$$
$$q_{ij}=\exp\left(-\|\boldsymbol{v}_i-\boldsymbol{v}_j\|^2/2\sigma^2\right) \tag{4-1b}$$

where $\sigma\in\mathbb{R}^+$ is the kernel bandwidth, and notation $\|\cdot\|$ stands for ℓ_2 norm. Note that the kernel bandwidth allows defining the similarity metric in the reproducing kernel Hilbert space (RKHS), which is the basis of our inference. Namely, selecting a different bandwidth in either space, distinct similarities will be performed by the same kernel for the same data, resulting in a more complex cost function to be optimized. To relax the formulation in Equations (4-1a) and (4-1b), in practice, the same kernel bandwidth is selected on both spaces, assuming the same notion of neighborhood dispersion and therefore, meaning that we aim to preserve the similarities found on the RKHS spaces rather than similarities on the input space (Liu et al., 2011).

Therefore, the kernel-based framework is devised so that the more correctly the points $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ explain the similarity between the high-dimensional data points $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$, the more alike the kernel values p_{ij} and q_{ij} become. In other words, the principal rationale behind the suggested similarity framework in Equations (4-1a) and (4-1b) is to find a low-dimensional data representation $\boldsymbol{V}$ so that the mismatch between p_{ij} and q_{ij} can be minimized. To this end, the following cost function is proposed:

$$C=\frac{1}{N-1}\mathbb{E}\left\{|p_{ij}-q_{ij}|/p_{ij}:\forall i,j\in N,j\neq i\right\},\,C\in\mathbb{R} \tag{4-2}$$

where notation $\mathbb{E}\left\{\cdot\right\}$ denotes the expectation operator. It is worth noting that Equation (4-2) aims at preserving the global structures rather than the local ones, enabling extraction of the most relevant samples (from the viewpoint of embedding preservation) while maintaining a competitive performance in machine learning applications like in regression and classification (Zhong and Enke, 2017; Zhao et al., 2015). In particular, the proposed framework can be employed to determine whether a new sample should be used to estimate the nonlinear model. We suggest to perform the cost function minimization using a gradient descent method, yielding the learning rule described as below:

$$\boldsymbol{v}_i^t=\boldsymbol{v}_i^{t-1}-\eta\frac{\partial C}{\partial\boldsymbol{v}_i^{t-1}} \tag{4-3}$$

where $\boldsymbol{v}_i^{t-1}$ is the low-dimensional representation of $\boldsymbol{u}_i$ at iteration $t-1$ and $\eta\in\mathbb{R}^+$ is the step-size parameter. Taking into account Equations (4-1a), (4-1b) and (4-2), the gradient update results in the following rule (for a detailed explanation, see in Appendix A.1):

$$\boldsymbol{v}_i^t=\boldsymbol{v}_i^{t-1}-\eta'\mathbb{E}\left\{\frac{\left(\boldsymbol{v}_i^{t-1}-\boldsymbol{v}_j^{t-1}\right)\left(p_{ij}q_{ij}^{t-1}-\left(q_{ij}^{t-1}\right)^2\right)}{p_{ij}\left|p_{ij}-q_{ij}^{t-1}\right|}:\forall j\in N,j\neq i\right\} \tag{4-4}$$

where $\eta'=\eta/\sigma(N-1)$. Consequently, the updating rule in Equation (4-4) shows that using kernel-based similarity measures, the low-dimensional representations are updated iteratively. Likewise, during the optimization process, the low-dimensional similarities will change, while high-dimensional similarities stay constant. That is, the mismatch between high and low dimensional spaces will be reduced as long as q_{ij} tends to p_{ij}. Thus, the learning rule quantifies the embedding preservation in dimensionality reduction through the introduced kernel-based cost function. Note, the strategy used to compute the non-linear data relationships within the spaces (see Equations (4-1a) and (4-1b)) has similarities to some previously proposed methods as in (Spathis et al., 2018). As the key difference between both methods, our proposal handles the kernel function to compute similarities not only in the high-dimensional space but also in the low-dimensional one. That is, although both approaches compute in a similar way the similarities of the high-dimensional space, our proposed objective function aims at reducing the mismatch between the similarities provided by the kernel evaluations on each space. In addition, it may be possible to find some relationships with SNE-based techniques (Hinton and Roweis, 2003; Maaten and Hinton, 2008). However, the following are the key differences between both strategies: *i)* our proposal computes the similarity preservation using kernel evaluations rather than asymmetric probability measures; *ii)* the proposed framework uses a kernel-based cost function, while the SNE cost function relies on Kullback-Leibler divergence.

4.2. Experimental Design

To validate the proposed framework of high-dimensional data reduction, we measure the embedding quality as follows: *i)* Visual inspection (Álvarez-Meza et al., 2017); *ii)* Quality assessment (Lee and Verleysen, 2008). The task is to get a two-dimensional representation of each dataset (Lee et al., 2013).

4.2.1. Datasets

Testing is carried out on the following four publicly available datasets[2]:

– ***Swiss-Roll*** [3]*:* This collection contains samples that share nonlinear structures, generating an input space with size M=3 and N=500 (Yu et al., 2018). Here, the challenge is to cut the manifold so that the main nonlinear data structures are clearly revealed.

– ***S-Curve*** [4]*:* This collection is a standard benchmark for manifold learning, providing an input space with size M=3 and N=500 (Li et al., 2011).

– ***Wine*** [5]*:* This dataset is the result of a chemical analysis of wines grown in the same region in Italy by three different cultivators, generating an input space with size M=13 and N=178 (Fischer and Poland, 2005).

– ***MNIST*** [6]*:* This dataset contains 60000 gray-level images of scanned handwritten digits sizing 28×28. Here, as suggested in (Lee et al., 2013), a random subset of 6000 images are selected. These images are vectorized, providing an input space with M=784 and N=6000 (Deng, 2012).

4.2.2. Reference methods

For comparison purposes, the following dimensionality reduction methods are contrasted:

1. PCA (Abdi and Williams, 2010), which performs the linear projection applying the spectral decomposition of the covariance matrix;

2. Isomap (Tenenbaum et al., 2000), that uses the geodesic distance as metric;

3. SNE (Hinton and Roweis, 2003), where the similarity preservation is based on Kullback-Leibler divergence;

[2]Note, in the simulations, datasets are normalized for computation convenience.
[3]This dataset publicly available at https://scikit-learn.org
[4]This dataset is publicly available at https://scikit-learn.org
[5]This dataset is publicly available at https://archive.ics.uci.edu/ml/datasets/wine
[6]This dataset is publicly available at http://yann.lecun.com/exdb/mnist/

4. t-SNE (Maaten and Hinton, 2008), which is a SNE extension based on student's t-distribution;
5. NeRV (Venna et al., 2010), type 1 mixture of Kullback-Leibler divergences;
6. JSE (Lee et al., 2013), type 2 mixture of Kullback-Leibler divergences;
7. KEDR (Álvarez-Meza et al., 2017), which is a recently proposed entropy dimensionality reduction method based on kernels.

4.2.3. Parameter settings

Table **4-1** summarizes the set-up of compared methods. In particular, as suggested in (Lee et al., 2013), the perplexity value K is fixed to $N/20$. The Renyi's entropy α and trade-off γ parameters were selected using the strategy proposed in (Álvarez-Meza et al., 2017). In addition, the kernel bandwidth σ was adjusted using the strategy proposed in (Liu et al., 2011). Note, the performance of our proposal is sensitive to the selection of η and T. However, values for these parameters can be selected as follows: *i)* η–step size, where a value of 0,01 has shown stable performance on all tested datasets; *ii)* T–number of iterations, based on our experimentation, an appropriate value is in the interval $[1000, 2000]$.

Table 4-1.: Parameter setting of compared methods. K–perplexity value, λ–number of neighbors, α–Renyi's entropy, γ–trade-off parameter, σ–kernel bandwidth, η–step size, T–number of iterations.

Dataset	**Parameter**	**Method**							
		PCA	**Isomap**	**SNE**	***t*-SNE**	**NeRV**	**JSE**	**KEDR**	**Proposal**
	K	-	-	25	25	25	25	-	-
	λ	-	25	-	-	-	-	-	-
	α	-	-	-	-	-	-	2	-
Swiss-Roll	γ	-	-	-	-	-	-	0	-
	σ	-	-	-	-	-	-	-	0.04
	η	-	-	-	-	-	-	-	0.01
	T	-	-	-	-	-	-	-	1500
	K	-	-	25	25	25	25	-	-
	λ	-	25	-	-	-	-	-	-
	α	-	-	-	-	-	-	2	-
S-Curve	γ	-	-	-	-	-	-	0	-
	σ	-	-	-	-	-	-	-	2
	η	-	-	-	-	-	-	-	0.01
	T	-	-	-	-	-	-	-	1500
	K	-	-	9	9	9	9	-	-
	λ	-	9	-	-	-	-	-	-
	α	-	-	-	-	-	-	2	-
Wine	γ	-	-	-	-	-	-	0	-
	σ	-	-	-	-	-	-	-	0.06
	η	-	-	-	-	-	-	-	0.01
	T	-	-	-	-	-	-	-	1500
	K	-	-	300	300	300	300	-	-
	λ	-	300	-	-	-	-	-	-
	α	-	-	-	-	-	-	2	-
MNIST	γ	-	-	-	-	-	-	0	-
	σ	-	-	-	-	-	-	-	0.006
	η	-	-	-	-	-	-	-	0.01
	T	-	-	-	-	-	-	-	1500

4.3. Results

In practice, the high-dimensional input data $\boldsymbol{U}$ is provided and for a desired value m, an initial m-dimensional training data set, noted as $\boldsymbol{V}^0$, is performed by generating a random set with centered Gaussian distribution, having a small variance as suggested in (Maaten and Hinton, 2008). After that, the kernel matrices $\boldsymbol{P}$ and $\boldsymbol{Q}^0$ are computed, for which the optimization of $\boldsymbol{V}$ is performed.

4.3.1. Dimensionality reduction on synthetic datasets

Figure **4-1** shows the two-dimensional embeddings of the *Swiss roll* dataset. On the one hand, our proposal and Isomap tend to preserve the "spiral shape"(see Figures 4.1(a) and 4.1(h)). As seen, both methods aim to conserve global neighborhoods. On the other hand, all other methods aim to cut the *Swiss roll* and unfold it (see Figures 4.1(b) to 4.1(g)), meaning that they favor the preservation of local neighborhoods. Thus, unlike methods based on Euclidean distances, the approximations are computed along the manifold. Additionally, t- SNE highlights the local structures due to its Student's t-distribution. Figure **4-2** shows the two-dimensional embeddings of the *S-curve* dataset. Once again, the methods follow two very different strategies to embed the high-dimensional manifold. That is, global (see Figures 4.2(a) and 4.2(h)) and local (see Figures 4.2(b) to 4.2(g)) neighborhoods.

Figure **4-3** shows the rank-based qualities of the *Swiss-Roll* and *S-Curve* datasets. Note, all curves are shown in a diagram with a logarithmic scale for the abscissa. This gives to each neighborhood size the most appropriate weight, which is important for visual representation, but also allows to each curve to be summarized into a scalar value (Lee and Verleysen, 2008). In addition, each area under the curve is reported in the diagram legend. This number reflects the overall quality of the embedding at all scales, i.e., for all K-ary neighborhood sizes. In particular: (1) both JSE and t-SNE score very high for the preservation of small neighborhoods. In turn, PCA and our proposal show a different shape, i.e., these methods are totally unable to preserve local structures; (2) our proposal achieves the highest scores for large neighborhoods, which supports the results found in Figures 4.1(h) and 4.2(h).

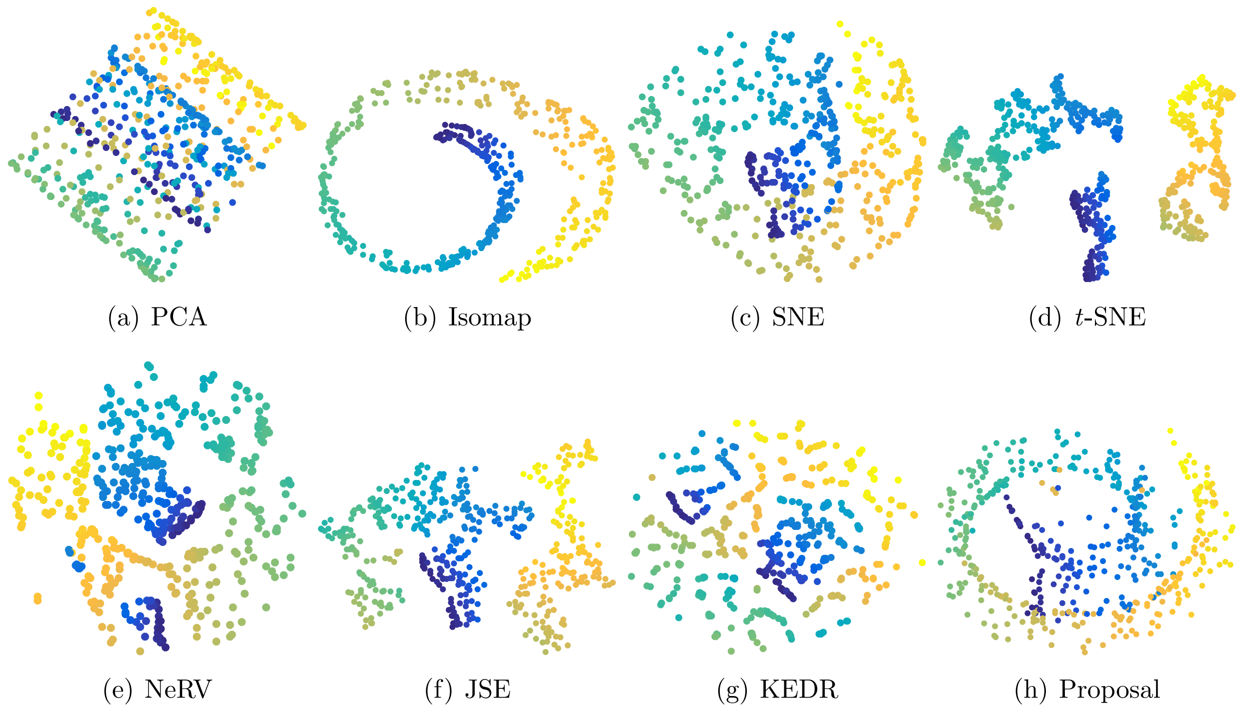


Figure 4-1.: Embeddings of *Swiss-Roll* dataset.

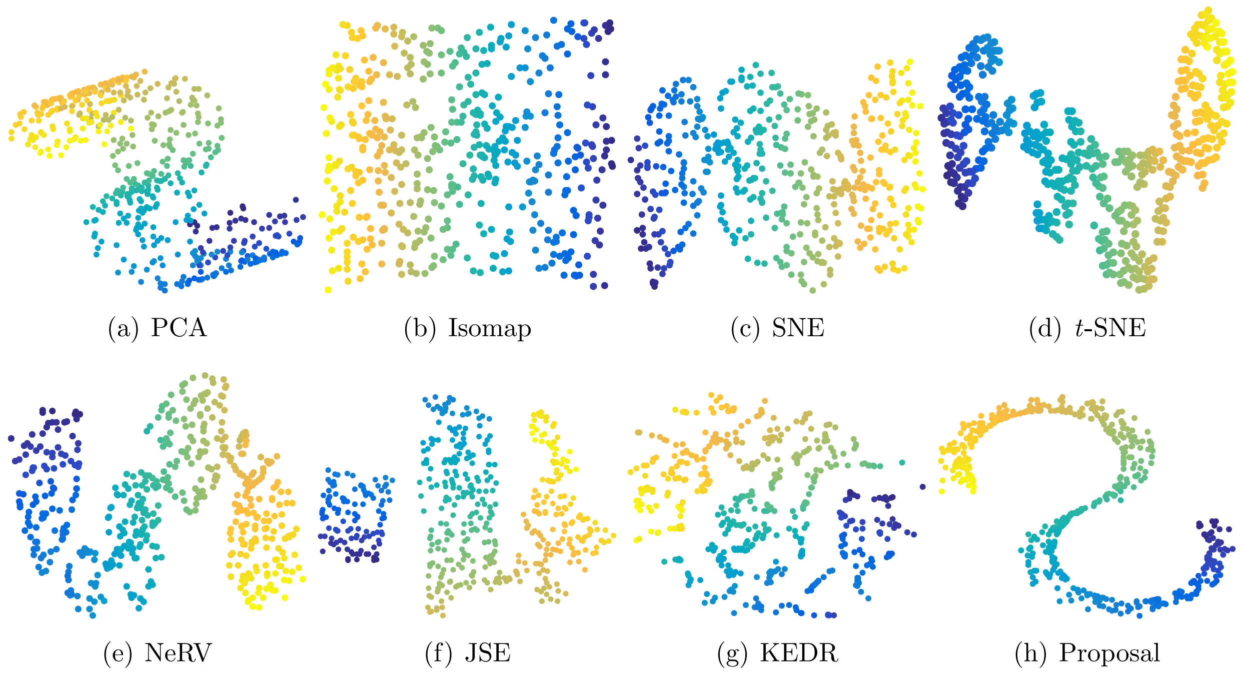
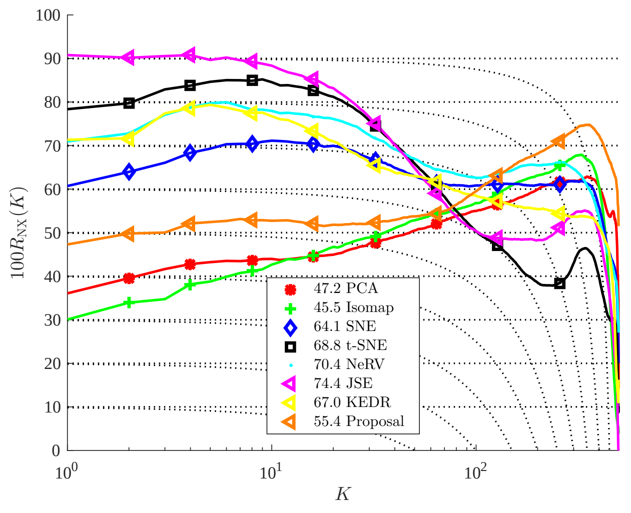
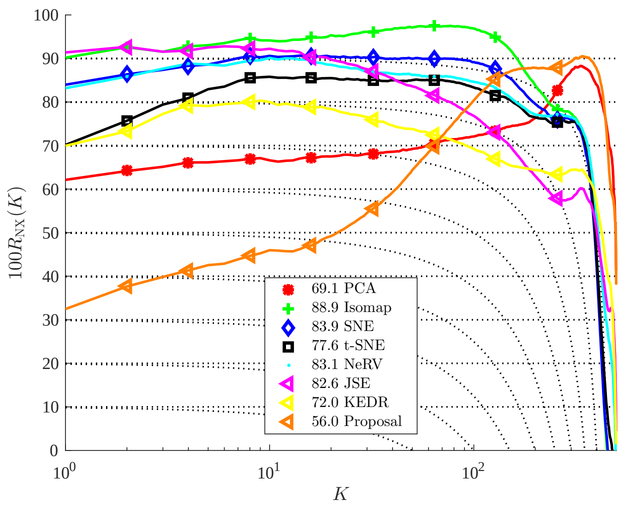


Figure 4-2.: Embeddings of *S-Curve* dataset.



(a) Swiss-Roll



(b) S-Curve

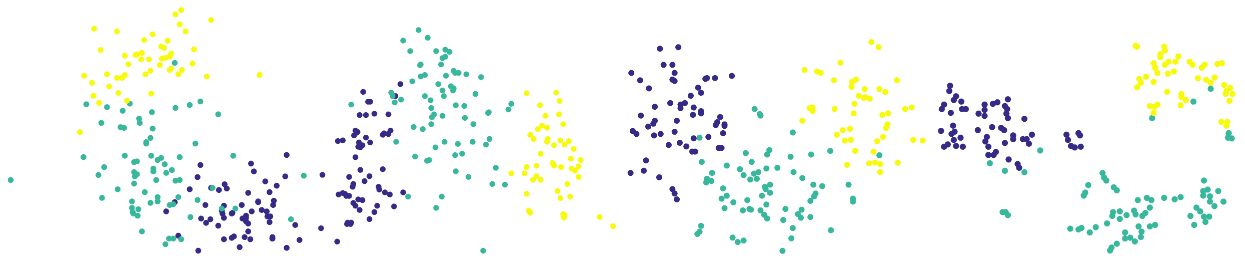
Figure 4-3.: Quality assessments in synthetic datasets.

4.3.2. Dimensionality reduction on real-world datasets

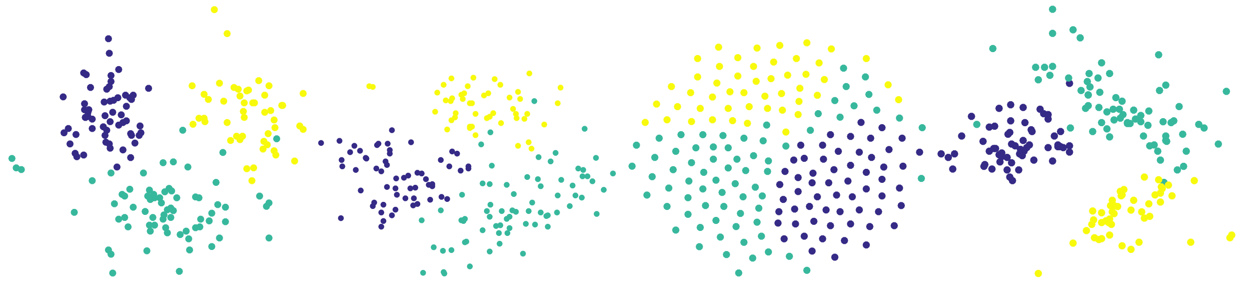
Figure **4-4** shows the two-dimensional embeddings of the *Wine* dataset, where each color indicates a different cultivator. Note that the labels are not used to determine the spatial coordinates of the embeddings. These labels just provide a way to evaluate the preservation of similarities within each class.

A quick glance shows that the methods follow the same strategy to embed the data (see Figure **4-4**). That is, they tend to create a cluster per each cultivator. However, some of these clusters have samples of their neighbors, which is a situation that occurs in all contrasted methods. This may be due to: (1) the definition of pairwise relationships in both the high-dimensional and low-dimensional spaces; (2) the two-dimensional embeddings reveal a hidden dynamic, which is not observable in the input space. Further, Figure **4-5** shows the two-dimensional embeddings of the *MNIST* dataset. Again, we use these labels just for viewing purposes. Four methods (Figures 4.5(c) to 4.5(f)) try to create clusters in the embeddings, meaning that they tend to find similarities in local neighborhoods. In contrast, it is not clear whether Isomap is trying to keep local or global structures (see Figure 4.5(b)).

Finally, Figure **4-6** shows the rank-based qualities of the real-world datasets. The t-SNE method shows the best performance in small neighborhoods. In addition, as in Section 4.3.1, our proposal achieves the highest scores for the global structures. The superiority of t-SNE, for small neighborhoods, is evident. However, this method needs many hyper-parameters and "tricks" (Maaten and Hinton, 2008). For example, besides the step-size η, the t-SNE method requires: (1) perplexity value; (2) initial momentum; (3) final momentum; (4) in the early stages of the optimization, Gaussian noise should be added to the samples at each iteration; (5) a variance value per each sample. In contrast to t-SNE, an advantage of our proposal is its simplicity. This makes it easy to implement in different data sets. That is, besides the step-size, the only hyper-parameter is the kernel bandwidth σ.

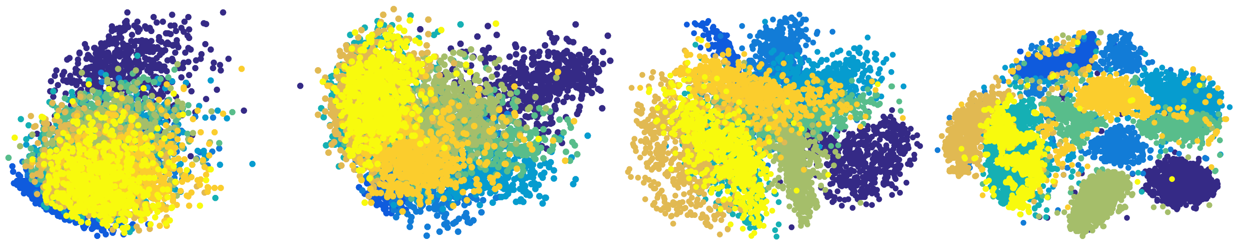


(a) PCA (b) Isomap (c) SNE (d) *t*-SNE

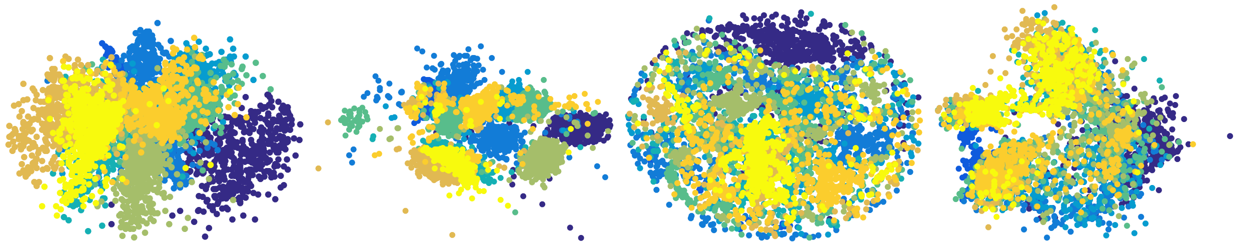


(e) NeRV (f) JSE (g) KEDR (h) Proposal

Figure 4-4.: Embeddings of *Wine* dataset.

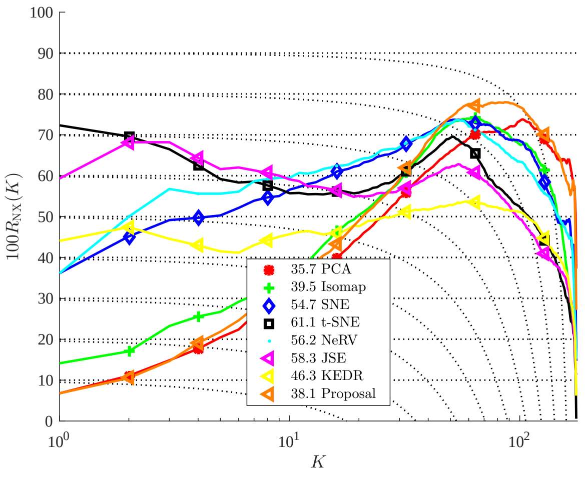


(a) PCA (b) Isomap (c) SNE (d) *t*-SNE

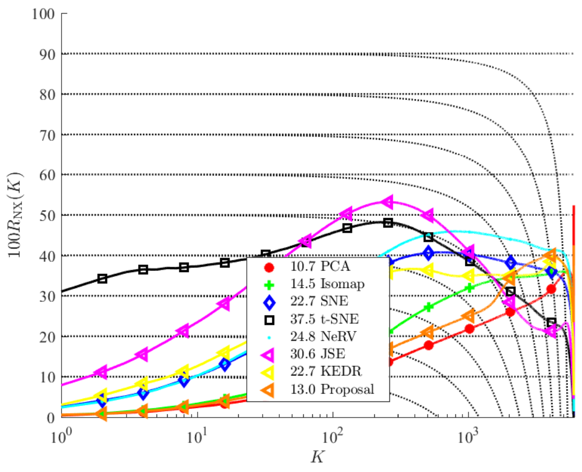


(e) NeRV (f) JSE (g) KEDR (h) Proposal

Figure 4-5.: Embeddings of *MNIST* dataset.



(a) *Wine*



(b) *MNIST*

Figure 4-6.: Quality assessments in real-world datasets.

4.4. Summary

In this study, we have introduced and evaluated a kernel-based dimensionality reduction method. To test its performance, we used both synthetic and real-world data sets. Simulation results demonstrated that: (1) our proposal preserves large neighborhoods; (2) the small number of hyper-parameters, in our proposal, makes it easy to implement in different datasets. This work addressed two challenges of dimensionality reduction methods, i.e., how to measure the manifold structures and embedding preservation. Consequently, we used the Gaussian kernel to: (1) measure the manifold structures in both the high-dimensional and low-dimensional spaces; (2) propose a kernel-based cost function, which is minimized using a gradient descent method.

5. Proposed framework for time series prediction using dimensionality reduction

Kernel-based adaptive filters are sequential learning algorithms, operating in reproducing kernel Hilbert spaces. Their learning performance is susceptible to the selection of appropriate values for kernel bandwidth and step-size parameters. Additionally, as these algorithms train the model using a sequence of input vectors, their computation scales with the number of samples. In this chapter[1], we propose a framework that addresses the previous open challenges of kernel-based adaptive filters. In contrast to similar methods, our proposal sequentially optimizes the bandwidth and step-size parameters using stochastic gradient algorithms that maximize the correntropy function. To remove redundant samples, a sparsification approach based on dimensionality reduction is introduced. The framework is validated on both synthetic and real-world datasets. Results show that our proposal converges to relatively low values of mean-square-error while provides stable solutions in real-world applications.

The remainder of the chapter has the following structure: Section 5.1 introduces the proposed framework; Sections 5.2 and 5.3 describe the simulation results; Section 5.4 presents a brief summary of the chapter.

[1]The outcomes of this chapter were published in *IEEE International Conference on Acoustics, Speech, and Signal Processing – ICASSP 2019* (Garcia-Vega et al., 2019).

5.1. Implementation of kernel-based adaptive filters

The goal is to learn a continuous input-output mapping $f{:}\mathbb{U} \to \mathbb{R}$ based on a paired sequence of input-output examples $\{\boldsymbol{u}_1, y_1\}, \ldots, \{\boldsymbol{u}_t, y_t\}$, where $\boldsymbol{u}_t$ is an m-dimensional input vector that belongs to the input set $\mathbb{U}{\subset}\mathbb{R}^m$, and $y_t{\in}\mathbb{R}$ is the output time series over the time domain $t{\in}N$. Because its ability to model non-linear systems, the input-output mapping function f can be learned using a kernel-based adaptive filter, yielding the following sequential rule through the time domain (As detailed in Section 2.3):

$$f_t = \begin{cases} f_{t-1} + \eta e_t \kappa_\sigma(\boldsymbol{u}_t, \cdot), & \forall t \neq 0 \\ 0, & t = 0 \end{cases} \tag{5-1a}$$

$$e_t = y_t - f_{t-1}(\boldsymbol{u}_t) \tag{5-1b}$$

where $\eta{\in}\mathbb{R}^+$ is the learning-rate, $\kappa_\sigma(\cdot,\cdot){\in}\mathbb{R}^+$ is a Mercer kernel with a bandwidth $\sigma{\in}\mathbb{R}^+$ that controls the mapping smoothness. We propose to optimize both η and σ by minimizing the prediction error $e_t{\in}\mathbb{R}$, using the following stages of adaptive filter implementation.

5.1.1. Kernel bandwidth optimization using correntropy

Based on nonlinear similarity measures, the adaptive filter parameters are proposed to be optimized using the correntropy cost function expressed over time as follows (Wang et al., 2017b):

$$J_t = \underset{\forall \sigma, \eta}{\arg\,\text{máx}} \{\exp\left(-e_t^2(\sigma_t, \eta_t)/2\lambda^2\right)\} \tag{5-2}$$

where $\lambda{\in}\mathbb{R}^+$ is the correntropy bandwidth that rules similarity between data points. Correntropy generalizes the conventional correlation function to nonlinear spaces, which has proven useful in many areas such as regression (Liu et al., 2007), adaptive filtering (Zhao et al., 2011), classification (Singh and Principe, 2010), and spectral characterization (Garde et al., 2010). The primary rationale behind the suggested strategy in Equation (5-2) is to extract more information from the data structure for the adaptation process, yielding solutions that are more accurate for non-Gaussian processes (Liu et al., 2006). In the first optimizing value, we perform the Kernel bandwidth estimation in Equation (5-2) using the gradient descent method, yielding the learning rule given as:

$$\sigma_t = \sigma_{t-1} + \beta \partial J_t / \partial \sigma_{t-1} \tag{5-3}$$

where σ_t is the bandwidth at iteration t and $\beta\in\mathbb{R}^+$ is the step-size parameter. Thus, using Equations (5-1a), (5-2) and (5-3), the kernel bandwidth estimation results in the following rule (see Appendix A.2):

$$\sigma_t = \sigma_{t-1} + \alpha\eta e_t e_{t-1}||\boldsymbol{u}_t - \boldsymbol{u}_{t-1}||^2 \kappa_{\sigma_{t-1}}(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}) \tag{5-4}$$

where $\alpha{=}J_t\beta/\lambda^2\sigma^3_{t-1}$, and notation $||\cdot||$ stands for ℓ_2 norm.

5.1.2. Learning-rate estimation based on correntropy

Likewise in Equation (5-3), the gradient-descent estimation yields the following learning-rate update at iteration t:

$$\eta_t = \eta_{t-1} + \beta\partial J_t/\partial\eta_{t-1} \tag{5-5}$$

where $\beta\in\mathbb{R}^+$ is the step-size parameter. Then, considering Equations (5-1a), (5-2) and (5-5), the learning-rate update results as below (see Appendix A.2):

$$\eta_t = \eta_{t-1} + \beta'' e_t e_{t-1}\kappa_\sigma(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}) \tag{5-6}$$

being $\beta''{=}\beta\exp(-e_t^2/2\lambda^2)$.

5.1.3. Dimensionality reduction through a sparsification strategy

In dimensionality reduction, a low-dimensional representation, $\boldsymbol{V}{=}\{\boldsymbol{v}_i{\in}\mathbb{R}^m{:}i{\in}[1,t{-}1]\}$, must be obtained from a provided high-dimensional finite set $\boldsymbol{U}{=}\{\boldsymbol{u}_i{\in}\mathbb{R}^M{:}i{\in}[1,t-1]\}$ that holds M features extracted at $t-1$ samples, under the dimensionality restriction $m{<}M$. To this end, given a training pair $\{\boldsymbol{u}_t, y_t\}$ fed at the kernel-based adaptive filter input, sparsification methods can be employed to decide whether a new sample $\boldsymbol{u}_t$ should be added to a dictionary (that is, a reduced set of input samples) used to estimate nonlinear models (Saide et al., 2015), decreasing the computational complexity.

For encoding all non-linear data relationships within spaces, therefore, a couple of kernel matrices are introduced: *i)* Input kernel, $\boldsymbol{P}{\in}\mathbb{R}^{t-1\times t-1}$ that holds elements $p_{ij}{=}\kappa_{\sigma_U}(\boldsymbol{u}_i,\boldsymbol{u}_j)$, $\kappa_{\sigma_U}{:}\mathbb{R}^M\times\mathbb{R}^M{\to}\mathbb{R}^+$; *ii)* Output kernel, $\boldsymbol{Q}{\in}\mathbb{R}^{t-1\times t-1}$ with elements $q_{ij}{=}\kappa_{\sigma_V}(\boldsymbol{v}_i,\boldsymbol{v}_j)$, $\kappa_{\sigma_V}{:}\ \mathbb{R}^m\times\mathbb{R}^m{\to}\mathbb{R}^+$. Both real-valued kernels are assumed to be Gaussian due to their universal approximating capability, desirable smoothness, and numeric stability (Liu et al., 2011). So, the similarity measures of high and low dimensional spaces are respectively as below:

$$p_{ij} = \exp\left(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2/2\sigma_U^2\right) \tag{5-7a}$$
$$q_{ij} = \exp\left(-\|\boldsymbol{v}_i - \boldsymbol{v}_j\|^2/2\sigma_V^2\right) \tag{5-7b}$$

where $\sigma_U,\sigma_V{\in}\mathbb{R}^+$ are the corresponding kernel sizes. Therefore, the kernel-based framework is devised so that the more correctly the points ($\boldsymbol{v}_i$ and $\boldsymbol{v}_j$) explain the similarity between the high-dimensional data points ($\boldsymbol{u}_i$ and $\boldsymbol{u}_j$), the more alike the kernel values p_{ij} and q_{ij} become.

Thus, the principal rationale behind the suggested similarity framework in Equations (5-7a) and (5-7b) is to find a low-dimensional data representation $\boldsymbol{V}$ so that the mismatch between p_{ij} and q_{ij} can be minimized. So, the following cost function is proposed:

$$C = \frac{1}{(t-1)-1}\mathbb{E}\left\{|p_{ij} - q_{ij}|/p_{ij} : \forall i, j \in t-1, j \neq i\right\}, C \in \mathbb{R} \tag{5-8}$$

where notation $\mathbb{E}\{\cdot\}$ denotes the expectation operator. In particular, we suggest to perform the cost function minimization using a gradient descent method, yielding the learning rule described as below:

$$\boldsymbol{v}_i^k = \boldsymbol{v}_i^{k-1} - \mu \partial C / \partial \boldsymbol{v}_i^{k-1} \tag{5-9}$$

where $\boldsymbol{v}_i^{k-1}$ is the low-dimensional representation of $\boldsymbol{u}_i$ at iteration $k-1$ and $\mu \in \mathbb{R}^+$ is the step-size parameter. Relying on Equations (5-7a), (5-7b) and (5-8), the gradient update results in the following rule (for more details see Chapter 4):

$$\boldsymbol{v}_i^k = \boldsymbol{v}_i^{k\text{-}1} - \mu' \mathbb{E}\left\{\frac{\left(\boldsymbol{v}_i^{k\text{-}1} - \boldsymbol{v}_j^{k\text{-}1}\right)\left(p_{ij}q_{ij}^{k\text{-}1} - (q_{ij}^{k\text{-}1})^2\right)}{p_{ij}|p_{ij} - q_{ij}^{k\text{-}1}|} : \forall j \in t\text{-}1, j \neq i\right\}$$

where $\mu' = \mu/(\sigma_V^2((t-1)-1))$. Consequently, introducing the quantization-size $\varepsilon \in \mathbb{R}^+$ (Chen et al., 2012b), the following *sparsification* strategy is proposed:

i) $\min\limits_{1 \le i \le t-1} \|\boldsymbol{v}_t - \boldsymbol{v}_i\| \le \varepsilon$: Update the closest sample weight to $\boldsymbol{u}_t$.

ii) $\min\limits_{1 \le i \le t-1} \|\boldsymbol{v}_t - \boldsymbol{v}_i\| > \varepsilon$: Add the input sample $\boldsymbol{u}_t$ to the dictionary.

In terms of embedding preservation, the previous imposed restraints aim to select only the input data that encodes the global structures extracted from training samples. Thus, the main rationale behind the sparse dictionary building is to hold, as much as possible, those samples, which are more likely to appear.

5.2. Experimental Design

We validate the proposed kernel-based adaptive framework in the case of prediction tasks, using the mean-square-error (MSE) as a measure of performance. At each iteration of the training set, therefore, the learned filter is used to compute the MSE value on each test set as carried out in (Liu et al., 2011).

5.2.1. Datasets

Testing is carried out on the following two benchmarking datasets used in prediction tasks:

– ***Mackey-Glass chaotic time-series***. Prediction performance is validated on a short-term signal set, which is generated by a chaotic system whose states are governed by a set of time-delayed differential equations. The task is to predict the current value using the previous ten consecutive samples. As experimented in (Liu et al., 2011), the data are normalized for the computation convenience, and for implementing the validation strategy, 500 samples are used as the training subset, while another 100 consecutive samples are the test subset.

– ***Wind Speed data***. This collection holds hourly wind speed records from the northern region of Colombia[2]. In this case, the performance is also evaluated in predicting the current value using the previous ten consecutive samples. The considered training set ranges from September-24-2008 to October-31-2008, and the test set ranges from May-28-2009 to June-02-2009.

5.2.2. Reference methods

For a comparison purpose, the proposed variable bandwidth, adaptive step-size, and *sparsification* strategy are contrasted with the following kernel-based adaptive filters:

1. Kernel least-mean-square (noted as KLMS) as the simplest kernel-based adaptive strategy (Liu et al., 2008);

2. Quantized kernel least-mean-square (QKLMS) that introduces an online vector quantization method into KLMS (Chen et al., 2012b);

3. Kernel least-mean-square with variable kernel bandwidth (KLMS-VKS) described in (Chen et al., 2016a);

4. Kernel-least-mean-square tested with a variable learning rate (KLMS-VSS) discussed in (Niu and Chen, 2018).

[2]The dataset is publicly available at http://www.ideam.gov.co/solicitud-de-informacion

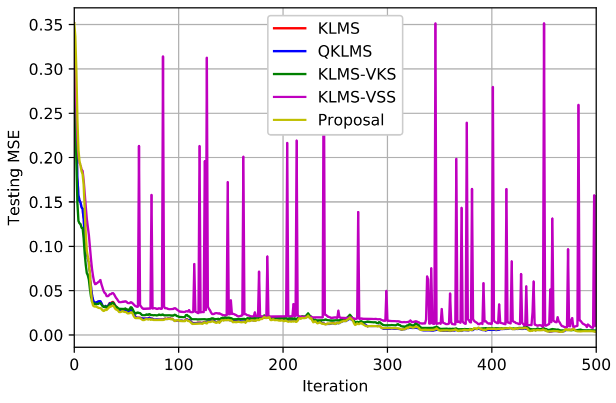
5.2.3. Parameter settings

The set-up of compared adaptive filters is as follows: *i*) the step-size is adjusted at η=0,2 for KLMS and QKLMS, while the initial learning-rate η_1 is set at 0 for KLMS-VKS, KLMS-VSS, and our proposal; *ii*) the kernel bandwidth is set at $\sigma=\sqrt{1/2}$ for KLMS and QKLMS, which is also the initial bandwidth in our proposal, KLMS-VKS, and KLMS-VSS i.e., $\sigma_1=\sqrt{1/2}$; *iii*) the quantization size ε is set at 0,05 and 0,1 for QKLMS and our proposal, respectively; *iv*) the learning-rate β is set at 0,1; *v)* the correntropy bandwidth is set at λ=1; *vi*) dimensionality reduction method, k=1000, m=2, μ=0,1 and σ_U, σ_V=0,2. All used kernel parameters had been adjusted heuristically.

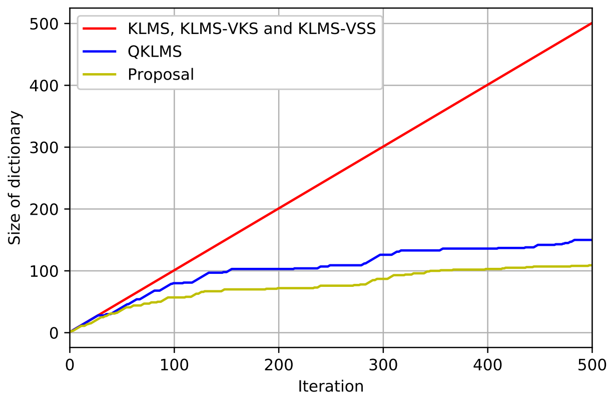
5.3. Results

5.3.1. Mackey-Glass time-series

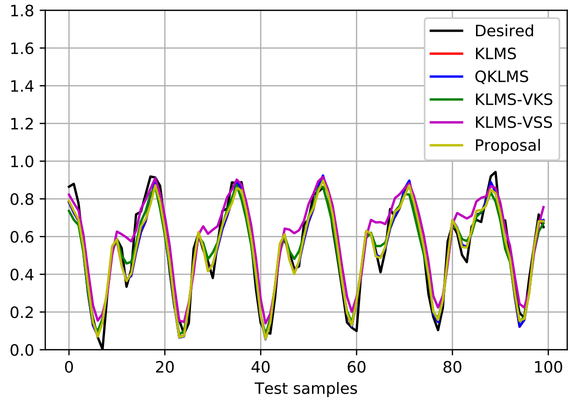
Figure 5.1(a) displays the learning curves, plotting the mean-square-error results performed by each compared solution versus the number of iterations. A quick glance shows that KLMS-VSS performs the worst MSE values, having abrupt changes during training, which may suggest that the algorithm is easily trapped on local minimums. As seen, KLMS and KLMS-VKS methods show a relatively good performance since they achieve more stable MSE values through iterations. However, the evolution curves of network size in Figure 5.1(b) make clear that their dictionary sizes linearly grows during training. This issue may be explained since both algorithms do not incorporate any sparsification technique, resulting in a significant drawback for implementation in online applications. By contrast, the number of samples of QKLMS algorithm grows very slowly, resulting in a final network that sizes only 150. Even that QKLMS and our proposal achieve similar MSE values, the former method demands a dictionary size significantly higher as seen in Figure 5.1(b), and therefore, increasing the computational burden of online applications. As seen in Figure 5.1(c), the proposed framework achieves a competitive performance, reaching the lowest network size through iterations and suggesting that its *sparsification* strategy (based on dimensionality reduction) helps to hold the most relevant samples to perform prediction tasks.
As regards the kernel bandwidth and learning-rate influence on the performed prediction, Table **5-1** displays the MSE evolution over the test set, showing that the proposed framework achieves the lowest MSE at iteration 100. Thus, there is an improvement in convergence time while competitive performance is maintained in future iterations, proving that our proposal converges to relatively low values of MSE, avoids overfitting, and provide stable solutions in real-world applications.



(a) Learning curves



(b) Dictionary size



(c) Test set prediction

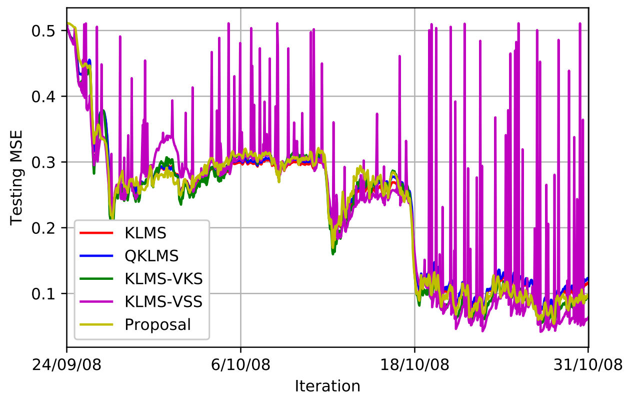
Figure 5-1.: Performed results by each compared adaptive filter on Mackey-Glass.

Table 5-1.: Performed results on Mackey-Glass time-series prediction at different iterations. The best overall method of each column are marked with bold notation. *MSE*-mean square error. *DS*-Dictionary Size.

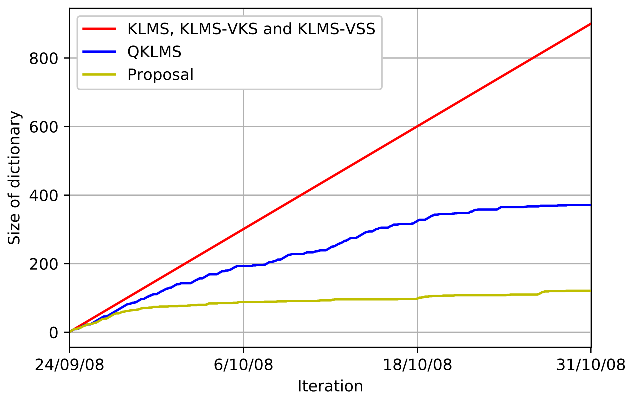
Method	**Measure**	**Iteration**				
		100	*200*	*300*	*400*	*500*
KLMS	MSE	0.016	0.017	0.007	0.006	0.004
	DS	100	200	300	400	500
QKLMS	MSE	0.017	0.017	0.007	0.006	0.004
	DS	80	103	126	136	150
KLMS-VKS	MSE	0.021	0.019	0.011	0.008	0.005
	DS	100	200	300	400	500
KLMS-VSS	MSE	0.028	0.021	0.016	0.013	0.011
	DS	100	200	300	400	500
Proposal	MSE	0.016	0.007	0.007	0.006	0.004
	DS	**57**	**71**	**86**	**100**	**104**

5.3.2. Wind Speed

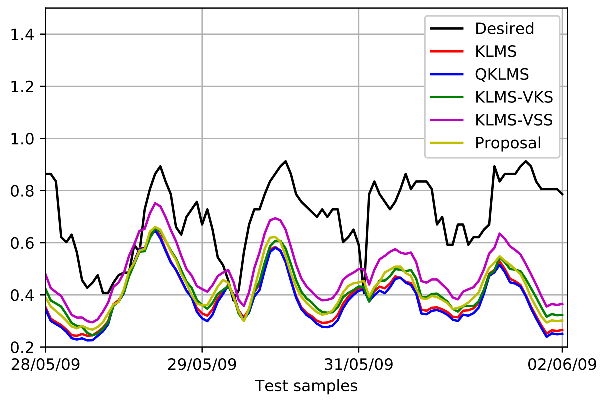
Figure 5.2(a) shows the learning curves estimated for the test set. The contrasted algorithms provide a robust performance through iterations. Although KLMS-VSS produces the poorest MSE values as in the previous dataset, the displayed MSE evolution shows that its performance becomes even worse because of the increased complexity of real-world data. By contrast, the other contrasted methods provide a more robust performance through iterations. It is worth noting that the testing MSE decreases slower in all methods when compared with the learning curves of synthetic results (see Figure 5.1(a)), clearly pointing out on the presence of highly non-stationary dynamics. This situation makes the kernel-based adaptive filters demand more time to encode the most relevant samples of this time-series correctly. The variable bandwidth and step-size, incorporated by our framework, promote the kernel-based adaptive filter to converge faster without significant loss of accuracy. As seen in Figure 5.2(b), the evolution curves make clear also that our proposal reaches the lowest dictionary size during training while maintains a competitive MSE performance (see Figure 5.2(c)). However, if their initial values are inappropriately chosen at the beginning, the converging speed can be very slow. In this case, the suitable initial values of bandwidth and step-size can be selected using one of the methods developed on this account like the Silverman's rule of thumb.
Furthermore, the results presented in Table **5-2** suggest that the proposed framework is an adequate alternative to increase the convergence speed while maintains a high accuracy with the benefit of demanding a condensed dictionary size, and therefore, improving the performance of on-line prediction tasks.



(a) Learning curves



(b) Dictionary size



(c) Test set prediction

Figure 5-2.: Performed results by each compared adaptive filter on wind speed prediction.

Table 5-2.: Performed results on Wind Speed at different iterations. The bold notation indicates the best overall method. *MSE*-mean square error. *DS*-Dictionary Size.

Method	Measure	Iteration				
		28/09/08	06/10/08	14/10/08	23/10/08	31/10/08
KLMS	MSE	0.253	0.299	0.249	0.084	0.115
	DS	100	300	500	700	900
QKLMS	MSE	0.252	0.302	0.255	0.087	0.122
	DS	81	193	280	357	371
KLMS-VKS	MSE	0.241	0.311	0.253	0.066	0.094
	DS	100	300	500	700	900
KLMS-VSS	MSE	0.307	0.481	0.263	0.056	0.063
	DS	100	300	500	700	900
Proposal	MSE	0.262	0.311	0.272	0.074	0.095
	DS	**62**	**88**	**96**	**108**	**121**

5.4. Summary

In this study, a framework for kernel-based adaptive filters is introduced that addresses three main challenges of their online implementation: selection of appropriate kernel bandwidth, step-size parameter, and training samples. In particular, the first two stages are optimized based on nonlinear similarity cost function expressed over time. To reduce the dictionary size, we also include a dimensionality reduction method that incorporates a *sparsification* strategy, employing a kernel-based cost function that quantifies the global structures of training samples. Validation on both datasets, synthetic and real-world, proves that the proposed framework converges to relatively low values of mean-square-error, avoiding while provides stable solutions in real-world applications.

6. Conclusions and Future Work

6.1. Concluding Remarks

– The proposed framework determines the low-dimensional data representation by reducing the mismatch between the non-linear data relationships within the spaces, meaning that the optimization procedure considers the whole training set to find the most suitable embeddings. Thus, under an out-of-sample scenario, the optimization has to start from scratch as the similarity measures may vary slightly.

– It is worth noting that the introduced dimensionality reduction method comprises an $\mathcal{O}(N^3)$ computational complexity, resulting in an expensive execution time when compared to conventional SNE-based algorithms.

– The proposed framework sequentially updates the bandwidth and step-size parameters using a stochastic gradient algorithm that maximizes the correntropy function. Thus, the estimation error decreases along iterations, which means an improvement in convergence time while maintaining the robustness and simplicity of kernel-based adaptive filters.

– As the correntropy function is inherently insensitive to outliers, the proposed adaptive bandwidth and step-size provide an effective mechanism to eliminate the detrimental effect of outliers, and they are intrinsically different from the use of a threshold in conventional techniques.

– The proposed *sparsification* strategy is trained with the samples that are most likely to appear during the prediction task, starting with an empty dictionary and gradually adding new samples. As a result, the prediction task is performed by extracting the most relevant input data – concerning the embedding preservation – while maintaining a competitive performance. However, we must clarify that our *sparsification* strategy may be adversely affected with few training samples, due to it is more difficult to identify global structures under this scenario.

6.2. Future Work

We are in the process of expanding our research in the following areas:

- More elaborate optimization algorithms or GPU-based implementations must be considered to reduce the computational burden in the proposed dimensionality reduction method;
- Extend the results to the case where a more elaborate hyper-parameter tuning procedure is introduced into the compared kernel-based adaptive filters;
- Considering additional Mercer kernels, i.e., not restricted to the Gaussian kernel;
- Evaluating the discriminative ability of low-dimensional representations in classification and regression tasks;
- Consider additional information theoretic measures;
- Integrate the proposed methods with additional classification and regression tasks.

6.3. Academic Discussion

- S. Garcia-Vega, and G. Castellanos-Dominguez. “Similarity preservation in dimensionality reduction using a kernel-based cost function”. In: *Pattern Recognition Letters*. Volume: 125. Pages 318-324. DOI: ttps://doi.org/10.1016/j.patrec.2019.05.009.

- S. Garcia-Vega, E. A. Leon-Gomez, and G. Castellanos-Dominguez. “Time Series Prediction for Kernel-based Adaptive Filters using Variable Bandwidth, Adaptive Learning-rate, and Dimensionality Reduction”. In: *International Conference on Acoustics, Speech, and Signal Processing — ICASSP 2019*. Country: United Kingdom. Pages 3892–3896. Online ISBN: 978-1-4799-8131-1. DOI: https://doi.org/10.1109/ICASSP.2019.8683117.

- P. A. Muñoz-Gutierrez, J. D. Martinez-Vargas, S. Garcia-Vega, E. Giraldo, and G. Castellanos-Domínguez. “EEG Based Brain Mapping by using Frequency Spatio Temporal Constraints”. In: *International Conference on Brain Informatics – BI 2018*. Country: United States. Pages: 13–21. Electronic ISBN: 978-3-030-05587-5. DOI: https://doi.org/10.1007/978-3-030-05587-5_2.

- S. Garcia-Vega, G. Castellanos-Dominguez, M. Verleysen, and J. A. Lee. “Multi-step-ahead forecasting using kernel adaptive filtering”. In: *International Joint Conference on Neural Networks – IJCNN 2016*. Country: Canada. Pages: 2132-2139. Electronic ISBN: 978-1-5090-0620-5. Electronic ISSN: 2161-4407. DOI: https://doi.org/10.1109/IJCNN.2016.7727463.

A. Appendix

A.1. Gradient descent based optimization of cost function

The minimization of cost function C is performed using a gradient descent method as follows:

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} - \eta \frac{\partial}{\partial \boldsymbol{v}_i^{t-1}} \left[C \right] \tag{A-1}$$

where $\boldsymbol{v}_i^{t-1}$ is the low-dimensional representation of $\boldsymbol{u}_i$ at iteration t–1 and $\eta \in \mathbb{R}^+$ is the step-size parameter,

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} - \eta \frac{\partial}{\partial \boldsymbol{v}_i^{t-1}} \left[\frac{1}{N\left(N-1\right)} \sum_{j \neq i} \frac{\left| p_{ij} - q_{ij}^{t-1} \right|}{p_{ij}} \right]$$

the absolute value $|\cdot|$ can be defined as $|\cdot| = \sqrt{(\cdot)^2}$,

$$\begin{aligned}
\boldsymbol{v}_i^t &= \boldsymbol{v}_i^{t-1} - \eta \frac{\partial}{\partial \boldsymbol{v}_i^{t-1}} \left[\frac{1}{N\left(N-1\right)} \sum_{j \neq i} \frac{\sqrt{\left(p_{ij} - q_{ij}^{t-1}\right)^2}}{p_{ij}} \right] \\
\boldsymbol{v}_i^t &= \boldsymbol{v}_i^{t-1} - \eta \frac{\partial}{\partial \boldsymbol{v}_i^{t-1}} \left[\frac{1}{N\left(N-1\right)} \sum_{j \neq i} \frac{\sqrt{\left(p_{ij}\right)^2 - 2p_{ij}q_{ij}^{t-1} + \left(q_{ij}^{t-1}\right)^2}}{p_{ij}} \right] \\
\boldsymbol{v}_i^t &= \boldsymbol{v}_i^{t-1} - \eta \left[\frac{1}{\sigma^2 N\left(N-1\right)} \sum_{j \neq i} \frac{\left(\boldsymbol{v}_i^{t-1} - \boldsymbol{v}_j^{t-1}\right) \left(p_{ij}q_{ij}^{t-1} - \exp\left(-\left\|\boldsymbol{v}_i^{t-1} - \boldsymbol{v}_j^{t-1}\right\|^2 / \sigma^2\right)\right)}{p_{ij}\sqrt{\left(p_{ij}\right)^2 - 2p_{ij}q_{ij}^{t-1} + \left(q_{ij}^{t-1}\right)^2}} \right]
\end{aligned}$$

multiplying the exponent of $\exp(\cdot)$ by $2/2$,

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} - \eta \left[\frac{1}{\sigma^2 N(N-1)} \sum_{j\neq i} \frac{\left(\boldsymbol{v}_i^{t-1} - \boldsymbol{v}_j^{t-1}\right)\left(p_{ij} q_{ij}^{t-1} - \exp\left(-\left\|\boldsymbol{v}_i^{t-1} - \boldsymbol{v}_j^{t-1}\right\|^2 / \sigma^2 \times 2/2\right)\right)}{p_{ij}\sqrt{(p_{ij})^2 - 2p_{ij}q_{ij}^{t-1} + \left(q_{ij}^{t-1}\right)^2}} \right]$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} - \eta \left[\frac{1}{\sigma^2 N(N-1)} \sum_{j\neq i} \frac{\left(\boldsymbol{v}_i^{t-1} - \boldsymbol{v}_j^{t-1}\right)\left(p_{ij} q_{ij}^{t-1} - \left(\exp\left(-\left\|\boldsymbol{v}_i^{t-1} - \boldsymbol{v}_j^{t-1}\right\|^2 / 2\sigma^2\right)\right)^2\right)}{p_{ij}\sqrt{(p_{ij})^2 - 2p_{ij}q_{ij}^{t-1} + \left(q_{ij}^{t-1}\right)^2}} \right]$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} - \eta \left[\frac{1}{\sigma^2 N(N-1)} \sum_{j\neq i} \frac{\left(\boldsymbol{v}_i^{t-1} - \boldsymbol{v}_j^{t-1}\right)\left(p_{ij} q_{ij}^{t-1} - \left(q_{ij}^{t-1}\right)^2\right)}{p_{ij}\sqrt{\left(p_{ij} - q_{ij}^{t-1}\right)^2}} \right]$$

Finally, the gradient update is given by

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} - \frac{\eta}{\sigma^2 N(N-1)} \sum_{j\neq i} \frac{\left(\boldsymbol{v}_i^{t-1} - \boldsymbol{v}_j^{t-1}\right)\left(p_{ij} q_{ij}^{t-1} - \left(q_{ij}^{t-1}\right)^2\right)}{p_{ij}\left|p_{ij} - q_{ij}^{t-1}\right|} \tag{A-2}$$

Remark. *Equation (A-2) aims to find the points $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ that minimizes the mismatch between p_{ij} and q_{ij}. Note, the p_{ij} similarity will not change during the optimization process (see Equation (4-1a)), but this is not the case for q_{ij}. Thus, the following scenarios may appear:*

1. *$0 < q_{ij} < p_{ij}$: The sum argument in Equation (A-2) will give values between 0 and 1, i.e., $0 < \left(p_{ij}q_{ij}^{t-1} - (q_{ij}^{t-1})^2\right)/\left(p_{ij}|p_{ij} - q_{ij}^{t-1}|\right) < 1$, which means a small change between $\boldsymbol{v}_i^{t-1}$ and $\boldsymbol{v}_i^t$.*

2. *$q_{ij} > p_{ij}$: Here, $\boldsymbol{v}_i$ is updated with relatively high values at each iteration, i.e., $\left(p_{ij}q_{ij}^{t-1} - (q_{ij}^{t-1})^2\right)/\left(p_{ij}|p_{ij} - q_{ij}^{t-1}|\right) < 0$.*

A.2. Kernel bandwidth and step-size optimization

Provided a step-size value $\beta \in \mathbb{R}^{+}$, the cost function J_t in Equation (5-2) is maximized in terms of either optimizing parameter $\zeta = \{\sigma, \eta\}$ through the gradient descent method as follows:

$$\zeta_t = \zeta_{t-1} + \beta \partial J_t / \partial \zeta_{t-1} \tag{A-3}$$

The learning rule Equation (A-3) can be unfolded as below:

$$\begin{aligned}
\zeta_t =& \zeta_{t-1} + \beta \frac{\partial}{\partial \zeta_{t-1}} \exp(-e_t^2/2\lambda^2)\\
\zeta_t =& \zeta_{t-1} + \beta \exp(-e_t^2/2\lambda^2) \frac{\partial}{\partial \zeta_{t-1}} (-e_t^2/2\lambda^2)\\
\zeta_t =& \zeta_{t-1} - (\beta/2\lambda^2) \exp(-e_t^2/2\lambda^2) \frac{\partial}{\partial \zeta_{t-1}} e_t^2\\
\zeta_t =& \zeta_{t-1} - (\beta/2\lambda^2) \exp(-e_t^2/2\lambda^2) \frac{\partial}{\partial \zeta_{t-1}} \Big(y_t - f_{t-1}(\boldsymbol{u}_t)\Big)^2\\
\zeta_t =& \zeta_{t-1} - \beta' \frac{\partial}{\partial \zeta_{t-1}} \big(y_t^2 - 2 y_t f_{t-1}(\boldsymbol{u}_t) + f_{t-1}^2(\boldsymbol{u}_t)\big)
\end{aligned}$$

where $\beta' = (\beta/2\lambda^2) \exp(-e_t^2/2\lambda^2)$

From Equation (2-11), it holds for the σ parameter that

$$f_{t-1}(\boldsymbol{u}_t) = f_{t-2}(\boldsymbol{u}_t) + \eta e_{t-1} \kappa_{\sigma_{t-1}}(\boldsymbol{u}_t, \boldsymbol{u}_{t-1})$$

then, the following expression takes place:

$$\sigma_t = \sigma_{t-1} - \beta' \left(-2 y_t \frac{\partial}{\partial \sigma_{t-1}} (\eta e_{t-1} \kappa_{\sigma_{t-1}}(\boldsymbol{u}_t, \boldsymbol{u}_{t-1})) + \frac{\partial}{\partial \sigma_{t-1}} \Big(\eta e_{t-1} \kappa_{\sigma_{t-1}}(\boldsymbol{u}_t, \boldsymbol{u}_{t-1})\Big)^2 \right)$$

Besides, we assumme the Mercer kernel $\kappa_{\sigma_{t-1}}$ be Gaussian kernel, that is,

$$\kappa_{\sigma_{t-1}} = \exp\left(-||\boldsymbol{u}_t - \boldsymbol{u}_{t-1}||^2 / 2\sigma_{t-1}^2\right),$$

so that we obtain:

$$\begin{aligned}
\sigma_t =& \sigma_{t-1} - \beta' \left(-2 y_t \eta e_{t-1} \kappa_{\sigma_{t-1}}(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}) \frac{1}{\sigma_{t-1}^3} ||\boldsymbol{u}_t - \boldsymbol{u}_{t-1}||^2 \right.\\
& \left. + 2 f_{t-1}(\boldsymbol{u}_t) \eta e_{t-1} \kappa_{\sigma_{t-1}}(\boldsymbol{u}_t, \boldsymbol{u}_{t-1}) \frac{1}{\sigma_{t-1}^3} ||\boldsymbol{u}_t - \boldsymbol{u}_{t-1}||^2 \right)
\end{aligned}$$

Lastly, the gradient update yields as follows:

$$\sigma_t = \sigma_{t-1} + \alpha\eta e_t e_{t-1} ||\boldsymbol{u}_t - \boldsymbol{u}_{t-1}||^2 \kappa_{\sigma_{t-1}} (\boldsymbol{u}_t, \boldsymbol{u}_{t-1}) \tag{A-4}$$

where $\alpha{=}J_t\beta/\lambda^2\sigma_{t-1}^3$, and notation $\|\cdot\|$ stands for ℓ_2 norm. In the case of η parameter, the sequential rule in Equation (2-11) is as follows:

$$f_{t-1}(\boldsymbol{u}_t) = f_{t-2}(\boldsymbol{u}_t) + \eta_{t-1} e_{t-1} \kappa_\sigma (\boldsymbol{u}_t, \boldsymbol{u}_{t-1})$$

Therefore, the gradient update of η yields as below:

$$\eta_t = \eta_{t-1} + \beta^{''} e_t e_{t-1} \kappa_\sigma (\boldsymbol{u}_t, \boldsymbol{u}_{t-1}) \tag{A-5}$$

where $\beta^{''}{=}\beta\exp(-e_t^2/2\lambda^2)$.

Bibliography

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Ali, J. B., Fnaiech, N., Saidi, L., Chebel-Morello, B., and Fnaiech, F. (2015). Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals. *Applied Acoustics*, 89:16–27.

Alvarez-Meza, A. M., Daza-Santacoloma, G., and Castellanos-Dominguez, G. (2012). Biomedical data analysis by supervised manifold learning. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 41–44. IEEE.

Álvarez-Meza, A. M., Lee, J. A., Verleysen, M., and Castellanos-Dominguez, G. (2017). Kernel-based dimensionality reduction using renyi's α-entropy measures of similarity. *Neurocomputing*, 222:36–46.

Álvarez-Meza, A. M., Valencia-Aguirre, J., Daza-Santacoloma, G., Acosta-Medina, C. D., and Castellanos-Domínguez, G. (2013). Video analysis based on multi-kernel representation with automatic parameter choice. *Neurocomputing*, 100:117–126.

An, S., Liu, W., and Venkatesh, S. (2007). Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition*, 40(8):2154–2162.

Ashiquzzaman, A., Tushar, A. K., Islam, M. R., Shon, D., Im, K., Park, J.-H., Lim, D.-S., and Kim, J. (2018). Reduction of overfitting in diabetes prediction using deep learning neural network. In *IT Convergence and Security 2017*, pages 35–43. Springer.

Bao, Y., Xiong, T., and Hu, Z. (2014). Multi-step-ahead time series prediction using multiple-output support vector regression. *Neurocomputing*, 129:482–493.

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591.

Ben Taieb, S., Hyndman, R. J., and Bontempi, G. (2014). Machine learning strategies for multi-step-ahead time series forecasting.

Bishop, C. M. (2006). Machine learning and pattern recognition. *Information Science and Statistics. Springer, Heidelberg.*

Bontempi, G. (2008). Long term time series prediction with multi-input multi-output local learning. *Proc. 2nd ESTSP*, pages 145–154.

Bontempi, G. and Taieb, S. B. (2011). Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International journal of forecasting*, 27(3):689–699.

Borg, I. and Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

Brock, W. A. (2018). Causality, chaos, explanation and prediction in economics and finance. In *Beyond Belief*, pages 230–279. CRC Press.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167.

Cardenas, D., Orbes-Arteaga, M., Castro-Ospina, A., Alvarez-Meza, A., and Castellanos-Dominguez, G. (2014). A kernel-based representation to support 3d mri unsupervised clustering. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 3203–3208. IEEE.

Cassisi, C., Montalto, P., Aliotta, M., Cannata, A., and Pulvirenti, A. (2012). Similarity measures and dimensionality reduction techniques for time series data mining. In *Advances in data mining knowledge discovery and applications*. InTech.

Chen, B., Liang, J., Zheng, N., and Príncipe, J. (2016a). Kernel least mean square with adaptive kernel size. *Neurocomputing*, 191:95–106.

Chen, B., Zhao, S., Seth, S., and Principe, J. C. (2012a). Online efficient learning with quantized klms and l 1 regularization. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–6. IEEE.

Chen, B., Zhao, S., Zhu, P., and Príncipe, J. C. (2012b). Quantized kernel least mean square algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 23(1):22–32.

Chen, K. and Yu, J. (2014). Short-term wind speed prediction using an unscented kalman filter based state-space support vector regression approach. *Applied Energy*, 113:690–705.

Chen, P.-A., Chang, L.-C., and Chang, F.-J. (2013). Reinforced recurrent neural networks for multi-step-ahead flood forecasts. *Journal of Hydrology*, 497:71–79.

Chen, Y.-L., Wu, X., Li, T., Cheng, J., Ou, Y., and Xu, M. (2016b). Dimensionality reduction of data sequences for human activity recognition. *Neurocomputing*, 210:294–302.

Chevillon, G. (2007). Direct multi-step estimation and forecasting. *Journal of Economic Surveys*, 21(4):746–785.

Chitsazan, M. A., Fadali, M. S., Nelson, A. K., and Trzynadlowski, A. M. (2017). Wind speed forecasting using an echo state network with nonlinear output functions. In *American Control Conference (ACC), 2017*, pages 5306–5311. IEEE.

Coussement, K., Lessmann, S., and Verstraeten, G. (2017). A comparative analysis of data preparation algorithms for customer churn prediction: A case study in the telecommunication industry. *Decision Support Systems*, 95:27–36.

Cunningham, P. and Delany, S. J. (2007). k-nearest neighbour classifiers. *Multiple Classifier Systems*, 34:1–17.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142.

Eldan, R. and Shamir, O. (2016). The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940.

Engel, Y., Mannor, S., and Meir, R. (2004). The kernel recursive least-squares algorithm. *IEEE Transactions on signal processing*, 52(8):2275–2285.

Erdem, E. and Shi, J. (2011). Arma based approaches for forecasting the tuple of wind speed and direction. *Applied Energy*, 88(4):1405–1414.

Evgeniou, T., Pontil, M., and Poggio, T. (2000). Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1.

Fischer, I. and Poland, J. (2005). Amplifying the block matrix structure for spectral clustering. In *Proceedings of the 14th annual machine learning conference of Belgium and the Netherlands*, pages 21–28. Citeseer.

Fuglede, B. and Topsoe, F. (2004). Jensen-shannon divergence and hilbert space embedding. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, page 31. IEEE.

Gallicchio, C. and Micheli, A. (2017). Deep echo state network (deepesn): A brief survey. *arXiv preprint arXiv:1712.04323*.

Gao, J., Shi, Q., and Caetano, T. S. (2012). Dimensionality reduction via compressive sensing. *Pattern Recognition Letters*, 33(9):1163–1170.

Garcia-Vega, S. and Castellanos-Dominguez, G. (2019). Similarity preservation in dimensionality reduction using a kernel-based cost function. *Pattern Recognition Letters*, 125:318–324.

Garcia-Vega, S., Castellanos-Dominguez, G., Verleysen, M., and Lee, J. A. (2016). Multi-step-ahead forecasting using kernel adaptive filtering. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 2132–2139. IEEE.

Garcia-Vega, S., León-Gómez, E., and Castellanos-Dominguez, G. (2019). Time series prediction for kernel-based adaptive filters using variable bandwidth, adaptive learning-rate, and dimensionality reduction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3892–3896. IEEE.

Garde, A., Sörnmo, L., Jané, R., and Giraldo, B. F. (2010). Correntropy-based spectral characterization of respiratory patterns in patients with chronic heart failure. *IEEE Transactions on Biomedical Engineering*, 57(8):1964–1972.

Giraldo-Suarez, E., Martínez-Vargas, J. D., and Castellanos-Dominguez, G. (2016). Reconstruction of neural activity from eeg data using dynamic spatiotemporal constraints. *International journal of neural systems*, 26(07):1650026.

Han, Y., Yang, K., and Zhou, G. (2013). A new anomaly: The cross-sectional profitability of technical analysis. *Journal of Financial and Quantitative Analysis*, 48(5):1433–1461.

Härdle, W. (1990). *Applied nonparametric regression.* Number 19. Cambridge university press.

Hardy, N. and Buonomano, D. V. (2018). Encoding time in feedforward trajectories of a recurrent neural network model. *Neural computation*, 30(2):378–396.

Haykin, S. S., Widrow, B., and Widrow, B. (2003). *Least-mean-square adaptive filters*, volume 31. Wiley Online Library.

Hinton, G. E. and Roweis, S. T. (2003). Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Honeine, P. (2015). Analyzing sparse dictionaries for online learning with kernels. *IEEE Transactions on Signal Processing*, 63(23):6343–6353.

Hu, R., Cao, J., Cheng, D., He, W., Zhu, Y., Xie, Q., and Wen, G. (2017). Self-representation dimensionality reduction for multi-model classification. *Neurocomputing*, 253:154–161.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501.

Isufi, E., Loukas, A., Simonetto, A., and Leus, G. (2017). Autoregressive moving average graph filtering. *IEEE Transactions on Signal Processing*, 65(2):274–288.

Jaeger, H. (2007). Echo state network. *Scholarpedia*, 2(9):2330.

Jolliffe, I. T. (1986). Principal component analysis and factor analysis. In *Principal component analysis*, pages 115–128. Springer.

Joyce, J. M. (2011). Kullback-leibler divergence. In *International Encyclopedia of Statistical Science*, pages 720–722. Springer.

Ketkar, N. (2017). Feed forward neural networks. In *Deep Learning with Python*, pages 17–33. Springer.

Lee, J. and Verleysen, M. (2008). Quality assessment of nonlinear dimensionality reduction based on k-ary neighborhoods. In *New Challenges for Feature Selection in Data Mining and Knowledge Discovery*, pages 21–35.

Lee, J. A., Lendasse, A., and Verleysen, M. (2004). Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis. *Neurocomputing*, 57:49–76.

Lee, J. A., Renard, E., Bernard, G., Dupont, P., and Verleysen, M. (2013). Type 1 and 2 mixtures of kullback–leibler divergences as cost functions in dimensionality reduction based on similarity preservation. *Neurocomputing*, 112:92–108.

Lee, J. A. and Verleysen, M. (2007). *Nonlinear dimensionality reduction*. Springer Science & Business Media.

Li, H., Jiang, H., Barrio, R., Liao, X., Cheng, L., and Su, F. (2011). Incremental manifold learning by spectral embedding methods. *Pattern Recognition Letters*, 32(10):1447–1455.

Li, Y. and Hamamura, M. (2015). Zero-attracting variable-step-size least mean square algorithms for adaptive sparse channel estimation. *International Journal of Adaptive Control and Signal Processing*, 29(9):1189–1206.

Liu, B. (2009). Some research problems in uncertainty theory. *Journal of Uncertain Systems*, 3(1):3–10.

Liu, B., Zhou, Y., Xia, Z.-g., Liu, P., Yan, Q.-y., and Xu, H. (2018). Spectral regression based marginal fisher analysis dimensionality reduction algorithm. *Neurocomputing*, 277:101–107.

Liu, W., Park, I., Wang, Y., and Príncipe, J. C. (2009). Extended kernel recursive least squares algorithm. *IEEE Transactions on Signal Processing*, 57(10):3801–3814.

Liu, W., Pokharel, P. P., and Principe, J. C. (2006). Correntropy: A localized similarity measure. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 4919–4924. IEEE.

Liu, W., Pokharel, P. P., and Príncipe, J. C. (2007). Correntropy: Properties and applications in non-gaussian signal processing. *IEEE Transactions on Signal Processing*, 55(11):5286–5298.

Liu, W., Pokharel, P. P., and Principe, J. C. (2008). The kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 56(2):543–554.

Liu, W. and Príncipe, J. C. (2008). Kernel affine projection algorithms. *EURASIP Journal on Advances in Signal Processing*, 2008(1):784292.

Liu, W., Principe, J. C., and Haykin, S. (2010). *Kernel Adaptive Filtering: A Comprehensive Introduction*. Wiley Publishing, 1st edition.

Liu, W., Principe, J. C., and Haykin, S. (2011). *Kernel adaptive filtering: a comprehensive introduction*, volume 57. John Wiley & Sons.

Luo, X., Deng, J., Liu, J., Wang, W., Ban, X., and Wang, J.-H. (2017). A quantized kernel least mean square scheme with entropy-guided learning for intelligent data analysis. *China Communications*, 14(7):1–10.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Makridakis, S. and Hibon, M. (2000). The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476.

Martinez-Vargas, J., Cardenas-Pena, D., and Castellanos-Dominguez, G. (2012). Extraction of stationary components in biosignal discrimination. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 1–4. IEEE.

Na, S.-H. and Lee, J.-H. (2014). Partial-update dimensionality reduction for accumulating co-occurrence events. *Pattern Recognition Letters*, 36:62–73.

Nadler, B., Lafon, S., Kevrekidis, I., and Coifman, R. R. (2006). Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Advances in neural information processing systems*, pages 955–962.

Niu, Q. and Chen, T. (2018). A new variable step size lms adaptive algorithm. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1–4. IEEE.

Ojha, V. K., Abraham, A., and Snášel, V. (2017). Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60:97–116.

Oliveira, A. G., Baumgartner, M. T., Gomes, L. C., Dias, R. M., and Agostinho, A. A. (2018). Long-term effects of flow regulation by dams simplify fish functional diversity. *Freshwater Biology*, 63(3):293–305.

Paul, R. and Chalup, S. K. (2017). A study on validating non-linear dimensionality reduction using persistent homology. *Pattern Recognition Letters*, 100:160–166.

Peluffo Ordoñez, D. H., Lee, J. A., and Verleysen, M. (2014). Recent methods for dimensionality reduction: A brief comparative analysis. In *2014 European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2014)*.

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238.

Pietrzak, M. B., Faldzinski, M., Balcerzak, A. P., Meluzín, T., and Zinecker, M. (2017). Short-term shocks and longterm relationships of interdependencies among central european capital markets. *Economics & Sociology*, 10(1):61.

Popov, V. L. and Heß, M. (2016). *Method of dimensionality reduction in contact mechanics and friction*. Springer.

Ren, Y., Suganthan, P., and Srikanth, N. (2015). A comparative study of empirical mode decomposition-based short-term wind speed forecasting methods. *IEEE Trans. Sustain. Energy*, 6(1):236–244.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

Sadatnejad, K. and Ghidary, S. S. (2016). Kernel learning over the manifold of symmetric positive definite matrices for dimensionality reduction in a bci application. *Neurocomputing*, 179:152–160.

Saide, C., Lengelle, R., Honeine, P., Richard, C., and Achkar, R. (2015). Nonlinear adaptive filtering using kernel-based algorithms with dictionary adaptation. *International Journal of Adaptive Control and Signal Processing*, 29(11):1391–1410.

Salaken, S. M., Khosravi, A., Nguyen, T., and Nahavandi, S. (2017). Extreme learning machine based transfer learning algorithms: A survey. *Neurocomputing*, 267:516–524.

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409.

Scholkopf, B. and Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Sheather, S. (2004). Density estimation. *Statistical science*, pages 588–597.

Shi, J., Jiang, Q., Zhang, Q., Huang, Q., and Li, X. (2015). Sparse kernel entropy component analysis for dimensionality reduction of biomedical data. *Neurocomputing*, 168:930–940.

Simão, M., Neto, P., and Gibaru, O. (2017). Using data dimensionality reduction for recognition of incomplete dynamic gestures. *Pattern Recognition Letters*, 99:32–38.

Singh, A. and Principe, J. C. (2010). A loss function for classification based on a robust similarity metric. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–6. IEEE.

Slakter, M. J. (1965). A comparison of the pearson chi-square and kolmogorov goodness-of-fit tests with respect to validity. *Journal of the American Statistical Association*, 60(311):854–858.

Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., and Lendasse, A. (2007). Methodology for long-term prediction of time series. *Neurocomputing*, 70(16-18):2861–2869.

Spathis, D., Passalis, N., and Tefas, A. (2018). Fast, visual and interactive semi-supervised dimensionality reduction. In *European Conference on Computer Vision*, pages 550–563. Springer.

Sun, Y. and Wen, G. (2017). Cognitive facial expression recognition with constrained dimensionality reduction. *Neurocomputing*, 230:397–408.

Suykens, J. A., Lukas, L., and Vandewalle, J. (2000). Sparse approximation using least squares support vector machines. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 2, pages 757–760. IEEE.

Szafraniec, F. H. (2015). The reproducing kernel property and its space: the basics. *Operator Theory*, pages 3–30.

Taieb, S. B., Bontempi, G., Atiya, A. F., and Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert systems with applications*, 39(8):7067–7083.

Taieb, S. B., Sorjamaa, A., and Bontempi, G. (2010). Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing*, 73(10):1950–1957.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

Tiao, G. C. and Xu, D. (1993). Robustness of maximum likelihood estimates for multi-step predictions: the exponential smoothing case. *Biometrika*, 80(3):623–641.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244.

Tsai, F. (2010). Comparative study of dimensionality reduction techniques for data visualization. *Journal of artificial intelligence*, 3(3):119–134.

Valencia-Aguirre, J., Álvarez-Meza, A. M., Daza-Santacoloma, G., Acosta-Medina, C. D., and Castellanos-Domínguez, G. (2012). Human activity recognition by class label lle. In *Iberoamerican Congress on Pattern Recognition*, pages 260–267. Springer.

Van Der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

Vapnik, V. and Izmailov, R. (2017). Knowledge transfer in svm and neural networks. *Annals of Mathematics and Artificial Intelligence*, 81(1-2):3–19.

Venna, J., Peltonen, J., Nybo, K., Aidos, H., and Kaski, S. (2010). Information retrieval perspective to nonlinear dimensionality reduction for data visualization. *Journal of Machine Learning Research*, 11(Feb):451–490.

Wang, J., Yue, S., Yu, X., and Wang, Y. (2017a). An efficient data reduction method and its application to cluster analysis. *Neurocomputing*, 238:234–244.

Wang, W., Zhao, J., Qu, H., Chen, B., and Principe, J. C. (2017b). Convergence performance analysis of an adaptive kernel width mcc algorithm. *AEU-International Journal of Electronics and Communications*, 76:71–76.

Weigend, A. S. (2018). *Time series prediction: forecasting the future and understanding the past*. Routledge.

Wu, Z., Shi, J., Zhang, X., Ma, W., Chen, B., and Senior Member, I. (2015). Kernel recursive maximum correntropy. *Signal Processing*, 117:11–16.

Xiong, T., Bao, Y., and Hu, Z. (2013). Beyond one-step-ahead forecasting: evaluation of alternative multi-step-ahead forecasting models for crude oil prices. *Energy Economics*, 40:405–415.

Yang, H., Pan, Z., Tao, Q., and Qiu, J. (2018). Online learning for vector autoregressive moving-average time series prediction. *Neurocomputing*.

Yang, X., Yu, F., and Pedrycz, W. (2017). Long-term forecasting of time series based on linear fuzzy information granules and fuzzy inference system. *International Journal of Approximate Reasoning*, 81:1–27.

Ye, Q., Yin, T., Gao, S., Jing, J., Zhang, Y., and Sun, C. (2016). Recursive dimension reduction for semisupervised learning. *Neurocomputing*, 171:1629–1636.

Yu, L., Wang, S., and Lai, K. K. (2008). Forecasting crude oil price with an emd-based neural network ensemble learning paradigm. *Energy Economics*, 30(5):2623–2635.

Yu, W., Wang, R., Nie, F., Wang, F., Yu, Q., and Yang, X. (2018). An improved locality preserving projection with 1-norm minimization for dimensionality reduction. *Neurocomputing*, 316:322–331.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zhang, G., Wu, Y., and Liu, Y. (2014). An advanced wind speed multi-step ahead forecasting approach with characteristic component analysis. *Journal of Renewable and Sustainable Energy*, 6(5):053139.

Zhang, J., Chen, L., Zhuo, L., Liang, X., and Li, J. (2018a). An efficient hyperspectral image retrieval method: Deep spectral-spatial feature extraction with dcgan and dimensionality reduction using t-sne-based nm hashing. *Remote Sensing*, 10(2):271.

Zhang, S., Cao, H., Yang, S., Zhang, Y., and Hei, X. (2018b). Sequential outlier criterion for sparsification of online adaptive filtering. *IEEE Transactions on Neural Networks and Learning Systems*.

Zhao, M., Chow, T. W., Wu, Z., Zhang, Z., and Li, B. (2015). Learning from normalized local and global discriminative information for semi-supervised regression and dimensionality reduction. *Information Sciences*, 324:286–309.

Zhao, S., Chen, B., and Principe, J. C. (2011). Kernel adaptive filtering with maximum correntropy criterion. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2012–2017. IEEE.

Zhao, S., Chen, B., Zhu, P., and Príncipe, J. C. (2013). Fixed budget quantized kernel least-mean-square algorithm. *Signal Processing*, 93(9):2759–2770.

Zheng, Y., Wang, S., Feng, J., and Chi, K. T. (2016). A modified quantized kernel least mean square algorithm for prediction of chaotic time series. *Digital Signal Processing*, 48:130–136.

Zhong, X. and Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, 67:126–139.

Zhuo, L., Cheng, B., and Zhang, J. (2014). A comparative study of dimensionality reduction methods for large-scale image retrieval. *Neurocomputing*, 141:202–210.

Zou, Y., Wang, T., Xiao, J., and Feng, X. (2017). Temperature prediction of electrical equipment based on autoregressive integrated moving average model. In *Automation (YAC), 2017 32nd Youth Academic Annual Conference of Chinese Association of*, pages 197–200. IEEE.