MODÈLE D'ANALYSE DE SURVIE POUR LA PRÉDICTION DE L'ÉVOLUTION DES GRAPHES DYNAMIQUES

—

APPLICATIONS AUX RÉSEAUX SOCIAUX ET SERIES MULTIPLES

par

Étienne Gaël Tajeuna

Thèse présentée au Département d'informatique
en vue de l'obtention du grade de philosophiæ doctor (Ph.D.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 4 novembre 2020

Le 4 novembre 2020

Le jury a accepté la thèse de Étienne Gaël Tajeuna dans sa version finale.

Membres du jury

Professeur Shengrui Wang
Directeur
Département d'informatique

Professeur Mohamed Bouguessa
Codirecteur
Département d'informatique
Université du Québec à Montréal (UQAM), Montréal, QC, CA

Professeur Manuel Lafond
Membre interne
Département d'informatique

Professeur Patrick Gallinari
Membre externe
Département d'informatique
Sorbone Université, Paris, France

Professeur Djemel Ziou
Président-rapporteur
Département d'informatique

Sommaire

Dans le domaine de l'analyse des graphes dynamiques, l'état de l'art démontre qu'il n'existe pas assez d'approches formelles pour la modélisation et la prédiction des phénomènes de changements que subirait un sous-graphe qui évolue dans le temps. Dans cette thèse, nous développons une approche formelle pour l'analyse de l'évolution des sous-graphes. De manière générale, nous concevons une approche basée sur le principe des fenêtre coulissantes pour analyser l'évolution des sous-graphes. Plus précisément, suivant la dynamique évolutive d'un graphe, nous définissons une fenêtre d'observation à partir de laquelle, on pourra mieux observer l'évolution de ce graphe et ainsi mieux apprécier les différents changements que pourrait subir les différentes sous-structures de ce graphe. Afin de modéliser l'évolution des sous-structures de graphe, nous nous sommes inspirés du modèle statistique basé sur l'analyse de survie. Nos contributions relatives au modèle d'analyse de survie et à l'analyse des graphes dynamiques sont mises en exergue dans deux champs d'applications, à savoir, les réseaux sociaux et les séries multiples.

Dans le cas de l'analyse des réseaux sociaux (ARS), nous utilisons les structures de graphes pour relater les éventuelles relations que pourraient avoir les différentes entités que constituent un réseau social donné. Ici, on s'intéresse particulièrement à la formation des communautés (ensemble d'entités densément liées) et de leurs évolutions dans le temps. Une communauté dans son évolution est susceptible de subir plusieurs modifications structurelles. En effet, au fil du temps, plusieurs communautés pourraient fusionner en une seule communauté ou alors une communauté pourrait devenir plus petite (du fait de la réduction du nombre d'entités appartenant à cette communauté), plus grande (du fait de l'augmentation du nombre d'entités appartenant à cette communauté), se diviser en plusieurs autres communautés ou rester identique

(stable) à elle-même. Ces différents phénomènes de changement que subirait une communauté sont des évènements importants à analyser afin de pouvoir prédire quels seront les prochains changements auxquels une communauté serait exposée. Pour ce faire, nous définissons un modèle de régression basé sur l'analyse de survie pour modéliser l'évolution des communautés par rapport au temps et ainsi prédire aux instants ultérieurs quels seront les différents phénomènes de changements que subirait une communauté.

Dans le cas de l'analyse des séries multiples (ASM), nous supposons que chaque série d'un ensemble de séries peut s'écrire comme une suite de profils qui se répètent à des intervalles de temps distincts. C'est ainsi que nous proposons une approche locale pour étudier les différentes variations d'un ensemble de séries chronologiques données. L'étude des différentes variations nous permet en effet de mieux comprendre l'évolution de cet ensemble de séries et ainsi faire des prévisions avec un faible taux d'erreur. Plus précisément, suivant des intervalles de temps réguliers, nous projetons nos séries chronologiques dans un espace topologique (graphe) où il sera désormais possible de mieux appréhender les dépendances relationnelles entre les séries et ainsi déterminer les profils significatifs qui caractérisent ces dernières. Les profils significatifs extraits des sous-structures de graphes sont suivis par rapport au temps afin de déterminer celles qui ont tendance à être plus récurrentes ou à disparaître par rapport au temps. Afin de modéliser l'évolution des profils et calculer le risque qu'ils apparaissent, nous utilisons la régression de Cox et les réseaux de neurones. Les réseaux de neurones exploités nous permettent de générer et prédire de manière permanente et automatiquement des descripteurs qui pourraient expliquer l'évolution des profils identifiés.

Remerciements

Mes remerciements sont adressés à mes directeurs de recherche, les Professeurs Shengrui Wang et Mohamed Bouguessa pour leurs encouragements et surtout pour la totale confiance qu'ils m'ont accordée tout au long de cette période de recherche. Je ne saurais ignorer les efforts qu'ils ont portés à mon égard afin de me permettre d'avoir des compétences tant en recherche fondamentale qu'en recherche appliquée. D'ailleurs, je remercie le Professeur Mohamed Bouguessa pour les opportunités de stages en entreprise.

Mes remerciements vont aussi à l'ensemble des enseignants du Département d'informatique à savoir le Professeur Luc Lavoie pour ses conseils et son encadrement dans la formation en enseignement, les Professeurs Djemel Ziou et Froduald Kabanza pour leurs encouragements et conseils d'orientations. Leurs conseils et encouragements ont impacté sur ma persévérance durant toute cette longue période de recherche.

Je remercie également le Conseil de Recherches en Sciences naturelles et en Génie du Canada et SAP Canada pour le soutien financier qui m'a accompagné durant ces années de recherche.

Je tiens à remercier toute ma famille qui a œuvré de prêt ou de loin de par leur encadrement et soutien moral pour le succès de ce travail de recherche.

Des remerciements particuliers vont à l'endroit de ma très chère épouse qui par sa patience a démontré qu'elle a toujours cru en moi. Lors de toute cette période de recherche, elle a su être présente pour m'accompagner dans les moments difficiles.

Abréviations

ARS Analyse des Réseaux Sociaux

ASM Analyse des Séries Multiples

AIC *Akaike Information Criteria*

AR *Auto Regressive*

ARIMA *AutoRegressive Integrated Moving Average*

AS *Autonomous System*

AD *Anderson-Darling*

CSS-MLE *Conditional Sum of Squares - Maximum Likelihood Expectation*

CCD *Chlorine Concentration Data set*

CDF *Cumulative Density Function*

CNN *Convolutional Neural Network*

CVM *Cramer Von-Mises*

DBLP *Digital Bibliography & Library Project*

ELD *Electricity Load Data set*

ELF *Electricity Load Forecasting Data set*

EOG *ElectroOculoGraphy signal data*

EQD *Earth Quake Data set*

ESR *Epileptic Seizure Recognition data set*

FAG-F *Finch Aggregation Forecasting*

GPS *Gesture phase Segmentation data set*

KLDIV *Kullback Divergence*

KNN *K-Nearest Neighboor*

KNNR *K-Nearest Neighboor Regression*

KS *Kolmogorov-Smirnov*

LR *Logistic Regression*

LSTM *Long-Short Term Memory*

MA *Moving Average*

MAE *Mean Average Error*

MLP *Multi-Layer Perceptron*

NEF *NEtwork based Forecasting*

NSERC *Natural Sciences and Engineering Research Council of Canada*

NBAG-F *Network Based Aggregation Forecasting*

OBM *OrBitMap*

PCA *Principal Components Analysis*

RMSE *Root Mean Square Error*

RNN *Recurrent Neural Network*

RDS *Rock Data Set*

SARIMA *Seasonal AutoRegressive Integrated Moving Average*

SDSS *Smart Data Set for Sustainability*

SVM *Support Vector Machine*

SVR *Support Vector Regression*

SDSS *Smart Data Set for Sustainability*

SyD *Synthetic Data set*

VAR *Vector Auto Regressive*

Table des matières

Liste des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction

1.1 Mise en contexte et problématique

1.1.1 Mise en contexte

Le monde réel est constitué d'un ensemble d'entités homogènes et hétérogènes qui peuvent être mises en commun selon leurs différentes relations qui, en général sont variables dans le temps : on parle de système complexe. Les questions posées sur l'analyse des systèmes complexes ont suscité l'intérêt de plusieurs chercheurs [1], [2], [3], [4]. Dépendamment des domaines d'expertises, on peut constater que, de manière fondamentale, les structures de graphes sont pour la plupart du temps employées pour représenter un système complexe. Un graphe est une structure de données composée d'un ensemble de noeuds et d'un ensemble de liens relatant la relation entre les noeuds. Dans le cas d'un système complexe, les entités sont représentées par les noeuds du graphe tandis que la mise en commun entre les entités de ce système complexe est représentée par l'ensemble des liens du graphe. Dans la figure 1.1, nous avons un exemple de graphe constitué de 12 noeuds et 22 liens.

L'usage du graphe tel que présenté dans la figure 1.1 pose un défi majeur en ce qui concerne la dynamique ou l'évolution des systèmes complexes. En effet, d'un instant à un autre, dans un système complexe, le nombre d'entités peut varier et par conséquent influencer sur les relations existentielles entre les entités. Dans la précédente définition

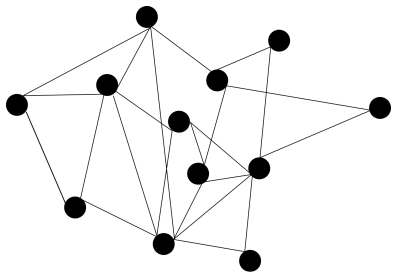


figure 1.1 – Exemple de graphe.

du graphe, utilisée pour représenter un système complexe, il est impossible de faire ressortir l'aspect dynamique que présente un réel système complexe. En d'autres termes, le graphe dans sa définition de base, est une structure de données statique qui ne met pas en exergue l'aspect temporel et dynamique que constitue un système complexe réel. Afin de prendre en compte l'aspect évolutif des systèmes complexes, la notion de graphe dynamique a été mise sur pied [5], [6], [7], [8]. Dans le domaine de l'Analyse des Réseaux Sociaux (ARS) [9], [10], [11], [12] et dans plusieurs autres domaines applicatifs [13], [14], [15], [16] on appelle graphe dynamique une séquence ordonnée ou série chronologique de graphes statiques. Dans une telle séquence ou série, chaque graphe statique est une instance du graphe dynamique à un instant donné. Dans la figure 1.2 nous avons une série de 4 graphes statiques G_1, G_2, G_3 et G_4 illustrant chacun l'état d'un graphe dynamique aux intervalles de temps t_1, t_2, t_3 et t_4 respectivement.

À partir de la représentation illustrée dans la figure 1.1, il est clair qu'on ne saurait capter les différents phénomènes de disparitions et d'apparitions des liens et noeuds à différents intervalles de temps. Par contre dans la figure 1.2, on peut observer par exemple, lors du passage de l'intervalle de temps t_1 à l'intervalle de temps t_2, les disparitions des liens et noeuds (de couleur bleue). De même, on peut observer les apparitions de nouveaux noeuds et liens (de couleur noire) à l'intervalle de temps t_4.

Il est important de noter que les graphes dynamiques illustrés par les séries ordonnées de graphes statiques (comme dans la figure 1.2) ne sont qu'une représentation discrète simulant l'évolution temporelle des systèmes complexes. Bien qu'ils aient une évolution continue, l'utilisation des graphes dynamiques pour simuler la dynamique des systèmes complexes présente de nombreux avantages. Par le biais des théories

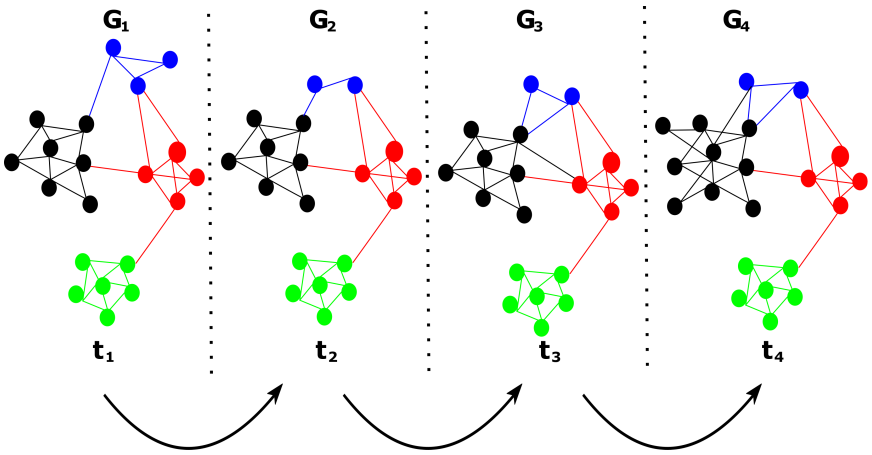


figure 1.2 – Exemple de graphe dynamique. $G_1 - G_4$ sont les instances du graphe dynamique aux intervalles de temps t_1 à t_4 respectivement.

de l'analyse des graphes, il est possible de découvrir d'importantes informations et révélatrices sur la nature du système complexe mis en investigation. Dans l'analyse des graphes dynamiques, on note deux grandes orientations de recherche. Pour certains, ils s'intéressent principalement à l'étude de la dynamique de l'apparition et de la disparition des liens et des noeuds dans un graphe qui évolue dans le temps. D'autres, s'intéressent à l'étude de la formation des sous-structures de graphes et de l'analyse de leurs évolutions dans le temps.

Dans le cas où l'analyse du graphe dynamique est portée sur la disparition et / ou l'apparition des noeuds et des liens, l'objectif est de comprendre la dynamique globale du système complexe investigué. L'investigation ici faite a pour but de construire un modèle mathématique / statistique qui facilitera la prédiction de la disparition / apparition des noeuds et des liens du graphe dynamique à des instants futures. De telles recherches sont exploitées dans de nombreux domaines. Dans le domaine de la santé par exemple, une telle analyse peut être utilisée pour comprendre le phénomène de propagation d'une maladie ou d'un virus dans un environnement [17], [18], [19]. Dans l'analyse des médias sociaux, les graphes dynamiques sont utilisés pour comprendre le processus menant à la radicalisation des individus dans le temps.

Dans le cas où les investigations portées sur les graphes dynamiques sont plus locales, il est question de détecter des sous-structures de graphes ou sous-graphes et

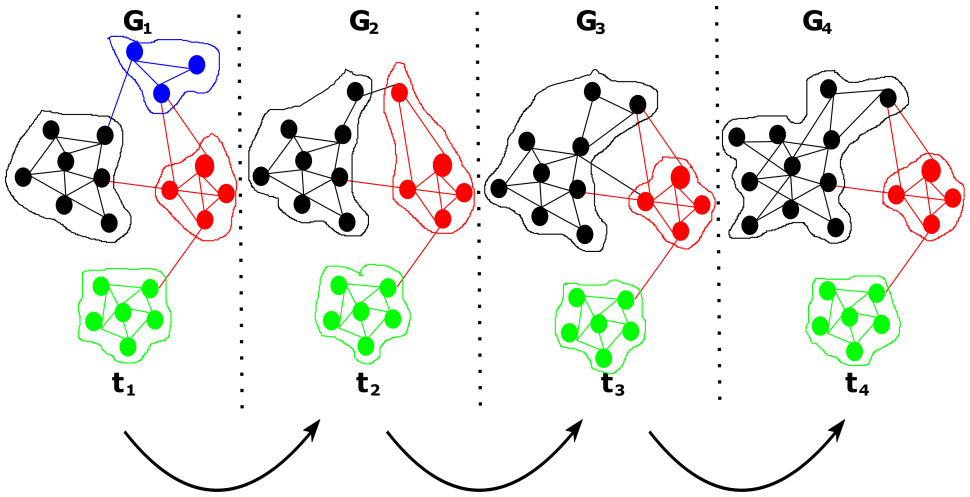


figure 1.3 – Évolution des sous-structures de graphe.

ensuite étudier leurs évolutions afin de prédire les éventuels changements auxquels ces sous-graphes sont prédisposés. Dans le domaine de l'ARS par exemple, les sous-graphes connus sous le nom de communautés sont détectés et suivis dans le temps afin de prédire les divers phénomènes de changements structurels qu'elles pourraient subir [20], [21], [22]. De façon plus explicite, dans la figure 1.3 nous avons un exemple de sous-graphes (étiquettés respectivement de couleurs noire, verte, rouge et bleue) observés aux intervalles de temps t_1 à t_4. Ici on peut constater que certains sous-graphes ont subi des modifications dans le temps à l'instar des sous-graphes bleu, noir et rouge.

Il est important de savoir qu'on entend par sous-structure de graphe, un ensemble de noeuds fortement connectés les uns aux autres et faiblement connectés aux autres noeuds du graphe. Dans le cas de la figure 1.3 au premier interval de temps t_1, le graphe est formé de quatre sous-graphes identifiés par les couleurs bleue, noire, verte et rouge. De manière visuelle, dans cet exemple, il est facile de pointer du doigt les différents sous-graphes. Toutefois, lorsqu'on fait face à des graphes de plus grande taille il est plus difficile d'identifier à vu d'oeil de telles sous-structures. Heureusement, dans la littérature, il existe plusieurs algorithmes développés pour identifier les sous-structures de graphes densément connectées. On note par exemple les algorithmes de Girvan et Newman [23], *fast greedy modularity* [24], *Cfinder*, *Infomap* [25] et [26].

Dans [23], une approche hiérarchique de type *top-down* (du haut vers le bas) est mise en place pour subdiviser une structure de graphe en sous-structures de graphes. En fait, les liens ayant une forte mesure de centralité sont itérativement supprimés du graphe jusqu'à ce qu'on obtienne une partition de sous-structures de graphes disjoints. Les sous-structures ici sont obtenues suivant une contrainte définie par la fonction de modularité de Newman et Girvan. Dans [24], nous avons aussi une approche hiérarchique, mais cette fois-ci de type *bottom-up* (du vers le haut), où chacun des noeuds est initialement considéré comme une sous-structure de graphe. Les liens sont itérativement ajoutés pour former des sous-graphes sous la même contrainte de la fonction de modularité de Newman et Girvan. Dans [25], une fonction de mappage dual est définie pour identifier les sous-graphes disjoints. De manière spécifique, la fonction duale décrit la trajectoire d'une marche aléatoire effectuée dans le graphe en effectuant une compression (en utilisant le *minimum descriptive length*) du message diffusé. De cette manière il devient possible de détecter les zones où le message diffusé a tendance de traîner. De telles zones correspondent en général aux sous-structures de graphes. À la différence des modèles définis dans [23], [24] et [25] ou une partition du graphe en entrée est effectuée, dans [26], les sous-structures de graphes identifiés peuvent être superposées. Les sous-structures de graphes ici sont des cliques de noeuds formant un graphe complet.

Dans le cadre de cette thèse, nous nous intéresserons exclusivement au suivi des sous-structures de graphes formées dans les graphes dynamiques. Nous présenterons des applications liées à notre étude sur les réseaux sociaux et séries multiples.

Dans l'analyse de l'évolution des graphes, on peut constater que les sous-structures de graphes qui s'y forment évoluent également avec le temps (figure 1.3). L'étude de cette évolution suscite l'intérêt de nombreux chercheurs suivant sa portée dans plusieurs domaines. En effet, l'on peut constater que les sous-structures de graphes dans leurs évolutions sont susceptibles à plusieurs modifications structurelles qui peuvent avoir des interprétations majeures en fonction du contexte dans lequel on s'y trouve. On note par exemple qu'une sous-structure de graphe dans son évolution pourrait prendre plus des proportions et devenir plus grande qu'elle ne l'était ou alors le phénomène inverse pourrait se produire. On pourrait aussi noter entre autres qu'une sous-structure de graphe dans son évolution peut se diviser en plusieurs autres

sous-structures de graphe ou alors plusieurs sous-structures de graphe pourraient se mettre ensemble pour former une seule sous-structure de graphe. La figure 1.4 illustre les phénomènes de changements à savoir la *fusion*, la *division*, *le rétrécissement* et la *stabilité* dont une sous-structure de graphe pourrait être sujette lors de son évolution. Il est important de noter que, lors de l'évolution d'une sous-structure de graphe, celle-ci pourrait subir plusieurs modifications qui causeraient la *disparition* de cette dernière. La *disparition* tout comme l'*apparition* d'une sous-structure de graphe sont aussi des informations cruciales dans l'évolution d'une sous-structure de graphe.

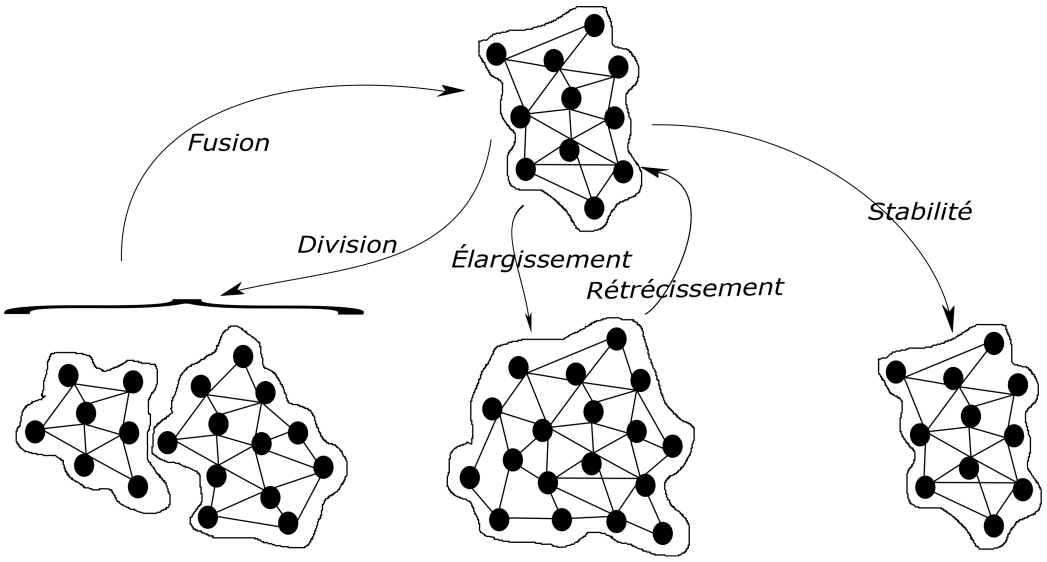


figure 1.4 – Phénomènes de changements auxquels les sous-structures de graphe peuvent être sujettes.

Connaître d'avance quels phénomènes de changement subira une sous-structure de graphe est une question importante pour les chercheurs utilisant les graphes comme structure de données pour analyser et / ou résoudre une problématique donnée. Pour prédire les phénomènes de changement que subirait une sous-structure de graphe, il se pose à priori la question de savoir comment faut-il suivre son évolution. Dans le domaine de l'ARS, de nombreuses approches [20], [27], [28], [22], [29] ont été développées pour répondre à cette question. De manière générale, étant donné un graphe dynamique, à chaque instance de ce graphe dynamique sont détectées des sous-structures de graphes. Par la suite, les sous-structures de graphes identifiés sont

deux-à-deux comparées à différents intervalles de temps. Il est important de noter que la comparaison entre deux sous-structures de graphe, dans la plupart des cas, est faite en terme de noeuds et les liens entre les noeuds ne sont pas considérés. Ainsi, lors des comparaisons, deux sous-structures de graphes sont mises en ensemble (ou alors considérées similaires) si elles ont des noeuds en commun.

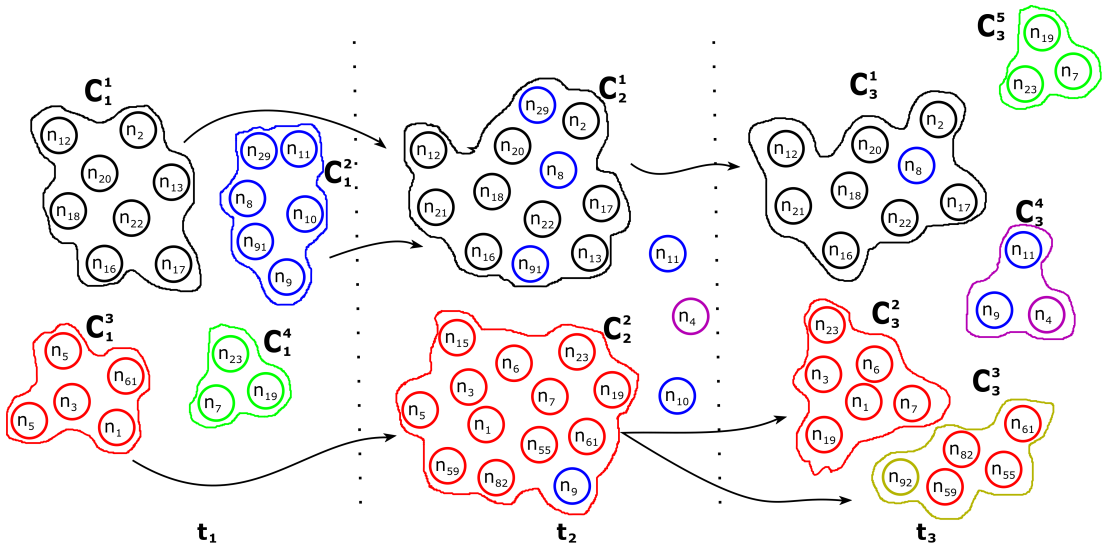


figure 1.5 – Suivi de l'évolution de sous-structures de graphes. Les séquences $C_1^1 \rightarrow C_2^1 \rightarrow C_3^1$, $C_1^2 \rightarrow C_2^1 \rightarrow C_3^1$, $C_1^3 \rightarrow C_2^2 \rightarrow C_3^2$, $C_1^3 \rightarrow C_2^2 \rightarrow C_3^3$ illustrent l'évolution des sous-structures des graphes C_1^1, C_1^2 et C_1^3 respectivement.

Pour être plus explicite, expliquons le principe de suivi de l'évolution des sous-structures de graphes en prenant l'exemple illustré dans la figure 1.5. Dans ce cas, à chacun des intervalles de temps allant de t_1 à t_3 les ensembles de sous-structures de graphes $\{C_1^1,\ C_1^2,\ C_1^3,\ C_1^4\}$, $\{C_2^1,\ C_2^2\}$ et $\{C_3^1,\ C_3^2,\ C_3^3,\ C_3^4,\ C_3^5\}$ sont détectées respectivement. Par la suite, toujours allant de t_1 à t_3, les sous-structures de graphes détectées à l'intervalle de temps t_1 sont comparées à celles détectées dans l'intervalle de temps t_2 et par la suite, les sous-structures de graphes détectées à l'intervalle de temps t_2 sont comparées à celles détectées à l'intervalle de temps t_3. Au terme des comparaisons, on obtient par exemple les séquences $C_1^1 \rightarrow C_2^1 \rightarrow C_3^1$, $C_1^2 \rightarrow C_2^1 \rightarrow C_3^1$, $C_1^3 \rightarrow C_2^2 \rightarrow C_3^2$, $C_1^3 \rightarrow C_2^2 \rightarrow C_3^3$ illustrant l'évolution des sous-structures des graphes C_1^1, C_1^2 et C_1^3 respectivement. Pour chacune des séquences obtenues, on peut constater que les sous-structures de graphes la constituant sont similaires sur le

point de vue des noeuds. Une fois que sont obtenues les séquences de sous-structures de graphes reflétant l'évolution des sous-structures de graphes, ces dernières (les séquences) sont utilisées comme base de connaissances d'un modèle d'apprentissage machine pour prédire les éventuels phénomènes de changements qu'elles pourraient subir. Il vaut la peine de mentionner que les modèles d'apprentissage machine ici employés prennent en entrée un ensemble de caractéristiques reflétant la structure organisationnelle des noeuds : on parle de caractéristiques topologiques telles que le *degré de centralité*, la *transitivité*, etc. Dépendamment des domaines d'applications, d'autres caractéristiques statistiques ou sémantiques lorsqu'elles existent peuvent être prises en compte.

En résumé, pour prédire les phénomènes de changement que subirait une sous-structure de graphe, une approche 2-tiers est généralement définie. Dans le premier tiers, un algorithme est utilisé pour détecter et suivre l'évolution des sous-structures de graphes dans le temps. À la sortie de ce premier tiers sont obtenues des séquences de sous-structures de graphes reflétant leurs évolutions chronologiques. Dans le deuxième tiers, plusieurs caractéristiques topologiques et / ou statistiques / sémantiques (lorsqu'elles existent) sont extraites des séquences et plutard utilisées comme base de connaissances d'un modèle d'apprentissage machine. Il est important de noter que dans le domaine de l'ARS, dans la majorité des travaux existants, une sous-structure de graphe correspond à un groupe de noeuds densément connectés et faiblement connectés au reste de noeuds du graphe. Identifier les sous-structures de graphes revient encore à retrouver les groupes de noeuds qui sont fortement connectés les uns aux autres et faiblement connectés aux restes de noeuds du graphe.

1.1.2 Problématique

Bien que des solutions aient été apportées pour répondre au problème de prédiction des phénomènes de changement auxquels les sous-structures de graphes seraient sujettes, il existe toujours de nombreuses situations réelles où l'usage des techniques existantes a des difficultés à prédire avec une bonne précision. Il est tout d'abord important de rappeler que dans les méthodes existantes, où une stratégie 2-tiers est employée, les résultats de la prédiction des phénomènes de changement (effectués au

niveau du deuxième) sont fortement influencés par la stratégie utilisée pour suivre l'évolution des sous-structures de graphes (méthode effectuée dans le premier tiers). En pratique, lorsque les sous-structures de graphes sont suivies dans leurs évolutions, on peut constater que, d'un instant (intervalle de temps) à un autre certaines sous-structures de graphes peuvent être observées et d'autres pas. Cela dit, une sous-structure de graphe pourrait avoir une évolution discontinue dans le temps.

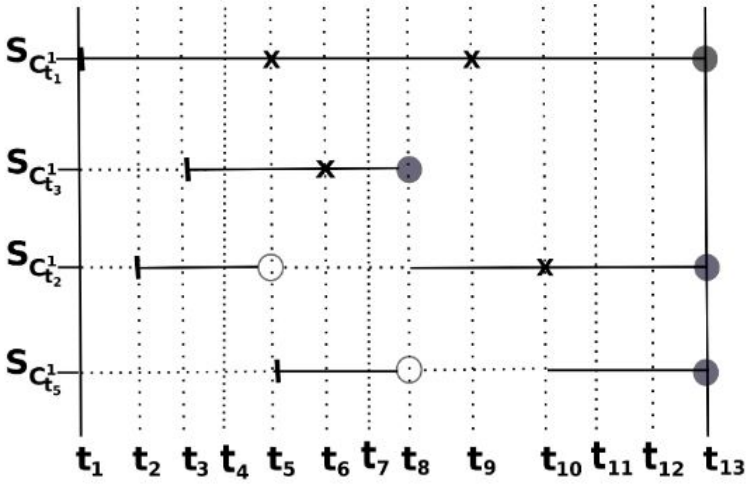


figure 1.6 – Étude de l'évolution des sous-structures de graphes. Chacun des $S_{C^1_{t_i}}$ spécifie l'évolution d'une sous-structure de graphe. « | » pour signifier le *début des observations*. « **X** » pour signifier qu'un *évènement* s'est produit. « ∘ » pour signifier un *arrêt momentané des observations*. « • » pour signifier *la fin des observations. Enfin « ... » pour signifier qu'on a aucune observation.*

Prenons l'exemple présenté dans la figure 1.6. Ici, nous avons l'évolution de quatre sous-structures de graphes dénotées respectivement par $S_{C^1_{t_1}}$, $S_{C^1_{t_2}}$, $S_{C^1_{t_3}}$ et $S_{C^1_{t_5}}$ pour la période allant de t_1 à t_{13}. On peut constater que l'évolution de certaines sous-structures sont continues tels sont les cas de $S_{C^1_{t_1}}$ et $S_{C^1_{t_3}}$. Par contre les évolutions $S_{C^1_{t_2}}$ et $S_{C^1_{t_5}}$ sont discontinuent dû au fait que nous ayons un manque d'observation à certains laps de temps. Les caractéristiques extraites d'une sous-structure de graphe qui évolue de manière non-discontinue constituent une base de connaissances consistente qui renseignerait sur le processus évolutif de cette dernière. Par contre, lorsque cette sous-structure présente une évolution discontinue, nous avons un manque d'informations qui malheureusement empêcherait de comprendre le processus évolutif de cette dernière.

Cette problématique liée au manque d'observations semble peu pertinente si de tels cas ne sont pas assez récurrents dans l'évolution d'une sous-structure de graphe. Cependant, comme ci-haut mentionné, en fonction de l'algorithme de suivi de l'évolution des sous-structures de graphes, on peut constater que certaines sous-structures dans leurs évolutions respectives présentent un fort taux d'absence d'observations à différents laps de temps. Avec une récurrence de manque d'observations, il devient difficile de mettre en pratique des techniques d'apprentissage automatique. Il est aussi important de noter que le manque d'observations à différents laps de temps pourrait ne pas être lié à l'algorithme à priori utilisé, mais plutôt à la nature des données mises en investigation. Peu importe que l'on ait une problématique liée à la nature des données ou l'algorithme de suivi de l'évolution des sous-structures de graphes, la question de prédiction des phénomènes de changements reste complexe lorsque nous avons une récurrence de manque d'observations. Cette problématique de prédiction se pose également lorsque nous avons des sous-structures dont l'évolution est assez brève dans le temps tels que l'évolution $S_{C^1_{t_3}}$ dans la figure 1.6.

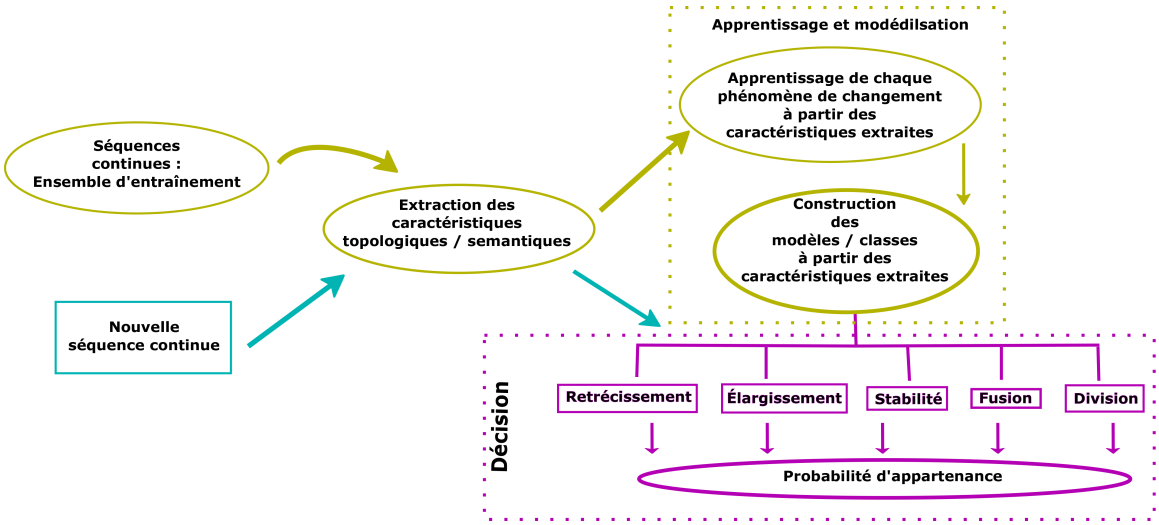


figure 1.7 – Méthodologie de prédiction de phénomènes de changements que subiraient les sous-structures de graphes dans leurs évolutions dans le temps.

Mis à part les problèmes de discontinuité que présenteraient les sous-structures dans leurs évolutions, on pourrait aussi questionner la méthodologie employée pour prédire les différents phénomènes de changements. En effet, au regard de la figure 1.7,

on pourrait résumer le principe de prédiction utilisé à un problème de classification. De ce fait, si par exemple on veut prédire à plusieurs instants futurs quels seront les différents phénomènes de changements auxquels s'expose une sous-structure de graphe, il est difficile de répondre à cela à partir de cette représentation. Ceci est dû au fait que, les informations prises en entrée du processus ne tiennent pas compte de l'aspect temporel. Pour prédire quels seront les phénomènes de changements que subira une sous-structure de graphe, il faudra avoir une nouvelle connaissance de l'état de l'évolution de la sous-structure en question. En absence de cette information, il est impossible de prédire à plusieurs instants futurs. De plus, comme mentionnée ci-haut, cette représentation ne tient que compte de l'évolution continue de sous-structures de graphe.

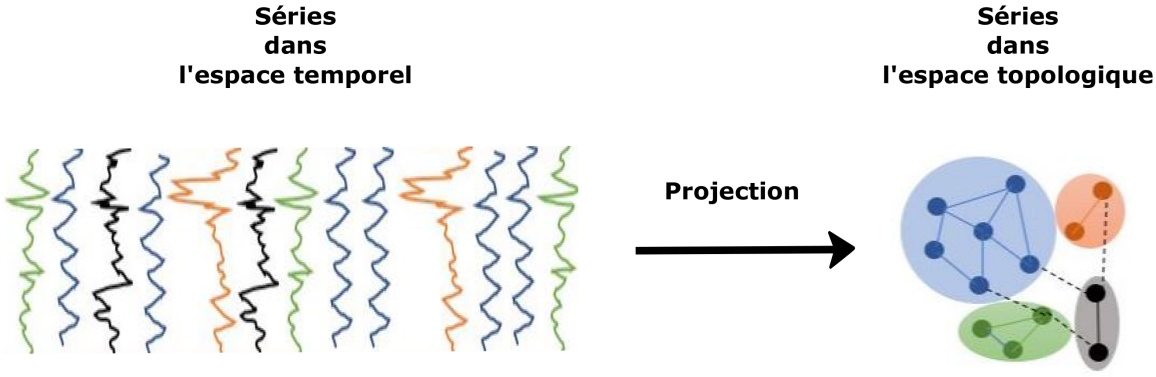
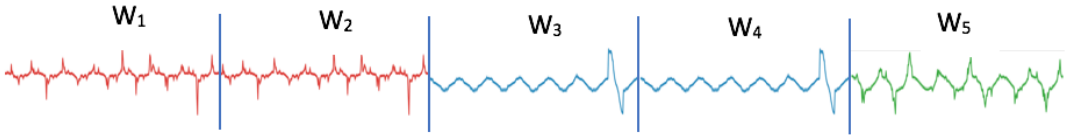


figure 1.8 – Projection d'un ensemble de séries chronologiques d'un espace temporel vers un espace topologique.

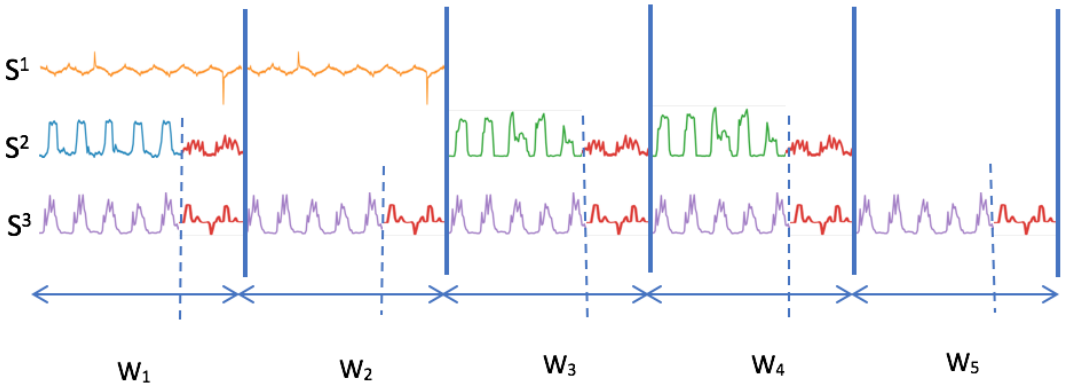
Il est important de souligner que les limites jusqu'ici soulevées sont très rencontrées dans le domaine de l'ARS [20]. Pour ce qui est du domaine de l'analyse des séries multiples (ASM), les structures de graphes sont très souvent exploitées [30], [31] pour projeter les séquences de données d'un espace temporel à un espace topologique afin de facilement retrouver des groupes de séries partageant le même profil. La figure 1.8 illustre un exemple de séries chronologiques qui sont projetées dans un espace topologique. Les groupes formés dans l'espace topologique représentent les groupes de séries ayant le même profil dans l'espace temporel. Afin de prédire quelles prochaines valeurs prendront les séries chronologiques, des groupes de séries constitués sont extraits des séries représentatives qui sont par la suite utilisées comme base de

connaissance d'un modèle d'apprentissage automatique.

Dans l'exemple illustré dans la figure 1.8, la projection topologique obtenue est une représentation statique de l'état de similirité entre les différentes séries chronologiques. Cependant, certaines séries au cours du temps peuvent présenter de nombreuses variantes ou alors présenter différents profils à différents intervalles de temps. Si l'on prend par exemple le cas illustré dans la figure 1.9(a), on constate que cette série aux intervalles de temps W_1 et W_2 présente le même profil et brusquement aux intervalles de temps W_3 et W_4 un autre profil est présenté. Finalement, dans l'intervalle de temps W_5 un autre profil distinct des précédents est présenté.



(a) Exemple de série présentant différents profils à différents intervalles de temps.



(b) Exemples de profils présentés par différents séries à différents intervalles de temps.

figure 1.9 – Illustration des profils (profils étiquettés par les différentes couleurs) présentés par les séries chronologiques à différents intervalles de temps.

Si l'on adopte le même principe illustré dans la figure 1.8 où les séries entières sont mises en commun, on fera face à deux problèmes majeurs : (1) la difficulté de comparer les séries présentant plusieurs profils à différents intervalles de temps, (2) on ne saurait capter la dynamique de changement de profils qui s'opère à travers le temps. De plus, si l'on considère un cas beaucoup plus complexe incluant non seulement les profils variant dans le temps, mais aussi l'absence d'observation telle que présenté

dans la figure 1.9(b), il est difficile de mettre sur pied une métrique de comparaison entre séries qui nous permet de construire notre graphe dû au manque d'observations que nous avons dans certains intervalles de temps. En d'autres termes, l'usage d'un graphe statique pourrait donner un aspect global des relations existantes entre les séries, mais ne saurait présenter les états de relations à différents intervalles de temps. Par contre, si l'on considère la projection dans un espace topologique dynamique (graphe dynamique) où chaque intervalle de temps seraient les intervalles W_1 à W_5 (dans la figure 1.9) on aurait certainement différents types de groupes (sous-structures de graphes) que l'on peut suivre dans le temps tel que fait dans le domaine de l'ARS.

En résumé, de manière générale, nous avons présenté l'importance et l'intérêt de l'usage des structures de graphes lorsque nous faisons face aux systèmes complexes. Dans l'étude de ces systèmes complexes qui peuvent être ici représentés par les graphes dynamiques, lorsqu'on se focalise sur les sous-structures de graphes et leur évolution dans le temps, d'importantes informations peuvent être exploitées. D'ailleurs, plusieurs auteurs de differents domaines, se sont intéressés à la prédiction des phénomènes de changements auxquels s'exposeraient des sous-structures de graphes dans leurs évolutions chronologiques. On note en particulier les travaux effectués dans les domaines de l'ARS [32], [21], [33] et l'ASM [34], [35] qui font ici office aux cas d'application des structures de graphes dynamiques.

1.2 Objectif de recherche

Dans la section précédente, nous avons présenté un certain nombre de problèmes existant dans l'analyse des graphes dynamiques particulièrement lorsqu'il s'agit du suivi des sous-structures de graphes. Nous avons présenté des problématiques rencontrées dans les domaines de l'ARS et l'ASM. Dans cette thèse nous adressons ces problèmes. Plus précisément, nous touchons du doigt les problèmes liés au suivi des sous-structures de graphes. Rappelons que nous avons évoqué le fait que dans certains domaines d'applications, certaines sous-structures de graphes évoluent de manière discontinue ce qui ne faciliterait pas la tâche lors de l'analyse de l'évolution de telles structures. Et bien dans cette thèse nous proposons une nouvelle façon de suivre l'évolution des sous-structures de graphes dans le domaine de l'ARS qui permettrait de réduire de

taux de manque d'observations que l'on pourrait rencontrer lors de l'évolution d'une sous-structure de graphe.

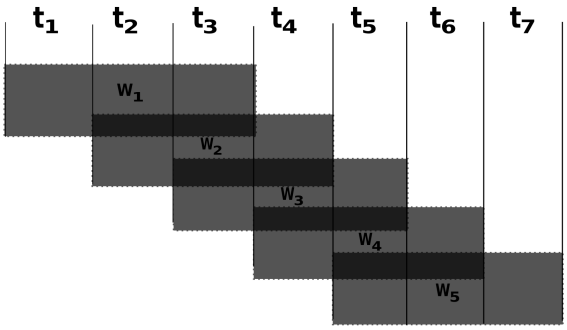


figure 1.10 – Exemple de fenêtre coulissante. Ici nous avons un exemple de fenêtre W ayant une taille équivalente à trois laps de temps. La fenêtre glisse de la gauche vers la droite avec un pas équivalent à un laps de temps. De la gauche vers la droite, la fenêtre est représentée par les instances W_1, W_2, W_3, W_4 et W_5.

En effet, un graphe dynamique généralement représenté par une série de graphes statiques, où chaque graphe statique représente l'état de la structure du graphe dynamique à un instant t. En faisant le suivi à partir d'une fenêtre coulissante comme présentée dans la figure 1.10, il est a été démontré que la qualité de l'évolution des sous-structures de graphes seraient nettement meilleure que lorsque que les intervalles de temps complètement disjoints. Cependant, le choix de la taille de fenêtre reste aussi une question rarement abordée dans le cadre de l'ARS en particulier et même dans le cadre de l'analyse des graphes dynamiques en général. Dans cette thèse nous abordons cette problématique.

Dans le cadre de l'ASM, au lieu d'avoir une approche globale qui malheureusement ne permet pas d'avoir un suivi sur les différents profils présentés par un ensemble de séries, nous proposons une approche locale qui nous permettra d'avoir un regard effectif sur les profils présentés par différentes séries à différents intervalles de temps. Il est important de noter ici que le choix de l'intervalle de temps tout comme dans l'ARS n'est toujours pas tâche facile. Pour le cas de l'ASM, dans cette thèse, nous présentons aussi comment le choix de l'intervalle de temps pourrait se faire sans avoir besoin de l'intervention d'un utilisateur et plutôt se faire de manière automatique.

Au regard de l'approche abordée pour prédire les phénomènes de changements, il

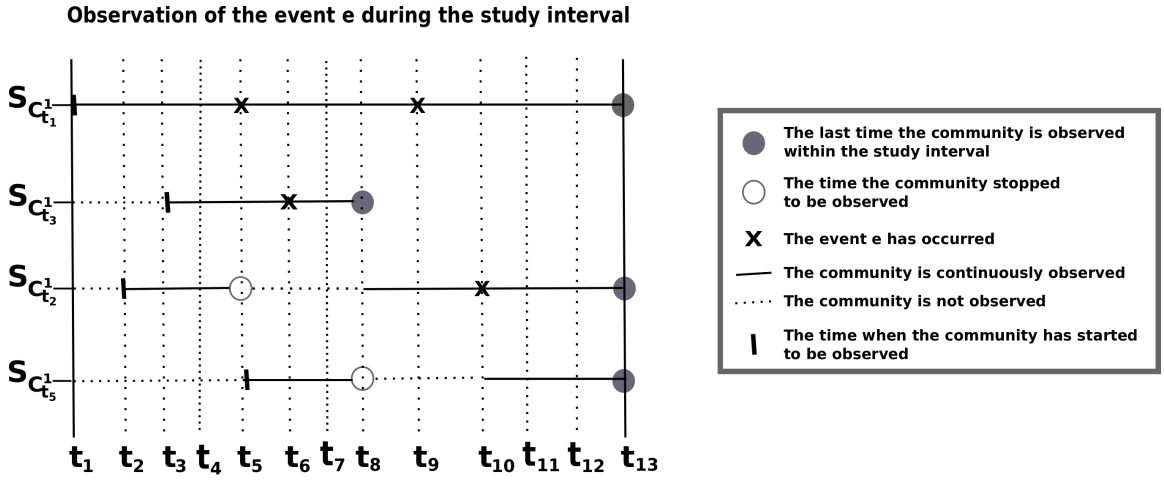


figure 1.11 – Analyse de survie pour la prédiction des phénomènes de changements que subirait une sous-structure de graphe qui évolue dans le temps [36].

est clair que prédire à plus d'un instant futur nécessite à chaque fois la connaissance de l'état structurel de la sous-structure de graphe en question. Dans cette thèse, nous présentons une autre façon d'aborder la question qui nous permet de non seulement prédire l'instant suivant que prendrait l'état d'une sous-structure dans son évolution, mais aussi à plusieurs instants futurs. En effet, reprénons l'exemple présenté dans la figure 1.6 et supposons que les phénomènes de changements peuvent se produire de manière indépendante. En d'autres termes si l'on suppose qu'un phénomène qui a eu lieu lors de l'évolution d'une sous-structure de graphe n'a aucune influence sur les autres types de phénomènes que cette sous-structure pourrait subir dans son évolution, alors une étude pourrait être faite au cas par cas. Ainsi, en prenant le cas de la figure 1.11 où l'on regarde l'évolution des sous-structures de graphes par rapport à un phénomène de changement e (e pouvant être le phénomène de *rétrécissement*, de *élargissement*, de *stabilité*, de *fusion* ou de *division*.), deux phénomènes distincts ne sont pas pris en compte dans l'intervalle de temps allant de t_1 à t_{13}. Avec ce type de réprensentation, il devient donc possible de faire une analyse statistique consistant à décompter ou énumérer de nombre de fois que l'évènement e se produit par rapport au temps et partir de cette relation pour construire une fonction qui nous permettra d'évaluer le risque que l'évènement e se produise. Ce type d'analyse est fréquemment rencontré dans les études cliniques où il est question de calculer le temps de survie

d'un patient : on parle d'analyse de survie.

Tout au long de cette thèse, nous utilisons le modèle statistique connu sous le nom d'analyse de survie pour répondre aux problèmes de prédictions mentionnés dans les domaines de l'ARS et l'ASM. Pour parler des problèmes de prédiction évoqués dans le domaine de l'ARS, nous utilisons l'analyse de suivie pour modéliser les différents phénomènes de changements que subirait une sous-structure qui évolue dans le temps. Cette modélisation nous permettra de non seulement de mieux comprendre l'impact des caractéristiques extraites des sous-structures de graphes, mais aussi de prédire à plusieurs instants futurs quels seraient les phénomènes de changements auxquels s'exposerait une sous-structure de graphe dans son évolution. Parlant des problèmes de prédiction évoqués dans le domaine de l'ASM, nous utilisons l'analyse de survie pour modéliser l'évolution des profils présentés par les séries chronologiques et par la suite prédire si ces profils seront exhibés ou pas par une série dans les instants futurs.

Il vaut la peine de rappeler que, de manière pratique, les profils encore connus sous le nom de *régimes* sont très souvent rencontrés dans les domaines de la santé, la climatologie, la finance, etc. Il est important de savoir qu'un *régime* est un profil présenté par un ensemble de séries co-évolutives à différents intervalles de temps. En d'autres termes, étant donné un ensemble de séries chronologiques évoluant dans un même contexte, on appelle *régime*, tout profil exhibé par plusieurs séries à différents intervalles de temps. Défini de cette façon, pour un jeu de séries multiples, un *régime* serait un profil qui se répète verticalement et horizontalement tels qu'illustré dans la figure 1.12. Dans cet exemple, on peut constater que le profil de couleur marron, bien qu'il soit répétitif ne sera pas considéré comme un *régime* puisqu'il n'y a qu'une série qui présente ce profil aux intervalles de temps I_1 et I_3. Pourtant, pour ce qui est des autres profils, ils sont tous considérés comme des profils vu qu'ils sont répétitivement vus au sens horizontal et vertical. C'est donc ce type de profils que nous étudions par le biais de l'analyse de survie, le risque qu'ils (les régimes) continuent d'être observés dans l'ensemble de données.

Il faut noter que l'identification des régimes dans les séries chronologiques multiples a pour ultime but la modélisation des phénomènes de transitions entre les régimes qui expliqueraient la non-stationnarité que présente chacune des séries dans un tel contexte. On note d'ailleurs les travaux de Hoschtein et al. [37] qui propose un modèle Markovien

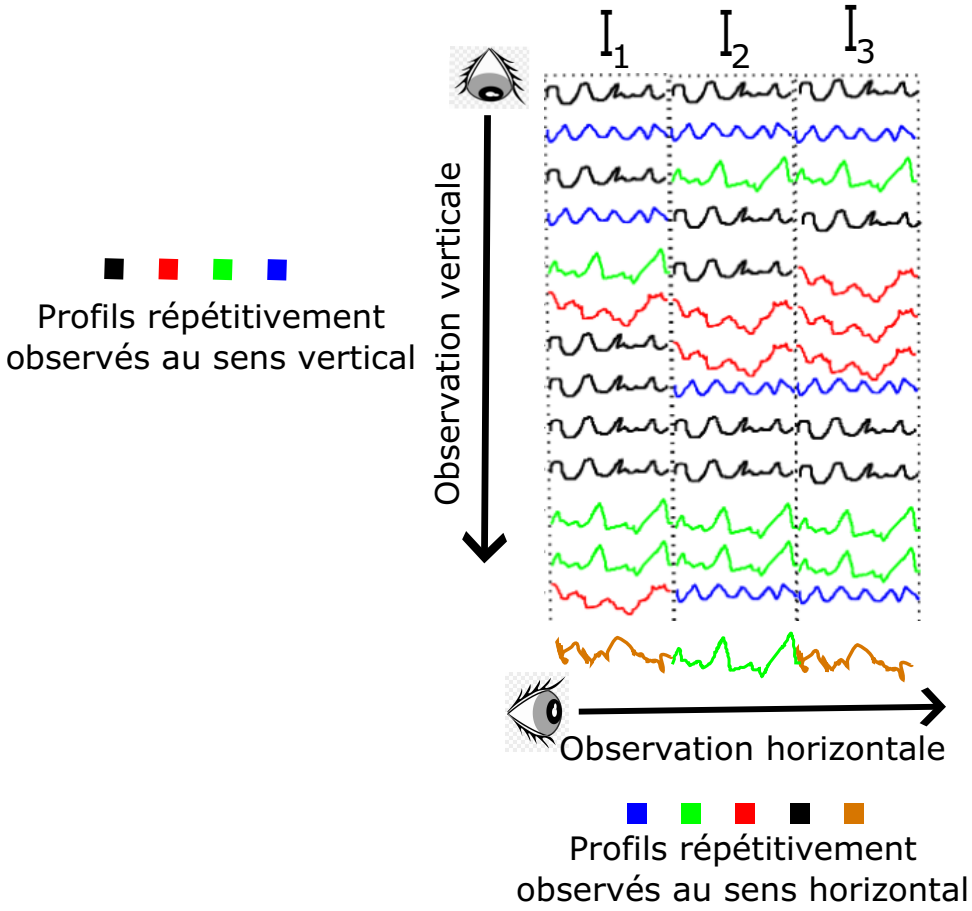


figure 1.12 – Exemples de profils répétitifs aux sens vertical et horizontal. Les profils portant les couleurs noir, bleu, vert et rouge sont des régimes. Tant dis que le profil de couleur marron n'est pas considéré comme un régime bien qu'il soit répétitif.

pour modéliser le mécanisme de transition entre les différents régimes ici pris pour les états de transitions. Dans l'approche décrite dans [37], les régimes sont connus d'avance. De plus une seule matrice de transition est définie pour matérialiser les probabilités de transition entre les régimes observées dans l'ensemble des séries. Dans les travaux de Hallac et al. [38], une modélisation basée sur les graphes dynamiques est faite pour identifier les profils co-existants entre les séries multiples. Toutefois, les auteurs dans [38] ne se sont pas appesantis sur le mécanisme de transition observé dans les séries chronologiques.

Rappelons que, l'analyse de survie est une approche statistique généralement

utilisée pour évaluer le risque d'apparition d'un évènement à partir d'un échantillon de population/observations. Dans l'analyse de survie, nous avons des modèles non paramétriques tels que la méthode de Kaplan Meier et des modèles paramétriques tels que la régression de Cox. Dans l'approche de survie non paramétrique de Kaplan Meier [39] il est question d'estimer le risque d'apparition dudit évènement à partir de l'ensemble d'observations. Formellement, étant donné un échantillon de n observations dont chacune des observations est susceptible de subir un évènement e à tout temps t. Si l'on appelle $\mathcal{N}_e(t)$ la fonction qui décompte le nombre d'observation(s) ayant subit l'évènement e. Pour un interval d'observation allant de t_1 à t_m, $m > 1$, selon Kaplan Meier, la fonction de survie $S(t)$ est définie comme suit :

$$S(t) = \begin{cases} 1 & si \quad t < t_1 \\ \prod_t \left(1 - \frac{\mathcal{N}_e(t)}{n - \mathcal{N}_e(t)}\right) & si \quad t \in [t_1,\ t_m] \end{cases} \tag{1.1}$$

La fonction d'estimation de risque d'apparition d'un évènement tel que présenter dans (1.1) nous donne une information générale sur la possibilité d'observation de cet évènement. On n'a aucun contrôle sur les facteurs qui pourraient influencer sur la possibilité d'observer un évènement. Dans l'approche de survie paramétrique de Cox [40] la fonction de survie par rapport à un évènement e est estimée en fonction d'un ensemble de facteurs (variables) qui influenceraient sur la possibilité que ledit évènement se produise. Formellement, étant donné une observation $\mathcal{O}$ pour laquelle on a recensé un ensemble de facteurs $\mathcal{Z}$ pouvant influencer sur le risque que cette observation soit sujette de l'évènement e, suivant Cox, le risque pour cette observation $\mathcal{O}$ est donné par :

$$S(t) = exp\left(-\int_{t_1}^{t} \lambda_0(\tau) exp(\gamma \mathcal{Z}) d\tau\right) \tag{1.2}$$

où γ est le vecteur de paramètres quantifiant l'importance des facteurs $\mathcal{Z}$. Ces paramètres peuvent être estimés par une approche numérique basée sur l'algorithme de Raphson-Newton. La fonction $\lambda_0()$ est la fonction de base qui permet de généraliser le risque d'apparition dudit évènement si on n'a aucune connaissance des facteurs $\mathcal{Z}$. De

manière expérimentale, la densité de probabilité de Weibull est souvent prise pour approximer cette fonction de base.

Dans l'approche de survie de Cox, on peut donc constater que, pour estimer le risque d'observation d'un évènement, on a besoin de l'ensemble des facteurs prélevés sur les observations. Dans le cadre de cette thèse, vu qu'il est question d'identifier les covariantes ou caractéristiques qui expliqueraient les phénomènes de changements que subirait une sous-structure de graphe, nous opterons pour approche paramétrique. Toutefois, au lieu d'utiliser un modèle de Cox tel que présenté dans (1.2) où les caractéristiques ou covariantes sont statiques, nous supposerons dans notre cas étude que ces caractéristiques sont également évolutives dans le temps. Ceci nous permet ainsi de construire un modèle de *Cox dynamique*. Les caractéristiques sont dans un premier temps supposés évoluer de manière linéaire et dans un second temps on supposera que ces caractéristiques suivent une évolution non linéaire.

1.3 Contributions

Dans les sections précédentes, nous avons identifié des problèmes liés en général dans le suivi des structures de graphes dynamiques et en particulier dans les domaines de l'ARS et l'ASM. Nous proposons de nouvelles façons d'aborder les différentes problématiques. Ces différentes façons concourent ici non seulement aux contributions liées à l'analyse des graphes dynamiques, mais aussi au modèle d'analyse de survie. Pour chacun des domaines ARS et ASM nos contributions se prśentent comme suit.

1.3.1 Première contribution : Nouvelle approche de détection de fenêtre dans les graphes dynamiques

Comme nous l'avons mentionné, lors du suivi des sous-structures de graphes dans le domaine de l'ARS on se rend compte que ces dernières pourraient avoir une évolution discontinue. Le fait de présenter de tels discontinuités et de manière abondante n'est pas favorable dans le processus d'apprentissage. Pour limiter ou réduire le taux de non-observations, nous proposons une nouvelle approche basée sur le principe de fenêtre coulissantes. Le principe de fenêtre coulissante certe existe déjà dans de

nombreux domaines y compris dans le domaine de l'ARS [41], [42], [43]. Seulement, particulièrement dans le domaine de l'ARS, au meilleurs de nos connaissances, aucune proposition à ce jour n'a été faite pour détecter de manière automatique la taille de la fenêtre adéquate à utiliser pour suivre l'évolution des graphes. Il est important de rappeler ici que, lorsqu'on parle de fenêtre, on fait allusion à un ensemble d'intervalles de temps. Une fenêtre coulissante peut être encore vue comme étant simplement des intervalles de temps qui se chevauchent.

En effet, nous partons du principe que la dynamique d'un graphe pourrait être formalisée à partir des phénomènes de changements de bases observés au niveau des noeuds du graphe dynamique en question. Les phénomènes de changements de bases ici étant le fait que, entre deux instants consécutifs, un nouveau noeud pourrait apparaître, disparaître ou rester dans le graphe. Plus on a les noeuds qui apparaissent et disparaissent avec peu de noeuds qui restent, plus le graphe dynamique en investigation présente une évolution rapide. Plus on a des noeuds qui restent et avec peu de noeuds qui apparaissent et disparaissent, plus le graphe dynamique en investigation présente une évolution lente. De ce principe, nous construisons une nouvelle fonction appelée *fluctuation* dépendante de la taille de la fenêtre et de la proportion de noeuds qui restent / disparaissent du graphe dynamique. À partir d'une fenêtre donnée, la fonction de *fluctuation* permet de cristalliser la dynamique d'un graphe à partir d'un score.

Tout comme dans le domaine de l'ARS, nous avons également évoqué dans le domaine de l'ASM la notion de fenêtre. En effet, à titre de rappel, dans le domaine de l'ASM, la définition de fenêtre renvoie à un intervalle de temps pour lequel un profil est exposé : les profils encore vus comme étant des périodes. Dans la littérature, nous avons pu constater qu'il existe bon nombre de méthodes liées à la détection des périodes dans les séries chronologiques. Cependant, lorsqu'on a affaire à plus d'une série chronologique dont l'investigation est simultanément soumise, nous n'avons pas rencontré des méthodes permettant de détecter ces profils. Certes à chacune des séries on pourrait appliquer une méthode de détection de profils. Seulement, cela devient fastidieux lorsque nous avons affaire à de nombreuses séries. De plus, les séries en investigation pourraient présenter de nombreuses irrégularités telles que les manques d'observations à différents intervalles de temps. Encore plus, certaines séries pourraient exposer de nombreux profils distincts au fil du temps.

Dans cette thèse nous partons du postulat qu'un ensemble de séries (séries multiples) simultanément mises en investigation puisse être issu d'un ensemble d'observations générées par des variables aléatoires partageant ou pas un nombre d'informations communes (tout comme dans un écosystème). Ainsi, nous pouvons emprunter le modèle de structure de graphe dynamique pour le domaine de l'ASM qui nous permettra de représenter les relations existantes entre les séries en invistigation. Cependant, tout comme dans le domaine de l'ARS, le problème du choix de l'intervalle de temps (fenêtre) pose également un enjeu majeur dans la mesure où l'on voudrait être capable de retrouver le plus petit profil exhibé communément par les séries. À la différence de la fonction de *fluctuation* donné dans le domaine de l'ARS, dans le domaine de l'ASM, nous définissons une autre fonction qui nous permettra d'évaluer la fluctuation de l'ensemble des séries mises en investigations. Cette fonction que nous appelons *variation de séries* est basée sur la proportion du nombre de sous-structures rencontrées par rapport au temps et est inversement proportionnelle au choix de la taille de la fenêtre. Plus la taille de la fenêtre est grande, plus le nombre de sous-structures identifiés devient petit. En d'autres termes, plus la taille de la fenêtre est grande, moins nous avons des profils qui pourraient se repéter dans le temps. De même, plus la fenêtre est petite plus nous avons profils qui pourraient ne pas se repéter dans le temps.

1.3.2 Deuxième contribution : Modélisation de l'évolution des sous-structures de graphes basée sur l'analyse de survie

Dans toutes les méthodes existentes de supervision utilisées dans le domaine de l'ARS pour prédire les phénomènes de changement, les suites continues de sous-structures de graphes sont utilisées dans la phase d'apprentissage pour modéliser chacun des phénomènes de changement. Ainsi seulement des sous-structures de graphes dont les évolutions sont continues sont prises en compte pour prédire le prochain phénomène de changement auquel s'exposerait une sous-structure de graphe qui évolue dans le temps. Or en réalité, les sous-structures de graphes pourraient présenter une évolution discontinue. Pour prendre en compte cela, nous proposons une étude statistique

faite sur chacun des phénomènes de changements par rapport au temps. Ainsi, nous sommes capable, à partir d'un ensemble de sous-structures de graphes qui évoluent dans le temps, d'estimer le risque qu'un évènement se produise ou pas. Le risque est ici modélisé par l'analyse de survie. L'analyse de survie est utilisée dans plusieurs domaines incluant l'économie et la santé. Dans le domaine de l'ARS nous n'avons pas rencontré des travaux qui ont été développés dans ce sens à l'exception des travaux dans [44], [45]. Cependant, dans [44], [45], les auteurs s'intéressent à la prédiction de la structure globale du graphe. En d'autres termes, ils s'intéressent à la prédiction des liens qui pourraient apparaître entre deux noeuds à des instants subséquents. Formellement, dans leurs approches, à l'aide de la régression de Cox, ils évaluent le risque d'apparition d'un lien entre deux noeuds ; ce qui expliquerait le sens de diffusion de l'information dans un graphe donné. Il important de noter que dans [45], les auteurs prennent deux cas de figure de la fonction de risque de propagation vers un noeud i du graphe décrite dans (1.2) : Le cas où la fonction de Cox est additive (c'est-à-dire $S_i(t) = 1 - exp(-\int_{t_1}^{t_m} \lambda_0(\tau) \cdot \mathcal{Z} d\tau)$) et le cas où elle est multiplicative (c'est-à-dire $S_i(t) = 1 - exp(-\int_{t_1}^{t_m} \lambda_0(\tau) \prod_j \mathcal{Z}_{j,i} d\tau)$ avec $\mathcal{Z}_{j,i}$ les variables de propagations du noeud j vers le noeud i).

Dans notre modélisation basée sur l'analyse de survie, nous faisons surtout usage de la régression de Cox qui nous permet non seulement d'estimer le risque général qu'un phénomène de changement se produise à un instant donné, mais aussi de savoir le risque individuel qu'une sous-structure de graphe soit sujette à un phénomène de changement. La régression de Cox ici utilisée étant une fonction temporelle, elle nous permet d'estimer le risque qu'un phénomène de changement se produise non pas seulement à un instant futur immédiat, mais à plusieurs instances temporelles futures. L'un des avantages principaux de cette fonction de Cox est qu'elle tient compte du fait que les sous-structures de graphes dans leurs évolutions pourraient présenter des discontinuités. Il est également important de mentionner que le modèle de Cox nous permet de faire usage de l'historique des connaissances topologiques des sous-structures de graphes dans leurs évolutions.

1.3.3 Troisième contribution : Apports au modèle de regression de Cox

Le modèle conventionnel de la régression de Cox est formé de deux parties : une partie connue sous le nom de fonction de risque de base et l'autre partie que nous appelons risque individuel. Le risque de base permet en cas d'absence d'observation ou alors en cas de manque de caractéristiques, de modéliser le risque / la probabilité qu'un évènement se produise. Le risque individuel est une fonction indépendante du temps qui permet à partir des caractéristiques connues (l'observation connue ici étant la sous-structure observée) d'estimer le risque / la probabilité qu'une sous-structure de graphe soit sujette d'un évènement ou phénomène de changement. La fonction de base étant inconnue, certains auteurs fixent le choix de cette fonction de manière intuitive. Dans le domaine de la santé, ce choix est souvent fixé à la densité de probabilité de Weibull [46], [47]. Dans notre contexte, on ne saurait faire un tel choix quelconque de peur que le choix de la densité de probabilité comme fonction de bases ne soit pas assez adéquate. Dans cette thèse, nous proposons une méthode expérimentale nous permettant d'aller chercher parmi un certain nombre de densités de probabilités celle-là plus adéquate.

Dans la partie de la fonction de risque individuelle, vu qu'elle est indépendante du temps, à des intervalles de temps (notamment les intervalles de temps futurs à prédire) on n'a aucune connaissance des caractéristiques liées aux sous-structures. Par conséquent juste la partie de la fonction de risque de base est exploitée. Or, en utilisant uniquement le risque de base, la prédiction d'un évènement quelconque est davantage moins précise dans le temps. Afin de pallier cette contrainte, nous proposons de générer les caractéristiques liées à l'évolution des sous-structures de graphes à ces instants futurs. Pour générer les caractéristiques à des instants futurs, nous supposons dans un premier temps que celles-ci soient linéaires. C'est ainsi que nous définissons une nouvelle fonction de survie dynamique qui utilise simultanément les régressions de Cox et multivariées linéaires. Dans un deuxième temps, nous supposons que ces caractéristiques évoluent de manière non-linéaires. Dans ce second cas, le modèle de survie dynamique proposé exploite simultanément les régressions de Cox et les K plus proches voisins (KNNR). Enfin, nous générons également les caractéristiques des

sous-structures de graphes par le biais d'un réseau de neurones récurrents, *Long Short Term Memory* (LSTM).

1.4 Structure et organisation de cette thèse

Le travail élaboré dans cette thèse est divisé en trois principaux chapitres qui constituent l'ossature des contributions au modèle de survie pour la prédiction de l'évolution des graphes dynamiques en général, et en particulier, dans leurs applications aux réseaux sociaux et séries multiples.

Dans le Chapitre 2, nous abordons les questions relatives au suivi des communautés dans les réseaux sociaux. Le suivi des communautés est fait pour prédire les éventuels phénomènes de changement qu'elles subiraient dans le temps. Plus précisément, en fonction du réseau social, nous commençons par construire une fenêtre à partir de laquelle on pourrait mieux observer son évolution ainsi que la formation des communautés et leurs évolutions dans le temps. Par la suite, suivant l'ensemble des communautés évolutives découvertes, nous utilisons une approche statistique basée sur la régression de Cox pour modéliser le risque qu'un phénomène de changement se produise dans cet ensemble. De manière particulière, nous supposons que les caractéristiques extraites des communautés évolutives sont stationaires. C'est ainsi que, nous définissons un modèle de Cox où les cofacteurs peuvent à tout temps être prédits en utilisant un modèle autoregressif multivarié. Pour justifier la pertinence du modèle théorique proposé, nous avons effectué des tests sur le réseau social de copublication extrait de la librairie numérique DBLP. Des tests ont été aussi effectués sur les réseaux de routeurs AS et AS-Caida pris sur le site de l'Université de Stanford et sur les relations des internautes du réseau Yelp.

Dans le Chapitre 3, nous abordons le problème de prévisions dans les séries chronologiques multiples. Ici, nous supposons que les séries chronologiques peuvent s'écrire comme une suite de profils qui se répètent à des intervalles de temps distincts. En effet, d'une série à une autre, on se rend que celles-ci peuvent présenter des profils répétitifs et similaires à différents intervalles de temps. Par l'usage des structures de graphes dynamiques, nous faisons une recherche heuristique pour cibler la meilleure taille de fenêtre pour laquelle on pourra retrouver des régimes qui se répètent tout au

long des séries et dans le temps. Les régimes sont ici représentés par les sous-structures de graphes et suivis dans le temps. Ainsi, nous obtenons des suites de sous-structures de graphes qui font états de l'ensemble des séries qui ont exhibé un régime donné par rapport au temps. L'évolution et l'extraction des caractéristiques sont soumises aux régressions de Cox et K-plus proches voisins (KNNR) pour modéliser le risque qu'un régime soit exhibé ou pas. Par la connaissance de ce risque, on peut donc prévoir quelles seront les prochaines valeurs que présentera une série chronologique dans le future. Des tests sur des données synthétiques et réelles sont faits pour démontrer l'utilité du modèle théorique proposé.

Dans le Chaptire 4, nous présentons un cas d'application portée sur l'analyse des consommations électriques des clients. Nous faisons une analyse du comportement quotidiens des clients selon leurs consommations électriques respectives. Contrairement aux deux chapitres précédents où la recherche d'un intervalle de temps est faite, ici elle fixée à une journée vu que l'analyse des comportements est quotidienne. Les comportements des clients dans leurs consommations électriques sont définis par des sous-structures de graphes qui sont par la suite suivis dans le temps. Ici les phénomènes de changements que l'on observe dans l'ARS sont supposés comme étant des caractéristiques importantes et liées aux caractéristiques topologiques extraites des sous-structures de graphes. C'est ainsi que nous construisons une architecture neuronale basée sur le LSTM pour générer les futures caractéristiques liées à l'évolution des sous-structures de graphes. Cette architecture est conjointement liée au modèle de survie pour prédire le risque qu'un comportement se reproduise ou pas aux instants futurs. Notre modèle a été expérimenté sur des données réelles extraites de EnerNOC et SDSS.

Chapitre 2

Modélisation et prédictions des changements de structures de communautés dans les réseaux sociaux dynamiques

Résumé

Au fil du temps, les structures de communautés observées dans les réseaux sociaux pourraient être sujettes à de nombreux phénomènes de changements connus sous le nom d'évènements critiques. Une communauté pourrait par exemple se *diviser* en plusieurs autres communautés, *s'accroître* en une communauté plus grande, *décroître* en une communauté plus petite, rester *stable* ou se *fusionner* à une autre communauté. On peut constater que dans de nombreux travaux récents, la question de prédiction de ces phénomènes de changements suscite de plus en plus l'intérêt de nombreux chercheurs. L'étude de l'évolution des communautés dans le temps est une étape importante dans le processus de prédiction des évènements critiques que celles-ci (les communautés) pourraient subir. D'ailleurs, ce problème de prédiction de phénomènes de changements est un défi majeur dans le domaine d'analyse des réseaux sociaux. En effet dans les travaux récents, on peut constater qu'il y a peu d'approches formelles dans la modélisation et la prédiction des évènements critiques que pourraient subir les communautés. Cette carence en approches formelles a motivé nos efforts à concevoir une nouvelle méthode statistique dans le but de prédire des évènements critiques. La méthode statistique ici proposée nous permet, entre autres, de faire un meilleur usage des évènements critiques antérieurs dont une communauté évolutive a été sujette. Dans cette méthode statistique, nous définissons le concept de fenêtre coulissante à partir duquel un modèle autorégressif et les techniques d'analyse de survie sont simultanément exploités. Le modèle autorégressif employé nous permet de simuler l'évolution des structures de communautés tandis que les techniques d'analyse de survie permettent de prédire les futurs phénomènes de changements.

Commentaires

Dans ce chapitre, il est intégralement présenté l'article intitulé *Modeling and Predicting Community Structure Changes in Time-Evolving Social Networks*, article publié dans la revue *IEEE Transactions on Knowledge and Data Engineering* (TKDE) en Juin 2019, volume 36, numéro 6, pages 1166-1180. Les auteurs sont Etienne G. Tajeuna, Mohamed Bouguessa et Shengrui Wang.

Dans la réalisation de cet article, j'ai (Etienne G. Tajeuna) contribué en tant que premier auteur dans la conception du modèle théorique y compris les expérimentations validant le modèle théorique proposé. La rédaction du document et la vérification des équations ont entièrement été accompagnées par mes directeurs de recherche Pr. Shengrui Wang et Pr. Mohamed Bouguessa.

Il est important de noter que le travail élaboré dans ce chapitre découle de l'article intitulé *Survival Analysis for Modeling Critical Events that Communities May Undergo in Dynamic Social Networks*, article publié en 2017 dans *Proceedings of the Symposium on Applied Computing* (SAC) des auteurs Etienne G. Tajeuna, Mohamed Bouguessa et Shengrui Wang. Dans cet article, nous faisons usage du modèle conventionnel de Cox pour modéliser les différents phénomènes de changements que pourraient subir les communautés évolutives. L'une des contraintes rencontrées dans le travail publié dans SAC est due au fait que, plusieurs communautés présentent une évolution discontinue et par conséquent rendent la prédiction des phénomènes de changements difficiles. Grâce au principe de fenêtre coulissante proposé dans notre travail publié dans TKDE, nous réduisons le taux d'absence d'information et par conséquent améliorons la qualité de prédiction.

Modeling and Predicting Community Structure Changes in Time-Evolving Social Networks

Étienne G. Tajeuna
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
etienne.gael.tajeuna@usherbrooke.ca
Mohamed Bouguessa
Département d'informatique, Université du Québec à Montréal,
Montréal, Québec, Canada H2X 3Y7
bouguessa.mohamed@uqam.ca
Shengrui Wang
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
shengrui.wang@usherbrooke.ca

Keywords: Dynamic Network; Community Evolution; Critical Events; Prediction; Autoregressive Model; Survival Analysis.

Abstract

As time evolves, communities in a social network may undergo various changes known as critical events. For instance, a community can either *split* into several other communities, *expand* into a larger community, *shrink* to a smaller community, remain *stable* or *merge* into another community. Prediction of critical events has attracted increasing attention in the recent literature. Learning the evolution of communities over time is a key step towards predicting the critical events the communities may undergo. This is an important and difficult issue in the study of social networks. In the work to date, there is a lack of formal approaches for modeling and predicting critical events over time. This motivates our effort to design a new statistical method for event prediction in order to make better use of histories of past changes. To this end, this paper proposes a sliding window analysis from which we develop a model that simultaneously exploits an autoregressive model and survival analysis techniques. The autoregressive model is employed here to

simulate the evolution of the community structure, whereas the survival analysis techniques allow the prediction of future changes the community may undergo.

2.1 Introduction

A social network is usually conceived as a graph in which individuals in the network are represented by nodes. Nodes are connected to each other by links which depict the relationships among the individuals. In real life, most social networks are dynamic in nature due to the appearance or disappearance of individuals and relationships among them. To model a dynamic social network, recent approaches [20], [48], [21], [28], [49], [50], [51] have used series of frozen networks each of which corresponds to a particular point in time. Such a modelization is useful to simulate the evolution of the social network and detect structural changes [20], [48], [28], [50] in order to predict the future structure of the network [28], [52], [53], [32], [54], [55].

The use of series of static graphs to simulate dynamic social networks has been motivated by the needs of social network analysis. There has been increased interest in analyzing the formation and evolution of communities of friends in networks, and also in understanding individuals' (or communities') behavior over time in order to predict their interactions. There is extensive literature on predicting interactions in social networks. Some authors [56], [57], [58] adopt a global approach, by investigating the whole network, in order to predict whether an existing link will appear or disappear at the next time point. Others [21], [53], mainly focus on the evolution of community structures in order to predict changes they may undergo in the future. In this paper we adopt a local approach in which we focus on modeling and predicting the evolution of communities over time.

Tracking community structures over time and predicting their future changes has important applications in various domains. For instance, in criminology [59], social network methodologies are used to discover and track groups of delinquent individuals over time in order to control them. In public health [60], social network strategies can be applied to discover the dynamics of certain subpopulations that are susceptible to a disease or to predict the early stages of an epidemic. Community structure prediction methods can also be applied on clinical data to predict how long a group of patients affected by a specific cancer will survive. In education, tracking and predictive modeling approaches can be used to reflect students' academic trajectories over different time periods and elaborate a general predictive framework for academic success and student

retention.

Note that we use the term "community" for any group of nodes that are densely connected among themselves and sparsely connected to others. A community structure can be drastically affected by changes in nodes and variations in their links, such as appearing and disappearing over time. Hence, from one time point t_i to another t_j, $t_j > t_i$, a community can *split* into several other communities, *expand* into a larger community, *shrink* to a smaller community or remain *constant*. In the same way, several communities can *merge* into one community. We call these possible phenomena (*split*, *expand*, *shrink*, *stable*, *merge*) critical events which communities may undergo over time.

To predict critical events that a community may undergo, existing approaches adopt a two-step methodology. The first step involves modeling and tracking the changes: a dynamic network is modeled as a series of frozen networks, each representing the instance of the network at a given time point. Then, a community detection algorithm is applied on each frozen network to identify the communities at the corresponding time-stamp. Finally, a pairwise comparison is performed to identify critical events and track the communities over time. Note that the tracking of a community yields a sequence of communities to illustrate its evolution. For example, in Figure 2.1(a) we show communities identified at different time-stamps. Figure 2.1(b) shows the tracking results, where it can be seen that the evolution of community $C^1_{t_1}$ is given by $C^1_{t_1} \rightarrow C^1_{t_3} \rightarrow C^1_{t_5} \rightarrow C^1_{t_6}$. The second step usually involves using a supervised learning approach, based on the features extracted from the identified communities, to predict the most probable next event an evolving community may undergo.

Advances have been made in predicting the critical events; these will be summarized in Section 2. We would mention here that the existing approaches suffer from one or more of the following shortcomings:

1. There is an intrinsic handicap when making investigations on consecutive snapshots of the dynamic social network, defined by time-stamps. Indeed, there is no way to define "optimal" time-stamps to represent the snapshots of the social network in order to capture the interrelation between nodes. Therefore, tracking communities in such conditions can yield communities evolving in a non-consecutive way. Moreover, some critical events that communities may

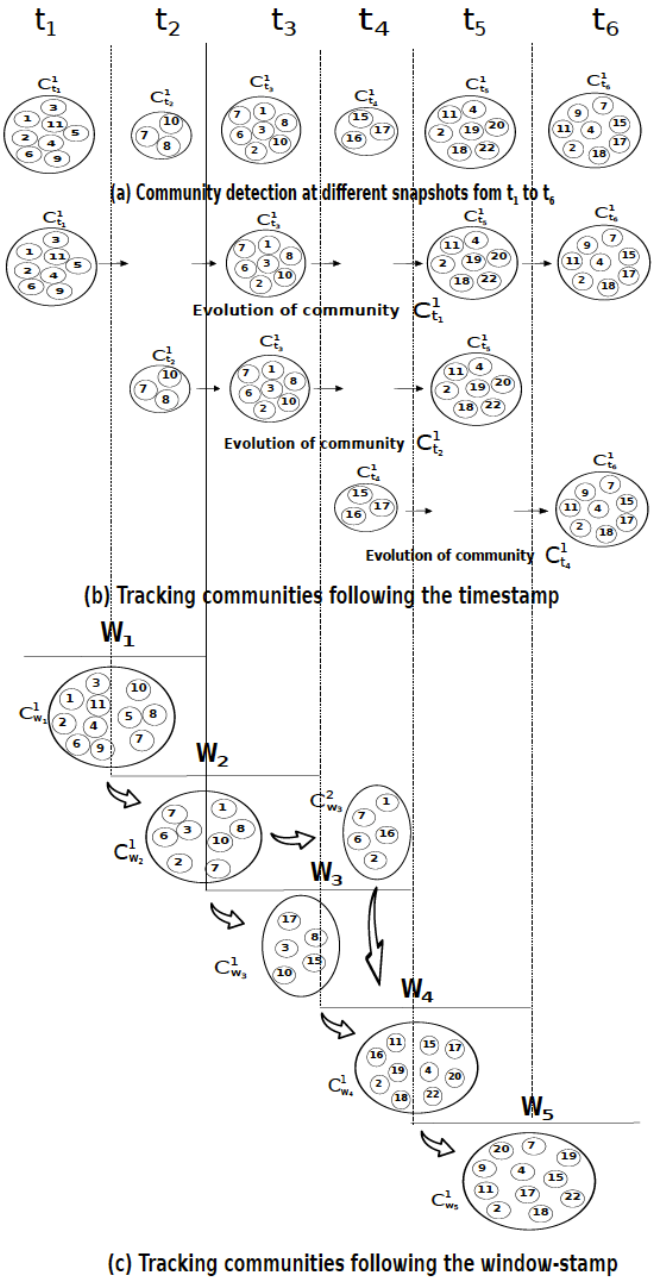


Figure 2.1 – An example of tracking communities over time. (a) The identified communities at different time-stamps from t_1 to t_6, where the time-stamps are independent. (b) Tracking result with non-overlapping time-stamps. (c) Tracking result with overlapping time-stamps.

undergo across time-stamps will not be detected and will be difficult to predict.

2. The majority of the existing methods have not addressed the problem of predicting critical events when communities are evolving in a non-consecutive way. The exception is Takaffoli et al. in [21]. We note that the authors in [21] defined features that take into consideration temporal aspects of evolving communities, enabling them to handle non-consecutive evolutions. However, predicting critical events with non-consecutive evolving communities requires a long existence history. Unfortunately, in certain real data sets, some evolving communities do not have such a long lifespan.
3. Most of the existing approaches address the problem of predicting critical events by finding the features that explain the changes the evolving communities may undergo and then using supervised classifiers to determine whether the event will occur at the next time point. Except for the model used in [54], however, such approaches cannot be extended to predict events at time-stamps later than the one immediately following.

To be more explicit, let us return to the example of the identified communities in Figure 2.1(a). Tracking the identified communities, we obtain the results presented in Figure 2.1(b). We can see that not all of the tracked communities are evolving in a consecutive way: for instance, looking at community $C^1_{t_1}$, we can see that it has no representation at times t_2 and t_4. The same observation can be made for the evolving community $C^1_{t_2}$ which has no representation at times t_4 and t_6 and community $C^1_{t_4}$ which has no representation at time t_5. Moreover, evolving community ($C^1_{t_4}$) presents a short lifespan which is difficult to handle in attempting to predict what it will undergo at a more distant time.

If, instead of identifying communities at different time-stamps before tracking, we modify the time interval over which the communities are observed, we can better ensure that communities will evolve with fewer time-stamps in which there is no observation. For example, instead of observing the graph at consecutive time-stamps, say we decide to track a community in overlapping time-stamps. We can thus define an advancing window that covers a number of time-stamps. For instance, Figure 2.1(c) illustrates a window of two time-stamps. As time evolves, the window slides from left to right, advancing one time-stamp at each slide. Using this window concept, tracking

the community $C^1_{W_1}$ identified in the first window-stamp W_1 yields a stable evolution till the last window-stamp W_5. Moreover, we can see that the communities are more consistent in terms of nodes over time as compared to those in Figure 2.1(b). Note that defining a window and making it slide by a step of one time-stamp does not change the fact that the graph still evolves by a step of one time-stamp.

While the sliding window concept is useful to track consistent evolving communities, defining the size of window to use remains a challenge. In practical terms, it is possible to set the size of the window manually. However, using such an *ad hoc* window size is hard to justify. One of the goals of this paper is to automatically identify the optimal window size in order to address the aforementioned drawbacks of existing approaches. To this end, we propose a principled approach in which we first define the optimal window size. Once the window is defined, it is then used to identify and track communities over time and detect critical events they may undergo. Then, we model the critical events as temporal equations in order to predict future critical events based on autoregression and survival analysis theories. Our approach utilizes topological features extracted from the evolving communities.

The significance of our work can be summarized as follows:

1. We propose an approach for automatically detecting the size of the window to adopt when identifying and tracking communities over time. The size of the window here is estimated based on the numbers of nodes appearing, disappearing and remaining in the dynamic network at two consecutive, independent time-stamps. The window obtained is then slid over time in order to identify and track evolving communities and detect the critical events they may undergo.
2. Having obtained the critical events, we develop a novel approach for modeling critical events over time for a given dynamic social network. Our approach utilizes both autoregressive modeling and survival analysis theory to investigate the temporal relations among evolving communities. With the temporal relations reflecting the critical events, we will be able to predict not only the next event an evolving community may undergo, but future events more distant in time.
3. We conducted detailed experiments on several real datasets extracted from four social networks: DBLP, Autonomous System (AS), Yelp and AS-Caida.

The results suggest that the proposed approach achieves competitive results in comparison to mainstream classifiers.

2.2 Related Approaches

Much work has been done to address the problem of predicting critical events that evolving communities may undergo in dynamic social networks. The existing approaches generally use a two-step supervised learning process. In the first step, a tracking algorithm is implemented to follow the evolution of a community over different time points in order to discover a sequence of critical events this community may undergo. Specifically, in the first step, the dynamic graph is first represented as a series of static snapshot graphs at different time points. A community detection algorithm is then used to identify community structures at each of these snapshots independently. Finally, a pairwise comparison between the detected community structures is performed to identify the sequence of communities that reflects the evolution of a community along different time-stamps. (Note that the focus of this paper is on modeling and predicting structure changes in the communities. For the problem of tracking community structures over time, interested readers can refer to [20], [48], [28], [29] to obtain a good idea of the state of the art.) In the second step, a supervised approach is used to predict the next event that will occur for a given evolving community. To this end, authors extract several features from the evolving communities to train classifiers according to the different critical events. Then, using the new features extracted from an evolving community, they predict (classify) the next event this community may undergo. Below, we discuss how the identified sequence of communities is used in a supervised process to predict the future events.

In order to predict the next event a community may undergo, existing approaches [21], [53], [32], [54], [55] utilize the sequence reflecting the community's evolution over different time-stamps in order to extract various community-related topological features which are considered to be relevant for the purpose of prediction. These features are then used as attributes in a supervised machine learning process to predict the next critical event. Note that each critical event is initially defined as an independent class and the classifiers are then trained according to these predefined

classes.

In [32], Brodka et al. made use of seven features, including the size of the community and three previous transition event types at four time-stamps. Specifically, given an evolving community observed till time t_n, they recorded as features the following information: (1) the community's size at time t_{n-3}; (2) the event undergone during the transition $t_{n-3} \rightarrow t_{n-2}$; (3) the community's size at time t_{n-2}; (4) the event undergone during the transition $t_{n-2} \rightarrow t_{n-1}$; (5) the community's size at time t_{n-1}; (6) the event undergone during the transition $t_{n-1} \rightarrow t_n$; and (7) the size of the community at time t_n. The extracted features were used as input to train several classifiers (BayesNet, NaiveBayes, KNN, AdaBoost, DecisionTree, RandomForest, etc.) to predict (classify) the next event type that will occur during the transition time $t_n \rightarrow t_{n+1}$, that is *growing*, *continuing*, *shrinking*, *dissolving*, *merging* or *splitting*. The experiments conducted by Brodka et al. in [32] suggest that the RandomForest and DecisionTree classifiers usually provide more accurate results. However, their results are highly influenced by two user-supplied similarity thresholds that are employed to compare communities. The dependence on these two parameters makes proper tuning difficult and limits the applicability of the approach in practice. Moreover, the selected topological feature (community size) may not be sufficient to explain or predict the next transition event.

To address the limitations of the approach proposed by Brodka et al. in [32], Gliwa et al. [53] extended their work by using more topological features (such as *leadership*, *density*, and *cohesion*). Their assumption was that adding more features may increase the prediction accuracy. Moreover, rather than using four previous time observations as in [32], Gliwa et al. [53] instead observed the current time-stamp and the two previous time-stamps of the evolving community to record the features, which were then used to train different classifiers than the ones used in [32]. It is necessary to mention that in the approach reported in [53], to predict the next event an evolving community will undergo, Gliwa et al. simply assigned the dominating event the community may undergo. Note that the dominating event is one of the events assigned for a given community. This event is estimated based on the priority of existing events. In their approach, Gliwa et al. sorted the events in terms of priority (from highest to lowest) as follows: *constancy*, *change size*, *split*, *merge*, *addition*, *deletion*, *split_merge*, *decay*. The

experiments they conducted suggest that the DecisionTree, RandomForest, SimpleCart and DecisionTable classifiers achieved the best results. However, the results are still influenced by the user-defined sorting of event priorities.

It is important to note that the approaches proposed in [32] and [53] take for granted that consecutive observations of the evolving community at previous times must exist. In other words, a community is assumed to evolve in a continuous and consecutive manner: it is expected to be present across all time points, without disappearing at one and reappearing at another. Such a restriction is inconvenient and may be unrealistic, since in dynamic social networks communities may evolve in an interrupted fashion [20], [28]. Furthermore, using many or few features for the purpose of prediction may affect the classification process, due to the presence of highly correlated topological features (in the case of many features) or lack of information (in the case of few features). To address these limitations, the authors in [21] and [55] used a feature selection approach to eliminate the redundant features from the set of features initially identified. Moreover, Takaffoli et al. [21] defined temporal features estimated based primarily on the difference between two topological features at two different time-stamps of a given evolving community. Takaffoli et al. showed that their approach is capable of predicting the most probable next critical event for a community evolving in either a consecutive or a non-consecutive way.

In the above approaches, it can be seen that the authors in [53], [32] extracted only a few features from evolving communities in order to predict the next critical event they might undergo, whereas the authors in [21] extracted more features to achieve the same task. Taking many features into consideration is important to handle the fact that some communities are evolving in a non-consecutive way. However, the problem of defining useful features for predicting future events depends on the networks studied. Moreover, of all of the above approaches [21], [53], [32], [55], only one is applicable to predict critical events at subsequent unobservable time points: i.e., the approach of Ilhan et al [54], in which it is possible to predict at far distant times. Indeed, Ilhan et al. [54] defined an autoregressive model over each extracted feature to formalize its evolution. In their model, they assumed that the current value of a feature is defined as a linear combination of the whole past history of information for this feature. Hence, for a specific evolving community they are capable of predicting

the next feature values using the autoregressive integrated moving average (*ARIMA*) model. However, considering features independently as in [54] may bias the prediction results due to the fact that the extracted features might be highly correlated [21]. Moreover, the autoregressive model used by Ilhan et al. in [54] does not consider the fact that communities may evolve in a non-consecutive way.

In our previous work [36], we attempted to predict critical events not only at the next time point but at more distant ones. In that approach, we made use of survival analysis to model critical events an evolving community may undergo. Specifically, we used the *Cox* model [61] to define a temporal probabilistic function for each of the critical events (*shrink, expand, merge, split* and *stable*). We made the simplistic assumption that the past history of features observed from an evolving critical event is sufficient to predict critical events at further time-stamps. For instance, say we have an evolving community that has been observed from time t_1 to time t_5, and we want to predict whether this community will undergo a critical event at times t_6, t_7, t_8, etc. At first sight, predicting what will happen at time t_6 seems realistic, given that we have recent information from time t_1 to t_5, since the calculated probability is a cumulation of past probabilities that the event will occur. From time t_7 on, the probability of an event occurring will no longer depend on features extracted for the community, since we no longer have an observation of the evolution. At this point, our previous approach supposed that the event we want to forecast is only predicted by a probability density function. However, relying solely on a probability density function to fully calculate the probability of the event's occurring at each more distant time is not enough. In other words, using the conventional *Cox* model in this case will not be sufficient to forecast critical events.

To keep on having feature values at further unobservable time-stamps, we here propose to use an autoregressive model. This autoregressive model is merged with the conventional *Cox* model, in which we will need to re-estimate the parameters at each more distant time for which we want to predict a critical event a community will undergo. In a nutshell, we propose in this paper a *Dynamic Cox* model for predicting critical events that evolving communities may undergo. The model is called *Dynamic Cox* because the features and parameter values are time-dependent, whereas in the conventional *Cox* model, these values are fixed.

2.3 Background & notations

The work reported here utilizes various theories and techniques for (1) tracking communities, (2) estimating feature values by vector autoregression (VAR) and (3) predicting critical events by survival analysis. This section introduces them briefly. For further details, readers may refer to [21], [32], [62], [63].

2.3.1 Background information

Tracking communities can be roughly defined as the task of discovering a sequence of communities that reflects the evolution over time. In other words, the aim is to align communities at different time points in such a way as to represent an evolution. For instance, a sequence $S = \{C_{t_1}, C_{t_2}, C_{t_3}, C_{t_4}\}$ is considered as an evolution of community C_{t_1} if all communities C_{t_i}, $C_{t_{i+1}} \in S$ are similar in terms of nodes (i.e., if they share nodes). Note that in most cases the similarity is given in terms of nodes shared. As mentioned in the Introduction, an evolving community may undergo critical events during its evolution. Understanding how these occur requires an analysis of the past history of the topological features in relation to the critical events. One way to analyze the extracted features over time is to consider them to be continuous and linear [54]. Model-based approaches such as VAR could thus be used to generate feature values, as in Ilhan et al. [54].

VARs are multi-linear autoregressive models in which each vector observation is represented as a combination of previous observations (lag observations). Specifically, considering Y_n to be a random d-dimensional vector observed at time t_n, this vector can be expressed as a linear combination of the p lag vectors Y_{n-1}, Y_{n-2}, ..., Y_{n-p}, as follows:

$$Y_n = \mathcal{E}r_n + \sum_{i=1}^{p} Y_{n-i} M_i \tag{2.1}$$

where M_i is a *d-by-d* matrix representing the coefficients (weights) associated with the lag vectors Y_{n-i} and $\mathcal{E}r_n$ is the additive Gaussian noise with zero mean.

In order to estimate the appropriate features that best fit the autoregressive model given in Eq.(2.1), numerical approaches based on expectation maximization and the Newton-Raphson technique can be used. Analytical approaches such as variational Bayes expectation maximization can also be employed. In our experiment we used the conditional sum of squares maximum likelihood expectation (*CSS-MLE*) method as implemented in [63]. It is worth noting that *VAR* models can be extended to more sophisticated autoregressive models such as the autoregressive moving average (*ARMA*) or the autoregressive integrated moving average (*ARIMA*). Autoregressive models are primarily used for time-dependent data. In our case, however, the data depend not only on time but also on events. To handle such data, we therefore explored the statistical approach known as survival analysis [62], a popular method widely used in medical event prediction [64], which has been adopted in the study of social networks as well [65], [45], [36].

Survival analysis is a statistical method for studying the occurrence and timing of events. Its aim is to estimate, via a probability (generally called the *survivor function* $S()$), the risk of an event's occurring given the past history of a set of time-varying observations. Formally, given the risk or hazard ($\lambda(t)$) of an event's occurring at a specific time t, the survivor function is given as the cumulative risk over time:

$$S(t) = 1 - exp\left(-\int_0^t \lambda(\tau)d\tau\right) \tag{2.2}$$

If we assume that all the time-varying observations follow the same survival function, non-parametric approaches like the Kaplan-Meir estimator can be used, or the empirical distribution can be fitted to a theoretical cumulative distribution such as an Exponential, Weibull or Gamma distribution. In real data, however, the observations generally do not reflect the same survival function. Parametric approaches such as the *Cox* model have therefore been used [62] to extend the formal model given in Eq.(2.2). Here, the survival function is given in a conditional way according to an observation $\mathcal{O}$, as follows:

$$S(t|\mathcal{O}) = 1 - exp\left(-\int_0^t \lambda_0(\tau)\lambda(\tau|\mathcal{O})d\tau\right)$$
$$= 1 - exp\left(-\int_0^t \lambda_0(\tau)exp(\mathcal{F}.\gamma)d\tau\right) \tag{2.3}$$

where λ_0 stands for the baseline or general hazard all observations are subject to; $\mathcal{F}$ is the feature vector, also known as the cofactors extracted from $\mathcal{O}$; and γ is the parameter explaining the contribution of each cofactor to the observed event.

2.3.2 Notation

Let us take a dynamic social network from which individual interrelationships are collected at regular time-stamps for a duration going from t_1 to t_m. At any time t_i $(i = 1, \ldots, m)$, we use the graph structure $g_{t_i} = (V_{t_i}, E_{t_i})$ to represent the snapshot of the social network, where V_{t_i} stands for the set of nodes and E_{t_i} the set of edges. We then use the series $G = \{(V_{t_i},\ E_{t_i}) \mid 1 \leq i \leq m\} = (g_{t_i})_{1 \leq i \leq m}$ to denote the dynamic social network over the whole period $[t_1,\ t_m]$. We use $\mathcal{D} = \{1,\ 2,\ ...,\ m-1\}$ to denote the set of interval durations. For a fixed duration $s \in \mathcal{D}$, we use the notation $W^{s\rightarrow}$ to mean a window W of size s that slides from left to right by a step of one time-stamp. It is worth noting that $W^{s\rightarrow}$ can also be represented as the set $\Big\{\{t_1,\ ...,\ t_s\},\ \{t_2,\ ...,\ t_{s+1}\},\ ...,\ \{t_{m-s+1},\ ...,\ t_m\}\Big\} = \{W_1,\ W_2,\ ...,\ W_{m-s+1}\}$, where each W_j is an instance of $W^{s\rightarrow}$. Hence, the graph instance corresponding to the window instance W_j can be expressed as follows:

$$g_{W_j} = \bigcup_{i=j}^{j+s-1} g_{t_i} \tag{2.4}$$

For each graph g_{W_j}, we define a partition $\Big\{C^1_{W_j},\ C^2_{W_j},\ ...,\ C^{q_j}_{W_j}\Big\}$ representing the communities detected at window-stamp W_j using an existing community detection algorithm. Note that the focus of this paper is to model and predict critical events that

communities may face. Accordingly, any good algorithm for detecting communities could be used. It is important to mention that each detected community $C^l_{W_j}$ has $E^l_{W_j}$ and $V^l_{W_j}$ as its sets of edges and vertices, respectively.

For each detected community C at each window-stamp, we extract a set of n topological features given by the vector $\Upsilon = (\upsilon_1,\ \upsilon_2,\ ...,\ \upsilon_n)$. We adopt the notation S_C to denote the sequence of communities that reflects the evolution of a community C from window-stamp W_j to W_{m-s+1}. In other words, S_C is an evolving community each of whose components C_{W_b}, $j \leq b \leq m-s+1$ represents its instance at the specific window-stamp W_b or time-stamp t_{b+s-1}.

2.4 Proposed model framework

2.4.1 Determining window size.

Given a sliding window $W^{s\rightarrow}$ moving through the interval $[t_1,\ t_m]$, for any window-stamp W_j, the set of nodes observed at this instance are given by $\mathcal{S}_j = \bigcup_{i=j}^{j+s-1}\{V_{t_i}\}$. At the next window-stamp W_{j+1}, we have the set of nodes given by $\mathcal{S}_{j+1} = \bigcup_{i=j+1}^{j+s}\{V_{t_i}\}$. At each step in which a window of size s moves from step j to step $j+1$, we then calculate the numbers of nodes that remain (N^s_r), disappear (N^s_d) and appear (N^s_a) in the graph, as follows:

$$N^s_r(W_j,\ W_{j+1}) = \left|\mathcal{S}_j \bigcap \mathcal{S}_{j+1}\right| \tag{2.5}$$

$$\begin{aligned} N^s_d(W_j,\ W_{j+1}) &= \left|\mathcal{S}_j\ -\ \left(\mathcal{S}_j \bigcap \mathcal{S}_{j+1}\right)\right| \\ &= |\mathcal{S}_j|\ -\ N_r(W_j,\ W_{j+1}) \end{aligned} \tag{2.6}$$

$$N_a^s(W_j,\, W_{j+1}) = \Big|\mathcal{S}_{j+1} \,-\, \Big(\mathcal{S}_j \bigcap \mathcal{S}_{j+1}\Big)\Big| \\ = |\mathcal{S}_{j+1}| \,-\, N_r(W_j,\, W_{j+1}) \tag{2.7}$$

From the numbers of nodes that remain, disappear and appear (N_r^s, N_d^s, N_a^s), we can get an idea of how the social network fluctuates in terms of nodes over time. Note that the larger the value of N_r^s, the more the network tends to be static, which may make it impossible to capture changes such as *merge*, *split*, *shrink* and *expand* that evolving communities may undergo. In the same way, the smaller the value of N_r^s, the more the network changes over time, which may result in several cases where communities are evolving in a non-consecutive way. Note that the number of nodes remaining in the graph over time, N_r^s, increases according to the size of the window. This size must therefore be chosen to result neither in a quasi-stable graph nor in a situation where the graph is highly changeable over time. To this end, we first calculate the *fluctuation* fl_s of the graph given a size s of the sliding window, as follows:

$$fl_s(W_j,\, W_{j+1}) = \frac{N_a^s(W_j,\, W_{j+1}) \,\times\, N_d^s(W_j,\, W_{j+1})}{N_r^s(W_j,\, W_{j+1})^2} \\ = 1 \,-\, \frac{|\mathcal{S}_j| \,+\, |\mathcal{S}_{j+1}|}{N_r^s(W_j,\, W_{j+1})} \,+\, \frac{|\mathcal{S}_j| \,.\, |\mathcal{S}_{j+1}|}{N_r^s(W_j,\, W_{j+1})^2} \tag{2.8}$$

Note that if the network is static, i.e., when $N_r^s = |\mathcal{S}_j| = |\mathcal{S}_{j+1}|$, the fluctuation is 0. On the other hand, when the graph is highly changing, i.e., $N_r^s = |\mathcal{S}_j \bigcap \mathcal{S}_{j+1}| = \tilde{N} \ll |\mathcal{S}_j|, |\mathcal{S}_{j+1}|$, where $\tilde{N}$ is the minimum number of nodes remaining in the network, the fluctuation is $\frac{|\mathcal{S}_j|.|\mathcal{S}_{j+1}|}{\tilde{N}}$. Hence, we can conclude that the *fluctuation* is always bounded in $[0,\, \frac{|\mathcal{S}_j|.|\mathcal{S}_{j+1}|}{\tilde{N}}]$.

In our work, to study the various changes the evolving communities are undergoing, we selected the minimum window size $\hat{s}$ such that the *fluctuation* fl_s is bounded within [0, 1] with a deviation lower than a small value ϵ (in our implementation lower

Algorithm 1: Determining Window size.

```
Data: G ;                          /* The graph at different timestamp. */
Result: Size ;                     /* The window's size. */
begin
    for s ∈ 𝒟 do
        Size ← s;
        ℱl ← {} ;    /* Initialize the set of fluctuation's score. */
        for W_j, W_{j+1} ∈ W^{s→} do
            - Resize the graphs g_{W_j}, g_{W_{j+1}} ;
            - Calculate the fluctuation using Eq. (2.8) ;
            - ℱl ← ℱl ⋃ {fl_s} ;
        - Dev ← σ²(ℱl) ;           /* Calculate the variation. */
        if Dev < 10^{-2} and ∀ fl_s ∈ ℱl, 0 ≤ fl_s ≤ 1 then
            Break ;                /* Stop the algorithm. */
    Return Size ;
```

than 10^{-2}) [1], as follows:

$$\hat{s} = \min_{s \in \mathcal{D}} \left\{ fl_s \leq 1,\ \sigma^2(\mathcal{F}l) < \epsilon \right\} \tag{2.9}$$

where $\mathcal{F}l$ is the set of *fluctuations* given a sliding window $W^{s\rightarrow}$ and $\sigma(\mathcal{F}l)$ its standard deviation.

By using an incremental process we can easily approximate the optimal window size $\hat{s}$. Algorithm 1 can be run to estimate the minimal window size.

Having determined the window size which may best express how the graph evolves over time, we now identify communities, track the communities over time and detect the changes they may undergo following the defined window-stamp. We will now present a detailed approach for modeling and predicting the critical events the evolving

1. The reader should be aware, however, that the choice of this smallest value is not limited to 10^{-2} and the user can set other smallest values near zero (that is, the lower bound of fl_s). However, setting values much less than 10^{-2} can unnecessarily increase the execution time. Hence, we suggest using 10^{-2} as a default value.

communities may undergo.

2.4.2 Modeling critical events

Let $Cr_e = \{Split,\ Merge,\ Shrink,\ Expand,\ Stable\}$ be the set of critical events[2] an evolving community may undergo. We assume that these events are mutually independent, which means the probability that a community may pass through one event is not affected by another event. In other words, the probability that a community may undergo an event can be evaluated independently of other events. Hence, for a critical event $e \in Cr_e$, in observing a community evolving over time, we will either see this event occurring or we will not. The two possible responses will then be recorded when observing evolving communities: either the event e occurs (codified as 1) or it does not (codified as 0). Therefore, for the critical event e, we can represent whether a community underwent this event at any given time as a binary counting process. We generalize this process by modeling the instantaneous risk that a community may undergo an event e, that is, the *hazard rate* [62] of a community's facing event e, using its topological features as covariates.

Hazard function

Given an event e and a sliding window $W^{s\rightarrow}$, for each window-stamp $W \in W^{s\rightarrow}$, we calculate the number of times $N_e(W)$ the event e has occurred, which corresponds to the number of evolving communities that passed through the critical event e during the time transition from one window-stamp to the next. The function $N_e(W)$ represents the counting process for event e. We assume that the count of the number of events (N_e) is affected at each window-stamp by a harmful function M_e. The counting process can thus be decomposed as follows:

$$N_e(W) = \Lambda_e(W) + M_e(W) \tag{2.10}$$

2. Several authors have defined and developed approaches for automatically detecting the critical events. Interested readers may refer to [20], [48], [28], [29], [22] for descriptions of some of these approaches.

where $\Lambda_e(W)$ is a non-decreasing predictable process, called the *cumulative intensity process*, and $M_e(W)$ is a mean zero martingale [62].

Considering $\Lambda_e(W)$ to be continuous, there exists a predictable non-negative intensity process $\lambda_e(W)$ such that

$$\Lambda_e(W) = \int_{W_1}^{W} \lambda_e(\omega) d\omega \tag{2.11}$$

Given an evolving community S_C, at each window-stamp, this community may or may not be observed. Hence, the intensity of this community with regard to the event e at any window-stamp W is

$$\lambda_e(W|S_C) = Y_e(W|S_C) h_e(W|S_C) \tag{2.12}$$

where $Y_e(W|S_C)$ takes the value 1 if S_C has an instance at window-stamp W and 0 otherwise. $h_e(W|S_C)$ is the intensity or the hazard rate for evolving community S_C undergoing event e at window-stamp W.

As demonstrated by previous work [21], [53], [32], community-based topological features can indicate the critical events a community may undergo. Because correlations between topological features are possible [21], we apply principal component analysis (PCA) on the extracted topological features [36]. From the PCA result, we select the d first ($d \leq n_c$) principal components to represent the new d-dimensional space $\mathcal{E}$. Note that the value of d corresponds to the ranking of the knee-point of the curve of the sorted eigenvalues.

After constructing the new space $\mathcal{E}$, we now represent the features extracted from a given community $C_{W_b} \in S_C$ by the vector $X_{W_b} = (x_{1.b}, x_{2.b}, ..., x_{d.b})$. The vector X_{W_b} corresponds to the normalized projection of the topological features at window-stamp W_b. Hence, we can gather the set of features of a community to describe its risk of undergoing a critical event at a specific window-stamp W_b (i.e time-stamp t_{b+s-1}). To

this end, we use the *Cox* regression model, defined as follows:

$$
\begin{aligned}
h_e(W_b|S_C) &= h_e(W_b|X_{W_b}, \Gamma^e_{W_b}) \\
&= \begin{cases} h_e(W_b) exp(X^{\star}_{W_b}.\Gamma^e_{W_b}) & if \quad \exists C_{W_b} \in S_C \\ h_e(W_b) & otherwise \end{cases}
\end{aligned} \tag{2.13}
$$

where $X^{\star}_{W_b}$ represents the transpose of the vector X_{W_b}, $\Gamma^e_{W_b} = (\gamma_{1,b}, ..., \gamma_{d,b})^e$ is the parameter representing the contribution of each covariate at window-stamp W_b to the event e which will probably occur and $h_e(W_b)$ the generalized hazard of event e occurring.

Probability of an event occurring

The intensity process defined in Eq. (2.12) gives the risk that a community will pass through an event at a given time point. From this, we can then calculate, at any time point, the probability that a community will undergo an event by computing the cumulative probability $\mathcal{S}_e(W_b|S_C)$ [62], as follows:

$$
\mathcal{S}_e(W_b|S_C) = 1 - exp\Big\{ - \int_{W_1}^{W_b} Y_e(\omega|S_C) h_e(\omega|S_C) d\omega \Big\} \tag{2.14}
$$

Note that the calculation of risk of occurrence at a specific time given in Eq. (2.13) takes into consideration that the current time t_{b+s-1} is observable. To extend Eq. (2.13) to allow us to calculate the risk at a more distant unobservable time, we need to predict the feature values at that unobservable time. For this, we assume that the compositional data vector X_{W_b} can be written as a linear combination of the p lag compositional vectors (details on the model are given in the next sub-subsection).

$$
X_{W_b} = \mathcal{E} r_b + \sum_{a=1}^{p} X_{W_{b-a}} \Omega_a \tag{2.15}
$$

where Ω_a a *d-by-d* is a matrix representing the coefficients associated with the lag vectors $X_{w_{b-a}}$ and $\mathcal{E}r_b$ is the additive Gaussian noise with zero mean and covariance SD. In our case study we have used the Python package developed in [63] to estimate these parameters (Ω_a, SD), which we call autoregressive parameters. Below, we present the method used to estimate the *Cox* parameters.

2.4.3 Estimation of the *Cox* parameters

Given a set of evolving communities E_C, we can formalize the risk of an event e occurring at a specific time as

$$\begin{aligned}\mathcal{R}_e(W_b) &= \sum_{S_C \in E_C} \lambda_e(W_b|S_C) \\ &= \sum_{S_C \in E_C} Y_e(W_b|S_C) h_e(W_b|S_C)\end{aligned} \tag{2.16}$$

Partial log-likelihood

Suppose that we observe communities from W_1 to some W_b which is not the last window-stamp. Clearly, from W_1 to W_b we do not have a total view of all communities, which implies that the notion of likelihood cannot be explicit. Due to this constraint, we instead define the partial likelihood of the parameters $\bar{\Gamma}^e_W = \left(\gamma^e_{W_1}, ..., \gamma^e_{W_b}\right)$ as

$$\begin{aligned}\mathcal{P}(\bar{\Gamma}^e_W) &= \prod_{S_C \in E_C} \prod_{i=1}^{b} \left[\frac{h_e(W_i|S_C)}{\mathcal{R}_e(W_i)}\right]^{\delta_{W_i}} \\ &= \prod_{S_C \in E_C} \prod_{i=1}^{b} \left[\frac{exp(X^\star_{W_i}\Gamma^e_{W_i})}{\sum_{S_C} Y_e(W_i|S_C) exp(X^\star_{W_i}\Gamma^e_{W_i})}\right]^{\delta_{W_i}}\end{aligned} \tag{2.17}$$

where δ_{W_i} is a binary indicator which takes value 1 when community S_C has an instance

at window-stamp W_i and 0 otherwise. Taking the logarithm of the last equation, we obtain the partial log-likelihood expanded as follows:

$$\mathcal{L}og_p(\bar{\Gamma}^e_W) = \sum_{S_C}\sum_{i=1}^{b}\delta_{W_i}\Bigg\{X^{\star}_{W_i}\Gamma^e_{W_i} - \mathcal{L}og\Big[\sum_{S_C} Y_e(W_i|S_C)exp(X^{\star}_{W_i}\Gamma^e_{W_i})\Big]\Bigg\} \tag{2.18}$$

Newton-Raphson approximation

Having obtained the partial log-likelihood, we can approximate the parameter Γ^e_W using the Newton-Raphson algorithm by solving the following recursive equation:

$$(\bar{\Gamma}^e_W)^{(it+1)} = \left(\bar{\Gamma}^e_W\right)^{(it)} - \mathcal{L}og_p^{''}\left((\bar{\Gamma}^e_W)^{(it)}\right)\mathcal{L}og_p^{'}\left((\bar{\Gamma}^e_W)^{(it)}\right) \tag{2.19}$$

where (it) denotes the current iteration step, $\mathcal{L}og_p^{''}(.)$ corresponds to the second derivative (Hessian) of the partial log-likelihood, and $\mathcal{L}og_p^{'}(.)$ to the first derivative (gradient) of the partial log-likelihood. It is worth noting that we used the Taylor expansion to reduce the calculation cost. Therefore, when taking an expansion of order 2 at $\Gamma = 0$, we obtain the gradient and the Hessian, given as follows:

$$\mathcal{L}og_p^{'}(\Gamma^e_{W_i}) = \sum_{S_C}\sum_{i=1}^{b}\delta_{W_i}\Bigg\{X^{\star}_{W_i} - \frac{\sum_{S_C} Y_e(W_i|S_C)X^{\star}_{W_i}}{\sum_{S_C} Y_e(W_i|S_C)} - \left(\frac{\sum_{S_C} Y_e(W_i|S_C)X^2_{W_i}}{\sum_{S_C} Y_e(W_i|S_C)}\right)\Gamma^e_{W_i} + \left(\frac{\sum_{S_C} Y_e(W_i|S_C)X^{\star}_{W_i}}{\sum_{S_C} Y_e(W_i|S_C)}\right)^2\Gamma^e_{W_i}\Bigg\} \tag{2.20}$$

$$\mathcal{L}og_p^{''}(\Gamma_{W_i}^e) = \sum_{S_C}\sum_{i=1}^{b}\delta_{W_i}\left\{\left(\frac{\sum_{S_C} Y_e(W_i|S_C)X_{W_i}^{\star}}{\sum_{S_C} Y_e(W_i|S_C)}\right)^2 - \left(\frac{\sum_{S_C} Y_e(W_i|S_C)X_{W_i}^{2}}{\sum_{S_C} Y_e(W_i|S_C)}\right)\right\} \quad (2.21)$$

Note that the iterative formula of the Newton-Raphson algorithm, as expressed by Eq.(2.19), requires starting values for $(\bar{\Gamma}_W^e)^{(0)}$. In our implementation, we used the K-means algorithm to define these initial values. The Newton-Raphson algorithm converges, as our estimates of $(\bar{\Gamma}_W^e)^{(0)}$ change by less than a small positive value ϵ with each successive iteration, to $\bar{\Gamma}_W^e$. Moreover, since the projected features used are independent of the event, we note that the calculation of the estimated parameters is independent of the type of event used.

2.4.4 Predicting an event

Once the parameters of our model have been estimated and the hazard function identified, we can calculate the probability that an evolving community will undergo a critical event at any time (observable or not) by applying Eq.(2.14). However, at each more distant unobservable time, all the parameters (both autoregressive and *Cox* parameters) need to be re-estimated in order to calculate the new probability value. At any unobservable time, we thus predict that a given community will undergo an event e if its probability (Eq.(2.14)) of occurrence has the highest value compared to the probability of occurrence of the other events. Hence, assuming that we have observed the evolution of communities from t_1 to t_{b+s-1}, which corresponds to the set of observable window-stamps $O_W = \{W_1, ..., W_b\}$, at later unobservable times corresponding to the set of unobservable window-stamps $\tilde{O}_W = \{W_{b+1}, W_{b+2}, ..., W_{b+k}, ...\}$, we can run Algorithm 2 to predict critical events at more distant times.

Algorithm 2: Predicting critical events

Data: E_C, Cr_e, O_W, $\tilde{O}_W$; /* Sets of evolving communities, critical events, observable and non-observable window-stamps. */

Result: Pr_C ; /* Set of predicted events. */

begin

1- Initialization:

begin

- $M \leftarrow O_W$; /* First period used to estimate the parameters */
- $Pr_C \leftarrow \{\}$;
- $Par \leftarrow \{\}$; /* Empty set of parameters */

2- Estimating parameters

begin

-(a) Calculate the VAR parameters;

-(b) Estimate the Cox parameters at the period M for each events using Eq.(2.19);

-(c) $Par \leftarrow Par \bigcup \Big\{ \{(a),(b)\} \Big\}$;

3- Updating & Predicting

begin

begin

for $S_{C^l_{W_j}} \in E_C$ **do**

for $W \in \tilde{O}_W$ **do**

- $M \leftarrow M \bigcup \{W\}$; /* Update period */
- Calculate the community's features using Eq.(2.15) at W ; /* Update features values */
- Run step 2 to update parameters;

begin

for $W \in \tilde{O}_W$ **do**

for $S_{C^l_{W_j}} \in E_C$ **do**

for $e \in Cr_e$ **do**

- calculate the probability S_e using Eq.(2.14);

- $Pr_C \leftarrow Pr_C \bigcup max\{S_e, e | \, e \in Cr_e\}$; /* select the event having highest S_e */

To summarize, we can run the steps below to implement the overall proposed approach:

1. Calculate the size of the window to use and redefine the network by running Algorithm 1;
2. At each window-stamp, split the instance of the network into a partition to obtain the communities;
3. Extract the topological features of each community identified in 2);
4. Project the features obtained in 3) in a new d-dimensional space using the PCA method;
5. Using the model proposed in [20], track the communities and identify the critical events they may undergo;
6. For the evolving communities obtained in 5), for each event e, estimate the corresponding hazard function;
7. Predict the critical events by running Algorithm 2.

2.5 Experiments

To evaluate our proposed approach, we first determined the appropriate window size to use, as discussed in subsection 2.4.1. After obtaining the appropriate window size for each of our networks, we detected the communities and tracked them over time in terms of nodes and the extracted features (presented later in subsection 2.5.1). Finally, to predict the critical events, we first detected these critical events and modeled them as discussed in subsection 2.4.2. Then, we predicted the events and compared the accuracy of the predicted results with some classifiers. Before presenting the evaluation, we will begin by describing the data under investigation.

We tested the proposed approach on four real datasets: the seventh version of the DBLP dataset [3] [66], the Autonomous Systems (AS) dataset [4] [8], the YELP dataset [5] and finally the AS-Caida dataset [6] [8].

3. http://arnetminer.org/citation
4. http://snap.stanford.edu/data/as.html
5. https://www.yelp.ca/academic_dataset
6. http://www.caida.org/data/active/as-relationships/

Table 2.1 – Dataset description.

Network	#Avg_n	#Avg_e	#snaps
AS-Caida	4,773	9,164	36
AS	3,682	7,621	57
Yelp	8,211	10,548	49
DBLP	40,058	73,229	21

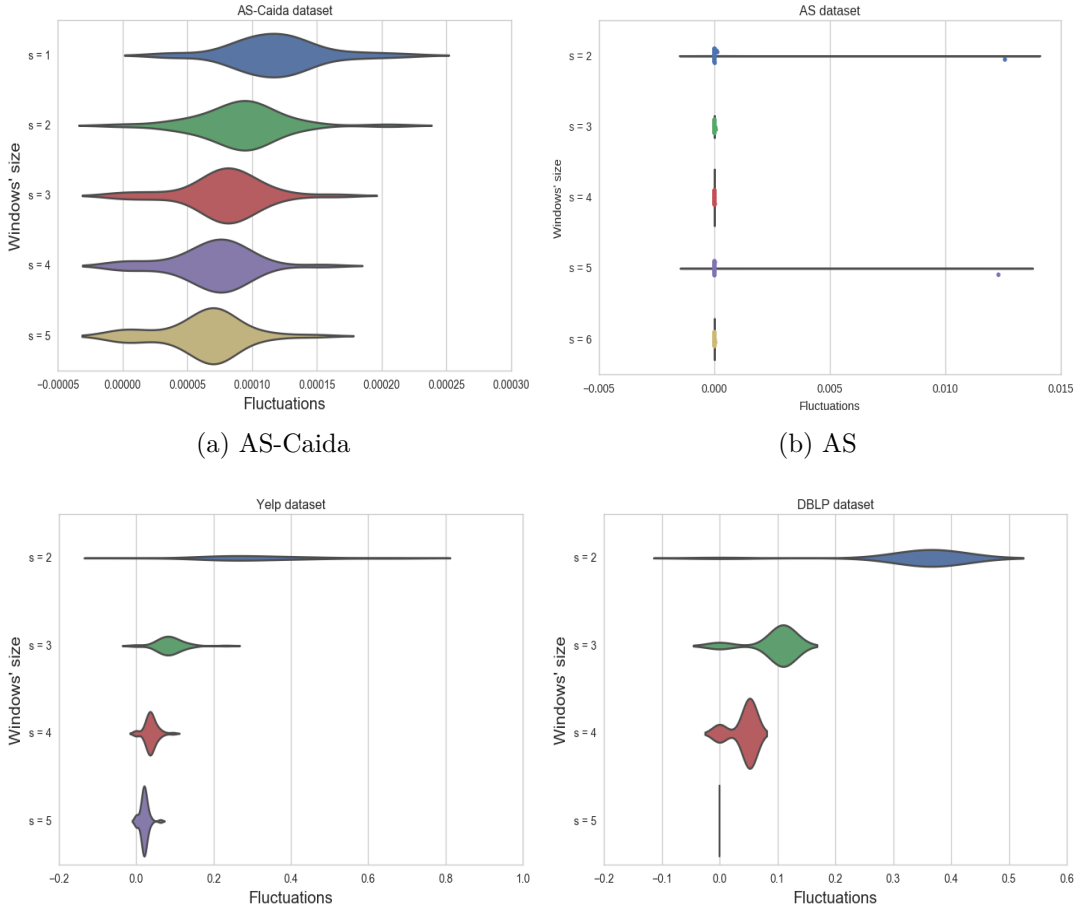
1) AS-Caida, like the AS dataset, contains Internet communications. However, AS-Caida contains two types of collaboration: provider-to-customer and provider-to-provider. We focused on the provider-to-customer collaboration, over which we created an undirected network of who-follows-whom on a monthly basis from January 2007 to December 2009.

2) The AS dataset contains the daily communication network of who-talks-to-whom from the Border Gateway Protocol logs. We built graphs on communication networks on a daily basis from 30 December 1998 to 27 February 1999, where each identifier is considered as a node and each relation between two identifiers is taken as an edge.

3) In the YELP dataset, there are three main objects, “Business", “Review" and “User", giving information on businesses reviewed by users. We focused on the “User" object: for each user having friend(s), we created edge(s) between this user and his or her friend(s). Graphs were constructed on a monthly basis from July 2010 to July 2014.

4) The DBLP dataset contains the co-publications of authors. For each paper published, it gives the paper title, the authors, the year, the publication venue, the index identification of the paper and the identifications of references to the paper. We built graphs of co-authorship from year 1995 to 2015, taking each year as a snapshot. Authors are represented by nodes, and co-authorships by edges.

Table 2.1 contains the descriptions of the datasets. The first column lists the different networks used in our investigation, the second gives the average number of nodes, the third, the average number of edges, per snapshot. The fourth column gives the number of snapshots per social network.



(a) AS-Caida

(b) AS

(c) YELP

(d) DBLP

Figure 2.2 – Violin plots of the variation in fluctuation value. The longer the violin, the sparser the fluctuation; the smaller the violin, the denser the fluctuation.

2.5.1 Window size estimation

To estimate the appropriate window size, we calculated the fluctuation as discussed in subsection 2.4.1. When running Algorithm 1: *Detecting Window size*, as depicted in Figure 2.2, we discovered in the cases of AS and DBLP that starting from a window of size $s = 3$ yields fluctuation values with very low deviations. In contrast, in the case of Yelp, we have low fluctuation starting from a window of size $s = 2$. In AS-Caida we instead discovered that, starting from a window of size $s = 1$, the network varies but with low fluctuation. Table 2.2 provides a summary of the extracted communities per

Table 2.2 – Network characteristics after defining the window size to use.

Network	#Avg_n	#Avg_e	#Avg_C	Size	#W-Stamp
AS-Caida	4,773	9,164	1,401	1	36
AS	3,810	23,090	645	3	55
Yelp	33,843	74,651	162	2	45
DBLP	104,295	233,406	340	3	19

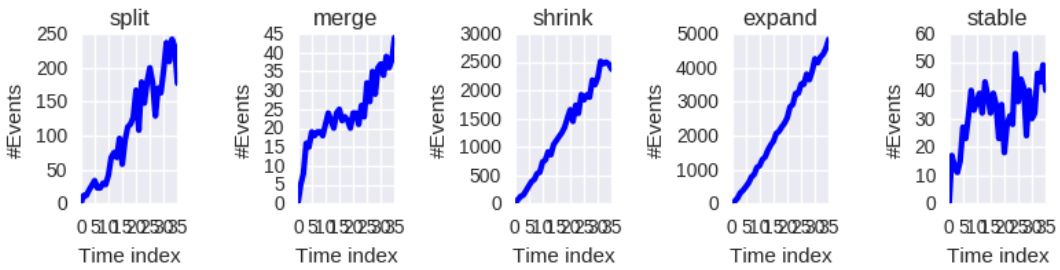
window-stamp when using the suggested window size. The second column (#Avg_n) gives the average number of nodes, the third (#Avg_e) the average number of edges, the fourth (#Avg_C) the average number of communities, the fifth the size of the window-stamp and finally, the last column gives the number of window-stamps. With the size of window defined, we can now extract and track communities over time.

Several methods exist for identifying communities in social networks. In the work described here, we used the Infomap [25] algorithm because it is an effective, parameter-free approach for detecting communities, as Lancichinetti et al. demonstrated in [67]. Tracking of communities over datasets was done using the approach in [20].

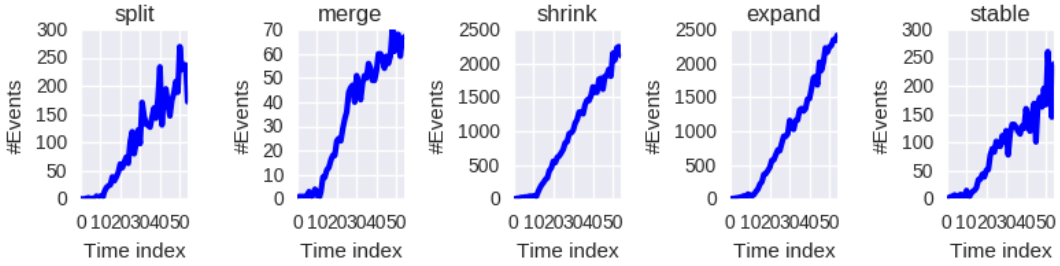
Having determined the appropriate window size to use in each of our cases, we can now proceed to calculate the probability of an evolving community undergoing a critical event. For this, we first need to determine the general hazard function and then estimate the parameters of the model.

2.5.2 Estimation of the general hazard

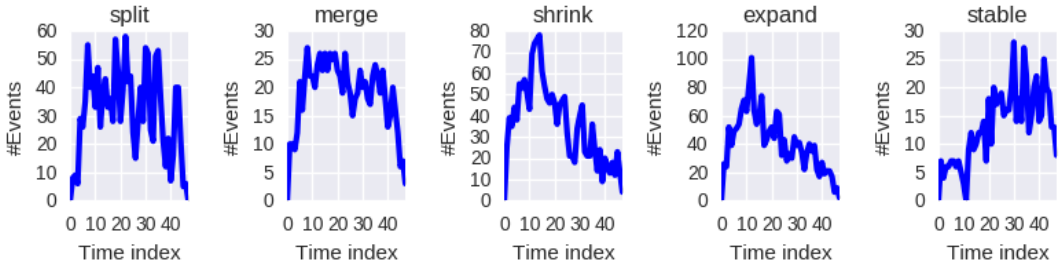
In order to estimate the general hazard, we first proceed by a counting process which consists of enumerating the number of events per time-stamp observed. For instance, Figure 2.3 gives the number of events occurring in the networks under investigation. From the communities we have selected, we can see that in most of the networks, the number of events occurring increases over time. However, this observation does not hold for the Yelp network Figure 2.3(c), where the time-event curve for all critical events presents a convex trend.



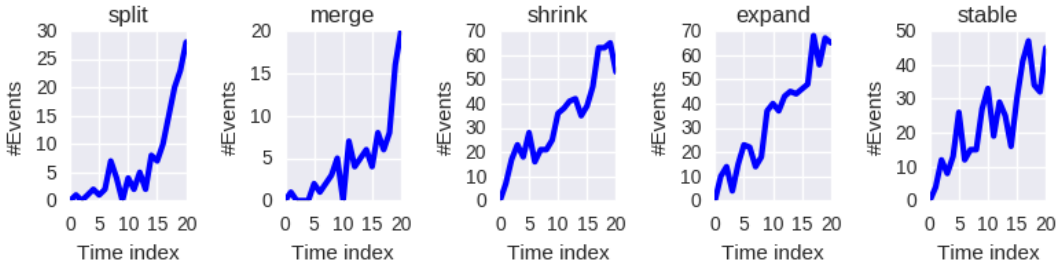
(a) AS-Caida



(b) AS



(c) Yelp



(d) DBLP

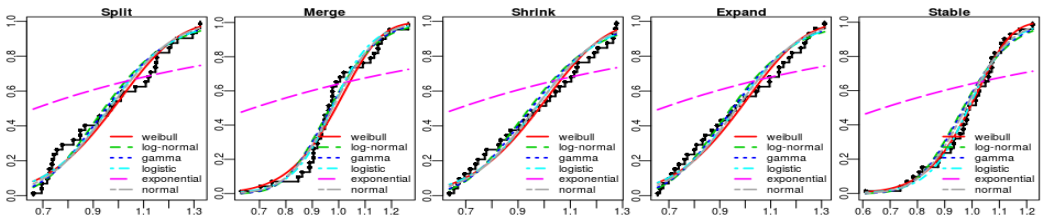
Figure 2.3 – The number of events (*split, merge, shrink, expand* and *stable*) per time-stamp in the AS-Caida (*a*), AS (*b*), Yelp (*c*) and DBLP (*d*) networks. It can be seen that the number of events increases with time in all networks except Yelp (*c*), where the number of events over time presents a convex trend.

Having determined the number of events over time, we assume that the corresponding hazard function can be modeled by a known statistical distribution. We therefore fitted the cumulative time variation (empirical distribution) of the number of events to several theoretical cumulative density functions (CDFs): Weibull (W), log-normal (LN), gamma (G), logistic (L), exponential (E) and normal (N). Then, to select the best distribution, we compared each of the theoretical distributions to the empirical distributions using the Cramer-von Mises (CVM) [68], Anderson-Darling (AD) [69] and Kolmogorov-Smirnov (KS) [70] tests. We selected as hazard function the optimal average goodness of fit (averaged over the three tests KS, CVM and AD) for each critical event.

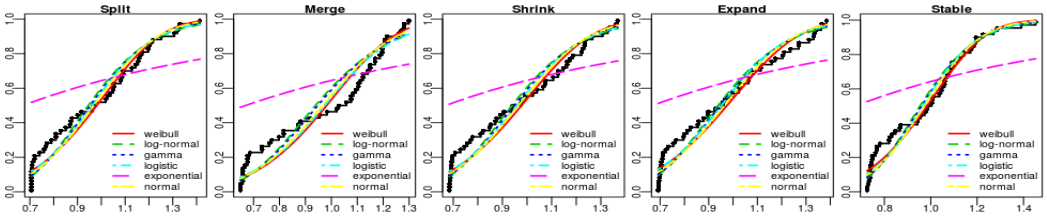
As depicted in Figure 2.4, the experimental CDFs (cumulative numbers of events over time) obtained show an increasing trend. Such an evolution suggests that the empirical distributions can be uniquely decomposed into two distributions [62], one being a uniform distribution and the other a martingale (as specified in Eq.(2.10)). Hence, we can assume that the uniform distribution corresponds to the theoretical distribution that best fits the cumulative empirical distribution from which the hazard will be deduced.

Plotting theoretical cumulative density functions with the empirical distributions as in Figure 2.4, it can be clearly seen that most of the theoretical distributions fit the experimental distributions well. However, calculating the goodness of fit (average over the three tests KS, CVM and AD) for each critical event yields a more detailed picture. In Table 2.3, the optimal value on each row, highlighted in bold, corresponds to the best distribution that should be selected for each event.

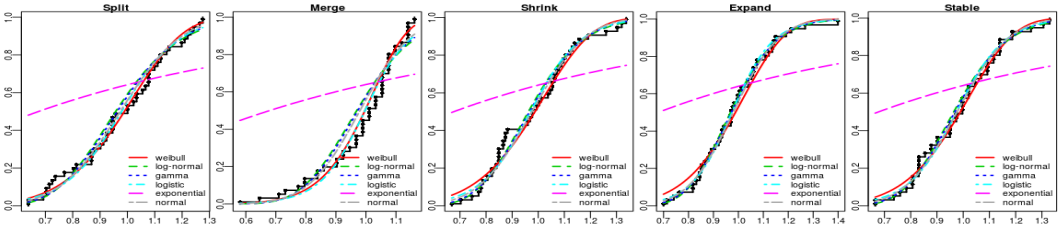
We can see that in the AS-Caida network (Table 2.3(a)), the Weibull distribution (W) fits the best for all the critical events except *merge*, where the logistic distribution (L) gives the best fit. These results show that one can select the Weibull distribution to define the general hazard for the *split, shrink, expand* and *stable* events, whereas the logistic distribution can be selected to define the general hazard of the merge event. Similarly, in the AS network (Table 2.3(b)), the Weibull distribution can be selected as general hazard function for the critical events *split, merge, shrink* and *stable* while, in the case of the *expand* event, the gamma distribution can be selected as hazard function. In the Yelp case (Table 2.3(c)), the Weibull distribution can be



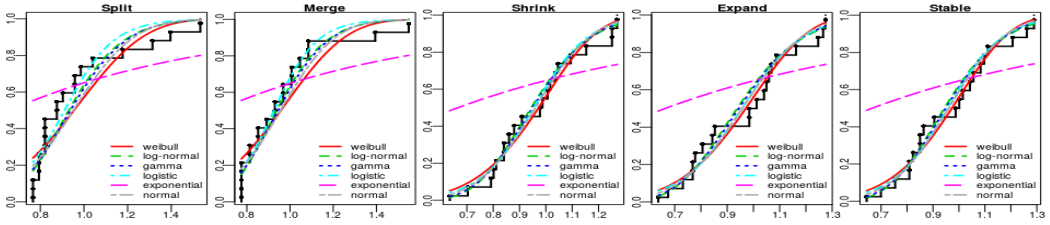
(a) AS-Caida



(b) AS



(c) Yelp



(d) DBLP

Figure 2.4 – Cumulative number of events (experimental cumulative density functions (CDFs)) over time in AS-Caida (a), AS (b), Yelp (c) and DBLP (d) networks against Weibull, Log-Normal, Gamma, Logistic, Exponential and Normal CDFs (theoretical CDFs).

Table 2.3 – Goodness of fit (experimental distributions against theoretical distributions) in the AS-Caida (a), AS (b), Yelp (c) and DBLP (d) networks. Boldface indicates the optimal fitting score.

(a) AS-Caida

	W	LN	G	L	E	N
Split	**0.344**	0.387	0.375	0.368	4.338	0.369
Merge	0.435	0.350	0.330	**0.296**	5.230	0.327
Shrink	**0.240**	0.285	0.264	0.253	4.306	0.250
Expand	**0.267**	0.278	0.269	0.278	4.344	0.271
Stable	**0.083**	0.365	0.308	0.155	5.239	0.212

(b) AS

	W	LN	G	L	E	N
Split	**0.513**	0.612	0.587	0.565	6.467	0.566
Merge	**0.867**	1.097	1.034	0.872	6.204	0.928
Shrink	**0.583**	0.660	0.638	0.609	6.200	0.629
Expand	0.604	0.586	**0.583**	0.584	6.204	0.619
Stable	**0.446**	0.597	0.551	0.517	6.852	0.500

(c) Yelp

	W	LN	G	L	E	N
Split	**0.147**	0.314	0.261	0.195	6.126	0.188
Merge	**0.327**	0.899	0.801	0.446	6.870	0.620
Shrink	0.299	**0.223**	0.224	0.272	6.394	0.246
Expand	0.256	0.071	**0.064**	0.078	6.734	0.085
Stable	0.173	0.162	0.146	0.167	6.469	**0.139**

(d) DBLP

	W	LN	G	L	E	N
Split	0.795	**0.595**	0.648	0.606	2.778	0.761
Merge	0.673	0.351	0.402	**0.344**	2.869	0.527
Shrink	0.188	0.120	**0.119**	0.145	2.795	0.147
Expand	**0.214**	0.243	0.229	0.227	2.665	0.218
Stable	0.167	0.144	**0.141**	0.165	2.776	0.153

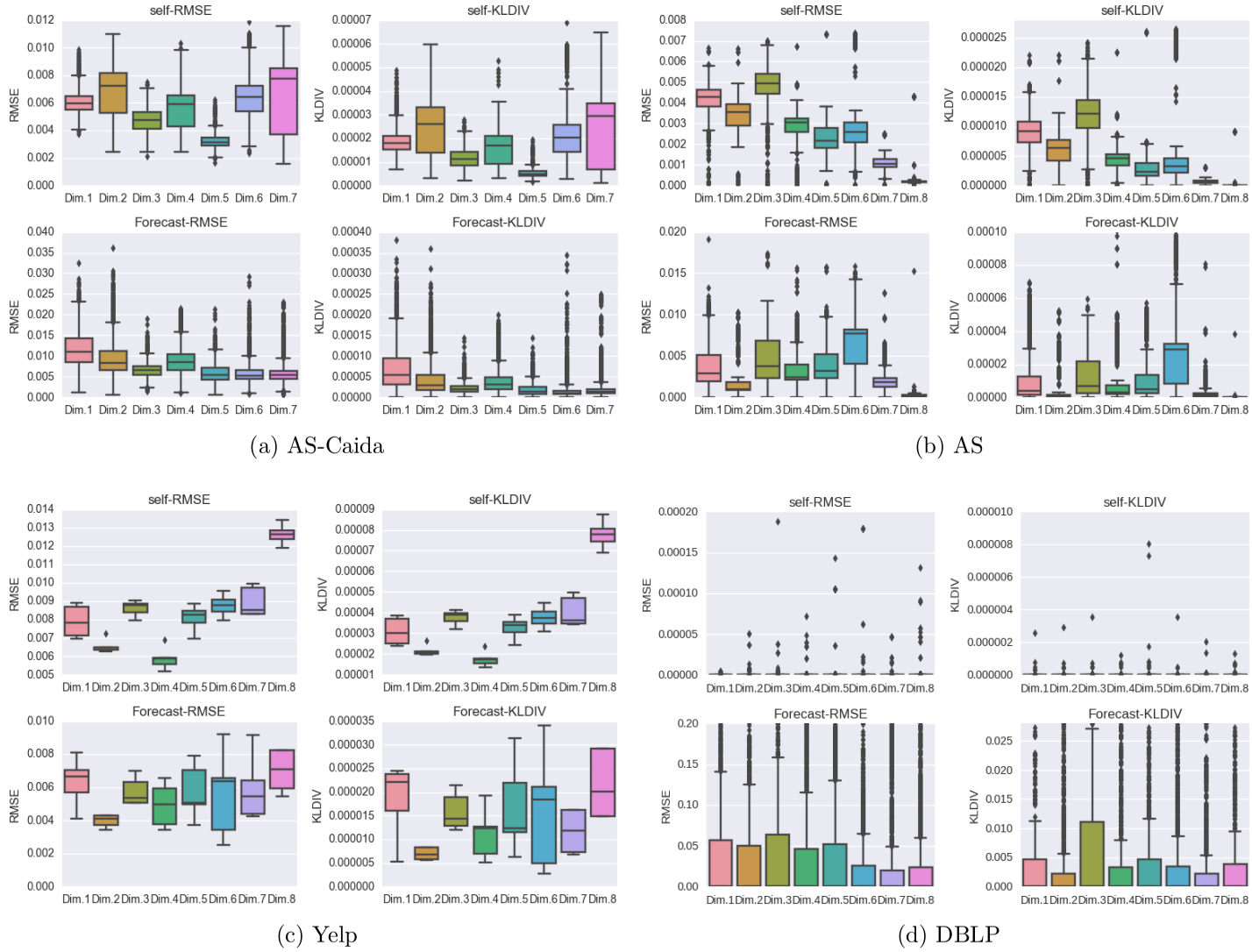
used as hazard function for the critical events *split* and *merge*, whereas the log-normal corresponds best for the *shrink*, *expand* and *stable* events. Finally, in the DBLP case (Table 2.3(d)), the Weibull distribution fits the best for the *expand* event, whereas for the opposite event *shrink*, the best fit is given by the gamma distribution, and for the *split* and *merge* events, respectively, the log-normal and logistic distributions can be chosen as general hazards.

2.5.3 Predicting feature values

Features used

Predicting critical events requires knowledge regarding the evolution of the topological features extracted from the communities [21], [52], [54], [55]. In our case studies we extracted the following features:

(1) The number of edges in the community and **(2) the number of nodes in the community** are self-explanatory; **(3) the average shortest path** provides general information about how far apart the nodes are in the community; **(4) the average closeness**, contrary to the average shortest path, gives general information about how close together the nodes are in the community; **(5) the average intra-degree** provides information on how the community is internally connected; whereas **(6) the average inter-degree** describes how the community is externally connected; **(7) the conductance**, introduced by [71], gives a global idea of how the nodes within the community are connected in the network; whereas **(8) the density** gives a local idea of how the nodes within the community are connected independently of the other communities in the network; **(9) the diameter** provides information about the longest path in the community; **(10) the attractive force** [72] indicates how capable the community is of attracting nodes to join its member nodes; and **(11) the transitivity** indicates the average number of cliques in the community.



(a) AS-Caida

(b) AS

(c) Yelp

(d) DBLP

Figure 2.5 – The accuracy of the model in forecasting feature values in different networks. In the first column of each figure ((*a*)-(*d*)) we have the root mean square error and the Kullback divergence of the model in recovering the values within the training intervals. In the second column we have the root mean square error and the Kullback divergence. The Dim.1-Dim.7 (in (*a*), (*b*) and (*d*)) and Dim.1-Dim.8 (in (*c*)) values correspond to the selected components when implementing *PCA* in different networks.

It should be noted that we used *PCA* to project our selected features into a new space in order to avoid handling correlated features. In our implementation of *PCA*, we selected $d = 7$ components for AS-Caida and $d = 8$ components for the other datasets to represent the new-space dimensions. The selected d in each network corresponds to the ranking of the knee-point of the curve of the sorted eigenvalues when implementing *PCA*.

Accuracy of the *VAR*-model

We used the data projected in the new space to predict the feature values. For this, in all datasets, we decided to train each of the projected time-varying vectors over the full time duration except for the last five time-stamps. We used the last five time-stamps to evaluate the accuracy of the autoregressive model in predicting feature values. It is important to note that we first filtered the data observations using the Hodrick-Prescott filter (*HP-filter*) [63], setting the *roughness parameter* (ρ) and the lag parameter (p) [7] for each network. The reason we chose the *HP-filter* is simply because it permits easy recovery of the initial observation with very low error.

We used two criteria to ascertain the accuracy of the selected *VAR* model. The first criterion we chose is the root mean square error (*RMSE*) and the second is a distance based on the Kullback-Leibler divergence (*KLDIV*). The latter metric has been used previously by Lefort et al. [73] to calculate the dissimilarity between two vectors. Specifically, given two d-dimensional vectors [8] $\mathcal{F}_1$ and $\mathcal{F}_2$, we define *KLDIV* as

$$KLDIV(\mathcal{F}_1, \mathcal{F}_2) = \frac{1}{2}\sum_{a=1}^{d}(f_{1a} - f_{2a})log\frac{f_{1a}}{f_{2a}} \tag{2.22}$$

Note that, for both of the metrics *RMSE* and *KLDIV*, the lower the scores returned, the better the predictive accuracy of the model.

After several attempts, we found in the AS-Caida case that with the values $\rho = 20$

7. The *roughness parameter* ρ is a value used in the *HP-filter* to separate the original series from its existing trend. The lag parameter p is the order of the autoregressive model.

8. Note that the vectors are normalized in such a way that all their components are above zero.

and $p = 9$ we achieved good results. For the AS dataset, these values (ρ and p) were set to 20 and 11, respectively. We found that the same setting used in AS also yielded good accuracy with the Yelp dataset. In DBLP, $\rho = 6.25$ and $p = 8$ allowed us to successfully forecast features at more distant times.

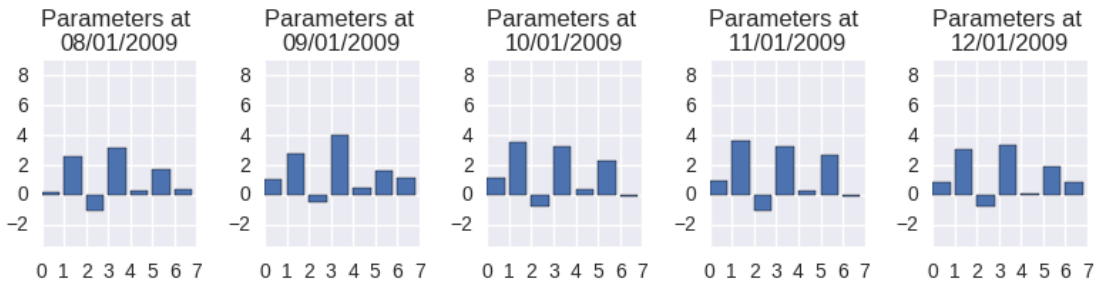
In our experiments, two tests were done to evaluate the accuracy of the chosen model. First, within the training interval, we tested the model to recover the observed value. We then compared the predicted values with the true ones. Figure 2.5 depicts the accuracy of the models used on the AS-Caida, AS, Yelp and DBLP datasets, respectively. The first row contains the boxplots of the *RMSE* and the *KLDIV* for each selected dimension within the training interval. The second row presents the RMSE and the KLDIV within the test interval.

As can be seen, we are able to predict feature values with very low errors in all datasets. However, we observed some contrasts in the results obtained; i.e., the quality of the prediction is much better in some datasets than in others. This might be due to the lifespan of the evolving communities. To give a specific example, the lifespans in the AS-Caida, AS and Yelp networks are longer than in DBLP. For the latter dataset, not only do the selected parameters (*roughness* and lag) enable us to recover the features with low errors during the intervals preceding the last five time-stamps (first row of Figure 2.5(d)), but we also discovered that the model shows good predictive accuracy (second row of Figure 2.5(d)) compared to the models in Figure 2.5(a)-Figure 2.5(c).

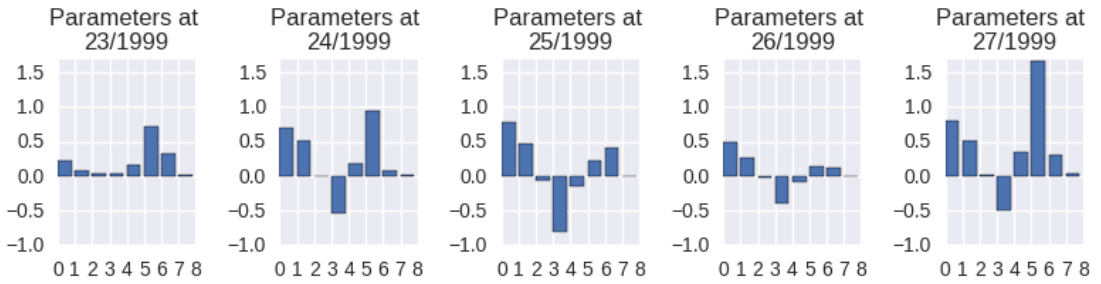
It is important to note that in our experiments, we discovered that when the lag parameter is set to a low value (under 5) or a high value (above the selected one), we cannot achieve the accuracy reported here. This might be an over-fitting issue.

2.5.4 Predicting critical events

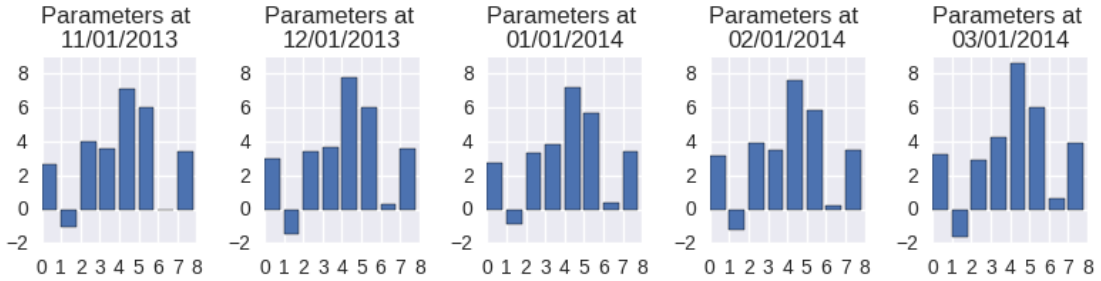
Having obtained the general hazard and the autoregressive model, we can now calculate the probability that a community may pass through an event at a more distant time. In evaluating performance in predicting the topological features, we assumed that evolving communities had been observed in all datasets till the fifth time-stamp preceding the last one. The last five time-stamps were used to test the



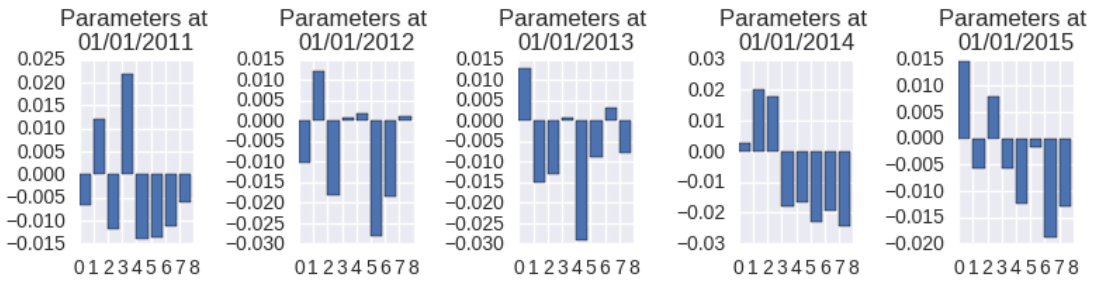
(a) AS-Caida



(b) AS



(c) Yelp



(d) DBLP

Figure 2.6 – Estimated *Cox* parameter at the last five time-stamps in the AS-Caida (*a*), AS (*b*), Yelp (*c*) and DBLP (*d*) networks.

model's ability to predict critical events. Note that in a period of five time-stamps, events can occur four times. Below, we will first present the results obtained for the different datasets. Then, restricting our aim to predicting only the very next event, we will compare our approach to some existing classifiers.

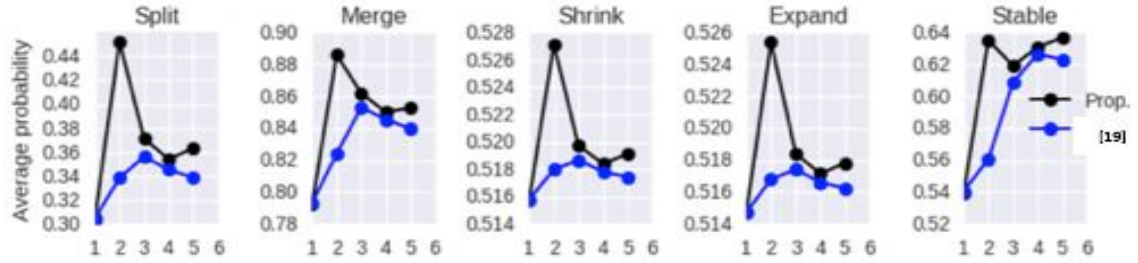
Cox parameters

As presented in subsection 4.4, predicting an event that an evolving community may undergo at various later times requires an iterative calculation of features and *Cox* parameters. The calculation of the features has already been shown in the previous subsection. Here, we present the time-varying *Cox* parameters at the last five time-stamps.

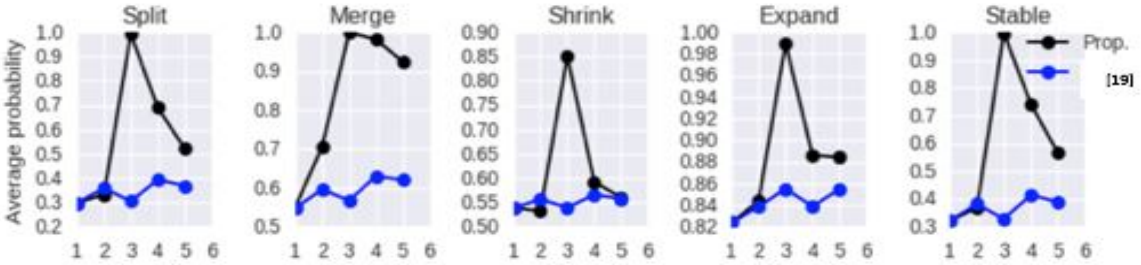
The subfigures in Figure 2.6 depict the *Cox* parameters obtained for different more distant times in the AS-Caida, AS, Yelp and DBLP datasets. Recall that, as presented in Algorithm 2, we need the predicted feature values to estimate the values of the *Cox* parameters at each later time-stamp. It can be seen in all cases that the bar charts (Figure 2.6) show that the *Cox* parameters are changing over time, which proves that the *Cox*-parameters are time-dependent. However, we note that the changes in the *Cox* parameters are not particularly pronounced (i.e., do not vary enough) in certain datasets. For example, in the Yelp dataset (Figure 2.6(c)), we can see that the parameters are quite stable over the first three predicted time-stamps. The same observation can be made for the last three predicted time-stamps in the AS-Caida dataset (Figure 2.6(a)). In contrast to the AS-Caida and Yelp datasets, the parameters in AS (Figure 2.6(b)) and DBLP (Figure 2.6(d)) display more change over time.

Probability of an event to occur

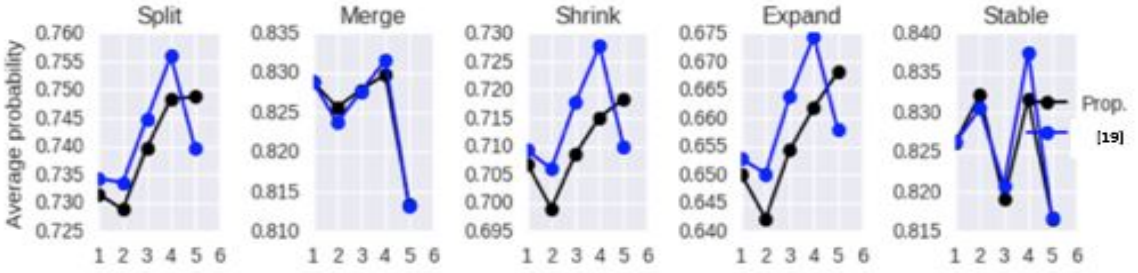
Here we compare the proposed model and our previous approach [36] in terms of the probability of an event occurring. Figure 2.7 shows the average probabilities of an event's occurrence over time. As we can see, the current model returns higher probabilities than our previous model in all experiments, with the exception of the Yelp case (Figure 2.7(c)), where it can be seen that the probabilities of the previous model are slightly better than those of the current model for all critical events from the



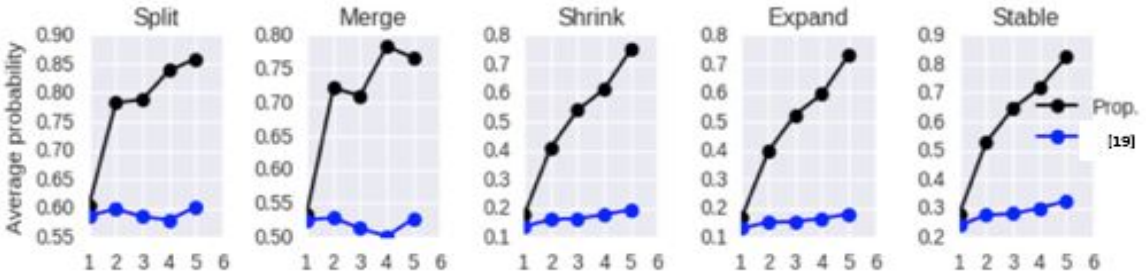
(a) AS-Caida



(b) AS



(c) Yelp



(d) DBLP

Figure 2.7 – Average probabilities of evolving communities undergoing critical events at the last five more distant time-stamps. The blue curve denotes the probabilities in [36], and the black curve the current model.

fourth-to-last time-stamp till the second-to-last time-stamp. At the last time-stamp, for the *split, shrink* and *expand* events, the proposed model performs better.

For the other datasets (figures 2.7(a),(b) and (d)), although the probabilities obtained by the proposed model are higher than those of our previous model, we note that the difference is not particularly great for more distant time predictions in certain cases. In Figure 2.7(a) in particular, both models' predictions for more distant times yield almost the same probabilities starting from the third-to-last time-stamp. Note that at the fourth-to-last time-stamp, both models always have the same probability with our previous approach because, at this time-stamp, both models still have an observation on the past history ofthe topological features.

Performance comparison

Several classifiers have been developed in the machine learning field. While it would have been interesting to see how each of these classifiers performs compared with the proposed approach, space limitations preclude an exhaustive comparison. Readers interested in a broader view of how classifiers perform may refer to [21], [52], [32], [53].

Most of the existing approaches predict what an evolving community will undergo at the next time-stamp using existing classifiers. Here, using the selected topological features, we are trying to assess how well the proposed model can predict the critical event an evolving community may undergo. To this end, we tested a number of classifiers and selected those that showed the best performance (i.e., SVM, LR and NB) to compare to our model. Note that in our case study we decided to observe the model's performance in predicting critical events at the fifth-to-last time-stamp in all datasets (i.e., window-stamps 32, 52, 42 and 16 in AS-Caida, AS, Yelp and DBLP respectively).

In our experiments on the classifiers, we adopted the approach of Brodka et al. [32] and selected sequences of length 4 representing the evolution of communities till the fourth-to-last time-stamp to train the SVM, LR and NB classifiers. Subsequently, we implemented a 10-fold cross validation in which, at each iteration, 90% of the selected sequences were used for training and 10% for the test. As final results, we averaged the probabilities obtained. The 10% at each iteration were used to calculate the probabilities in the proposed model.

Note that for all the models, we assumed that an evolving community undergoes a given event if the calculated probability is above a fixed threshold [9] set at .3. For instance, given an evolving community S_C, if the calculated probability of undergoing event e is above .3, we code the occurrence as 1, and indicate 0 in the other case to mean that the event did not occur. In this way, we generated binary values obtained by the different models.

Table 2.4 shows the performances of the models. In the first column are the critical events; the first row identifies the network under investigation and the second, the models being compared. The remainder of the table contains F-measures given within the $[0, 1]$ interval. In this table, it can be seen that the proposed model performed better than the classifiers in predicting the next time event in some cases. In certain others, however, the reverse was true: for example, in AS-Caida, the LR classifier returned better results for the events *shrink* and *expand*. The same observation can be made for the *stable* event in the AS dataset.

Generally, we can conclude that all of the tested models perform better for the events *shrink, expand* and *stable*. This may be due to the fact that the number of nodes within a community (a feature taken into consideration in our analysis) exerts the most influence in predicting these events.

9. The threshold can be automatically estimated as well.

Table 2.4 – Performance, according to the F-measure, of the proposed model (Prop.) compared with the Naive Bayes (NB), support vector machine (SVM) and logistic regression (LR) classifiers. Boldface indicates the best performance per event and dataset.

	AS-Caida				AS				Yelp				DBLP			
Events	SVM	LR	NB	Prop.	SVM	LR	NB	Prop.	SVM	LR	NB	Prop.	SVM	LR	NB	Prop.
Split	.72	.78	**.87**	.84	.64	.71	.76	**.85**	.51	.55	.50	**.73**	.67	.63	.66	**.75**
Merge	.67	.78	.85	**.89**	.58	.58	.66	**.72**	.41	.62	.58	.63	**.86**	.78	.73	.84
Shrink	.83	**.91**	.89	.83	.87	.89	.81	**.92**	.73	.71	**.91**	.82	.89	.94	**.96**	**.96**
Expand	.87	**.90**	.86	.86	.64	.71	**.85**	**.85**	.86	**.89**	.79	.87	.66	.70	**.87**	.85
Stable	.65	.81	**.83**	.77	.73	**.89**	**.89**	.86	.82	.84	**.89**	**.89**	**.85**	**.85**	**.85**	**.85**

2.6 Conclusion and future work

We have used two time-dependent models to define critical events evolving communities may undergo as temporal probability equations. In one of the models, based on survival analysis, we made use of the *Cox* regression function. In contrast to the conventional *Cox* model in which the input features and parameters are time-independent, we have considered this information to evolve with time. In order to determine the feature values over time, we have used the VAR model to represent each observed feature as a combination of lag observations. Moreover, as suggested in survival analysis theory, we define the hazard function using probability density functions.

When the features (topological features only) were taken into consideration, we succeeded in achieving interesting results. The results suggest that the proposed approach can be used to forecast critical events that evolving communities may undergo. While the model as presented can achieve interesting results, several points still need further work in order to improve the approach. For instance, a limitation of our approach is that we treat critical events with equal importance. However, in some applications, different communities may have their own life cycles. Critical events should thus be treated accordingly. One possible direction is to adjust the calculation of the fluctuation by incorporating a weighting scheme in which the importance of events such as “appear" and “disappear" is not considered equal but based on the dynamics of communities. Treating critical events according to communities’ life cycles is an important problem which, to our knowledge, has not yet been investigated in the current literature. Further research in this direction is called for.

Chapitre 3

Séries chronologiques et régimes multiples

Résumé

Les séries chronologiques sont des suites de points indexés suivant un ordre temporel. Dans de nombreux domaines d'applications, on peut constater que ces données peuvent être générées suivant une certaine cadence qui se répète à des intervalles de temps distincts : il s'agit des régimes. Dans ce chapitre, nous portons notre attention sur les prévisions des séries multiples qui présentent de nombreux régimes. L'un des problèmes majeurs rencontrés avec ce type séries est le fait que les régimes, lorsqu'ils existent, ne sont pas forcément contigus comme on peut le constater avec les séries périodiques. Avoir une connaissance sur le principe de changement de régime qui s'opère dans un jeu de séries multiples est important dans la prédiction des valeurs de ces séries. Pour ce faire, nous définissons un modèle qui sera capable de comprendre le mécanisme de changement de régime qui s'opère dans un jeu de séries multiples dans un but de faciliter la prédiction des valeurs de ces séries à des intervalles de temps subséquents. Le modèle ici proposé est basé sur un principe d'analyse de fenêtres glissantes à partir duquel une grille d'appartenance est construite. La grille d'appartenance nous permet de concrètement retrouver l'historique des régimes qui ont été exhibés par les séries d'un jeu de données. Par la suite, nous matérialisons le risque qu'un régime soit exhibé par une série en utilisant les régressions de Cox. L'originalité de notre approche réside sur le fait que nous concevons un modèle dépendant du temps pour comprendre le mécanisme de changements de régime qui s'effectuent de manière continue dans un jeu de données. Or, dans plusieurs travaux existants, ce mécanisme est souvent considéré comme statique. Les expérimentations faites dans le cadre de ce chapitre justifient la pertinence du modèle conçu.

Commentaires

Dans ce chapitre, il est intégralement présenté l'article intitulé *Modeling Regime Shifts for Multiple Time Series Forecasting*, article soumis dans *IEEE Transactions on Knowledge and Data Engineering* (TKDE). Etienne G. Tajeuna, Mohamed Bouguessa et Shengrui Wang sont les principaux auteurs de ce travail.

Dans la réalisation de cet article, j'ai (Etienne G. Tajeuna) contribué en tant que premier auteur dans la conception du modèle théorique y compris les expérimentations validant le modèle théorique proposé. Les rédaction du document et la vérification des équations ont entièrement été accompagnées par mes directeurs de recherche Pr. Shengrui Wang et Pr. Mohamed Bouguessa.

Modeling Regime Shifts for Multiple Time Series Forecasting

Étienne G. Tajeuna
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
etienne.gael.tajeuna@usherbrooke.ca
Mohamed Bouguessa
Département d'informatique, Université du Québec à Montréal,
Montréal, Québec, Canada H2X 3Y7
bouguessa.mohamed@uqam.ca
Shengrui Wang
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
shengrui.wang@usherbrooke.ca

Keywords: Time series forecasting, Regime shift, Time-evolving networks, Survival analysis.

Abstract

We investigate the problem of discovering and forecasting regime shifts in an ecosystem comprising multiple time series. Regime shifts refer to the changing behaviors exhibited by series at different time intervals. Learning the changing behaviors is a key step toward time series forecasting. While advances have been made, existing methods suffer from one or more of the following shortcomings: (1) relationships between time series are not taken into consideration for discovering regimes in multiple time series, (2) lack of an effective approach that models time-dependent behaviors exhibited by series, (3) difficulties in handling data discontinuities. Most of the existing methods are unable to handle all of these three issues in a unified framework. This, therefore, motivates our effort to devise a principled approach for modeling interactions and time-dependency in time series evolution and making better use of historical behaviors in time series forecasting. Specifically, we model an ecosystem of multiple time series by summarizing the heavy

ensemble of time series into a lighter and more meaningful structure via a mapping grid. By using the mapping grid, our model first learns time series behavioral dependencies through a dynamic network representation, and then learns the regime transition mechanism via a full time-dependent Cox regression model. The originality of our approach lies in modeling interactions between time series in regime identification and in modeling time-dependent regime transition probabilities, usually assumed to be static in existing work. Compared to well known state-of-the-art forecasting algorithms, we demonstrate that our approach yields better forecasting accuracy on both synthetic and real data sets.

3.1 Introduction

Time series are a type of time-dependent data found in many fields such as finance, medicine, meteorology, ecology, and the utility sector. In most of these fields, a time series may be dominated by several patterns, known as regimes [74], [75]. Identification of regimes allows better handling the behavioral variation of the series and improving time series forecasting. In a simple scenario, some series may periodically exhibit contiguous regimes over time. Fig. 3.1 [1] shows an example of a series dominated by three regimes, tagged respectively by blue, gray, and red boxes, which contiguously repeat over time. Such behavior can be seen for example in the utility sector, where a customer's electricity consumption may be dominated by two main regimes, business week consumption, and weekend consumption, which contiguously repeat over time. Series dominated by more than one regime that repeats over time are also known as multi-regime time series [75].

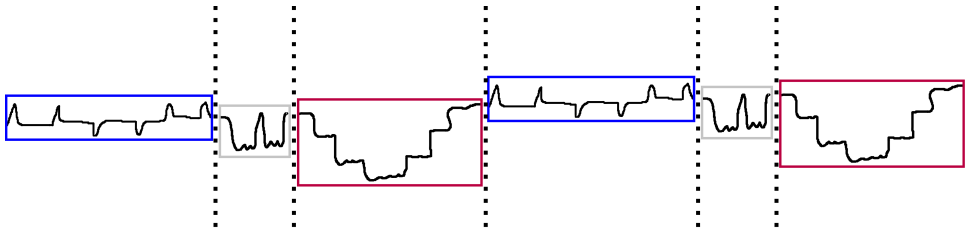


Figure 3.1 – An example of a series with three contiguous regimes. Vertical dashed lines delimit regime time intervals.

To learn the behavioral variation of a time series such as the one illustrated in Fig. 3.1, one can split the series into subseries each exhibiting a single regime, which, in fact, are easier to handle than multi-regime time series when attempting to forecast the series values at subsequent time [76]. Besides autoregressive [77], neuronal [78], [79] and other non-linear regression methods [80], [81], [82] are among the main learning models used for understanding series variation in the vast majority of applications. These methods are popular because they can capture non-stationarity due to the continuous changing behavior observed within the series. In practice, however, time series often include non-contiguous regimes, that is, regimes that do not follow a strict

1. *Most of the figures used in the paper are best viewed in color/screen.*

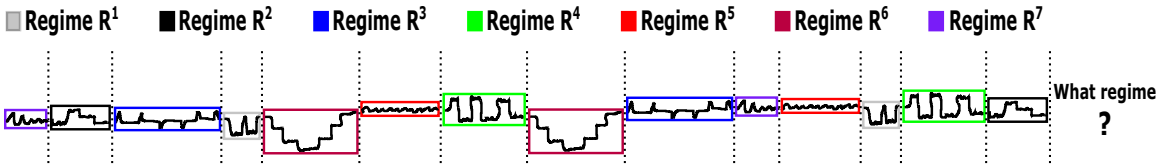


Figure 3.2 – Series comprising seven non-contiguous regimes. Vertical dashed lines delimit regime time intervals.

alignment. In such cases, it is difficult to determine how regimes are aligned compared to contiguous regimes as presented in Fig. 3.1.

Fig. 3.2 illustrates an example of a series containing seven non-contiguous regimes $R^1 - R^7$, respectively marked by gray, black, blue, green, red, brown and pink colors in the figure. In this example, regimes are non-contiguous. Such non-contiguous behavior occurs in many practical applications. For instance, in the environmental studies, a gradual rise in atmospheric greenhouse gas concentrations can provoke abrupt climate transitions [83]. In the area of health care, when patients with cognitive/heart problems are observed in their activities (e.g., walking, running, sitting), the signals collected may contain non-contiguous regimes whose discovery helps practitioners make further diagnostic and therapeutic decisions. In the financial area, unstable government policies and the mood of participants can influence on the financial market and yield infrequent discount rate changes [84].

Although it is possible to split the series into subseries to train a machine learning model to learn the series variation, the fact that regimes are non-contiguous causes a decision problem. For instance, at the 15^{th} time interval (the last one) in Fig. 3.2, it is difficult to know which regime will be applied, as it would be for a series dominated by contiguous regimes. It is easy to see that directly training a nonlinear/neuronal model over such series without taking regime shifts into consideration may yield poor forecasting accuracy if a pre-analysis of the regime shift mechanism is not performed.

Recently, a number of methods [85], [35], [86] addressing the forecast of multiple time series that evolve together have tackled the issue of discovering latent regimes that series may exhibit. In the majority of these approaches, a model-based method is first used to handle the general regime shifts occurring within the multiple time series under investigation. The obtained model is then used to foresee the subsequent

regimes that may be exhibited by the series at later time intervals. While advances have been made, the existing methods suffer from shortcomings in one or more of the following aspects:

1) Regime identification: In discovering regimes in multiple time series, existing methods usually do not consider the possible co-existence of series. In fact, the study of multiple time series as an ecosystem stipulates a co-evolutionary aspect between series which relates to the notion of complex systems and thus of possible interactions between these series at different time intervals. Exploring the interrelationships between series at different time interval is crucial for regime identification and prediction.

2) Regime shifts: Most of the existing methods attempt to capture the regime shift mechanism through a single transition matrix, which unfortunately can not reflect the continuous transitions occurring in series dominated by non-contiguous regimes. An exception is the recent work of Matsubara and Sakurai [35], in which the authors define transitions between regimes over time. However, they did not study how long a regime may persist. The time persistence of a regime is indeed important in calculating the effective transition probability between regimes.

3) Data discontinuity: In real cases of co-evolutionary time series, some of the series may be discontinuous due to the absence of observations at particular time intervals. In the forecasting process, most existing methods either fill in the missing observations with estimated data or completely ignore this issue. This may bias the co-evolution of series and potentially leads to substantial loss of information. We hypothesize that discontinuity in a time series is an informative aspect of the series that needs to be taken into consideration rather than being ignored or replaced with virtual values that may potentially alter the series' evolution.

For the sake of clarity, consider the set of series depicted by Fig. 3.3(a). Here, we can see that regimes $R^1 - R^7$ are exhibited by all series of the same data set at different time intervals. The observation of interchanging regimes across the whole data set shows that the series co-evolve over time. This is why possible interactions between series need to be considered in the process of regime identification. The observation of interchanging regimes in the data set further shows that most of the series continuously change behavior over time. Hence, defining a frozen matrix to account for the regime transition mechanism is not realistic since it cannot capture the temporal aspect of

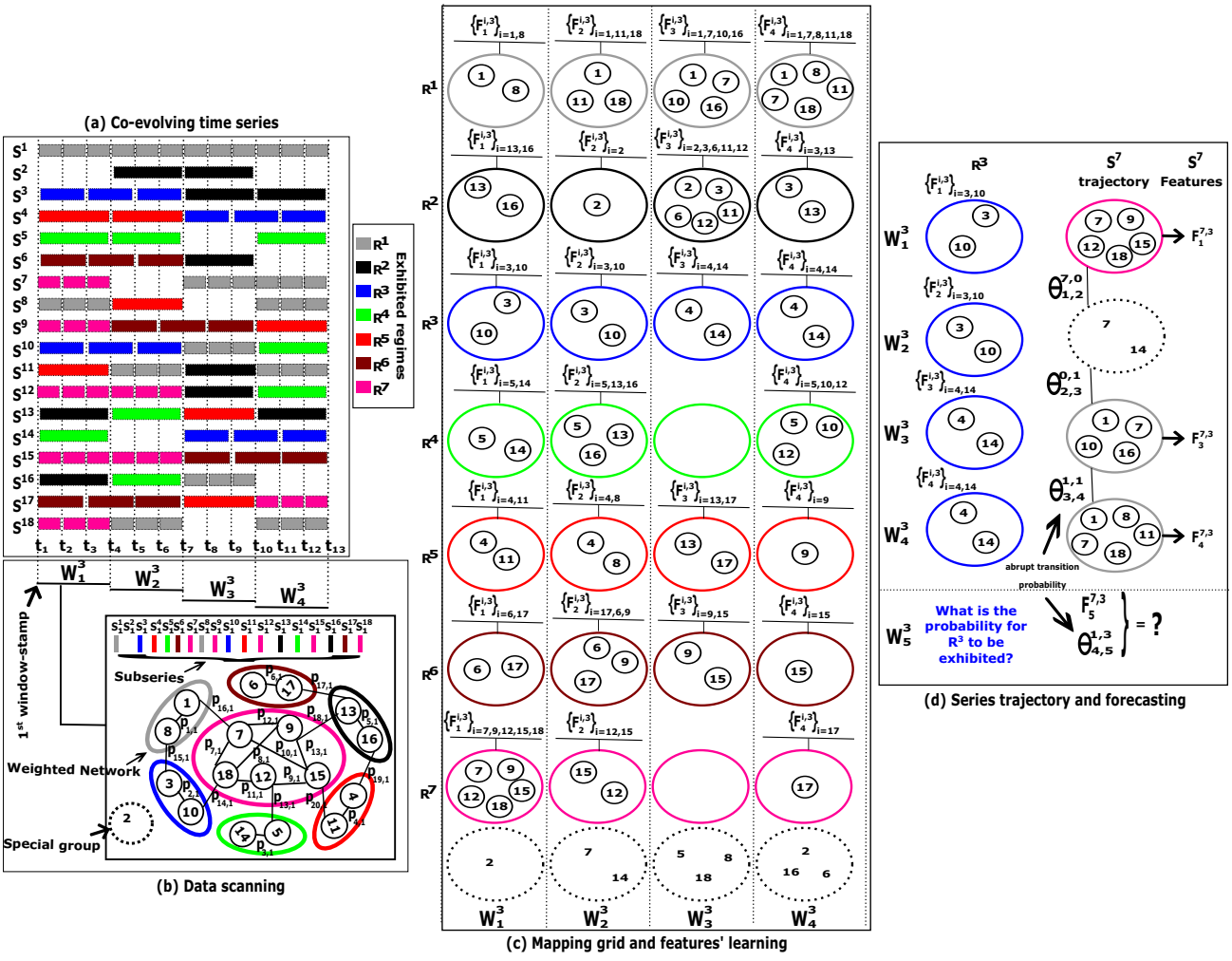


Figure 3.3 – Framework overview. (a) shows 18 time series exhibiting regimes $R^1 - R^7$ within the time ranging from $t_1 - t_{13}$. (b) illustrates a data *scanning* process with a window of size 3 that slides from left to right. The *scanning* process consists of building a weighted network ($p_{\zeta,1}$ are the weights of the edges at this first window-stamp W_1^3) structure from which patterns will be identified. (c) shows the mapping grid constructed using a sliding window W^3 and the learned features ($F_j^{i,\varrho}$). Edges between nodes are ignored for purpose of visibility. (d) depicts the trajectory followed by series S^7 and the transition probabilities from which it will be possible to forecast it values at window-stamp W_5^3. $F_5^{7,3}$ is the predicted features at the unknown window-stamp W_5^4. $\Theta_{1,2}^{\alpha,\beta} - \Theta_{4,5}^{\alpha,\beta}$, $0 < \alpha, \beta < 7$ are the transition probabilities of a switch from one behavior to another.

the regime shift. Furthermore, as can be seen, series are discontinuous due to missing data at different time points. As a regime is repeatedly exhibited, the repetitive gap observed at different time intervals could be informative. For instance, in series S^2, it could be said that after regime R^2 (tagged with black color) is twice repeated, we have three times a gap of missing information. These recurrent gaps should be considered in the forecasting process as such, rather than replacing them with virtual values that may bias series evolution.

To address the aforementioned drawbacks of existing methods, we propose the following:

1) A watcher for regimes identification. We start by automatically setting the size of a sliding window from which we may have suitable regimes (more details about identifying the optimal window size are given in Section 3.4.1). Then, we slide the settled window from left to right of the whole time series. Note that, at each slide of the window (window-stamp), a *scan* process is done. This process consists in searching for homogeneous patterns via network clustering, as illustrated in Fig. 3.3(b). In this example, we have a *window* of size 3 that slides over time. At the 1^{st} slide (window-stamp W_1^3), the observed subseries $S_1^1 - S_1^{18}$ are projected into a topological space, that is, a network (graph) structure where the nodes are labeled by numerals representing the observed subseries whereas the edges reflect the intensity of interaction or the similarity between the subseries through their weights values $p_{\zeta,1}$. The obtained network is then segmented into subnetworks whose node labels represent subseries exhibiting the same regimes.

2) Construction of a *mapping grid* that depicts regime time duration. Such a *mapping grid* enables an overview of regimes that have higher chance to be observed over time in the series. For illustration, in Fig. 3.3(c), we have for example a *mapping grid* built from the window whose size was set to 3 (W^3) in Fig. 3.3(b). In this example, each row r ($1 \leq r \leq 7$) of the grid illustrates the evolution of regime R^r. As can be seen, the number of nodes observed at each window-stamp of the seventh row (a number which gradually decreases), we may say that there is a high risk (probability) for regime R^7 to disappear. Rather than using such a naive approach to know whether a regime will no longer be observed or not, we apply a graph auto-encoder learning technique for learning features representations on networks (sets $\{F_j^{i,3}\}$, $1 \leq j \leq 4$

learned from each cell of a row in Fig. 3.3(b)) that can better explain regime survival.

3) Collection of historical data discontinuities via the sliding window. At each slide of the window, if there is a series for which no observation is found, that information is recorded in a special group. Note that the *mapping grid* contains a specific row for collecting such information. In Fig. 3.3(c), this last row shows dashed circles containing numerals representing series for which no observation is found. For example, at the first slide of window W^3 (window-stamp W_1^3), it can be seen that there is no observation for series S^2. By collecting this information in the *mapping grid*, it will be possible to foresee whether or not a series' values will be observed at subsequent window-stamps.

By applying the strategy described above, the forecasting process becomes more effective. Given a time series belonging to a collection of co-evolving series, to forecast its values, we use the *mapping grid* to get its trajectory (i.e., its historical behavior), and then estimate the probability of exhibiting a specific regime at subsequent window-stamps. In Fig. 3.3(d), for instance, we want to know the risk (probability) of regime R^3 being exhibited by series S^7 at window-stamp W_5^3. To this end, we first generate from the *mapping grid* the trajectory followed by S^7. In the example, the trajectory is illustrated by the sequence of groups (vertical sequence of colored circles (on the right) in Fig. 3.3(d)). From this sequence, we calculate the historical regime shift probabilities. In the example (Fig. 3.3(d)), these probabilities are given by $\Theta_{1,2}^{\alpha,\beta}$ to $\Theta_{3,4}^{\alpha,\beta}$. After identifying the trajectory of series S^7, we then calculate the regime lifespan (R^3 in the example) using the sets of features $\{F_j^{i,3}\}$, $1 \leq j \leq 4$, learned from the sequence of blue circles (3^{rd} row of the mapping grid). When forecasting, we estimate the next feature values $F_5^{7,3}$ and the probability of shifting to regime R^3, $\Theta_{4,5}^{1,3}$, based on the historical features and the regime shift probabilities. The forecasted regime that will be exhibited by series S^7 will correspond to the regime with the highest probability of being observed. Note that, the transition probabilities $\mathbf{\Theta}_{j,(j+1)}^{\alpha,\beta}$ capture also the risk of having no observation at a window-stamp as we can see in S^7 trajectory. Here, from W_1^3 to W_2^3, the abrupt transition probability is given by $\mathbf{\Theta}_{1,2}^{7,0}$.

The significance of this work can be summarized as follows:

1) We propose a principled approach for multiple time series forecasting capable of identifying and handling hidden regime shifts exhibited by these series. The mapping grid devised in our approach allows capturing the overall continuous regime shift

mechanism that occurs within the multiple time series data set.

2) We tackle the problem of continuous regime shifts by modeling a time-dependent probability of transition between regimes, using a full time-dependent Cox regression model. From the constructed mapping grid we calculate the survival of regimes and abrupt transition between regimes. The probability that a regime will be exhibited is calculated via a dynamic Cox regression model, whereas the abrupt transition is calculated based on the constructed mapping grid.

3) We validate the whole approach by implementing it on both synthetic and real data sets. We illustrate the suitability of the proposed method by comparing its performance to that of six learning algorithms. Our experiments show that the proposed approach significantly improves forecasting accuracy.

3.2 Mainstream related methods

In this section, we provide a high-level description of mainstream algorithms related to our work. While a complete survey is beyond the scope of this paper, we provide a critical review to put our work in perspective relative to existing methods.

Smooth transition autoregressions are widely used for forecasting time series that exhibit various regimes. In some studies [85], [87] it is first assumed that each exhibited regime can be recovered using an autoregressive model. Then, a weighted combination of these autoregressions is defined to foresee the series values at subsequent times. The linear combination is defined in a way that requires user intervention to manually fix the number of regimes. To sidestep this constraint, Sanquer et al. [74] proposed a hierarchical Bayesian framework for automatically identifying the hidden regimes exhibited by series. Note that in all of these methods, only one series can be handled at a time. To address the case of multiple time series, Hochstein et al. [37] proposed a multivariate smooth transition autoregression model in which each regime is modeled using a vector autoregressive model [88]. In [37], Hochstein also defined a higher-order Markov model to account for the regime shifts. This model allows capturing not only the immediate past regime shifts but also several previous regime shifts at once, while using a single transition matrix. Though the plurality of series is considered in this case, the order of regimes and the order of the Markov model need to be manually set

in advance.

In the methods mentioned above, it is important to note that the regime shifts have been defined through a single matrix that states the overall transition probabilities between the exhibited regimes, meaning that the probability of switching from one regime to another remains constant. For series that exhibit non-contiguous regimes, however, the regime shift mechanism may be time-dependent. Moreover, the information gaps that certain series may present are not taken into consideration. To handle data discontinuity (due to the presence of missing information), Hallac et al. [38] proposed a network-based approach for tracking the evolution of co-evolving time series. Although their model can dynamically infer the graph structure, they did not address the problem of regime transitions. In [80], the authors proposed the *RegimeCast* model which learns, at a given window-stamp, the various patterns that may exist in a co-evolving environment, and reports, at a subsequent time, the most likely pattern(s) that will be observed. Though the approach can report the subsequent patterns, it does not capture the possible dependencies that may exist between patterns. In an updated work [35], the authors proposed the deterministic *ORBITMAP* model for capturing the time-dependent transitions between exhibited regimes. However, in their setting, they work with regimes priorly labeled (known in advance). Moreover, the *ORBITMAP* model does not consider the case of data discontinuity.

To overcome the data discontinuity problem, Li et al. [89] proposed to complete missing values based on the neighboring trend followed by the data. Specifically, based on the observed part of the co-evolving time series, they defined a sequence of latent variables that follow probabilistic rules. Then, based on the probabilistic model, they generated back series values at all time points including time intervals where there is no observation. Cai et al. [86] also proposed to solve the missing values problem by building the corresponding network structure relating to the interrelation between time series. The constructed network is later projected in a latent space through a matrix factorization from which the missing information is no longer considered as a constraint since they are no longer perceived. Yet, these methods may make sense in the case that the time series have been accidentally harmed when being collected or manipulated, and thus cause the disappearance of values. However, in some real cases,

we may have multiple and repetitive missing values at different time intervals which are in fact instructive of series behaviors and must be taken as such.

Overall, in the current literature, various authors have contributed to the advances in handling regime shifts in multiple time series. Only abrupt transition probabilities are captured by these methods. However, some regimes may persist over time whereas others do not. Studying the time duration of regimes, or regime lifespan is an important aspect that needs to be considered in the calculation of regime transition probability. To study event lifespan, statistical methods based on survival analysis have demonstrated their effectiveness in predicting failure/death events [90], [91], [92]. In this paper, we will use the Cox regression model for evaluating the risk of a regime to be exhibited or not.

It is worth noting that, in the majority of applications of the Cox regression model, events are continuously tracked through a time-dependent baseline function which might be affected (continuously decreasing or increasing) by the effect of constant covariates. However, due to the continuous changing regime shifts mechanism occurring within the time series ecosystem, using a Cox regression model with constant covariates will not well reflect the regime's survival probability. Facing this, we instead devise a full time-dependent Cox regression for modeling regime lifespans. In other words, we devise a Cox regression model where the covariates are time-dependent.

3.3 Key concepts

In this section we describe key concepts and symbols used in our methods.

Time series: A time series is a set of points ordered by a time index as follows: $S^i = \{(t_l, e_l^i)\}_{l=1}^m$, where t_l are regular time-stamps, m the series length and e_l^i the series value at the specific time t_l. The index $i \in \mathbb{N}^*$ here refers to the i^{th} time series in a set of N univariate time series $\boldsymbol{S} = \{S^i\}_{i=1}^N$. The index set $\boldsymbol{\mathcal{N}} = \{i\}_{i=1}^N$ and its subsets are convenient representations of the whole set and groups of time series in $\boldsymbol{S}$. For each time series S^i, we use $\vec{S}^{\,i}$ to denote the vector corresponding to series S^i, where each component is the series value e_l^i.

Sliding Window: Given a set of time-stamps $\{t_l\}_{l=1}^m$ and a fixed duration $\varrho \in \mathcal{D}$, where $\mathcal{D} = \{2, ..., m-1\}$ is the set of possible interval durations, we use the notation

W^{ϱ} to denote a window of size ϱ. This window is required to slide from left to right following the time-stamps $\{t_l\}_{l=1}^{m}$. Let us consider a window W^3 of size 3 that slides from the left to the right. At the 1^{st} instance, this window covers the time set $\{t_1, t_2, t_3\} = W_1^3$. At the 2^{nd} instance, the window covers the time set $\{t_4, t_5, t_6\} = W_2^3$; etc. To generalize, for a window W^{ϱ} that slides, we use the notation W_j^{ϱ} to mean its j^{th} instance, or j^{th} window-stamp. The terms window-stamp and window instance will be used throughout the paper interchangeably to mean the same concept. Note that in this paper, we develop a new general framework to forecasting regime shifts while working with non-overlapping sliding windows. Generalizing the framework to overlapping sliding windows is our future work.

Subseries: We use the term subseries to denote the sequence of values observed from a series at a given window instance. Formally, for a given series S^i, the corresponding subseries observed at the window instance W_j^{ϱ} is given as $S_j^{i,\,\varrho} = \{(t_l,\, e_l^i)\}_{l=(j-1)*\varrho+1}^{j*\varrho}$. Hence, for a given sliding window W^{ϱ}, we can split each series into contiguous subseries time segments of length ϱ, as follows:

$$S^i = \bigcup_{j=1}^{b} S_j^{i,\,\varrho} \tag{3.1}$$

where $b = m/\varrho$ is the number of instances of window W^{ϱ} in the time interval $[t_1,\, t_m]$ and $\bigcup$ is used as a concatenation operator. At each j^{th} instance of W^{ϱ}, we use the notation $\vec{S}_j^{i,\,\varrho}$ to denote the vector representing the subseries $S_j^{i,\,\varrho}$.

Time series network: From each set of subseries $\{S_j^{i,\,\varrho}\}_{i=1}^{N}$ observed at window-stamp W_j^{ϱ}, a *time series network* is built as the projection of subseries $\{S_j^{i,\,\varrho}\}_{i=1}^{N}$ onto a topological space $G_j = (V,\, E_j,\, \mathcal{P}_j)$, a graph structure where V is the set of nodes each of which being labled by $i \in \boldsymbol{\mathcal{N}}$ corresponds to the subseries $S_j^{i,\,\varrho}$, E_j and $\mathcal{P}_j$ are respectively sets of edges and edge weights that reflect the relationship strength between series at window-stamp W_j^{ϱ}.

Regime: In this paper, we define a regime by the profile pattern of a group of similar subseries observed within a window instance. Within the whole set of series $\boldsymbol{S}$, we assume that there exist a maximum of $K \in \mathbb{N}^*$ possible regimes, denoted by $\boldsymbol{R} = \{R^r\}_{r=1}^{K}$. At each window-stamp W_j^{ϱ} we use the notation $S_j^{i,\,\varrho} \otimes R^r$ to mean that

subseries $S_j^{i,\,\varrho}$ exhibits regime R^r. Hence, to obtain regimes, for each of the time interval W_j^{ϱ}, we perform clustering analysis of the subseries into the set $C_j^r = \{S_j^{i,\,\varrho} \mid S_j^{i,\,\varrho} \otimes R^r\}$ whose profile pattern is given by T^r ($\vec{T}^r$ its vector representation). The profile pattern here is a subseries whose vector representation corresponds to the centroid of the set of vectors $\{\vec{S}_j^{\,i,\varrho} \mid S_j^{i,\,\varrho} \otimes R^r\}$. It is important to know that, at a given window-stamp W_j^{ϱ}, a regime R^r might not be observed (i.e., no series exhibiting this regime). In this case, we have $C_j^r = \emptyset$. In the same way of using the label R^r, $1 \leq r \leq K$ to denote existing regimes, we will use the label R^0 to denote no observation. We will use $\mathcal{O}_j = \{C_j^r\}_{r=1}^K$ to denote the set of groups of subseries identified at window-stamp W_j^{ϱ}.

The groups of subseries C_j^r are obtained by a clustering process which consists in splitting the network G_j into subnetworks $G_j^r = (V_j^r,\, E_j^r,\, \mathcal{P}_j^r)$, $1 \leq r \leq K$, where $V_j^r \subseteq V$ is the subsets of nodes whose labels are given by the subset of integers $\boldsymbol{\mathcal{N}}_j^r \subseteq \boldsymbol{\mathcal{N}}$ representing series that exhibited regime R^r at the window-stamp W_j^{ϱ}. $E_j^r \subseteq E_j$ and $\mathcal{P}_j^r \subseteq \mathcal{P}_j$ are the sets of edges and edge weights between these series (the one exhibiting regime R^r) at the window-stamp W_j^{ϱ} respectively. Note that, each subnetwork G_j^r is obtained in such a way that its nodes V_j^r are densely connected within the subnetwork, and are sparsely connected to the nodes of in the other parts of the network G_j. In a nutshell, we can say that, G_j^r relates the group of subseries that exhibit regime R^r projected in the topological space at W_j^{ϱ} whereas C_j^r relates the group of subseries that exhibit regime R^r at W_j^{ϱ} in the time space.

Regime lifespan: We use the term regime lifespan to denote the time period over which the regime continues to be observable. Noting that the regime is obtained from the group of subseries, the sole way of estimating the time existence of a regime is by tracking this group over time. The tracking is represented by a sequence of groups depicting the evolution of the regime. Formally, for a given regime R^r, the sequence representing its evolution over time (lifespan) is given as $\mathcal{E}^r = \{C_j^r\}_{j=1}^b$.

To investigate regime lifespan $\mathcal{E}^r$, we exploit the graph representation (series projected in the topological space) for identifying and computing features. In fact, at each window-stamp W_j^{ϱ}, we use a Graph Autoencoder for automatically learning from the graph G_j^r (graph representing the group of subseries C_j^r) the embedding $\{F_j^{i,\varrho}\}_{i \in \boldsymbol{\mathcal{N}}_j^r}$ (resp. $\{\vec{F}_j^{i,\varrho}\}_{i \in \boldsymbol{\mathcal{N}}_j^r}$, its corresponding set of vector representation) that captures the series relationships while preserving its content [93] (i.e., the subseries values). The

Table 3.1 – Main notation.

Notation	Definition	Notation	Definition
t_l	Time-stamp	R^r	Regime
W^{ϱ}	Window of size ϱ	R^0	None regime
W_j^{ϱ}	Window-stamp	$S_j^{i,\varrho} \otimes R^r$	$S_j^{i,\varrho}$ exhibiting R^r at W_j^{ϱ}
S^i	Time series	C_j^r	Group of subseries at W_j^{ϱ}
$\boldsymbol{\mathcal{N}}$	Set of integers	T^r	Profile pattern
$S_j^{i,\varrho}$	Subseries at W_j^{ϱ}	$\mathcal{M}$	Mapping grid
G_j	Network at W_j^{ϱ}	$\mathcal{E}^r$	Regime lifespan
G_j^r	Subnetwork at W_j^{ϱ}	$Tr(S^i \vert W_b^{\varrho})$	S^i trajectory till W_b^{ϱ}
$F_j^{i,\varrho}$	Node embedding at W_j^{ϱ}	$\Theta_{j\,,\,(j+1)}$	Transition probability

obtained embedding is used as knowledge feature that may help in understanding regime lifespan.

Trajectory: The term trajectory in our context refers to the changing behavior a given series displays over time: specifically, it corresponds to the different regimes the series exhibits. Here again, as in the definition of the regime lifespan, we recognize that a series may have missing values at certain window-stamps and use a sequence of groups to describe the various behavior displayed by a series over time. Formally, given a time series S^i that evolves till window-stamp W_b^{ϱ}, we define its trajectory as an order set or a sequence

$$Tr(S^i \mid W_b^{\varrho}) = \bigcup_{j=1}^{b} \left\{ R^{\alpha},\ 0 \leq \alpha \leq K \mid S_j^{i,\varrho} \otimes R^{\alpha} \right\} \tag{3.2}$$

Transition probability: The transition probability is defined as the risk of a series S^i of switching from a behavior R^{α}, $\alpha \in \{0,\ 1,\ 2,\ ...\ K\}$ it exhibits at window-stamp W_j^{ϱ} to another behavior R^{β}, $\beta \in \{0,\ 1,\ 2,\ ...\ K\}$ at window-stamp W_{j+1}^{ϱ}. In this paper, we use the term $\boldsymbol{\Theta}_{j\,,\,(j+1)}$ to refer to the matrix of transition probabilities from one behavior to another in two consecutive window-stamp W_j^{ϱ} to W_{j+1}^{ϱ}.

To conclude this section, Table 3.1 provides the main notation used throughout the paper.

3.4 Proposed approach

To model and forecast regimes shifts in multiple time series, we propose an approach that proceeds in three phases: (1) scanning the input multiple time series for identifying repetitive profile patterns, (2) restructuring the multiple time series into a grid that maps the series behaviors and regime lifespans and (3) building a survival model for multiple time series forecasting.

3.4.1 Data scanning

Given an ensemble of multiple time series, the main idea behind the data scanning consists in applying a network-based approach for discovering regimes that series may exhibit over time. To do this, we need to set the size of a suitable window from which we observe how the series evolve conjointly and identify the regimes that will be represented by the profile of similar series at each time stamps.

To set the window size, we use an incremental procedure in which we test, through a data scanning, various window sizes till getting an optimal one. Given a set of window sizes $\mathcal{D}$, for each value $\varrho \in \mathcal{D}$, we define a window W^{ϱ} which slides from left to right by scanning the series in the aim of identifying significant repetitive patterns. At the end of the sliding process, i.e., after passing through the series with the window, we define a series variation score which will help us to identify if we have reached the optimal window size.

Network construction

Given a window W^{ϱ}, based on Eq.(3.1) at each window-stamp W_j^{ϱ}, we build a network from the set of subseries derived from the series found in $\boldsymbol{S}$. The entities represented by nodes are subseries, while interrelations are represented by edges. To capture the relationship between subseries, we make use of the Symbolic Aggregate approXimation (SAX) [94] for discretizing our subseries into categorical data. Specifically, knowing that, the time series are scaled within the interval [0, 1], we uniformly codify values of this interval with a $10-symbols = \{g_a\}_{a=1}^{10}$ where the symbol g_a is considered as a random variable which can only take values in $[(a-1)/10,\ a/10)$. Thus, at each

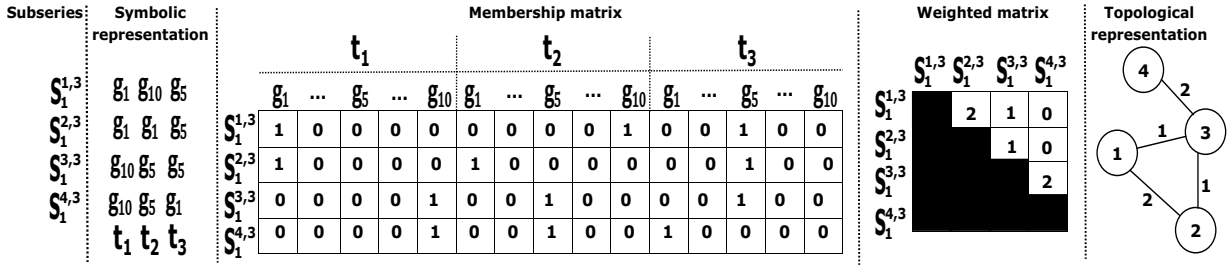


Figure 3.4 – Process flow for network construction. $S_1^{i,3}$ are subseries extracted at window-stamp W_1^3. The weighted matrix is symmetric, thus, the upper or lower triangle is considered. Integers $1-4$ represent subseries $S_1^{1,3} - S_1^{4,3}$ respectively in the topological representation. Edge weights are values of the weighted matrix.

window-stamp W_j^{ϱ}, each subseries can now be rewritten as a sequence of symbols. Having the symbolic representation of our subseries, we now build a membership matrix $\mathbf{\Delta}_j = (\delta_{j,\tau}^i)$, $1 \leq i \leq \boldsymbol{\mathcal{N}}$, $1 \leq \tau \leq 10\,\varrho$ where columns represent symbols and rows correspond to subseries. We use the values 1 and 0, respectively, to denote the presence and the absence of the symbol in the discretized series. Based on the membership matrix $\mathbf{\Delta}_j$, we then calculate the weighted matrix $\mathcal{P}_j = Upper(\mathbf{\Delta}_j \times \mathbf{\Delta}_j^*)$ where $\mathbf{\Delta}_j^*$ corresponds to the transpose of matrix $\mathbf{\Delta}_j$ and $Upper(\mathbf{\Delta}_j \times \mathbf{\Delta}_j^*)$ the upper triangle of matrix $\mathbf{\Delta}_j \times \mathbf{\Delta}_j^*$. In Fig. 3.4 we have an illustration of the procedure that we have implemented to build the network reflecting the relationship between subseries $S_1^{1,3} - S_1^{4,3}$.

Series variation & window size selection

At each window-stamp W_j^{ϱ}, we construct the network G_j, and then use a network-based clustering algorithm to split G_j into subnetworks of closely connected nodes. In our experiments, we used Infomap [25] because it is an effective parameter-free algorithm capable of clustering a networked data without any parameter setting as Lancichinetti et al. demonstrate in [67]. It is important to note that, at each window-stamp W_j^{ϱ}, the number of identified subnetworks may vary. Our aim is to find an optimal window size permitting highly similar, or repetitive, subnetworks across the networks at different window-stamps. The profile patterns of such subnetworks are defined as the regimes. The repetitiveness allows generating similar regimes at different

window-stamps for tracking and lifespan analysis. We propose a heuristic solution to this problem via the segmentation of a score function that is defined as follows:

$$\mathcal{V}ar(\varrho \,|\, \boldsymbol{S}) = \frac{1}{\varrho} \cdot max\left\{K_j,\ 1 \leq j \leq b\right\} \tag{3.3}$$

where K_j is the number of subnetworks identified at window-stamp W_j^{ϱ}, which means that, $max\left\{K_j,\ 1 \leq j \leq b\right\}$ corresponds to the maximum number of regimes observed in $\boldsymbol{S}$.

Eq. (3.3) defines a *regime-density-per-window-size* score. Intuitively and experimentally, we have the following observations. A narrow window size ϱ leads to high regime density score $\mathcal{V}ar()$, which corresponds to the cases where there exist regimes that are not repeating over time: the data set is unstable when looking through a narrow window. On the other hands, a wide window size ϱ leads to low regime density score $\mathcal{V}ar()$, which corresponds to the cases with highly repetitive regimes. Hence, it is important to select a window which is wide enough for finding suitable regimes. However, $\mathcal{V}ar()$ is a decreasing function that converges to the null density. A null density means that the window size is too large to permit capturing regime shift events. To avoid being in such situation, we select the smallest window size from which we can observe the quasi-stability in terms of regimes that repeat over time.

To automatically select the optimal window size $\hat{\varrho}$, we use the MDL pruning techniques [95], [96] over the set of regime density scores $\boldsymbol{\mathcal{V}ar} = \{\mathcal{V}ar(\varrho \,|\, \boldsymbol{S}),\ \varrho \in \mathcal{D}\}$. In [95], [96], authors use the MDL-selection to split their data into two subsets (sparse and dense subsets) then discard one of the two. In our case, we are selecting the border between the two subsets from which we can get the window size that minimizes the MDL criteria as follows:

$$\hat{\varrho} = \begin{cases} \underset{\varrho \in \mathcal{D}}{argmin}\{CL(\varrho)\} \\ CL(\varrho) = log_2(\overline{\boldsymbol{\mathcal{V}ar_1}}) + \sum_{\varrho \in \mathcal{D}_1} log_2(|(\mathcal{V}ar(\varrho \,|\, \boldsymbol{S}) - log_2(\overline{\boldsymbol{\mathcal{V}ar_1}})|) + \\ \qquad\qquad log_2(\overline{\boldsymbol{\mathcal{V}ar_2}}) + \sum_{\varrho \in \mathcal{D}_2} log_2(|(\mathcal{V}ar(\varrho \,|\, \boldsymbol{S}) - log_2(\overline{\boldsymbol{\mathcal{V}ar_2}})|) \end{cases} \tag{3.4}$$

where $\mathcal{D}_1 = \mathcal{D} \setminus \{\varrho+1, ..., m-1\}$, $\mathcal{D}_2 = \mathcal{D} \setminus \{2, ..., \varrho\}$, $\boldsymbol{\mathcal{V}ar_1} = \{\mathcal{V}ar(\varrho \,|\, \boldsymbol{S}),\ \varrho \in \mathcal{D}_1\}$, $\boldsymbol{\mathcal{V}ar_2} = \{\mathcal{V}ar(\varrho \,|\, \boldsymbol{S}),\ \varrho \in \mathcal{D}_2\}$ and $\overline{\boldsymbol{\mathcal{V}ar_1}}$ (resp. $\overline{\boldsymbol{\mathcal{V}ar_2}}$) the average variation score in $\mathcal{D}_1$ (resp. $\mathcal{D}_2$). The selected window size is taken as the optimal window size $\hat{\varrho}$. See Fig. 3.5 for an illustration.

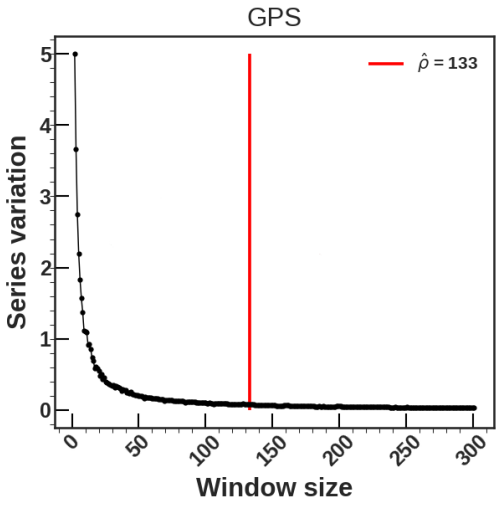


Figure 3.5 – Example of trends followed by $\mathcal{V}ar()$ in the gesture phase segmentation data set (GPS). The vertical line corresponds to the border between the large and small values. The window size corresponding to this border is selected as the optimal window size $\hat{\varrho}$.

Having estimated the optimal window size $\hat{\varrho}$, we then use this size to obtain the number of distinct regimes K by counting the number of distinct profile patterns across all window-stamps $W_j^{\hat{\varrho}}$, $1 \leq j \leq b$.

3.4.2 Mapping grid

In this section, we first present how to build and populate the mapping grid. Then, based on the constructed grid, we present how to learn features for explaining regime persistence and predicting regime survival. Finally, we illustrate how to estimate the regime survival probability using the learned features.

Grid construction and features learning

As it is shown in Fig. 3.3(c), the mapping grid is designed as a $(K+1) \times b$ grid $\mathcal{M}$ of cells each representing a group of subseries exhibiting a regime. The mapping grid enables an overview of regime persisting or disappearing over time and allows extraction of the trajectory of each series. In fact, with the size of the window set to $\hat{\varrho}$ and the number of regimes to K, we determine which regime a series exhibits at a window-stamps $W_j^{\hat{\varrho}}$, $1 \leq j \leq b$. This includes also the case of no regime when there is no observation. The last row of $\mathcal{M}$ contains the numerals indicating series for which there was no observation. For each regime R^r, we can extract from the grid the sequence $\mathcal{E}^r = \{C_j^r\}_{j=1}^b$ of groups of subseries representing the regime lifespan.

From the constructed mapping grid, features need to be learnt in order to better explain the regime lifespan by considering the interrelation between subseries of the same group. To take these inter-dependencies into consideration, we propose to use a graph autoencoder for automatically learning features from each node of our mapping grid. The principle behind the graph autoencoder consists in representing a network structure into a small latent space while preserving node interrelations and being possible of approximately reconstructing the network while using the latent representation [97], [98]. Dimensions of the latent representation (the embedding) is used as the network features. By running Algorithm 3 over a set of co-evolving time series, we get the corresponding mapping grid.

Regime survival

Now that we have constructed a grid capable of capturing various regime behaviors exhibited by series over time. We now turn to learning the probabilities with which these series change behaviors over time. Before getting on the regime shift occurring in each series, we need to first learn regime's lifespan. Given a regime R^r, at each instance j of the window $W^{\hat{\varrho}}$, we count the number of series $\mathcal{J}^r(j)$ exhibiting this regime, as follows:

$$\mathcal{J}^r(j) = H^r(j) + M^r(j) \tag{3.5}$$

Algorithm 3: Data scanning & grid construction

Data: $\boldsymbol{S} = \{S^i\}_{i=1}^N$, $\mathcal{D}$, ϵ ; /* Sets of co-evolving time series, window sizes and convergence criteria. */

Result: $\hat{\varrho}$, K, $\boldsymbol{SS}$, $\boldsymbol{\mathcal{V}ar}$, $\mathcal{M}$; /* Optimal window size, number of regimes, set of regimes, the series dynamic and the mapping grid. */

begin

A- Data scanning:

begin

A.1- Initialization:

begin

- $\varrho \leftarrow$ First value in $\mathcal{D}$;
- $\boldsymbol{\mathcal{V}ar} \leftarrow \emptyset$; /* Initialize the set of scores. */

A.2- Scanning:

begin

- At each window-stamp W_j^{ϱ} use Eq.(3.1) to get the set of subseries $\{S_j^{i,\varrho}\}_{i=1}^N$;
- Use $\{S_j^{i,\varrho}\}_{i=1}^N$ to Build the network G_j as discussed in section 3.4.1;
- Split G_j in to subnetworks and get the set of groups $\mathcal{O}_j$ at each window-stamp W_j^{ϱ} as discussed in section 3.4.1;
- Using Eq. (3.3), calculate the series variation $\mathcal{V}ar(\varrho \,|\, \boldsymbol{S})$;
- $\boldsymbol{\mathcal{V}ar} \leftarrow \boldsymbol{\mathcal{V}ar} \bigcup \{\mathcal{V}ar(\varrho \,|\, \boldsymbol{S})\}$;

A.3- Iteration:

begin

while *True* **do**

- $\varrho \leftarrow$ next value in $\mathcal{D}$;
- Using the new value ϱ scan series by running step **A.2** and get the *new* $\mathcal{V}ar(\varrho \,|\, \boldsymbol{S})$;
- If $|$*new* $\mathcal{V}ar(\varrho \,|\, \boldsymbol{S}) -$ *previous* $\mathcal{V}ar(\varrho \,|\, \boldsymbol{S})| < \epsilon$ stop the loop;

- Based on the set of scores $\boldsymbol{\mathcal{V}ar}$ get the optimal window $\hat{\varrho}$ as discussed in section 3.4.1;
- $\boldsymbol{SS} \leftarrow$ distinct regimes, as mentioned in section 3.4.1 and $K \leftarrow |\boldsymbol{SS}|$;

B- Mapping grid:

begin

- Initialize a $(K+1) \times b$ empty mapping grid $\mathcal{M}$, where b is the number of instances of $W^{\hat{\varrho}}$;
- At each window-stamp $W_j^{\hat{\varrho}}$, populate cells of the mapping grid with numerals $\mathcal{N}_j^r$ and $\mathcal{N}_j$ representing series exhibiting each regime R^r and series from which we have no observation respectively;
- For each cell of the mapping grid, learn the corresponding features as discussed in section 3.4.2

where $H^r()$ is a non-decreasing function, called *cumulative intensity process*, and $M^r()$ is a mean-zero martingale [62].

Considering $H^r()$ to be continuous, there exists a predictable non-negative intensity

process $h^r()$ such that

$$H^r(j) = \int_{\omega=1}^{\omega=j} h^r(\omega) d\omega \tag{3.6}$$

At each window-stamp $W_j^{\hat{\varrho}}$, we calculate the hazard for a regime R^r to be observed as:

$$h^r(j) = \begin{cases} 0 \quad if \quad |C_j^r| = 0 \\ \frac{\gamma^r(j)}{|\boldsymbol{\mathcal{N}}_j^r|} \sum_{S_j^{i,\,\hat{\varrho}} \in C_j^r} exp(\vec{F}_j^{\,i,\,\hat{\varrho}} \bullet \vec{\Gamma}_j^r) \quad otherwise \end{cases} \tag{3.7}$$

where the "$\bullet$" symbol stands for the dot product. In the above equation, $\vec{\Gamma}_j^r$ is the vector parameter representing the contribution of the feature vector $\vec{F}_j^{\,i,\,\hat{\varrho}}$ at $W_j^{\hat{\varrho}}$. This vector parameters are learnt based on the historical information collected till window-stamp $W_j^{\hat{\varrho}}$. $\gamma^r()$ is a Gamma probability density function that plays the role of the baseline hazard. Note that, any probability density function that fits with the distribution of the number of subseries exhibiting the given regime can be considered. Our choice has gone to the Gamma function due to its great shape flexibility.

Knowing the historical features till the actual feature vectors $\{\vec{F}_j^{\,i,\,\hat{\varrho}}\}_{i \in \boldsymbol{\mathcal{N}}_j^r}$, we have a partial view of the data ensemble. For this reason, we define the partial log-likelihood of parameters $\vec{\Gamma}_j^r$ over the observed feature values $\left\{\{\vec{F}_l^{\,i,\,\hat{\varrho}}\}_{i \in \boldsymbol{\mathcal{N}}_l^r},\ 1 \leq l \leq j\right\}$ as follows:

$$\begin{aligned} \mathcal{LL}(\vec{\Gamma}_j^r) &= \sum_{l=1}^{j-1} \sum_{S_l^{i,\hat{\varrho}} \in C_l^r} \nu_l^r \cdot \mathcal{L}\left(\frac{exp(\vec{F}_l^{i,\hat{\varrho}} \bullet \vec{\Gamma}_l^r)}{\sum_{S_l^{i,\hat{\varrho}} \in C_l^r} exp(\vec{F}_l^{i,\hat{\varrho}} \bullet \vec{\Gamma}_l^r)}\right) \\ &= \sum_{l=1}^{j-1} \sum_{S_l^{i,\hat{\varrho}} \in C_l^r} \nu_l^r \left[(\vec{F}_l^{i,\hat{\varrho}} \bullet \vec{\Gamma}_l^r) - \mathcal{L}\left(\sum_{S_l^{i,\hat{\varrho}} \in C_l^r} exp(\vec{F}_l^{i,\hat{\varrho}} \bullet \vec{\Gamma}_l^r)\right)\right] \end{aligned} \tag{3.8}$$

where ν_l^r is a binary indicator which takes 0 if $C_l^r = \emptyset$ and 1 otherwise and $\mathcal{L}$ the logarithm.

From the above partial log-likelihood, we learn the parameter $\vec{\Gamma}_j^r$ by using the

Newton-Raphson algorithm. The process is repeated each time the window slides. In this way, we always have all the parameters $\vec{\Gamma}_j^r$ updated with time. From Eq.(3.7) we can then calculate, the lifespan probability of a regime till the j^{th} instance of window $W^{\hat{\varrho}}$, by computing the cumulative probability $Surv(j \mid R^r)$ [62], as follows

$$Surv(j \mid R^r) = exp\Big\{ - \int_{\omega=1}^{\omega=j} h^r(\omega) d\omega \Big\} \tag{3.9}$$

3.4.3 Multiple time series forecasting

Given $\boldsymbol{S}$, a set of co-evolving time series, having constructed our mapping grid, to foresee the series values at a subsequent time, we need to know about regime transition from which we will be able to forecast the subsequent values of series in $\boldsymbol{S}$.

Regime transition

From the mapping grid $\boldsymbol{\mathcal{M}}$, for two consecutive window-stamps $W_j^{\hat{\varrho}}$ and $W_{j+1}^{\hat{\varrho}}$ we can calculate the probabilities $\mathcal{Q}_{j,(j+1)}$ of switching from behaviors R^α to behaviors R^β by using a counting process as follows:

$$\begin{aligned} \mathcal{Q}_{j,(j+1)} &= \left(\frac{|\boldsymbol{\mathcal{N}}_j^\alpha \cap \boldsymbol{\mathcal{N}}_{j+1}^\beta|}{|\boldsymbol{\mathcal{N}}_j^\alpha \cup \boldsymbol{\mathcal{N}}_{j+1}^\beta|} \right)_{\alpha,\, \beta \in \{0,1,...,K\}} \\ &= \left(\mathcal{Q}_{j,\,(j+1)}^{\alpha,\,\beta} \right)_{\alpha,\, \beta \in \{0,1,...,K\}} \end{aligned} \tag{3.10}$$

where $\boldsymbol{\mathcal{N}}_j^\alpha$ (resp. $\boldsymbol{\mathcal{N}}_j^\beta$) is the number of series presenting an R^α (resp. R^β) behavior at window-stamp $W_j^{\hat{\varrho}}$.

Note that, the component $\mathcal{Q}_{j,\,(j+1)}^{\alpha,\,\beta}$ relates again the instantaneous risk / probability of observing an R^β behavior while knowing that we were previously having an R^α behavior. In other words, the matrix $\mathcal{Q}_{j,(j+1)}$ relates the probability of suddenly switching from one behavior to another within the overall data set $\boldsymbol{S}$. However, if we want to calculate the probability of suddenly switching from one behavior to another for a single series $S^i \in \boldsymbol{S}$ we cannot uniquely exploit the overall risk phenomenon

calculated in Eq. (3.10).

In the absence of other series, for a specific series S^i, we believe that the risk of suddenly viewing an R^β behavior knowing that the previous exhibited behavior was R^α is calculated accordingly to its historical behaviors as follows:

$$\begin{aligned}\Pi(W_j^{\hat{\varrho}}, W_{j+1}^{\hat{\varrho}} \,|\, S^i) &= \left(\frac{\eta\left(R^\alpha \rightarrow R^\beta | \mathcal{T}r(S^i \,|\, W_{j+1}^{\hat{\varrho}})\right)}{|\mathcal{T}r(S^i \,|\, W_{j+1}^{\hat{\varrho}})| - 1} \right)_{\alpha,\beta \in \{0,1,...,K\}} \\ &= \left(\Pi_{j,(j+1)}^{\alpha,\beta}\right)_{\alpha,\ \beta \in \{0,1,...,K\}} \end{aligned} \tag{3.11}$$

where $\eta\left(R^\alpha \rightarrow R^\beta \mid \mathcal{T}r(S^i \mid W_{j+1}^{\hat{\varrho}})\right)$ is the number of time the sequence $\{R^\alpha,\ R^\beta\}$ appears in the trajectory $\mathcal{T}r(S^i \mid W_{j+1}^{\hat{\varrho}})$ and $|\mathcal{T}r(S^i \mid W_{j+1}^{\hat{\varrho}})|$ the length of the trajectory.

With relations given in Eq. (3.10) and Eq. (3.11), we can take into consideration the existence of other series in $\boldsymbol{S}$ and thus calculate the effective probability $\boldsymbol{\Theta}(W_j^{\hat{\varrho}}, W_{j+1}^{\hat{\varrho}} \,|\, S^i) = \left(\boldsymbol{\Theta}_{j\ ,\,(j+1)}^{\alpha,\beta}\right)_{\alpha,\ \beta \in \{0,1,...,K\}}$ of series S^i for suddenly switching from behaviors R^α to behaviors R^β as follows:

$$\boldsymbol{\Theta}_{j\ ,\,(j+1)}^{\alpha,\beta} = \begin{cases} 0 \quad if \quad \sum_{\star_1} \sum_{\star_2} \Pi_{j,(j+1)}^{\star_1,\star_2} \mathcal{Q}_{j,(j+1)}^{\star_1,\star_2} = 0 \\ \dfrac{\sum_{\star} \Pi_{j,(j+1)}^{\alpha,\star} \mathcal{Q}_{j,(j+1)}^{\star,\beta}}{\sum_{\star_1} \sum_{\star_2} \Pi_{j,(j+1)}^{\star_1,\star_2} \mathcal{Q}_{j,(j+1)}^{\star_1,\star_2}} \quad if \quad not \end{cases} \tag{3.12}$$

It is worth noting that, each component $\boldsymbol{\Theta}_{j\ ,\,(j+1)}^{\alpha,\beta}$ is a conditional probability that gives the risk of suddenly observing an R^β behavior in a given series $S^i \in \boldsymbol{S}$ knowing that this series was previously having an R^β behavior.

Forecasting

For each time series S^i observed from window-stamp $W_1^{\hat{\varrho}}$ to $W_b^{\hat{\varrho}}$, we want to predict the regime this series will exhibit at the next window-stamp $W_{b+1}^{\hat{\varrho}}$. To this end, we start by predicting the general switching $\mathcal{Q}_{b,(b+1)}$, the conditional transition $\Pi(W_b^{\varrho},\ W_{b+1}^{\varrho} \mid S^i)$ and the feature vector values $\{\vec{F}_{b+1}^{i,\hat{\varrho}}\}_{i \in \boldsymbol{\mathcal{N}}_{b+1}^r}$, $1 \leq r \leq K$. The

Algorithm 4: Forecasting.

Data: $\mathcal{M}$, $\boldsymbol{S}$, $W^{\hat{\varrho}}_{b+1}$; /* A mapping grid of size $(K+1)\times b$, set of time series we want to forecast values at window-stamp $W^{\hat{\varrho}}_{b+1}$. */

Result: $\widehat{\boldsymbol{S}}_{b+1}$, $\widehat{\mathcal{M}}$; /* Set of predicted values and the updated mapping grid. */

begin

- Define a new mapping grid $\widehat{\mathcal{M}}$ of size $(K+1)\times(b+1)$ where the $(K+1)\times b$ first values are equal to the one in $\mathcal{M}$ and the last column set to empty sets;
- $\widehat{\boldsymbol{S}}_{b+1} \leftarrow \emptyset$;
- Use a regression model as in Eq. (3.13) to predict the general transition $\mathcal{Q}_{b,b+1}$;

1- Update mapping grid:

begin

for $S^i \in \boldsymbol{S}$ **do**

- Using Eq. (3.2) get the trajectory $\mathcal{T}r(S^i|W^{\hat{\varrho}}_b)$, the historical features $\{F^{i,\varrho}_j\}^b_{j=1}$ of S^i and its latest behavior R^α;
- Use Eq. (3.13) to predict the feature $F^{i,\varrho}_{b+1}$;
- Use a regression model as in Eq. (3.13) to predict the transition $\Pi(W^{\varrho}_b, W^{\varrho}_{b+1} \,|\, S^i)$;
- Based on $\mathcal{Q}_{b,b+1}$ and $\Pi(W^{\varrho}_b, W^{\varrho}_{b+1} \,|\, S^i)$, use Eq. (3.12) to get the effective conditional transition $\boldsymbol{\Theta}_{b\,,\,(b+1)}(W^{\hat{\varrho}}_b, W^{\hat{\varrho}}_{b+1} \mid S^i) = \left(\boldsymbol{\Theta}^{\alpha,\,\beta}_{b\,,\,(b+1)}\right)_{\alpha,\,\beta\in\{0,\,1,\,\ldots,\,K\}}$;
- $l \leftarrow \underset{\beta\in\{0,1,\ldots,K\}}{argmax}\left(\boldsymbol{\Theta}^{\alpha,\,\beta}_{b\,,\,(b+1)}\right)$;
- $\widehat{\mathcal{M}}[l][b+1] \leftarrow \widehat{\mathcal{M}}[l][b+1]\bigcup\{i\}$; /* Populate cell $[l][b+1]$ of the new mapping grid. */

2- Forecasting:

for $S^i \in \boldsymbol{S}$ **do**

- Using Eq. (3.2) get the new trajectory $\mathcal{T}r(S^i|W^{\hat{\varrho}}_{b+1})$, the historical features $\{F^{i,\varrho}_j\}^{b+1}_{j=1}$ of S^i and its latest behavior R^α;
- Use Eq. (3.9) to calculate each regime survival till window-stamp W^{ϱ}_{b+1};
- Use Eq. (3.14) to forecast series values $\widehat{\vec{S}^{\,i,\,\hat{\varrho}}_{b+1}}$ at window-stamp $W^{\hat{\varrho}}_{b+1}$;
- $\widehat{\boldsymbol{S}}_{b+1} \leftarrow \widehat{\boldsymbol{S}}_{b+1}\bigcup\{\widehat{\vec{S}^{\,i,\,\hat{\varrho}}_{b+1}}\}$;

predicted matrices $\mathcal{Q}_{b,(b+1)}$ and $\Pi(W^{\varrho}_b, W^{\varrho}_{b+1} \mid S^i)$ are used for calculating the effective probability of suddenly changing $\boldsymbol{\Theta}(W^{\hat{\varrho}}_b, W^{\hat{\varrho}}_{b+1} \,|\, S^i)$ at window-stamp $W^{\hat{\varrho}}_{b+1}$ by applying Eq. (3.12). The predicted feature vector values are used for calculating the regime survival till window-stamp $W^{\hat{\varrho}}_{b+1}$ by applying Eq. (3.9). It is worth noting that, to predict the transition matrices values as well as the feature vector values, we use the regression form given as follows:

$$A_{b+1} = \lambda(A_1, ..., A_b) + \mu_{b+1} \tag{3.13}$$

where A_b could be a matrix or a feature vector at window-stamp $W_b^{\hat{\varrho}}$. $\lambda(A_1, ..., A_b)$ a given regression function and μ_{b+1} a white noise following a zero mean Gaussian function.

Knowing the survival $Surv(W_{b+1}^{\varrho}|R^r)$ of regime R^r and the conditional transition $\mathbf{\Theta}_{b,(b+1)}^{\alpha,\beta}$ of series S^i, we can then forecast the series S^i values at window-stamp $W_{b+1}^{\hat{\varrho}}$. If the most probable transition goes to one of regimes R^r, $1 \leq r \leq K$, then the predicted values $\widehat{\vec{S}_{b+1}^{\,i,\,\hat{\varrho}}}$ are calculated as:

$$\widehat{\vec{S}_{b+1}^{\,i,\,\hat{\varrho}}} = \vec{T}^{\,r} + \left[\overline{T}^{\,r} \pm \frac{\left(1 - \widehat{\mathcal{TP}}_{R^\alpha \to R^r}(W_{b+1}^{\hat{\varrho}})\right) \cdot \sigma(T^r)}{2}\right] \tag{3.14}$$

where $\vec{T}^{\,r}$ is the vector corresponding to the profile pattern of group C_{b+1}^r. $\overline{T}^{\,r}$ is the mean value of the profile pattern T^r . $\sigma(T^r)$ is the standard deviation of the profile pattern T^r. $\widehat{\mathcal{TP}}_{R^\alpha \to R^r}(W_{b+1}^{\hat{\varrho}})$ is the highest transition probability for S^i (corresponding to the regime it is most likely to switch to) given as:

$$\widehat{\mathcal{TP}}_{R^\alpha \to R^\beta}(W_{b+1}^{\hat{\varrho}}) = \underset{R^\beta \in \{R^0, R^1, ..., R^K\}}{max} \left\{\mathcal{TP}_{R^\alpha \to R^\beta}(W_{b+1}^{\hat{\varrho}})\right\} \tag{3.15}$$

with $\mathcal{TP}_{R^\alpha \to R^\beta}(W_{b+1}^{\hat{\varrho}})$ the transition probability which takes into consideration the fact that a regime might still be observed or not. This transition is calculated as follows,

$$\mathcal{TP}_{R^{\alpha}\rightarrow R^{\beta}}(W_{b+1}^{\hat{\varrho}}) = \begin{cases} * \;\; if \quad 1 \leq \alpha, \beta \leq K \\ \boldsymbol{\Theta}_{b\,,\,(b+1)}^{\alpha,\,\beta} \cdot \Big(1 - Surv(W_{b+1}^{\hat{\varrho}} \,|\, R^{\beta})\Big) \\ * \;\; if \quad 1 \leq \alpha \leq K, \;\; \beta = 0 \\ \boldsymbol{\Theta}_{b\,,\,(b+1)}^{\alpha,\,0} \\ * \;\; if \quad 1 \leq \beta \leq K, \;\; \alpha = 0 \\ \boldsymbol{\Theta}_{j\,,\,(j+1)}^{0,\,\beta} \cdot \Big(1 - Surv(W_{j+1}^{\hat{\varrho}} \,|\, R^{\beta})\Big) \end{cases} \tag{3.16}$$

If the most probable transition goes to none of the regimes R^r, $1 \leq r \leq K$ (i.e., switching to $\mathcal{N}_{b+1}$), then the forecast will correspond to a non-observation.

To forecast regimes at window-stamps later than $W_{b+1}^{\hat{\varrho}}$, we repeat the same process. By running Algorithm 4 we can forecast further regimes that will be exhibited by a given series.

3.5 Experiments

3.5.1 Data description

We evaluated the proposed approach on ten data sets: one synthetic data and nine real-world data. In the synthetic data set (SyD), the values of series were generated by 5 functions as follows:

$$Gn(t) = \sum_{r=1}^{5} a_r \, \eta_r(t) \, fct_r(t) \tag{3.17}$$

where $a_r \in \{0, 1\}$ ($\sum_{r=1}^{5} a_r = 1$) is a parameter that permits to have one regime exhibited at a time by the generated series. $\eta_k(t) \in [0, 1]$ a probability function given in such a way that, at each time-stamp, exhibited regimes by series of the data collection should always respect the following constraint $\sum_{r=1}^{5} \eta_r(t) = 1$. Functions $fct_r(t)$ are defined as follows:

Table 3.2 – Data set description.

Data	#Series	Length	Data	#Series	Length
SyD	450	$1,125$	RDS	70	$2,844$
ELD	100	$8,784$	EQD	461	512
SDSS	114	$5,112$	EOG	724	$1,250$
GPS	50	$9,873$	Pig	312	$2,000$
ESR	495	$4,094$	CCD	166	$4,307$

$$fct_r = \begin{cases} fct_1(t) = cos(\frac{2\pi t}{5}) + cos(\pi(t-3)) \\ fct_2(t) = sin(\frac{\pi t}{2} - 3) - sin(\frac{\pi t}{6}) \\ fct_3(t) = tan(\frac{\pi t}{2} - 3) - \frac{1}{2}cos(\frac{\pi(t-3)}{6}) + cos(\pi(t-13)) \\ fct_4(t) = sin(\frac{\pi t}{2} - 3) \times cos(\frac{\pi(t-3)}{6}) \times cos(\pi(t-13)) \\ fct_5(t) = cos(\frac{3\pi t}{5}) + sin(\frac{2\pi t}{5} - t) \end{cases} \tag{3.18}$$

From the EnerNOC website [2] (EnerNOC GreenButton Data), we collected the whole year 2012 customer electricity load (ELD) data. The reading of each customer was set to 1 hour. From the Smart Dataset for Sustainability (SDSS) [3], we collected apartment electricity consumption for the period ranging from 2016-01-01 to 2016-07-31 (213 days), with a read set to 1 hour. From the UCI archive [4], we collected the gesture phase segmentation (GPS) and the Epileptic Seizure Recognition data (ESR). From the UCR Time Series Classification Archive [5], we collected real world data sets from the rock data set (RDS), the earth quake data (EQD), electrooculography signal (EOG), pig data (Pig) and chlorine concentration data (CCD). As summarized in Table 3.2, the selected data sets have different characteristics, which allows us to conduct an objective empirical study to illustrate the suitability of our proposal. In these data sets, the number of series varies from 50 to 750, while the length of the series varies from 512 to $9,873$.

2. https://open-enernoc-data.s3.amazonaws.com/anon/index.html
3. http://traces.cs.umass.edu/index.php/Smart/Smart
4. https://archive.ics.uci.edu/ml/datasets/
5. https://www.cs.ucr.edu/~eamonn/time_series_data/

Note that, to have more cases with missing series values, for all data sets, we randomly select series and time intervals from which the value will be deleted. To do this we start by randomly selecting a number of series as follows, $Delete(\boldsymbol{S}) = \{S^i \mid i \in randint(1,\ N,\ \ randint(1,\ \frac{N}{30}))\}$, where $randint(a, b, c)$ is a uniform random function that generates c integers within the interval $[a,\ b]$, whereas $randint(a,\ b)$ generates one integer within the interval $[a,\ b]$. Then, randomly select time interval within which series values in $Delete(\boldsymbol{S})$ will be deleted as follows, $Delete(m) = \Big\{\{(t_j,\ t_{j+\varrho})\} \mid \varrho \in randint(\frac{m}{30},\ \frac{m}{10}),\ j \in randint(1,\ m - \varrho)\Big\}$, with m the length of series in $\boldsymbol{S}$.

In what follows, we start by illustrating the various important steps which are required for predicting the series values at further time points. Later on, we present the forecasting results and assess the accuracy of our model compared to some well-known state-of-the-art time series forecasting methods.

3.5.2 Data scanning

We identify the optimal window size for each of the ten data sets considered in the experiments. Specifically, we applied Algorithm 3 to build and split the networks related to subseries at each instance of a given window; and to identify the optimal window size and the groups of subseries exhibiting the different hidden regimes.

Window's size & regime identification

Based on the relation given in Eq. (3.3), we estimate how likely co-evolving series in each of our data sets do vary. For the purpose of illustration, in Fig. 3.6, we have the series variation with respect to the size of the window in the synthetic data set (SyD) and the Chlorine Concentration data set (CCD). As it is depicted, we see that the series variation in all cases is rapidly dropping when the size of the window is small. When the size of window is getting larger, the series variation tends to converge to a constant value. The values associated to the vertical red lines in Fig. 3.6 show the optimal window sizes for which we can better track regimes over time. We can note that, the obtained window size varies according to the data.

Using the optimal window sizes $\hat{\varrho}$, we then look for the distinct regimes exhibited by

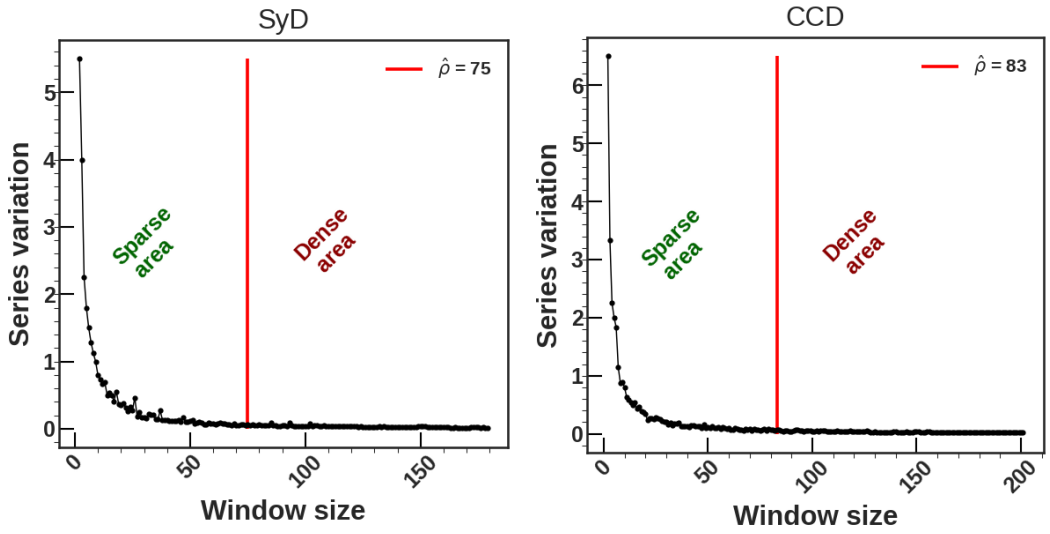


Figure 3.6 – Example of series variation in SyD and CCD data sets. The vertical red line gives the optimal window size in each data set.

series. In Fig. 3.7, for each row, we have the distinct regimes exhibited by co-evolving series in SyD, ESR and CCD data sets respectively. As can be seen, when running Algorithm 3, we found that, the number of regimes K in each of SyD, ESR and CCD data sets are respectively 5, 4 and 3. Note that, for the specific case of SyD the number of regimes $K = 5$ corresponds again to the number of functions used in Eq. (3.18) to generate the synthetic data. For real cases, we do not have a ground truth for validating the obtained regimes. Fortunately, according to the forecasting which depends on the identified regimes, we will be able to better validate whether the identified regimes are the good ones. In Table 3.3, we provide optimal window size and the number of regimes for all the tested datasets.

Table 3.3 – Window sizes and number of exhibited regimes.

Data	$\hat{\varrho}$	#Regimes	Data	$\hat{\varrho}$	#Regimes
SyD	75	5	RDS	109	5
ELD	94	2	EQD	53	3
SDSS	75	2	EOG	181	5
GPS	103	8	Pig	75	4
ESR	98	4	CCD	83	3

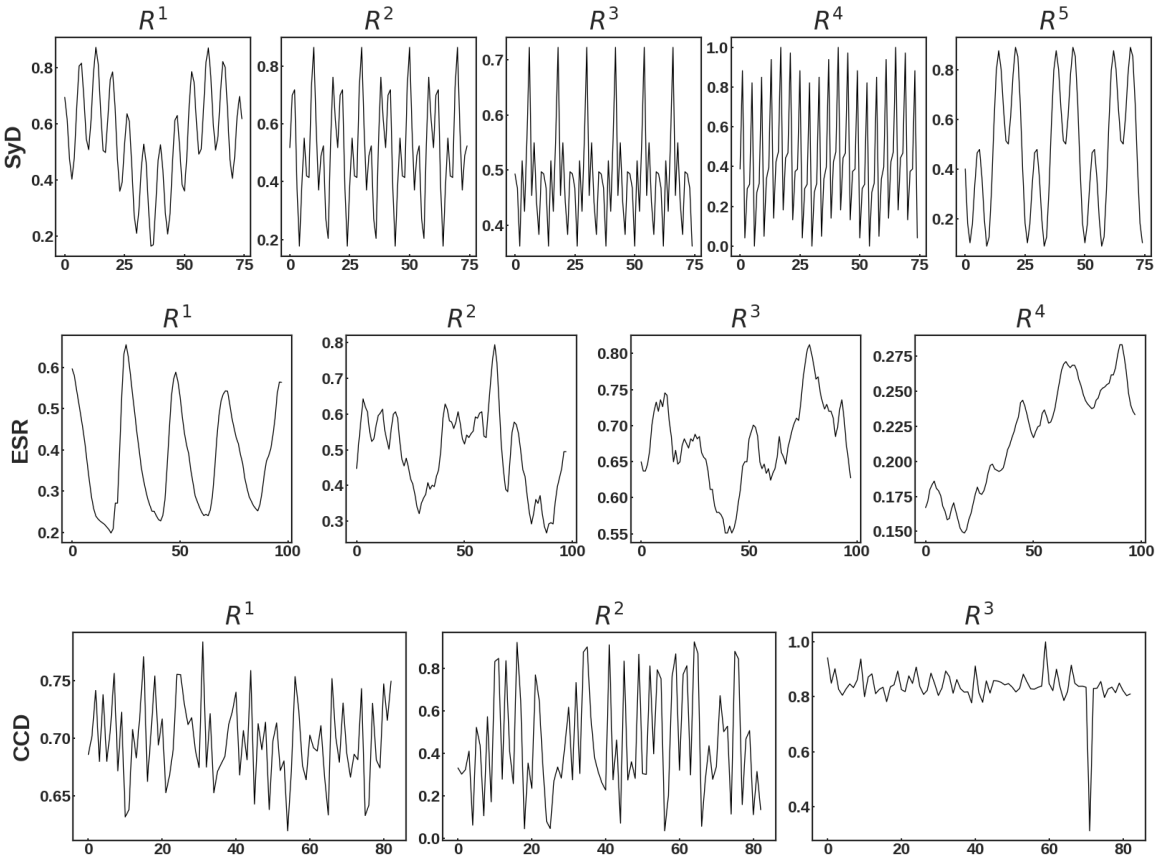


Figure 3.7 – Example of identified regimes in SyD, ESR and CCD.

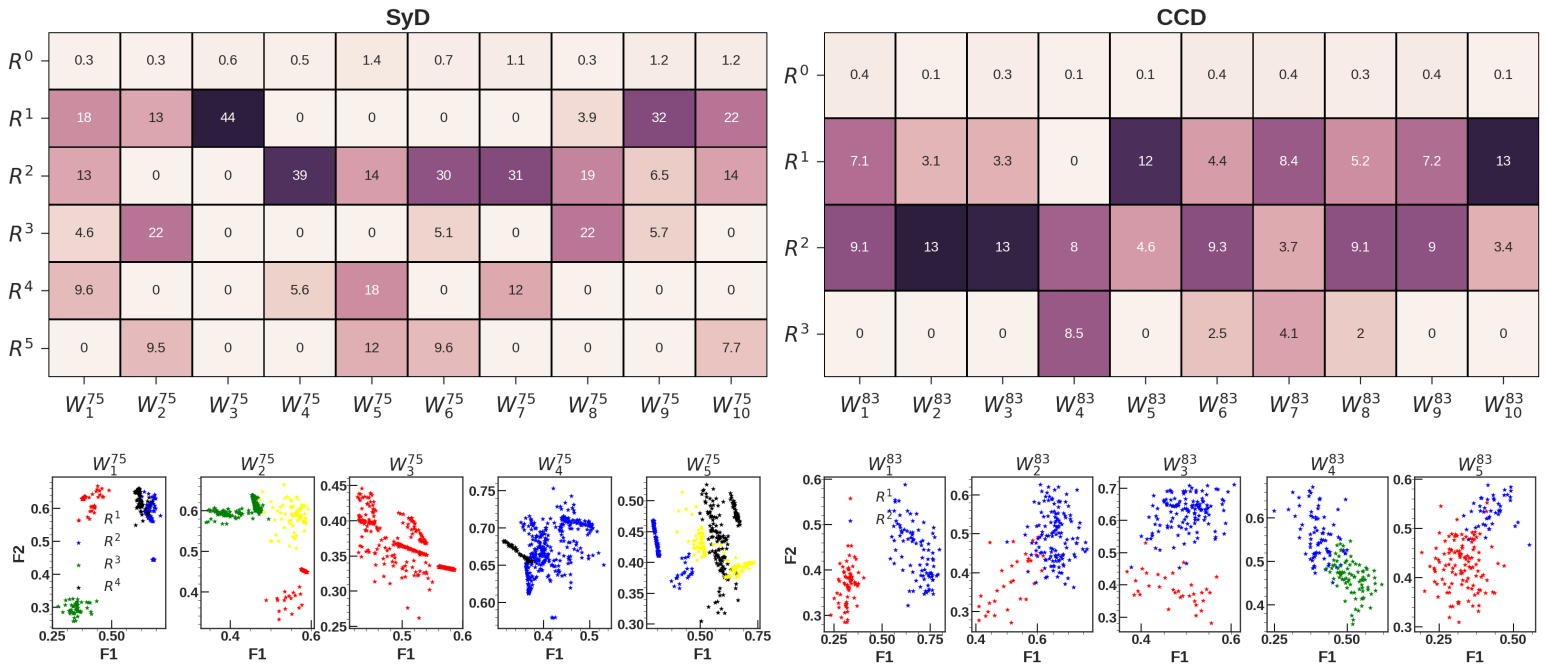


Figure 3.8 – Example of mapping grids in SyD and CCD at the 10 first window-stamps. Below each mapping grid, we have the embedding in a 2-D projection at the 5 first window-stamps.

3.5.3 Mapping Grid

Grid construction

With the discovered regimes in Fig. 3.7, we now identify series that contain each of these regimes at the different window instances which constitute our mapping grid. And, from each cell of the mapping grid, we learn features that encompass the subseries relationships and which may help in the understanding regime lifespans. For purpose of illustration, in Fig. 3.8 we have two examples of mapping grid in SyD and CCD with their corresponding embedding which stands for the learned features. The heatmaps show only the 10 first window-stamps in each cases. In these heatmaps, the darker the cell, the greater the number of series exhibiting the regime. It can be seen that the colors of heatmaps corresponding to data sets depicted in Fig. 3.8 change at different window-stamps. This shows that there are indeed series that contain different regimes at different window-stamps. We can note that, at some rows of our grids (e.g., R^2 and R^1 in SyD and CCD respectively), the color is gradually becoming dark when reading from left to right showing that the corresponding regime is more and more observed by most of series as time evolves. In the same way, for some rows of our grids (e.g., R^1 and R^2 in SyD and CCD respectively), the color becomes light when reading from left to right showing that the corresponding regime is less and less exhibited by series as the time evolves.

Features learning

To learn features from each cell of the mapping grid, we make use of a graph Autoencoder. Specifically, knowing that each cell of the mapping grid is populated by a subnetwork G_j^r, we use this subnetwork as the input of our graph autoencoder for generating an embedding $\{F_j^{i,\hat{\varrho}}\}$ which captures both the topological structure of G_j^r and its content. The obtained embedding is then used as the features of the mapping grid cell. It is good noting that, the embedding of each node of the subnetwork G_j^r corresponds to a $8-dimensional$ vector. In our architecture, we use as encoder a stack convolutional neural network with two hidden layers having 16 and 8 filters respectively.

Below each of the mapping grids in Fig. 3.8 we have the corresponding 2-D features

projection at five consecutive window-stamps mapping grids of series in SyD and CCD respectively, below, we have 2-D plots giving an overview of the series exhibiting regimes at the two first window-stamps in SyD and CCD.

Looking at the 2-D plots, we can note that, the embedded space is characterized by the presence of fairly distinguishable structures that allow us to discriminate between the various regimes. Moreover, from the embedded space, we also note that, some regimes are not represented. For example, regime R^1 (tagged in red) in CCD and regime R^5 (tagged in yellow) in SyD are not observed in the embedded space at the first window-stamp.

Having constructed our mapping grids, we can now exploit the embedded space as our learned features at each window-stamp to better understand the time duration for which a regime may be persist or not. In what follows, we illustrate the regime lifespans.

3.5.4 Regime survival

Recall that, to study regime lifespans, we are making use of the Cox regression model for which we need to identify the parameters. In this paper, we have devised our Cox regression (Eq. (3.7)) in such a way that it could integrate the time-dependent features as well as time-dependent parameters. From what we have presented so far, we can effectively say that, the learned features from our mapping grids do change with time. This is why we believe that, their importance may change as well from one time to another. To demonstrate this, in Fig. 3.9, we have an example of the extracted Cox parameters over five consecutive window-stamps for regimes identified in SyD. As can be seen, the bar plots are changing with time, showing that the role of each feature is changing over time.

Based on the Cox parameters and the feature vectors corresponding to series in different groups of subseries at different window-stamp, we can thus evaluate the lifespan probability of each regime. Recall that, in all our experiments, of each regime R^r, we make use of the Gamma distribution as the baseline hazard function ($\gamma^r()$ in Eq. (3.7)). The gamma distribution parameters are estimated by using the experimental distribution (the one corresponding to the number of series not exhibiting regime R^r

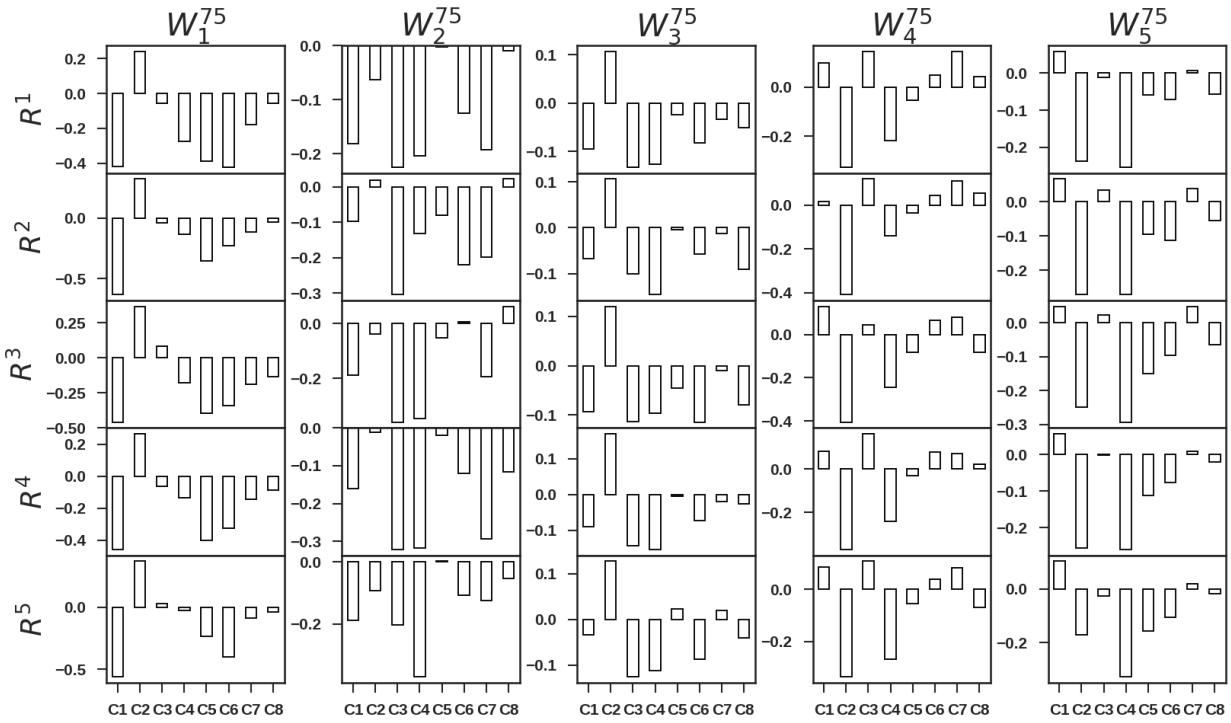


Figure 3.9 – Cox coefficients at the five first window-stamps in SyD data sets. The variation observed at each window-stamp for each regime demonstrates that the features are changing with time. Moreover, it shows that, the number of series exhibiting each of the regime is changing.

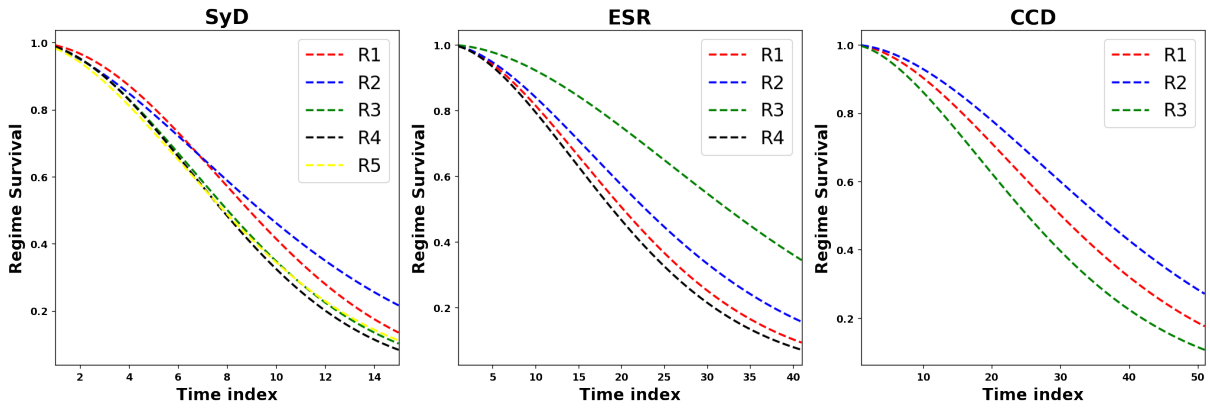


Figure 3.10 – Examples of regime lifespans in SyD, ESR and CCD.

over time).

Using the baseline function, the learned features, and the Cox parameters, we now apply Eq. (3.9) to have the probability for a regime to no longer be exhibited over time (the regime lifespan). In Fig. 3.10 we have examples of regime lifespans in SyD, ESR and CCD. As depicted, we note that, as time evolves, the probability for a regime to not be exhibited is quickly decreasing after a certain number of time in all cases. For some regimes, this lifespan probability is dropping very fast compare to other than remain stable after a certain time. In the ESR data set, for instance, the plot shows that the probability for regimes R^1 R^2 and R^4 to not be exhibited drops faster than R^3. This means that, as time evolves, regime R^3 will be less exhibited by the overall series in ESR compare to other regimes. The same interpretation is done in SyD and CCD data sets where we can note that, regime R^2 (in SyD and CCD data sets) has a higher probability to no longer be exhibited compare to other regimes.

3.5.5 Forecasting

For each data set, we consider all window-stamps to be known except the last four window-stamps for which we have to predict the series values. For the known window-stamps, we start by extracting series trajectories (with the corresponding features) and their respective conditional transitions. From extracted information, we will train a generative model in such a way that, this last one should be capable to predict features and transition probabilities at further window-stamps.

Series trajectory & regime shift

To track series behavior, we utilize our *mapping grids*. Fig. 3.11 presents examples of time series trajectories from the first window-stamp till the tenth window-stamp in ESR and CCD data set. The network below each trajectory corresponds to the transition matrix at the four first consecutive window-stamps. Each circle of the trajectory indicates the regime exhibited at a specific window-stamp. Here, from the sequence of circles, it can truly be said that the exhibited regimes are non-contiguous over time. The remark can be done when looking at the transition matrices for each of the series. Within two consecutive window-stamps, we can note that the transition

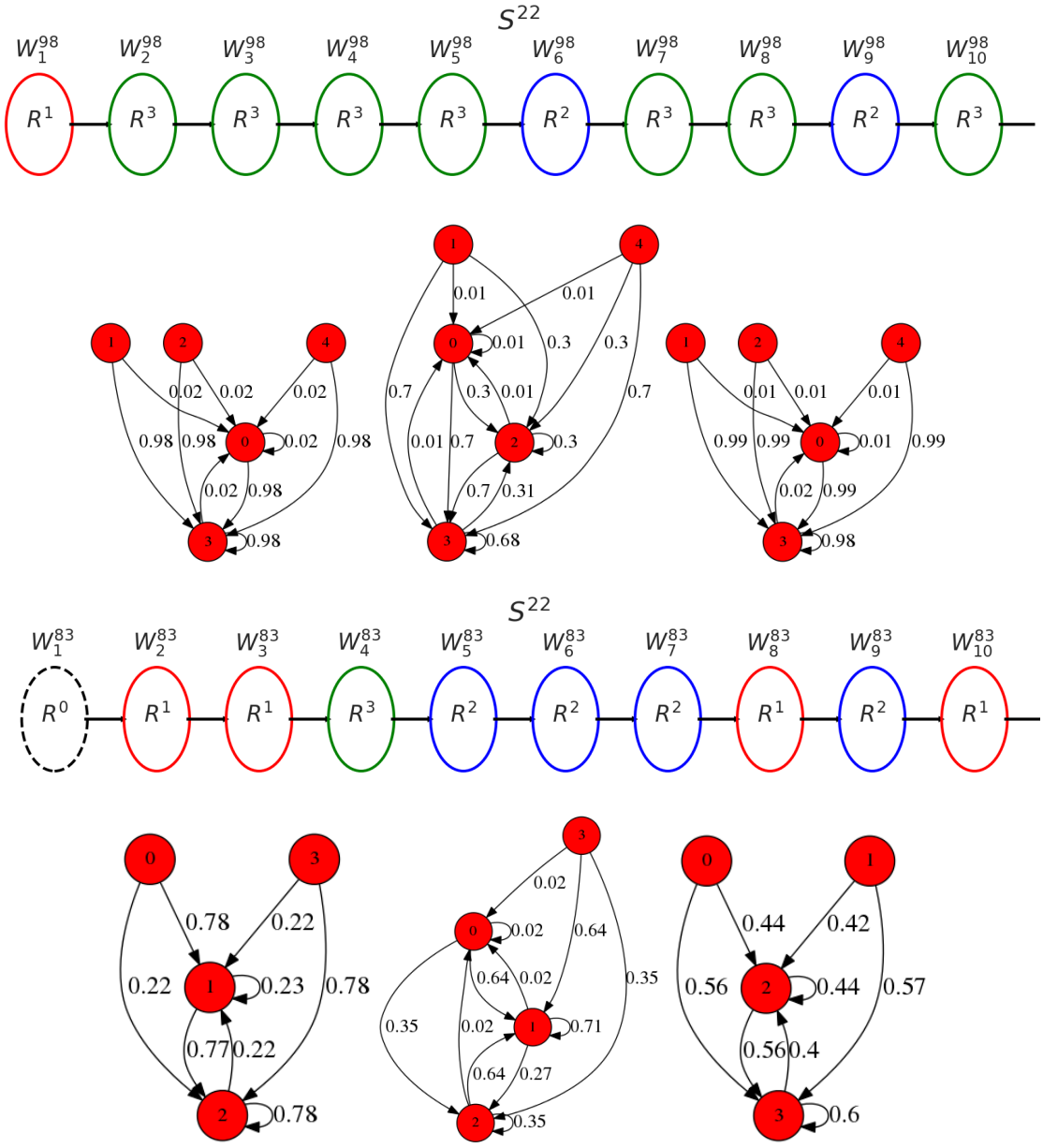
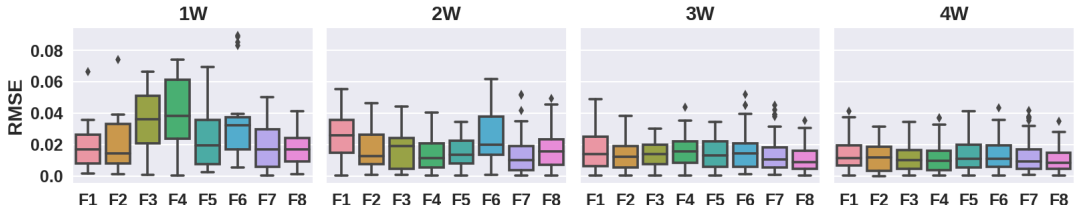
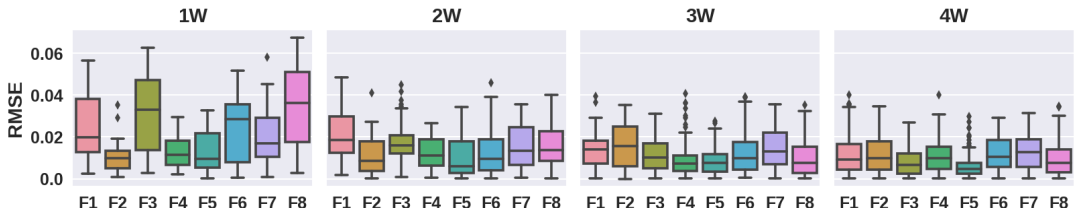


Figure 3.11 – Trajectory followed by series S^{22} and in ESR and CCD data sets at the ten first window-stamps ($W_1^{98} - W_{10}^{98}$ in ESR and $W_1^{83} - W_{10}^{83}$ in CCD). Graphs below each trajectories depict the three first abrupt transition matrices ($\mathbf{\Theta}_{1-2} - \mathbf{\Theta}_{3-4}$) extracted from each of the trajectories of S^{22}.

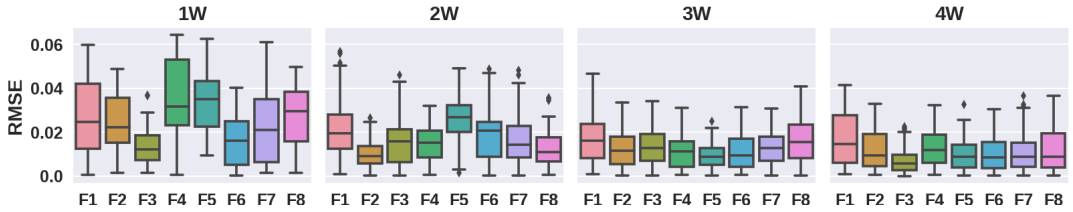
probability between regimes is changing with time. Moreover, the different transition



(a) SyD



(b) ESR



(c) CCD

Figure 3.12 – Example of error made (RMSE) when predicting features values at 1, 2, 3 and 4 window-stamps ahead in SyD, ESR and CCD.

matrices explain the behavioral sequence in each of the cases.

To be more clear, let take for instance the sequence S^{22} in the CCD data set. From window-stamp W_1^{83} to W_2^{83}, we note that series S^{22} has passed from behavior R^0 to R^1 and that the conditional transition $\mathbf{\Theta}_{1,2}^{0,1} = 0.78$ is greater than the $\mathbf{\Theta}_{1,2}^{0,2}$. This means that the transition matrix $\mathbf{\Theta}_{1,2}$ can well explain the changing behavior that happened from W_1^{83} to W_2^{83}. At the next consecutive window-stamps (W_2^{83} to W_3^{83}), we note that the behavior remains the same. If it was to use the same transition matrix (the one used at consecutive window-stamps W_1^{83} to W_2^{83}, $\mathbf{\Theta}_{1,2}$) the series was to switch to regime R^2 which does not correspond to the sequence. When instead looking at the transition matrix $\mathbf{\Theta}_{2,3}$, it corroborates with the fact that the series keeps the same behavior at window-stamp W_3^{83}. The same observations are made at further window-stamps in all cases (ESR and CCD).

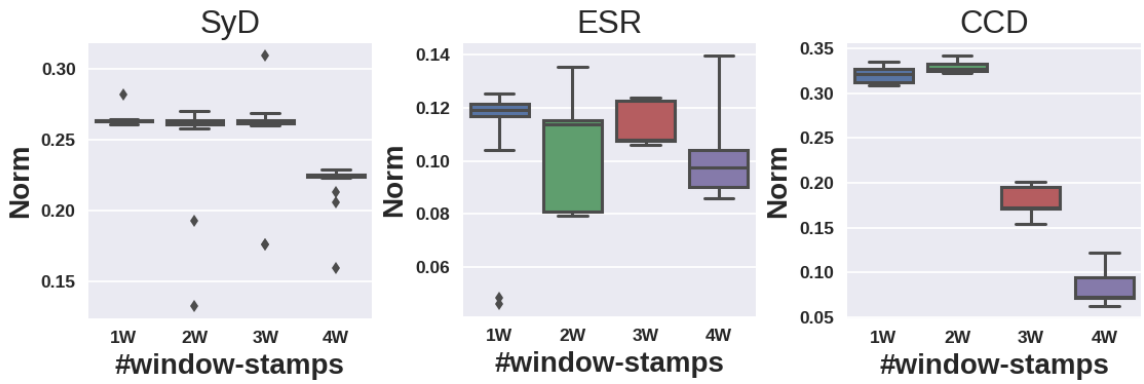


Figure 3.13 – Example of error made (the Frobenius Norm) when predicting transition matrice values at 1, 2, 3 and 4 window-stamps ahead in SyD, ESR and CCD.

Predicting feature vectors & abrupt transitions

To predict the feature vectors, any regression model can be put in practice to perform this task. In this paper, we are using the $k-$nearest neighbor regression (KNNR) because it is fast and adaptive when series incorporate linear as well as non-linear behavior [99], [100]. In figures 3.12 and 3.13 we have examples of the error made when predicting features and transition matrix values in SyD, ESR and CCD data sets respectively. As depicted in figures 3.12 and 3.13, we can note that, with respect to the Root Mean Square Error (RMSE), the KNNR can predict values with low error rates at 1, 2, 3 and 4 window-stamps ahead. The reported RMSE results show that the KNNR model is accurately predicting the values at a subsequent time. We note also that, when predicting feature values as well as transition matrices, the error made is not drastically increasing as it could be expected. Instead, we note that, when predicting values at further window-stamps, the error quite remains stable if not decreasing. This could be explained by the fact that the feature values (resp. transition matrix values) are less fluctuating as time evolves.

Forecasting missing information: Given a time series S^i, to evaluate if this regime will exhibit missing values, we use the binary codificaiton 0 and 1 to refer respectively to the absence and presence of a missing value. Hence, for N series for which we predicted the presence or absence of missing value, we will have a sequence of N binary values. The obtained predicted sequence is then compared to the real

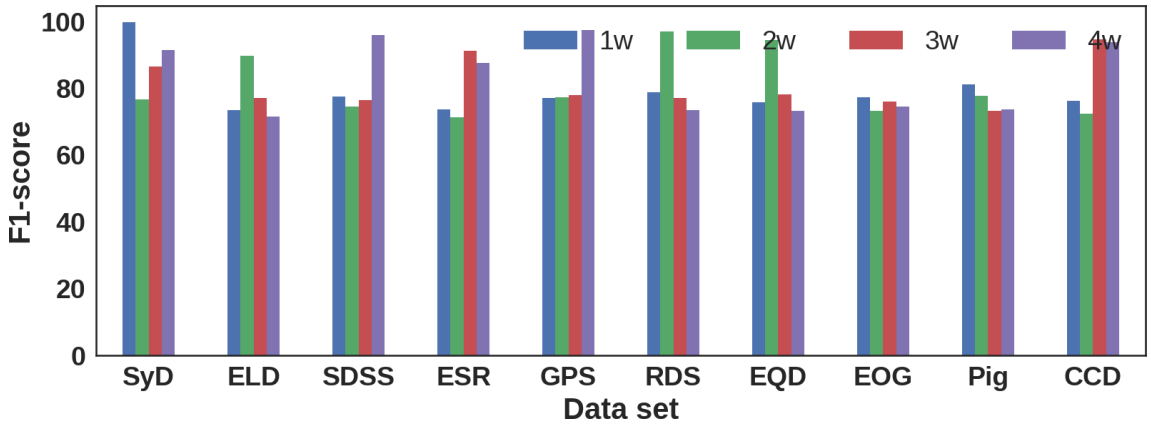


Figure 3.14 – Results in predicting missing values with respect to the F1 score.

sequence relating to the fact that missing value is exhibited or not by the N time series. We use the F1 score to evaluate the accuracy of the proposed model in predicting missing values. In Fig 3.14, we have the F1 score of the proposed model in predicting missing values in 1, 2, 3 and 4 window-stamps ahead respectively in each of the data set. Reported results show that the model can predict missing values with an average F1 score of 76% in general.

Forecasting series values: For the purpose of illustrations, we randomly picked three time series from SDSS, RDS, and EQD data sets and present how the model does forecast the series values are subsequent time. From Fig. 3.15, we can see how the model enables to follow the series behavior over time. From the reported values on each plot, for these cases, we can claim that the tracking of the regime over time did help in predicting series values at a subsequent time. Moreover, from the reported error, we can say that the identified regimes do correspond to the most exhibited within the ensemble of each time series. It should be noted that, though the trend can capture by the proposed approach, the predicted values at some time interval may much deviate from the truth series values. This is can be explained by the regime which can be quite modified over time.

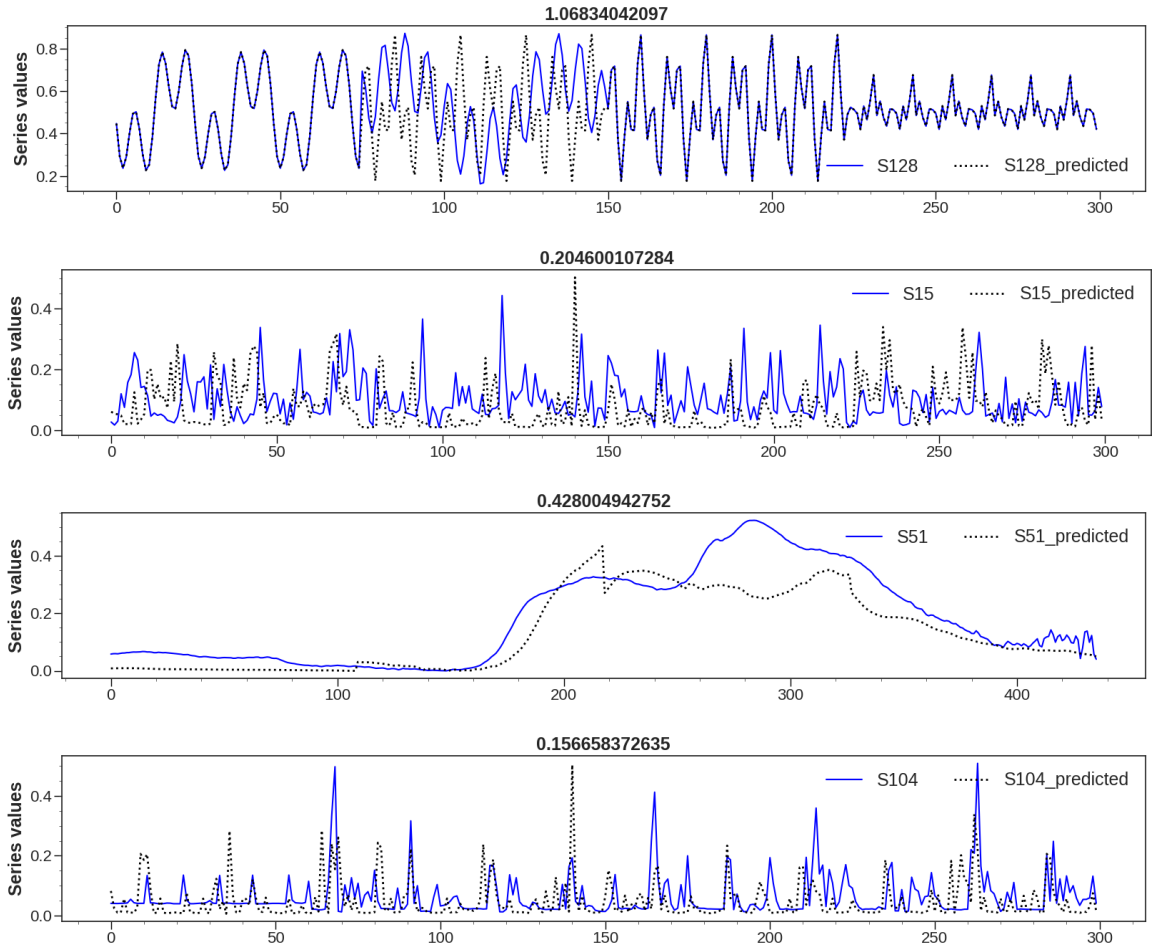


Figure 3.15 – Example of series prediction. One series was extracted respectively in SyD, SDSS, RDS, and EQD data sets. The forecasting is over the last four window-stamps in each of those data sets. The floating number on each plot corresponds to the Root Mean Square Error. The solid line are the current series values whereas the dash lines are the predicted series values.

Table 3.4 – Accuracy of the proposed approach against state-of-the-art methods evaluated by the Root Mean Square Error.

	SyD				ELD				SDSS				ESR			
Model	1w	2w	3w	4w	1w	2w	3w	4w	1w	2w	3w	4w	1w	2w	3w	4w
Proposed	**0.13**	**0.62**	**0.89**	**1.20**	0.47	1.08	**1.26**	**1.75**	0.28	0.78	**1.71**	1.84	**0**.81	**1**.06	**1**.36	**1**.89
KNNR	9.33	11.47	12.11	13.64	0.69	2.54	5.67	6.01	0.37	1.17	3.56	5.66	3.57	4.16	6.93	7.02
OrbitMap	3.96	3.98	3.11	4.48	0.59	0.94	1.46	**1.75**	0.23	**0.69**	2.06	**1**.84	1.54	2.04	2.36	2.67
CNN	8.75	13.64	13.62	16.75	0.37	0.58	3.64	3.64	0.35	1.16	3.29	3.66	2.06	2.36	3.24	3.87
VAR	13.26	14.87	18.12	18.76	0.43	2.47	6.48	6.68	**0.14**	1.67	5.61	6.02	3.69	4.39	5.39	7.17
NEF	3.61	4.67	9.72	10.43	**0.27**	**0.55**	**1.26**	**1.75**	0.31	0.94	2.11	2.54	2.68	3.94	4.28	4.67
Prophet	18.64	19.02	21.14	22.07	0.39	1.09	**1.26**	2.01	0.47	1.24	2.54	2.87	4.61	6.17	6.33	6.89
	GPS				RDS				EQD				EOG			
Proposed	2.14	2.26	**2**.37	**2**.85	**1**.25	**1**,47	**3**.12	**3**.88	**1**.11	**2**.44	**2**.77	**3**.12	**1**.86	**2**.36	**2**.84	**3**.48
KNNR	4.68	5.12	5.42	7.38	5.87	6.47	6.75	6.84	3.18	3.25	3.83	4.76	3.67	4.16	4.67	5.13
OrbitMap	**1**.98	**2**.13	**2**.37	3.03	4.12	5.28	5.39	5.95	1.84	**2**.44	3.12	3.59	3.12	3.33	3.44	**3**.48
CNN	2.94	3.17	3.95	4.25	4.44	6.13	6.95	8.25	2.87	3.02	3.23	3.89	4.52	4.62	4.98	5.26
VAR	3.15	3.98	4.23	4.93	6.18	6.77	7.04	7.36	3.12	3.22	3.48	3.89	3.77	4.20	4.25	4.97
NEF	2.56	2.94	3.74	3.95	4.02	4.59	4.78	5.17	2.63	2.75	3.02	3.25	3.09	3.21	3.56	4.12
Prophet	3.18	4.01	4.95	5.27	5.16	5.92	6.32	6.92	2.67	4.75	5.91	6.38	3.97	4.45	4.67	5.21

	Pig				CCD			
Model	1w	2w	3w	4w	1w	2w	3w	4w
Proposed	**1**.91	**2**.13	**2**.68	**3**.15	**0**.89	**1**.95	**2**.64	**2**.75
KNNR	**1**.91	2.56	4.23	5.97	2.34	2.89	3.18	3.95
OrbitMap	2.75	2.86	3.12	3.23	1.17	**1**.95	**2**.64	**2**.75
CNN	3.18	3.47	3.88	4.08	1.28	**1**.95	2.87	3.19
VAR	3.87	4.08	4.26	4.32	1.89	2.45	3.86	4.12
NEF	2.25	2.47	2.79	**3**.15	**0**.89	2.54	2.97	3.18
Prophet	3.09	3.54	3.94	4.17	1.28	2.43	2.97	3.29

Benchmark comparison

To further show the capability of the proposed approach in forecasting series values, we conduct a comparative analysis of our proposed model against well-known methods including the aggregated time series forecasting proposed in [101] where we make use of a network-based clustering to group series that at similar then apply an LSTM over the extracted representative pattern to predict the series values. We have chosen a network-based clustering because it will permit us to compare our approach (which requires a network-clustering as well) to one that does not use any features for forecasting series values (will use the term NEF for network-based forecasting to categorized this approach). We also compare the model to other methods including a k-nearest neighbor regression (KNNR), Prophet [102], OrBitMap [35], the Convolutional Neural Network (CNN) and the Vector AutoregRession (VAR). It is important to note that, for the LSTM, we set the number of input neurons to the size of each window. In addition to the input layer, two other layers, with 60 and 15 neurons respectively, were considered. For the CNN model, we used a stack model where the respective layers have 120, 30, and 15 filters.

To compare the above models with the proposed approach, we only work with series for which we have no missing values. Table 3.4 depicts the average Root Mean Square Error in each of the data sets when forecasting series values at one, two, three, and four window-stamps ahead. As depicted, all of the methods do have less performance when forecasting several time-stamps. However, we note that the KNNR and the CNN which are integrated into our model are fewer performings than the proposed approach. This may justify the reason for searching for more information that may explain the changing behavior observed within the series. These behaviors exhibited by series are hard to capture when looking at the performances of the VAR model which does not incorporate the changing behavior series may exhibit at different time. The same observations are made for the Prophet model though it is capable of capturing seasonality within series. However, when the various seasonality is non-contiguous the model faces difficulties in predicting subsequent values. The OrbitMap in contrary present interesting results but remains less competitive compared to the proposed approach. It should be noted that, in the particular cases of ELD and SDSS data sets, all models are quite having close performances. This can be explained by the fact that

we have only two regimes in both cases. Moreover, the two data sets (ELD and SDSS) include customers' electricity consumption which in general use to have persisting behaviors during a giving season.

3.6 Conclusion

In this paper, we devised a principled approach for mining and modeling regime shifts in an ecosystem comprising plenty of time series. The approach enables us to forecast the series values of this ecosystem at subsequent times. A notable feature of the proposed approach is that it can handle multiple time series dominated by a hidden endless regime shift mechanism. This is accomplished by devising a mapping grid, from which Cox regression and time-dependent transition matrices are utilized to determine regime shift probabilities. The proposed approach is able to automatically uncover various hidden regimes without any prior knowledge about the series under investigation, thanks to the sliding window mechanism we have proposed, which allows the automatic identification of the optimal window size associated with regimes. While the proposed approach achieves good results, some aspects still require further work in order to improve the model. For instance, enhancing the proposed approach in such a way that it should be capable of continuously discovering regimes as time evolves.

Chapitre 4

Prévision des charges électriques par analyse comportementale des consommateurs

Résumé

Pour prédire les consommations électriques des clients, la majorité des approches existantes utilise une méthodologie globale où l'historique des consommations électriques de ces clients est directement prise en compte. Dû au fait qu'un client peut changer sa façon de consommer à tout moment, en adoptant une telle méthodologie globale, on ne saurait tenir compte de ces changements comportementaux des clients dans leurs consommations électriques. Ne pas tenir compte du changement comportemental du client dans sa consommation électrique pourrait négativement influencer sur la précision du modèle de prédiction utilisé. Prévoir les charges électriques des clients tout en tenant compte de leurs comportements variables est un réel défi pour de nombreux modèles de prédiction existants. Dans ce chapitre, nous proposons un modèle capable de tenir compte du comportement variable du client dans son processus de consommation d'électricité. Le modèle proposé est basé sur les structures de graphes dynamiques où des sous-structures faisant office de groupes de consommateurs sont détectées et suivies dans le temps. Le suivi des groupes de consommateurs est ici fait par le biais de l'analyse de survie et un réseau de neurones récurrent à mémoires variables (LSTM). Les expérimentations faites sur les données réelles démontrent la pertinence du modèle proposé.

Commentaires

Dans ce chapitre, il est intégralement présenté l'article intitulé *Mining Customers' Changeable Electricity Consumption for Effective Load Forecasting*, article soumis dans la revue *ACM Transaction on Information Systems and Technology* (TIST) en avril 2019. Etienne G. Tajeuna, Mohamed Bouguessa et Shengrui Wang sont les principaux auteurs du travail élaboré dans cet article.

Dans l'élaboration de cet article, j'ai (Étienne G. Tajeuna) contribué en tant que premier auteur dans la conception du modèle théorique y compris les expérimentations validant le modèle théorique proposé. La rédaction et la vérification des équations quant à elles ont toutes été accompagnées par mes directeurs de recherche Pr. Shengrui Wang et Pr. Mohamed Bouguessa.

Le travail élaboré dans ce chapitre est une suite de notre travail présenté dans l'article intitulé *A Network-Based Approach to Enhance Electricity Load Forecasting*, publié en 2018 dans *IEEE International Conference on Data Mining Workshops* (ICDMW). Dans cet article, nous supposons que les profils de consommation des clients dépendent du fait qu'il s'agisse d'une fin de semaine (samedi et dimanche) ou alors d'une semaine de travail (lundi à vendredi). Suivant ses deux types d'intervalle de temps, nous identifions les différents profils que pourraient avoir les clients. Les profils sont identifiés par une approche de segmentation basée sur les graphes. Les profils représentés par les sous-structures de graphes sont suivis par rapport au temps et par la suite exploités pour prédire les consommations des clients. Pour prédire, seule l'évolution du profil est prise en compte pour déterminer les consommations prochaines du client.

Dans notre travail présenté dans TIST, nous partons du fait que, le client pourrait changer son mode de consommation électrique à n'importe quelle date selon plusieurs facteurs externes que l'on ne maîtrise pas toujours. C'est la raison pour laquelle nous exploitons les caractéristiques topologiques qui mettent en exergue la relation entre les clients suivant leurs profils de consommation électriques similaires. Nous exploitons ces caractéristiques topologiques comme information supplémentaire pour mieux comprendre le

comportement des clients dans leur consommation électrique et par conséquent mieux prédire leurs prochaines consommations.

Mining Customers' Changeable Electricity Consumption for Effective Load Forecasting

Étienne G. Tajeuna
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
etienne.gael.tajeuna@usherbrooke.ca
Mohamed Bouguessa
Département d'informatique, Université du Québec à Montréal,
Montréal, Québec, Canada H2X 3Y7
bouguessa.mohamed@uqam.ca
Shengrui Wang
Département d'informatique, Université de Sherbrooke,
Sherbrooke, Québec, Canada J1K 2R1
shengrui.wang@usherbrooke.ca

Keywords: Time Series, Dynamic Networks, Clustering, Survival Analysis, Forecasting.

Abstract

Most existing approaches for electricity load forecasting perform the task based on overall electricity consumption. However, using such a global methodology can affect load forecasting accuracy, as it does not consider the possibility that customers' consumption behavior may change at any time. Predicting customers' electricity consumption in the presence of such unstable behavior poses challenges to existing models. In this paper, we propose a principled approach capable of handling customers' changeable electricity consumption. We devise a network-based approach which consists in first building and tracking cluster of customer consumption patterns over time. Then, on the evolving clusters, we develop a framework that exploits long-short term memory (LSTM) recurrent neural network and survival analysis techniques to forecast electricity consumption. Our experiments on real electricity consumption data sets illustrate the suitability of the proposed approach.

4.1 Introduction

Recent technological developments have rendered energy meters smarter than the conventional models. The bidirectional communication capability of these smart meters allows customers to receive transparent feedback on their energy use and often reduce their electricity consumption [103]. Moreover, the data collected by these devices, in the form of time series data, can be analyzed by machine learning and statistical methods to yield better business insight. For instance, machine learning would allow predicting future events and conditions, which can then be used to reduce unexpected costs and provide potential input to decision making [104].

Most of the machine learning / statistical models used in energy analysis [105] are capable of forecasting customers' electricity consumption with quite high accuracy (e.g., $\geq 90\%$). A major reason for this may be that some customers tend to exhibit repetitive behavior from one time interval to another [103], yielding periodic electricity load profiles. Using this information, researchers have designed more sophisticated approaches to further improve the forecasting accuracy. To this end, several authors [106], [107], [108], [109], [110] have investigated hybrid approaches (clustering and forecasting) in which customers' electricity consumption data are first divided into clusters from which the forecasting task can be performed more effectively.

In spite of these advanced techniques, the problem of electricity load forecasting continues to pose challenges to existing approaches, especially in the situation that involves customers with non-stable consumption. As a matter of fact, from one day to another, customers may change their electricity consumption behavior. Therefore, learning how customers' electricity consumption behavior evolves over time is an important issue that needs to be addressed in order to perform the electricity forecasting in real application situations. This motivates our efforts to propose a principled approach capable of handling customer's changeable electricity consumption. In a nutshell, in our approach, we first identify clusters from customers' electricity consumption data on a daily basis and then track these clusters over time. The tracking process ends up with sequences of clusters reflecting the evolution of consumer consumption over time. Finally, these evolving clusters are analyzed using a survival analysis method and the LSTM (Long short-term memory) recurrent neural network to build a forecasting

model for customers' electricity consumption.

4.1.1 Background information

Recently, much work has been reported on hybrid approaches for time series forecasting. In these approaches, researchers first use a clustering strategy to help them improve forecasting accuracy and avoid the time-consuming task of forecasting based on thousands of observations. For instance, Yogesh et al. [108] proposed a scalable prediction approach based on a clustering strategy to circumvent the high time complexity of handling thousands of customers. The main idea behind their approach is to first aggregate customers who have similar electricity consumption profiles into clusters, from which they extract a representative pattern that is later used in a forecasting model. In [106], J. Kostrzewa first partitions series into two clusters of subseries having a lower potential prediction error, and later merges the predicted results as the prediction of the whole series. Adopting the approach used in [108], Laurinec et al. [107] proposed an incremental process for load forecasting. The aim of their clustering strategy is not limited to aggregating data but also involves finding a suitable data training window. They predict future consumption based on the next window interval.

It is important to note that most existing approaches, as in the aforementioned ones, are based on auto-regression. With auto-regressive models, it is possible to forecast series with very low error rates when the series is stationary or quasi-stationary. Unfortunately, this condition is not always satisfied in real applications. Rather than splitting customers' energy consumption into clusters, Yang et al. [110] partition each time series into k clusters based on shapes. The aim of their clustering is to extract important patterns from the series to improve forecasting accuracy. Instead of using an auto regressive model, they make use of support vector regression which indeed can handle non-stationary cases. In contrast to the work in [108] and [110], where no information other than customer electricity consumption was considered, Wang et al. [34] considered external factors such as weather, price, etc., in their clustering strategy. With this information, their method is capable of discovering the overall customer behavior and thus effectively forecasting electricity consumption. Note that

the forecasting task is performed via a multivariate Gaussian model. Such an approach, however, cannot capture customer behavior changes occurring at different time-stamps because they use a global clustering strategy.

In our previous work [109], a network-based approach was proposed to restructure customer electricity load profiles into networks according to business week days and weekend days. It was thus possible to track customer electricity consumption behavior based on a particular type of day. Even so, however, we discovered that customers did not always follow patterns observed on weekdays or weekends, as their electricity consumption behavior could change on any day. Moreover, the model in [109] can not handle changeable behavior a customer might have displayed in the past.

4.1.2 Rationale and contributions

In the two-step approach (clustering and forecasting) to electricity load forecasting, customers' changeable electricity consumption behavior remains a major challenge at both the clustering and the forecasting steps. In most of the existing methods, customers' overall electricity consumption is used as input to a time series clustering algorithm from which clusters of customers exhibiting similar electricity consumption are obtained as output. Using the entire load profile can be useful to identify the more global aspects of customers' electricity consumption. However, this approach tends not to capture the changeable electricity consumption behavior that a customer may present. For instance, if customers change their electricity use patterns from one time interval to another, this would not be captured by a global approach. Moreover, the changeable consumption behavior of customers may change cluster status at any time interval.

To sidestep the difficulties that existing methods encounter in capturing the temporal aspect of customers' electricity consumption, we propose a local approach to discovering clusters of customer electricity consumption patterns. Note that, in the field of energy analysis, day of the week is an explicit feature that may reveal a customer's cyclical consumption behavior. Thus, by following a given customer's electricity consumption day by day, we can discover whether or not the customer maintains the same pattern of consumption on a given day. This is why our local approach begins on

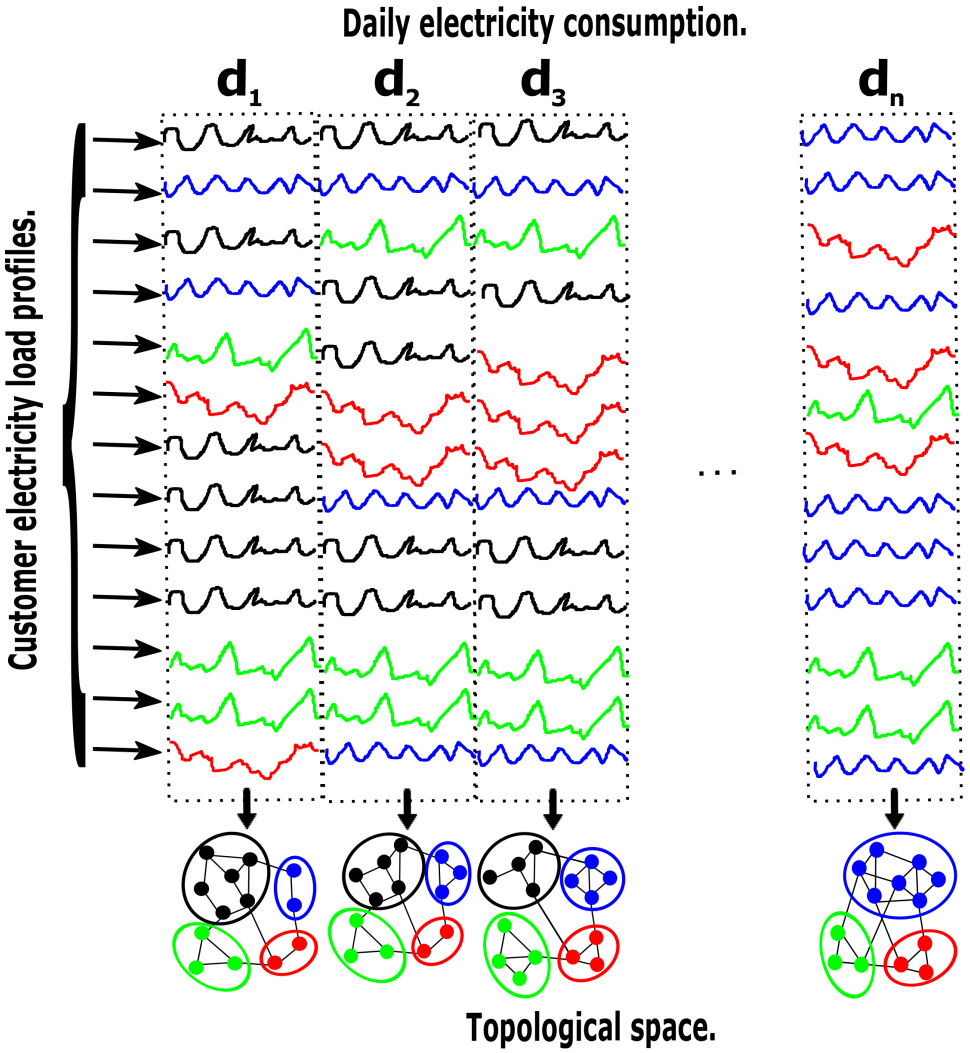


Figure 4.1 – Example of customers' electricity consumption projected into a topological space. Each colored subnetwork corresponds to a cluster of customers electricity consumption at a given day (each node observed within a subnetwork represents a customer daily consumption). Patterns observed at the top side of the figure and tagged to blue, black, green and red are the exhibited customer behaviors. For every single day, the number of patterns observed is equal to the number of subnetworks.

the first day by discovering clusters of customer electricity consumption patterns which are then tracked over subsequent days. We thus obtain evolving clusters, that is, the evolution of clusters over days whose survival can be investigated. Here, each evolving cluster again reveals the lifespan of customers' electricity consumption behavior. In

other words, an evolving cluster relates the temporal evolution of a specific behavior adopted by customers. In our work we emphasize on the study of customer behavior survivals because it enables to better foresee its electricity load values. Taking the fact that, within a given data set, customers may change their electricity consumption behavior, it is clear that the number of customers exhibiting a specific behavior may gradually increase/decrease over time . Such trend is due to the interchanging customer behaviors observed within the overall data set which entails create an *ecosystem*. With such an ecosystem, where we have customers' co-evolving electricity consumption, we believe that using a network structure at each day will enable us to better capture customer electricity consumption interrelations. Hence, at each day, we use a network structure to project daily customer electricity consumption from the time-space into a topological space. The network is then split into densely connected subnetworks that represent clusters of customer electricity consumption patterns. To be more clear, Figure 4.1 [1] depicts an example of customers' electricity consumption which are daily projected into a topological space. In this example, we can see that the topological representation quickly helps to see that the overall consumption behaviors is varying over time. For instance, when looking at the subnetwork of blue nodes, this subnetwork is gradually expanding from day d_1 to d_n showing that the behavior represented by the blue pattern is the more and more exhibited.

The network-based clustering used here enables us to capture the survival of a customer cluster via its evolving structure. Note that when we talk about structure, we refer to two main types: (1) topological features and (2) structural changes. The topological features, comprising aspects such as density, transitivity, etc. [111] (more details about the estimated topological features are given later in section 4.2.4) depict the time-dependent interrelation between customers' electricity consumption patterns. Structural changes describe particular customers' changing electricity consumption behavior. These behavior changes can be illustrated by basic movements of customers switching from one cluster to another. As basic movements, from day $d-1$ to day d, we have customers leaving a cluster, customers joining a cluster or customers remaining in the same cluster.

As a result of these basic movements that illustrate the changing behavior of

1. It is worth noting that most of the figures used in this paper are best viewed in color.

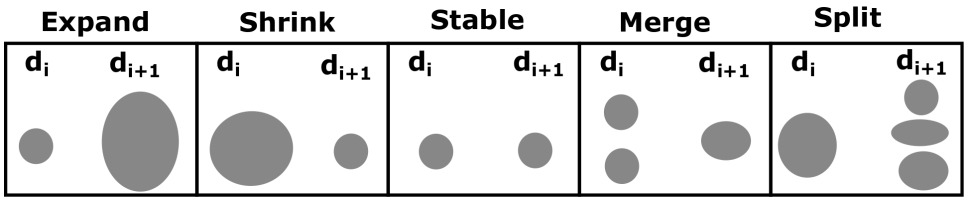
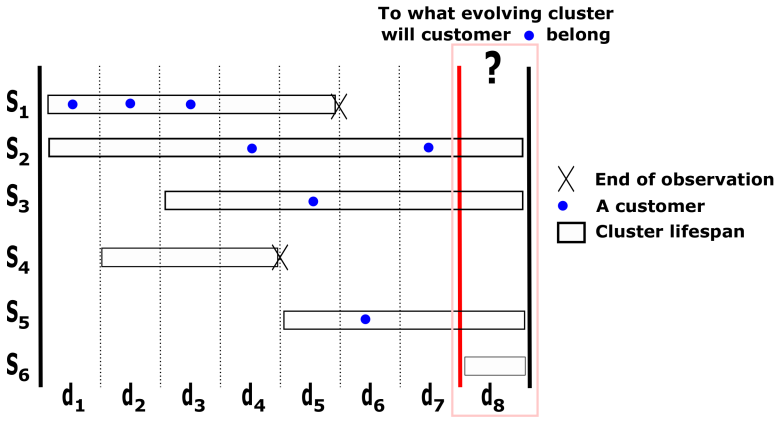


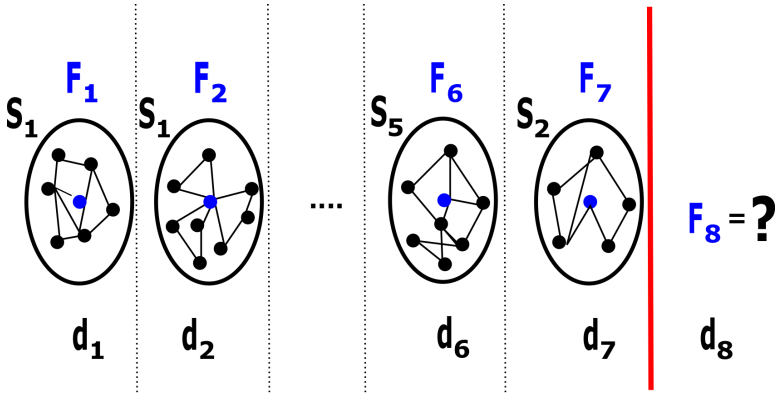
Figure 4.2 – Examples of structural changes an evolving cluster may undergo.

customers, an evolving cluster can significantly change its structure over time. For instance, we note that in its evolution from one day to another, a cluster may *expand* because more customers join it. The cluster may also *shrink*, meaning that some customers leave the evolving cluster between one day and another. Some clusters may *split* into several other clusters, just as several clusters may *merge* into a single cluster. Finally, certain clusters may remain *stable* (no change in terms of customers within the cluster from one day to another) in their evolution. We are convinced that learning such structural phenomena (*expand, shrink, split, merge, stable*) will be of benefit to understanding customers' electricity consumption behavior and can even enhance electricity load forecasting. The examples in Figure 4.2 depict possible changes an evolving cluster may undergo over time. In section 4.2.4, we formally define how these structural changes are captured. Topological features and structural changes are subsequently used to study the survival of customer electricity consumption clusters.

To predict a customer's electricity consumption on subsequent days, we trace the customer's trajectory through evolving clusters via the customer's topological features. The features are used to evaluate the evolving cluster the customer is most likely to belong to on later days and thus return the predicted electricity consumption for that customer. To further illustrate the proposed framework, Fig 4.3(a) shows an example of some evolving clusters, denoted by $S_1 - S_6$, which are observed over the days ranging from d_1 to d_7. Each cluster's lifespan is represented by a white horizontal band. An "X" seen at the end of a horizontal band indicates the end of the evolution. To forecast the electricity consumption of a customer (e.g., the one tagged in blue in Fig 4.3(a)) on future days, we first trace which cluster this customer belonged to on the preceding days. In Fig 4.3(b) the sequence S_1, S_1, S_1, S_2, S_3, S_5, S_2 shows the blue-tagged customer's trajectory through evolving clusters for the days $d_1 - d_7$. From each cluster (within the sequence) denoting the topological structure of an evolving



(a) An example showing lifespans of evolving clusters. The interval ranging from day d_1 to day d_7 (delineated by bold black and red vertical lines) corresponds to the current observable days. Day d_8 after the bold red vertical line is the future day to be predicted.



(b) The sequence S_1, S_1, ..., S_5, S_2 denotes the trajectory of a customer (tagged in blue) through clusters that are evolving over time. $F_1 - F_7$ are the features of the blue-tagged customer extracted from the evolving clusters on different days.

Figure 4.3 – Overview of the proposed framework.

cluster, features specific to the blue-tagged customer are extracted and utilized to generate further features F_8. These features (F_8) are used to identify the evolving cluster this customer is most likely to belong to on day d_8.

The significance of this work can be summarized as follows:

(1) We propose a principled approach for electricity load forecasting capable of handling customers' changeable electricity consumption behavior. In fact, the network-

based tracking strategy used in our approach allows us to model the time-dependent aspect of customers' electricity consumption into time-to-event sequences of clusters. The obtained sequences are more meaningful representation that describe the time-dependent aspect of electricity consumption behavior.
(2) We view the task of forecasting customer electricity consumption from survival analysis theory principle, based on which we devise a framework that exploits both the Cox regression model and the LSTM recurrent neural network for effective load forecasting. The Cox model allows us to calculate the probability that a cluster of customers will evolve over time. The role of the LSTM is to generate suitable features which are considered within the Cox model. To our knowledge, in the current literature, such information has not been considered in determining a cluster's probability of surviving.
(3) We validate the whole approach by implementing it on real customer energy consumption data collected from the EnerNOC GreenButton website and the Smart Dataset for Sustainability. We illustrate the suitability of the proposed method by comparing its performance to that of nine learning algorithms. Our experiments show that the proposed approach improve significantly the forecasting accuracy.

4.2 Proposed approach

4.2.1 Problem Statement

Let $E^i = \{(t_l, e_l^i)\}_{l=1}^m$ be the data (time series) of the electricity consumption of customer C^i recorded at regularly spaced timestamps from t_1 to t_m, where e_l^i stands for the recorded electricity consumption of customer C^i at time t_l. We use the vector notation $\vec{E}^{\,i} = (e_1^i, ..., e_m^i)$ to denote the overall consumption profile of customer C^i. $\boldsymbol{C} = \{C^i\}_{i=1}^N$ is used to denote the set of N customers and $\boldsymbol{E} = \{\vec{E}^{\,i}\}_{i=1}^N$ their respective electricity load profiles. Our goal is to forecast all customers $\boldsymbol{C}$ electricity consumption at subsequent time. To this end, we start by studying the daily changes in customer electricity consumption behavior. Hence, we split each vector $\vec{E}^{\,i}$ into subvectors $\vec{E}_d^{\,i}$ expressing the subprofile of customer C^i on day d. The value d is taken from the day set $\{1, 2, ..., D\}$, where $D < m$ is the total number of days within the

Table 4.1 – Notations used throughout the paper.

Notation	Definition
$\boldsymbol{C} = \{C^i\}_{i=1}^{N}$	Set of N customers
$E^i = \{(e_l^i,\, t_l)\}_{l=1}^{m}$	A customer C^i electricity consumption from time t_1 to time t_m, $m > 1$.
$\boldsymbol{E} = \{\vec{E}^i\}$	Set of customer electricity load profiles
$\vec{E}_d^i$	Customer electricity load profile at day d
D	Total number of known days within $[t_1,\, t_m]$
$G_d = \{\boldsymbol{C}, \boldsymbol{L}_d\}$	Constructed network at day d with $\boldsymbol{L}_d$ the set of links
$\boldsymbol{X}_d = \{X_d^\bullet\}_{\bullet=1}^{q_d}$	A partition of network G_d
$X_d^\bullet = \{\boldsymbol{C}_d^\bullet, \boldsymbol{L}_d^\bullet\}$	Subnetwork with $\boldsymbol{C}_d^\bullet \subseteq \boldsymbol{C}$ and $\boldsymbol{L}_d^\bullet \subseteq \boldsymbol{L}_d$
$F_d^\bullet$	Topological features extracted from $X_d^\bullet$
$S_{X_d^\bullet}$	An evolving cluster
$\mathcal{E}_{d\to d+1}^\bullet$	Structural change the evolving cluster $S_{X_d^\bullet}$ has undergone from day d to day $d+1$
$\mathcal{T}r^i$	Electricity consumption behavior trajectory followed by customer C^i

time interval $[t_1,\, t_m]$. For each day d, we restructure the set of subprofiles $\boldsymbol{E}_d = \{\vec{E}_d^{\,i}\}_{i=1}^{N}$ into a network $G_d = \{\boldsymbol{C},\, \boldsymbol{L}_d\}$, where $\boldsymbol{L}_d$ is the set of links describing interrelations between customers at day d. The constructed network G_d is then partitioned into a set $\boldsymbol{X}_d = \{X_d^\bullet\}_{\bullet=1}^{q_d}$ of clusters or subnetworks, where q_d is the number of clusters. Note that, for each cluster or subnetwork $X_d^\bullet$, we use $\boldsymbol{E}_d^\bullet$ to denote the corresponding subprofiles of customers within the subnetwork.

Like the network G_d, the cluster $X_d^\bullet$ is defined by the subset $\boldsymbol{C}_d^\bullet \subseteq \boldsymbol{C}$ of customers, where the interrelation among them is reported by the subset $\boldsymbol{L}_d^\bullet \subseteq \boldsymbol{L}_d$ ($X_d^\bullet = \{\boldsymbol{C}_d^\bullet,\, \boldsymbol{L}_d^\bullet\}$). For each cluster $X_d^\bullet$, we extract $F_d^\bullet$, a set of features describing its topological structure. In order to represent the evolution of a given cluster $X_d^\bullet$, we use $S_{X_d^\bullet} = \{X_d^\bullet,\, X_{d+1}^\bullet,\, ...\}$ to denote the sequence of the clusters that are derived from $X_d^\bullet$. In the rest of the paper, we will use the terminology "evolving cluster $S_{X_d^\bullet}$" to mean the evolution of cluster $X_d^\bullet$ over time and $\boldsymbol{SS}$ to mean the set of evolving clusters. In its evolution, a cluster may change structurally (i.e., *expand*, *shrink*, *split* or *merge*) or it may not (i.e., remain *stable*). For a given evolving cluster $S_{X_d^\bullet}$, on any two consecutive days d and $d+1$, we use $\mathcal{E}_{d\to d+1}^\bullet$ to mean the structural change the evolving

cluster $S_{X_d^\bullet}$ has undergone from day d to day $d+1$. Knowing the different electricity consumption behaviors exhibited by customers in different days, for each customer C^i, we then trace his trajectory (i.e., the different behavior(s) he/she displayed over days) as $\mathcal{T}r^i = \{X_d^\bullet = \{\boldsymbol{C}_d^\bullet, \boldsymbol{L}_d^\bullet\} \mid C^i \in \boldsymbol{C}_d^\bullet\}_{d=1}^{D}$ from which we can get the different features and thus predict his most probable behavior at subsequent days. To conclude this section, Table 1 provides the main notationused throughout the paper. To conclude this section, we summarize in Table 4.1 the main notations used throughout the paper.

4.2.2 Network construction

In this work, we make use of a time dependant network structure to relate customers who display similar behavior. For a given time interval, the network allows us to figure out how likely customers are to behave similarly in terms of electricity consumption. Given the set of subprofiles $\boldsymbol{E}_d$, the procedure for building the corresponding network consists in assessing relationships among these observations using a similarity measure. A number of metrics has been proposed for estimating the similarity between two vectors representing time series. Some of these metrics, e.g., the Euclidean distance, evaluate the proximity of the two series, that is, how close they are spatially. Others, e.g., the Kullback-Leibler distance, assess how similar the two series are in shape. In this paper, we consider both shape and proximity. Therefore, we define a binary similarity function as follows: two vectors $\vec{\mathcal{V}}^{\,1} = (v_1^1, v_2^1, ..., v_h^1)$ and $\vec{\mathcal{V}}^{\,2} = (v_1^2, v_2^2, ..., v_h^2)$ are similar if the following constraints are verified:

$$sim\left(\vec{\mathcal{V}}^{\,1}, \vec{\mathcal{V}}^{\,2}\right) = 1 \Leftrightarrow \begin{cases} mes_1\left(\vec{\mathcal{V}}^{\,1}, \vec{\mathcal{V}}^{\,2}\right) < \omega^1 \\ mes_2\left(\vec{\mathcal{V}}^{\,1}, \vec{\mathcal{V}}^{\,2}\right) < \omega^2 \end{cases} \tag{4.1}$$

with

$$\begin{cases} mes_1\left(\vec{\mathcal{V}}^{\,1}, \vec{\mathcal{V}}^{\,2}\right) & = \sqrt{\frac{1}{h}\sum_{\diamond=1}^{h}\left(v_\diamond^1 - v_\diamond^2\right)^2} \\ mes_2\left(\vec{\mathcal{V}}^{\,1}, \vec{\mathcal{V}}^{\,2}\right) & = \frac{1}{2h}\sum_{\diamond=1}^{h}\left(v_\diamond^1 - v_\diamond^2\right) log\left(\frac{v_\diamond^1}{v_\diamond^2}\right) \end{cases} \tag{4.2}$$

where ω^1 and ω^2 are thresholds to be determined.

Throughout the network construction process, we automatically estimate suitable thresholds for building the network at each time interval, which means that the thresholds in our context are not constant values. For each day d, the suitable threshold ω_d^1 (respectively ω_d^2) is obtained using a mixture of two gamma probability density functions [20]. In this paper, our threshold ω_d^1 (respectively ω_d^2) corresponds to the junction point between the two gamma curves estimated from the non-zero values obtained from mes_1 (respectively mes_2) in Eq.(4.2).

4.2.3 Tracking clusters

In our approach, we track clusters based on their representative electricity load patterns. We use the term *representative pattern* to denote the subprofile that reflects the common electricity consumption of customers within a cluster. To identify clusters within the network, we split the network into subnetworks in such a way that nodes within a subnetwork are densely connected to each other and sparsely connected to other subnetworks. Numerous algorithms have been proposed for partitioning a network into clusters (or communities) such as in [112], [113], [114]. In this paper, we make use of the Fastgreedy algorithm [115] to detect clusters, as it is fast and does not need any parameters setting.

To extract representative patterns from each of the identified clusters, we need to describe first the following notions. Let $\mathcal{D}_d^\bullet(C^i)$ be the degree of C^i on day d, which, corresponds to the number of edges linked to customer (the node) C^i in the cluster $X_d^\bullet$. We use also

$$\widetilde{\mathcal{D}}_d^\bullet = \frac{1}{|X_d^\bullet|} \sum_{C^i \in X_d^\bullet} \mathcal{D}_d^\bullet(C^i)$$

to represent the average degree within the cluster $X_d^\bullet$. The representative pattern of a cluster $X_d^\bullet$ can thus be defined as the average subprofiles whose corresponding customers have degree ($\mathcal{D}_d^\bullet(.)$) above the average degree ($\widetilde{\mathcal{D}}_d^\bullet$) within the cluster $X_d^\bullet$. In other words, we choose the subset $\mathcal{F}_d^\bullet \subseteq X_d^\bullet$ of customers $C^i \in X_d^\bullet$ such that

$\mathcal{D}_d^\bullet(C^i) \geq \widetilde{\mathcal{D}}_d^\bullet$. We formally define the representative pattern as follows:

$$\vec{R}_d^\bullet = \frac{1}{|\mathcal{F}_d^\bullet|} \sum_{C^i \in \mathcal{F}_d^\bullet} \vec{E}_d^i \tag{4.3}$$

Now that we know how to extract a representative pattern from a cluster, we can use it to track clusters over time. Specifically, given a cluster $X_1^\bullet$ identified on the first day, we want to know its real status on the next day. To this end, we first construct the network G_2 and detect the set of clusters $\boldsymbol{X}_2$, as discussed in section 4.2.2. Next, we find the suitable cluster within $\boldsymbol{X}_2$ that is expected to be the next status of the $X_1^\bullet$. In our approach, the next status of a cluster is determined based on the similarity between representative patterns. Therefore, the next status of cluster $X_1^\bullet$ is the cluster $X_2^\bullet$ whose representative pattern is, in a special sense, similar to the representative pattern $\vec{R}_1^\bullet$. We formally define the *next status* of cluster $X_1^\bullet$ as follows:

$$Next(X_1^\bullet) = \left\{ X_2^\bullet \middle| \, sim\left(\vec{R}_1^\bullet, \vec{R}_2^\bullet\right) = 1 \right\} \tag{4.4}$$

Note that the similarity is calculated based on the two threshold values ω_2^1 and ω_2^2. Moreover, the next status of a cluster given in (4.4) could be an empty set. In such a case, we assume that the cluster no longer exists. By running Algorithm 5 we identify new clusters and track them over days.

4.2.4 Feature extraction

In this section we present topological attributes and structural changes as features extracted from evolving clusters.

Topological features

In the field of network analysis, it has been demonstrated that the network structure of a cluster can be exploited to predict whether or not the cluster will disappear [21], [36]. In this paper, we study the topological structure of a cluster by examining node connectivity, which allows establishing a pattern of customer

Algorithm 5: Tracking clusters over days

Data: $\boldsymbol{C}$, $\boldsymbol{E}$, D /* Set of customers with their respective electricity load profiles and the total number of days. */

Result: $\boldsymbol{SS}$, $\boldsymbol{RP}$, $\boldsymbol{X}$ /* sets of evolving clusters, representative patterns and identified clusters. */

begin

$\boldsymbol{SS}$, $\boldsymbol{RP}$, $\boldsymbol{X} \leftarrow \emptyset$

$d \leftarrow 1$ /* first day */

A- Extract customer electricity subprofiles:

begin

$\boldsymbol{E}_d \leftarrow \emptyset$ /* set of customer electricity subprofiles at day d */

for $i \in \{1, ..., N\}$ **do**

$\boldsymbol{E}_d \leftarrow \vec{E}_d^{\,i}$

B- Clustering:

begin

- Calculate similarities among subprofiles in $\boldsymbol{E}_d$ and extract suitable thresholds ω_d^1 and ω_d^2 as discussed in section 4.2.2.
- Using Eq. (4.1), transform $\boldsymbol{E}_d$ into network G_d.
- Using a partitioning algorithm, split the network G_d into subnetworks $\boldsymbol{X}_d$.
- $\boldsymbol{X} \leftarrow \boldsymbol{X} \bigcup \boldsymbol{X}_d$

for $X_d^\bullet \in \boldsymbol{X}_d$ **do**

- Using Eq. (4.3), calculate the representative pattern $\vec{R}_d^\bullet$ of cluster $X_d^\bullet$
- $\boldsymbol{RP} \leftarrow \boldsymbol{RP} \bigcup \{\vec{R}_d^\bullet\}$

if $d = 1$ **then**

- $S_{X_d^\bullet} \leftarrow \{X_d^\bullet\}$
- $\boldsymbol{SS} \leftarrow \boldsymbol{SS} \bigcup S_{X_d^\bullet}$

C- Tracking:

begin

for $d \in \{2, ..., D\}$ **do**

- $temp \leftarrow \emptyset$ /* a temporary set of clusters at day d */
- Run step $\boldsymbol{A}$
- Run step $\boldsymbol{B}$ and obtain the set of clusters $\boldsymbol{X}_d$
- $temp \leftarrow \boldsymbol{X}_d$

for $S_{X_d^\bullet} \in \boldsymbol{SS}$ **do**

- Get the latest cluster $X_{d-1}^\bullet$ in $S_{X_d^\bullet}$
- Using Eq. (4.4) get the next status $X_d^\bullet$ of cluster $X_{d-1}^\bullet$
- $S_{X_d^\bullet} \leftarrow S_{X_d^\bullet} \bigcup \{X_d^\bullet\}$
- $\boldsymbol{SS} \leftarrow \boldsymbol{SS} \bigcup S_{X_d^\bullet}$
- $temp \leftarrow temp \setminus \{X_d^\bullet\}$

for $X \in temp$ **do**

- $S_X \leftarrow \{X\}$
- $\boldsymbol{SS} \leftarrow \boldsymbol{SS} \bigcup S_X$

interactions within the cluster. From each evolving cluster, we first extract a set of topological features, which can be briefly described as follows:
Two obvious features are **(1) the number of edges in the cluster** and **(2) the number of nodes in the cluster**, which are self-explanatory; **(3) the average shortest path**, which provides general information about how far apart the nodes are in the cluster; **(4) the average closeness**, which, contrary to the average shortest path, gives general information about how close together the nodes are in the cluster; **(5) the average intra-degree**, which provides information on how the cluster is internally connected; whereas **(6) the average inter-degree** describes how the cluster is externally connected; **(7) the conductance**, which, introduced by [71], gives an idea of how closely the nodes within the cluster are connected globally in the network; whereas **(8) the density** gives an idea of how the nodes within the cluster are connected independently of the other communities in the network; **(9) the diameter** which provides information about the longest path in the cluster; **(10) the attractive force** [72], which indicates how capable the cluster is of attracting nodes to join its member nodes; and finally **(11) the transitivity**, which indicates the average number of cliques in the cluster.

It is important to note that the features enumerated above are not the only topological features that could be extracted. Other features can be added to the ones we have presented. However, insofar as the extracted features reflect the interaction of node connectivity within the cluster, it might be possible to have highly correlated features. Using highly correlated features may not improve the forecasting accuracy for customers' electricity consumption. To avoid such a situation, we apply the Pearson correlation coefficient over the eleven features in order to eliminate features that are highly correlated and keep less correlated ones.

Structural changes features

An evolving cluster specifies a typical consumption behavior over time. Switching/moving from one cluster to another shows that the customer has changed her/his consumption behavior. As a result of all such customer movements, it will be observed that clusters can change their structure over time. There are three such movements, *leave*, *join* and *stay*, defined as follows:

(1) *leave*: A customer is said to have left an evolving cluster at a given day d if this customer was within the cluster at day $d-1$ and absent at day d.
Formally, given an evolving cluster $S_{X_d^\bullet}$, the set of customers leaving the cluster at day d is given as:

$$\mathcal{L}eave(S_{X_d^\bullet} \,|\, d) = \left\{ C^i \;\middle|\; \exists\, X_{d-1}^\bullet,\, X_d^\bullet \,\in S_{X_d^\bullet} \,/\, C^i \in X_{d-1}^\bullet,\; C^i \notin X_d^\bullet \right\} \tag{4.5}$$

(2) *join*: A customer is said to have joined an evolving cluster at day d if he or she was not present within the cluster at day $d-1$ and was present at day d.
Formally, given an evolving cluster $S_{X_d^\bullet}$, the set of customers joining the cluster at day d is given as:

$$\mathcal{J}oin(S_{X_d^\bullet} \,|\, d) = \left\{ C^i \;\middle|\; \exists\, X_{d-1}^\bullet,\, X_d^\bullet \,\in S_{X_d^\bullet} \,/\, C^i \notin X_{d-1}^\bullet,\; C^i \in X_d^\bullet \right\} \tag{4.6}$$

(3) *stay*: A customer is said to have stayed within an evolving cluster at a given day d if he or she is present within the cluster at days $d-1$ and d.
Formally, given an evolving cluster $S_{X_d^\bullet}$, the set of customers staying in the cluster at day d is given as:

$$\mathcal{S}tay(S_{X_d^\bullet} \,|\, d) = \left\{ C^i \;\middle|\; \exists\, X_{d-1}^\bullet,\, X_d^\bullet \,\in S_{X_d^\bullet} \,/\, C^i \in X_{d-1}^\bullet,\; C^i \in X_d^\bullet \right\} \tag{4.7}$$

As we did for the basic movements given in equations (4.5), (4.6) and (4.7), we define structural changes occurring to an evolving cluster as follows:
(1) *expand*: A cluster expands at day d if customers have joined this cluster at day d, such that the total number of customers within the cluster is higher than the total number of customers within the cluster at day $d-1$.
Formally, given an evolving cluster $S_{X_d^\bullet}$, the cluster is said to have expanded at date d if the following equation is respected:

$$\mathcal{E}xpand(S_{X_d^\bullet} \,|\, d) = \begin{cases} 1 \;\; if \;\; \exists X_{d-1}^\bullet \in S_{X_d^\bullet}, \exists X_d^\bullet \in Next(X_{d-1}^\bullet) \;/\; |X_d^\bullet| > |X_{d-1}^\bullet| \\ 0 \;\; otherwise \end{cases} \tag{4.8}$$

(2) *shrink*: A cluster shrinks at day d if customers have left this cluster at day d, such that the total number of customers within the cluster is lower than the total number of customers within the cluster at day $d-1$.
Formally, given an evolving cluster $S_{X_d^\bullet}$, the cluster is said to have shrunk if the following equation is respected:

$$\mathcal{S}hrink(S_{X_d^\bullet} \,|\, d) = \begin{cases} 1 \;\; if \;\; \exists X_{d-1}^\bullet \in S_{X_d^\bullet}, \exists X_d^\bullet \in Next(X_{d-1}^\bullet) \;/\; |X_d^\bullet| < |X_{d-1}^\bullet| \\ 0 \;\; otherwise \end{cases} \tag{4.9}$$

(3) *merge*: Clusters merge at day d into an evolving cluster if all these clusters are similar to the observed cluster status at day d.
Formally, given an evolving cluster $S_{X_d^\bullet}$, clusters are said to have merged into one cluster if the following equation is respected:

$$\mathcal{M}erge(S_{X_d^\bullet} \,|\, d) = \begin{cases} 1 \;\; if \;\; \left| \left\{ X_{d-1}^\bullet \in \boldsymbol{X}_{d-1} \,/\, X_d^\bullet \in Next(X_{d-1}^\bullet) \,\&\, X_d^\bullet \in S_{X_d^\bullet} \right\} \right| > 1 \\ 0 \;\; otherwise \end{cases} \tag{4.10}$$

(4) *split*: An evolving cluster splits at day d if there exists more than one cluster detected at day d such that all of these clusters are similar to the observed cluster status at day $d-1$.
Formally, given an evolving cluster $S_{X_d^\bullet}$, the cluster is said to have split into other clusters if the following equation is respected:

$$\mathcal{S}plit(S_{X_d^\bullet} \mid d) = \begin{cases} 1 \;\; if \;\; \exists X_{d-1}^\bullet \in S_{X_d^\bullet} \;/\; |Next(X_{d-1}^\bullet)| > 1 \\ 0 \;\; otherwise \end{cases} \tag{4.11}$$

(5) *stable*: A cluster remains stable at day d if all customers within the cluster at day $d-1$ stay in the cluster at day d, such that the total number of customers within the cluster remains constant from day $d-1$ to day d.
Formally, given an evolving cluster $S_{X_d^\bullet}$, the cluster is said to remain stable if the following equation is respected:

$$\mathcal{S}table(S_{X_d^\bullet} \mid d) = \begin{cases} 1 \;\; if \;\; \exists X_{d-1}^\bullet \in S_{X_d^\bullet}, \exists X_d^\bullet \in Next(X_{d-1}^\bullet) \;/\; X_d^\bullet = X_{d-1}^\bullet \\ 0 \;\; otherwise \end{cases} \tag{4.12}$$

Considering the cardinalities of the sets given in equations (4.5) - (4.7) and the Boolean functions of the structural changes given in equations (4.8) - (4.12), we then define the structural change features at day d as follows:

$$\mathcal{E}_{d-1\to d}^\bullet = \Big(|\mathcal{L}eave(S_{X_d^\bullet}|d)|, |\mathcal{J}oin(S_{X_d^\bullet}|d)|, |\mathcal{S}tay(S_{X_d^\bullet}|d)|, \mathcal{E}xpand(S_{X_d^\bullet}|d), \mathcal{S}hrink(S_{X_d^\bullet}|d), \mathcal{M}erge(S_{X_d^\bullet}|d), \mathcal{S}plit(S_{X_d^\bullet}|d), \mathcal{S}table(S_{X_d^\bullet}|d) \Big)$$

It is important to note that the calculation of structural changes requires the knowledge of cluster status at two consecutive time-stamps, whereas topological features are calculated based on one time-stamp. In order to have exactly the same number of features (topological and structural changes) extracted from an evolving cluster $S_{X_d^\bullet}$, for two consecutive days $d-1$ and d, if there exist two clusters $X_{d-1}^\bullet$, $X_d^\bullet \in S_{X_d^\bullet}$, we consider as topological features the values given by $\Delta F_{d-1\to d}^\bullet = F_d^\bullet - F_{d-1}^\bullet$. Hence, given an evolving cluster $S_{X_d^\bullet}$, the overall extracted features at day d are as follows:

Algorithm 6: Feature extraction

Data: $\boldsymbol{X} = \{\boldsymbol{X}_d\}_{d=1}^{D}$, $\boldsymbol{SS}$ /* sets of identified clusters from day 1 to D and evolving clusters. */
Result: $\boldsymbol{\mathcal{WF}}$ /* set of extracted features */
begin
  $\boldsymbol{\mathcal{WF}} \leftarrow \emptyset$
  A- Topological features:
  begin
    $F \leftarrow \emptyset$
    for $d \in \{1, ..., D\}$ **do**
      for $X_d^\bullet \in \boldsymbol{X}_d$ **do**
        - Roughly extract a set of 11 features $F_d^\bullet$ (as enumerated in section 4.2.4) from cluster $X_d^\bullet$
        - $F \leftarrow F \bigcup F_d^\bullet$
        - Using Pearson correlation coefficient (PCC(.)), eliminate correlated features and keep non-correlated ones
        - $F \leftarrow PCC(F)$
  B- Structural changes features:
  begin
    for $S_{X_d^\bullet} \in \boldsymbol{SS}$ **do**
      $\boldsymbol{\mathcal{WF}}^\bullet \leftarrow \emptyset$
      for $d \in \{2, ..., D\}$ **do**
        if $X_d^\bullet \in S_{X_d^\bullet}$ **then**
          - Using equations (4.5), (4.6), (4.7), (4.8), (4.9), (4.10), (4.11) and (4.12), extract structural changes features $\mathcal{E}^\bullet_{d-1\rightarrow d}$ as discussed in section 4.2.4
          - From topological features obtained in step $\boldsymbol{A}$, get the topological features $F^\bullet_{d-1}$ and $F^\bullet_d$ of clusters $X^\bullet_{d-1} \in S_{X_d^\bullet}$ and $X^\bullet_d$ respectively
          - Using Eq. (4.13), get the overall features $\boldsymbol{\mathcal{WF}}^\bullet_d$ at day d
          - $\boldsymbol{\mathcal{WF}}^\bullet \leftarrow \boldsymbol{\mathcal{WF}}^\bullet \bigcup \{\boldsymbol{\mathcal{WF}}^\bullet_d\}$
      $\boldsymbol{\mathcal{WF}} \leftarrow \boldsymbol{\mathcal{WF}} \bigcup \boldsymbol{\mathcal{WF}}^\bullet$

$$\boldsymbol{\mathcal{WF}}^\bullet_d = (\Delta F^\bullet_{d-1\rightarrow d},\ \mathcal{E}^\bullet_{d-1\rightarrow d}) \tag{4.13}$$

By running Algorithm 6 we can extract topological and structural changes features for all evolving clusters within the set of evolving clusters $\boldsymbol{SS}$.

4.2.5 Survival study of evolving clusters

In this section, based on the evolution of clusters, we elaborate a time-dependent probabilistic approach that quantifies the survival of these evolving clusters. This will enable us to calculate the risk for an evolving cluster to disappear at any future time point.

Survival probability

Consider the evolving cluster $S_{X_d^\bullet}$ (the set representing the evolution of cluster $X_d^\bullet$). As the days advance, it will be noted that this cluster may gradually weaken and finally disappear. Such a critical event might be due to variation in its number of customers over time. This process can be generalized by modeling the instantaneous risk of the evolving cluster's disappearing at a specific day, that is, the *hazard rate* [62], using its corresponding topological features as covariates.

At each day d, we calculate the number $\mathcal{N}(d)$ of clusters that disappear. We assume that the counting process given by $\mathcal{N}(d)$ can be decomposed as follows:

$$\mathcal{N}(d) = \Lambda(d) + M(d) \tag{4.14}$$

where $\Lambda(d)$ is a non-decreasing predictable process, called the *cumulative intensity process*, and $M(d)$ is a mean-zero martingale [62]. Considering $\Lambda(d)$ to be continuous, there exists a general predictable non-negative intensity process $\lambda(d)$ such that

$$\Lambda(d) = \int_1^d \lambda(x)dx \tag{4.15}$$

For the evolution of cluster $X_d^\bullet$ ($S_{X_d^\bullet}$), at each day, we calculate the intensity process using the Cox regression, defined as follows:

$$\lambda(d|S_{X_d^\bullet}) = \lambda(d)\ exp(\boldsymbol{\mathcal{W}}\boldsymbol{\mathcal{F}}_d^\bullet\,\Gamma_d) \tag{4.16}$$

where $\lambda(d)$ is the baseline function that generalizes the process. In this paper, we assume that the experimental distribution process (i.e., the number of clusters that disappears over days) can be approximated with a theoretical distribution function with parameters $\boldsymbol{\Theta}$ (more details on the selected distribution is given in the experimental section). Γ_d is the vector parameter representing the contribution of each covariate at day d. This vector parameter is estimated using the Newton-Raphson algorithm. From Eq.(4.16) we can then calculate, at any day, the probability that an evolving cluster $S_{X_d^\bullet}$ still exists by computing the cumulative probability $Surv(d \,|\, S_{X_d^\bullet})$ [62], as follows:

$$Surv(d \,|\, S_{X_d^\bullet}) = exp\Big\{ - \int_1^d \lambda(x|S_{X_d^\bullet})dx \Big\} \tag{4.17}$$

Estimation of the Cox parameters

Given the set of evolving clusters $\boldsymbol{SS}$, we can formalize the risk of an evolving cluster's $S_{X_d^\bullet}$ disappearing at a specific day d as

$$\mathcal{R}_{S_{X_d^\bullet}}(d) = \sum_{S_{X_d^\bullet} \in \boldsymbol{SS}} \lambda(d|S_{X_d^\bullet}) \tag{4.18}$$

Suppose that we observe clusters from the first day to a given day $d > 1$ which is not the last day. Clearly, from day 1 to day d we do not have a total view of all evolving clusters, which implies that the notion of likelihood cannot be explicit [36]. Due to this constraint, we instead define the partial likelihood of the parameters Γ_d as

$$\mathcal{P}(\Gamma_d) = \prod_{S_{X_d^\bullet} \in \boldsymbol{SS}} \prod_{x=1}^{d} \left[\frac{exp(\boldsymbol{\mathcal{WF}}_x^\bullet \Gamma_x)}{\sum_{S_{X_d^\bullet}} exp(\boldsymbol{\mathcal{WF}}_x^\bullet \Gamma_x)} \right]^{\delta_x} \tag{4.19}$$

where δ_x is a binary indicator which takes value 1 when cluster $S_{X_d^\bullet}$ has an instance at day d and 0 otherwise. Taking the logarithm of the last equation, we obtain the partial log-likelihood expanded as follows:

Algorithm 7: Customer ELectricity Load Forecasting

Data: $\boldsymbol{C}$, $\boldsymbol{SS}$, $\boldsymbol{\mathcal{WF}}$, $\mathcal{D} = \{D+1, D+2, ...\}$ /* sets of customers, evolving clusters, extracted features and days ahead to forecast. */

Result: $\hat{\boldsymbol{E}}$ /* set of customers' predicted electricity consumption. */

begin

- $\hat{\boldsymbol{E}} \leftarrow \emptyset$
- **for** $S_{X_d^\bullet} \in \boldsymbol{SS}$ **do**
  - Get the sequence of features $\boldsymbol{\mathcal{WF}}^\bullet$ from sequence $S_{X_d^\bullet}$ as discussed in section 4.2.4
  - Using the past features $\boldsymbol{\mathcal{WF}}^\bullet$ as input information to the LSTM neural network described in section 4.2.6, forecast features $\hat{\boldsymbol{\mathcal{WF}}}^\bullet = \{(\hat{\Delta F}^\bullet_{d-1\rightarrow d}, \hat{\mathcal{E}}^\bullet_{d-1\rightarrow d})\}_{d\in\mathcal{D}}$ at subsequent days $\mathcal{D}$
- **for** $C^i \in \boldsymbol{C}$ **do**
  - $\mathcal{T}r \leftarrow \emptyset$ /* set to record clusters customer C^i has passed through. */
  - $\mathcal{F}eat \leftarrow \emptyset$ /* set to record topological features related to customer C^i. */
  - **for** $S_{X_d^\bullet} \in \boldsymbol{SS}$ **do**
    - **for** $X_d^\bullet \in S_{X_d^\bullet}$ **do**
      - **if** $C^i \in X_d^\bullet$ **then**
        - $\mathcal{T}r \leftarrow \mathcal{T}r \bigcup \{X_d^\bullet\}$
        - Get topological features $F_d^\bullet$ of cluster $X_d^\bullet$
        - $\mathcal{F}eat \leftarrow \mathcal{F}eat \bigcup \{F_d^\bullet\}$
    - Over the set of topological features $\mathcal{F}eat$, get the set of features $\boldsymbol{\Delta F}^{\bullet\, i} = \{\Delta F^{\bullet\, i}_{d-1\rightarrow d}\}_{d=2}^{D}$, as discussed in section 4.2.6
    - Using the past topological features $\boldsymbol{\Delta F}^{\bullet\, i}$ as input information to the LSTM neural network described in section 4.2.6, forecast features $\hat{\boldsymbol{\Delta F}}^{\bullet\, i} = \{\hat{\Delta F}^{\bullet\, i}_{d-1\rightarrow d}\}_{d\in\mathcal{D}}$ at subsequent days $\mathcal{D}$
    - Using Eq. (4.21) find the evolving clusters that customer C^i will belong to at subsequent days $\mathcal{D}$, as discussed in section 4.2.6
    - Using Eq. (4.22) predict customer C^i's electricity consumption $\widehat{\vec{\boldsymbol{E}}^{\boldsymbol{i}}} = \{\widehat{\vec{E}^i_d}\}_{d\in\mathcal{D}}$ at subsequent days in $\mathcal{D}$
    - $\hat{\boldsymbol{E}} \leftarrow \hat{\boldsymbol{E}} \bigcup \widehat{\vec{\boldsymbol{E}}^{\boldsymbol{i}}}$

$$\mathcal{L}og(\mathcal{P}(\Gamma_d)) = \sum_{S_{X_d^\bullet}\in\boldsymbol{SS}} \sum_{x=1}^{d} \delta_x \left\{ \boldsymbol{\mathcal{WF}}^\bullet_x \Gamma_x - \mathcal{L}og\left[\sum_{S_{X_d^\bullet}} exp(\boldsymbol{\mathcal{WF}}^\bullet_x \Gamma_x) \right] \right\} \tag{4.20}$$

Having obtained the partial log-likelihood, we can approximate the parameter Γ_d using the Newton-Raphson algorithm.

4.2.6 Forecasting customer electricity consumption

Features forecasting

Note that the calculation of the intensity process given in Eq. (4.16) assumes that the current day d is observed. To extend Eq.(4.16) for calculating the risk at a future (unobserved) day (e.g., the next day), we need to predict the feature values at that unobserved day. Since extracted features are time-dependent, the problem of predicting their values can be reduced to a multivariate time series prediction [116]. Several algorithms have so far been proposed for predicting multivariate time series. We note, for instance, the vector auto-regressive model, which takes as input a vector of features and applies an auto-regression to each component of the vector. Such a model has proven successful if the feature evolution is linear and stationary. In many real cases, however, the observed features do not meet the conditions of linearity and stationarity. Non-linear approaches such as support vector regression (SVR) models are useful alternative methods that have proven their capacity for forecasting series with low error rates. However, in SVR models, user support is required to define the appropriate size of the historical information needed to train the model, and this size remains constant over time. In time-sequential data, the required amount of historical information may vary according to the target. Fortunately, computational approaches that consider the potential non-linearity and non-stationarity of the input data have been proposed. We note particularly the LSTM neural network, which has the capability of memorizing historical data in order to foresee subsequent observations.

In this paper we make use of an LSTM on features $\boldsymbol{\mathcal{WF}}_2^\bullet$, ...,$\boldsymbol{\mathcal{WF}}_d^\bullet$ ($d > 2$) in order to generate future feature values $\boldsymbol{\mathcal{W}}\hat{\boldsymbol{\mathcal{F}}}_{d+1}^\bullet = (\hat{\Delta F}_{d\to d+1}^\bullet, \hat{\mathcal{E}}_{d\to d+1}^\bullet)$ at day $d+1$ (respectively at subsequent days). In our case study, we consider the feature components to be non-correlated. For each feature dimension, we therefore construct an LSTM recurrent network using a dynamic sliding window method. The term "static sliding window" indicates a window of constant size that moves over time. For example, given the current day d, say we want to predict the value at the next day $d+1$. To do this we use the prior interval of days (window) $d-2$, $d-1$, d. To predict the value at day $d+2$ we then use the interval of days $d-1$, d, $d+1$, which overlaps with the previous interval. In this example, we have a window of static size 3 which has a one-day step

motion. For a dynamic sliding window, the size of the window is changing as the window moves.

Assuming that we know the feature values till day d, in order to identify the appropriate size of window to use to predict feature values at day $d+1$, we use an auto-regressive model of order p_d ($AR(p_d)$), where the p_d is taken such that the Akaike Information Criterion (AIC) is maximized over features from the first day till day d. For predicting feature values at day $d+2$, we make use of an $AR(p_{d+1})$, where p_{d+1} is chosen such that the AIC is maximized over features from the second day till day $d+1$. The process is then iterated over days to figure out the suitable window size to use. It should be noted that our LSTM network has as input layer 120 nodes and three layers, the first two having 60 and 30 neurons respectively. The last layer has one node which corresponds to the predicted feature values.

Electricity load forecasting

Assume we know the electricity consumption of customer C^i till a given date d. In order to forecast the customer's consumption at the next day $d+1$, we first trace the sequence of clusters the customer has passed through on the previous days. Over the sequence of clusters, we extract topological features $\Delta F^{\bullet\, i}_{d-1\to d}$ related to customer C^i. Over the set of obtained features, we train an LSTM recursive neural network, as discussed in section 4.2.6, in order to predict the next topological feature values $\hat{\Delta F}^{\bullet\, i}_{d\to d+1}$. We then assign this customer to the evolving cluster whose predicted topological features $\hat{\Delta F}^{\bullet}_{d\to d+1}$ are closest to $\hat{\Delta F}^{\bullet\, i}_{d\to d+1}$. Formally, the next evolving cluster to which customer C^i is supposed to belong is given as follows:

$$\mathcal{NE}(C^i) = \underset{S_{X^\bullet_d}\in \boldsymbol{SS}}{argmin}\Big\{ S_{X^\bullet_d} \equiv \hat{\Delta F}^{\bullet}_{d\to d+1},\ |\hat{\Delta F}^{\bullet}_{d\to d+1} - \hat{\Delta F}^{\bullet\, i}_{d\to d+1}|\Big\} \tag{4.21}$$

Once we know what evolving cluster customer C^i will belong to at $d+1$, we can predict this customer's electricity subprofile $\widehat{\vec{E}^{\,i}_{d+1}}$ based on the representative pattern $\vec{R}^{\bullet}_d$ for the evolving cluster at day d, as follows:

Table 4.2 – An overview of the proposed approach.

Step	Action	Parameter used
(1)	**Identifying and tracking customer electricity consumption behaviors:** Given a set of customer electricity consumption data, run Algorithm 5 to identify and track groups of customers' electricity consumption day-to-day.	Thresholds for constructing the daily network G_d: ω_d^1, ω_d^2
(2)	**Features extractions:** From the clusters and their evolution obtained in step (1), run Algorithm 6 to extract topological and structural changes features.	
(3)	**Electricity consumption behavior survivals and forecasting:** Based on the survival analysis approach developed in section 4.2.5 and the LSTM recurrent network discussed in section 4.2.6 run Algorithm 7 to predict customers' electricity load at subsequent days.	- Parameters of the used distribution for the baseline: $\boldsymbol{\Theta}$ - Setting of the LSTM network: 3 layers of 120, 30 and 15 neurons respectively stacked - Cox parameters: Γ_d

$$\widehat{\vec{E}_{d+1}^{\,i}} = Surv(d \,|\, S_{X_d^\bullet}) \;\times \vec{R}_d^\bullet \tag{4.22}$$

By running Algorithm 7 we can forecast customer electricity load at future days.

To conclude this section, in Table 4.2, we summarize the main steps of the whole approach with the corresponding parameters.

4.3 Experiments

4.3.1 Data description

In this section, we illustrate the suitability of our approach on electricity consumption data drawn from the Electricity Load Forecasting (ELF) data set collected

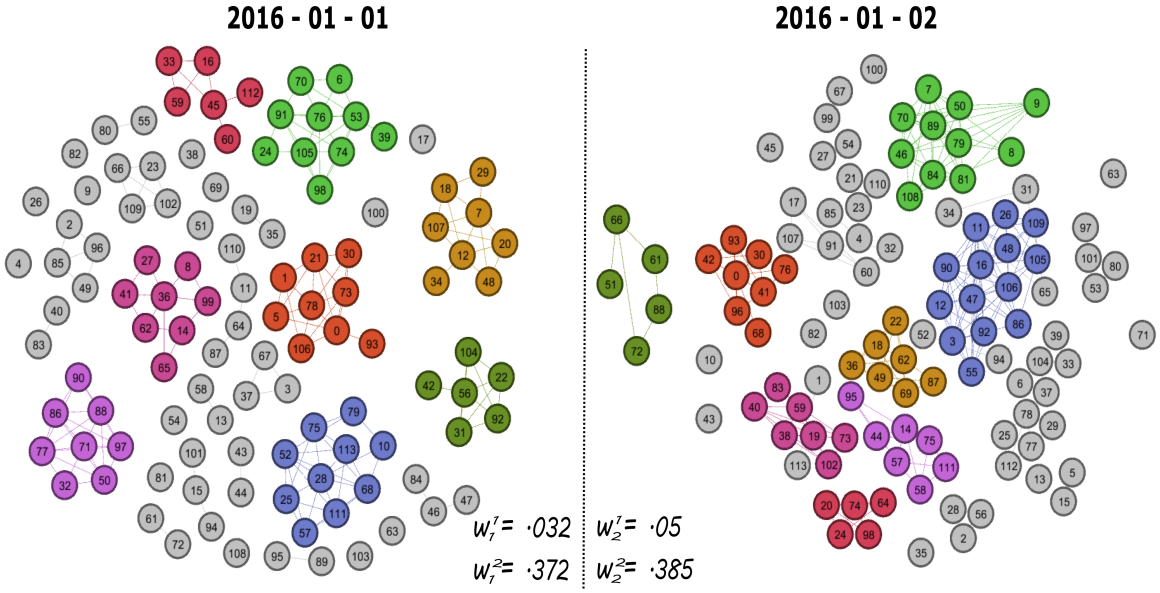
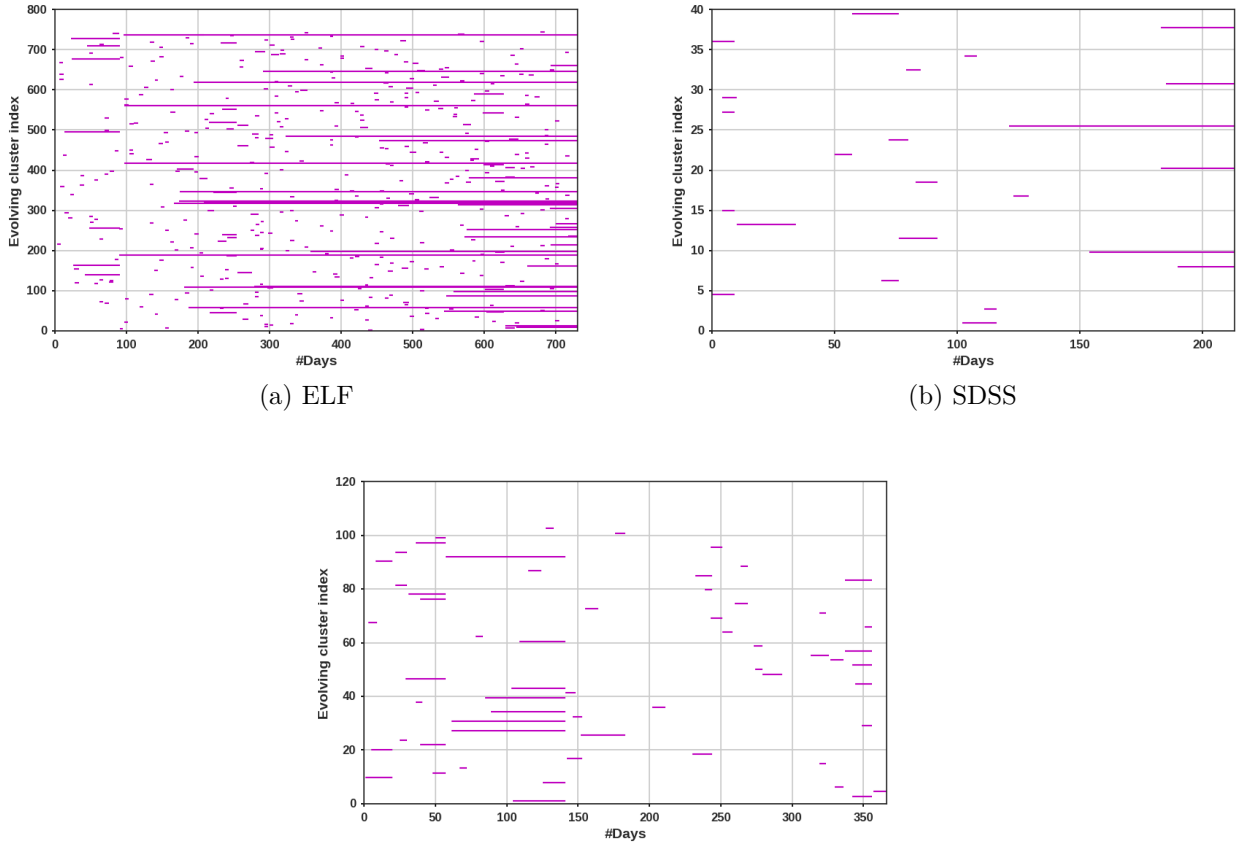


Figure 4.4 – SDSS network structure on dates $2016-01-01$ and $2016-01-02$. Labels on the nodes correspond to the customer labels. ω_1^1, ω_1^2 and ω_2^1, ω_2^2 are the identified thresholds used to build the network on these two first consecutive days. Nodes densely connected (subnetworks) and tagged with the same color exhibit the same electricity consumption behavior.

from the UCI archive [2], Smart Dataset for Sustainability (SDSS) [3] and the EnerNOC website [4] (EnerNOC GreenButton Data). The ELF data set comprises 370 customers' electricity consumption ranging from 2013-01-01 to 2014-12-31 (730 days) with the read set to 1 hour. This corresponds to a total of $730 \times 24 = 17,520$ reads per customer. The SDSS data set contains electricity consumption data for 114 single-family apartments for the period ranging from 2016-01-01 to 2016-07-31 (213 days), with a read set to 1 hour. Hence, for each customer we have a total of $213 \times 24 = 5,112$ reads. In EnerNOC data set, we have 100 customers. For each customer we have the whole year 2012 (366 days), with a read set to 1 hour. Hence, for each customer we have a total of $366 \times 24 = 8,784$ reads.

2. http://archive.ics.uci.edu/ml/datasets
3. http://traces.cs.umass.edu/index.php/Smart/Smart
4. https://open-enernoc-data.s3.amazonaws.com/anon/index.html



(a) ELF

(b) SDSS

(c) EnerNOC

Figure 4.5 – Lifespan of evolving clusters, in ELF, SDSS and EnerNOC. Only evolving clusters that last for at least 5 days long are reported.

4.3.2 Evolving network

To illustrate the variation in customers' electricity consumption, we investigate the interconnection between customers along time (expressed in days). In Figure 4.4 for instance, we have the interconnection between customers in SDSS data set on $2016-01-01$ and $2016-01-02$. From Figure 4.4, we can see that some subnetworks representing the cluster of customers having the same electricity consumption behavior vary with days in different ways. For instance, the cluster tags in blue initially observed on $2016-01-01$ has 10 nodes whereas on the next day $(2016-01-02)$ it has 14 nodes: the cluster has undergone a structural change which we call *expand*. Moreover,

in this structural change, we note that several nodes left the cluster whereas other nodes join this clusters. This demonstrates that there are some customers that might exhibit different consumption behavior in different days. In some way, some nodes remain in their respective clusters in the consecutive days; which shows that they are not changing their electricity consumption behavior.

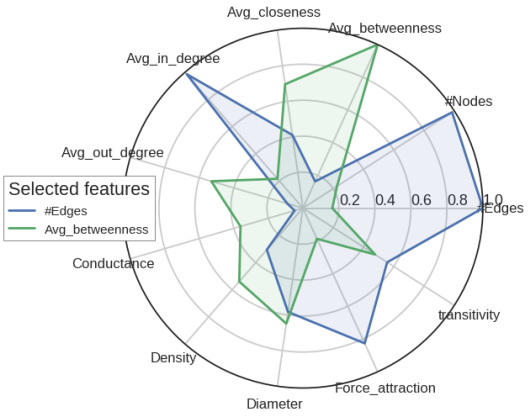
From what is illustrated in Figure 4.4, we can see that, when projecting the customer electricity consumption from the temporal space to the topological space, there are important structural information characterizing the cluster. Knowing that, each cluster relates a specific electricity consumption behavior, it becomes essential to exploit this structural information when tracking the evolution of customer electricity consumption behaviors. In Figure 4.5 we have cluster lifespans as horizontal lines, for ELF, SDSS and EnerNOC. Each line represents the lifespan of an evolving cluster. An interrupted line indicates that the evolving cluster is no longer observed after a certain day. In our context, such disappearance (case when a cluster is no longer observed) is considered as *death*, which means that the cluster's evolution ceases. Finally, it can be seen that the lifespan of some evolving clusters does not start from the first day, which means that these clusters were not identified at the beginning (first day) of the tracking.

Recall that the goal of cluster tracking in this paper is to describe customers' electricity consumption behavior over time. Hence, in Figure 4.5 each horizontal line from one point (one day) to another corresponds to the initiation of an electricity consumption behavior which lasts till a specific time point. In order to better understand each of these evolving behaviors, we investigated a set of features depicting the interrelation of customers constituting these clusters and all of the changes undergone by the clusters. The following section discusses the results obtained on these features in more detail.

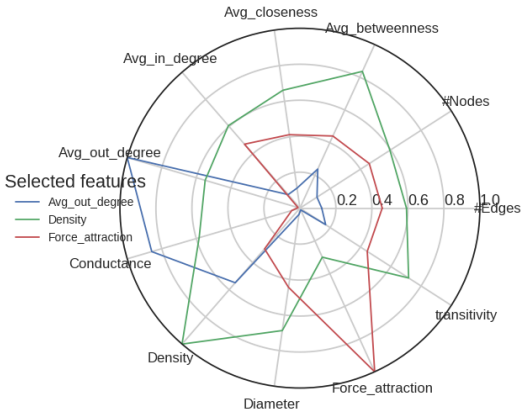
4.3.3 Features' identification

Topological features

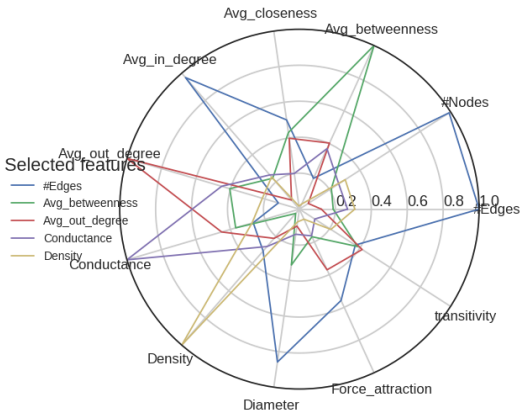
Based on the set of topological features described in section 4.2.4, we start by extracting all features from the detected clusters. Then, we apply the Pearson cor-



(a) ELF



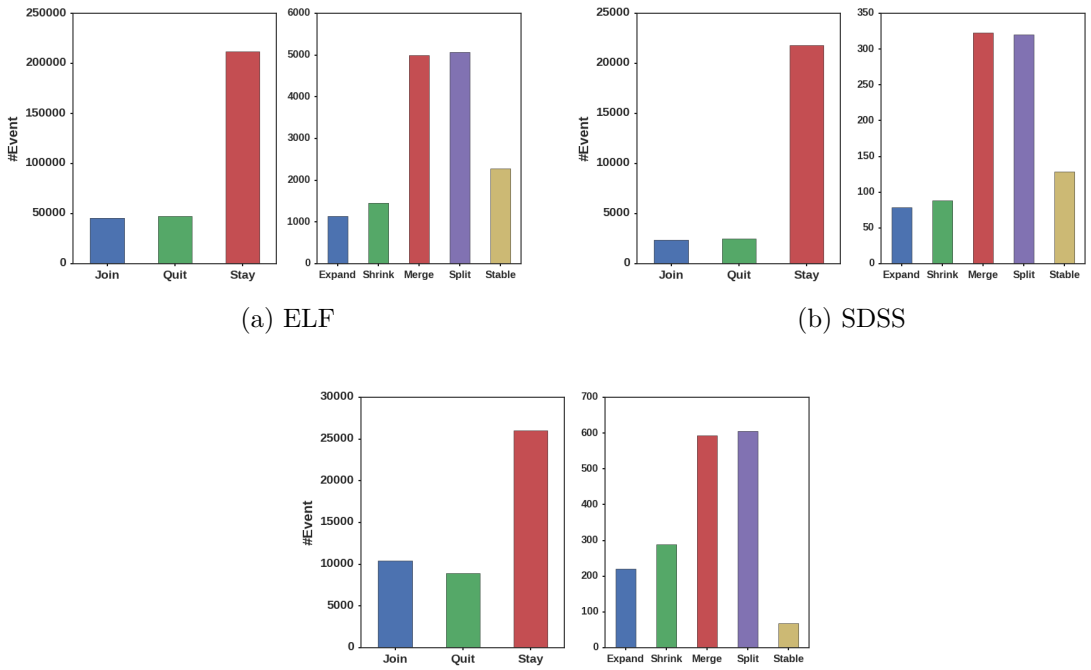
(b) SDSS



(c) EnerNOC

Figure 4.6 – Selected topological features in ELF (a), SDSS (b) and EnerNOC (c).

relation on the whole set of extracted topological features in each of the datasets (ELF, SDSS and EnerNOC). Fig 4.6 shows the radar plots depicting the correlation between features. Each circle within the radar represents a level of correlation between features. Polygons drawn within the radar indicate the correlation of selected features relative to other features. For instance, we can see for the ELF dataset (Fig 4.6(a)) that, out of 11 topological features, only two *#Edges* and *Avg_betweenness* were retained. Of these two, it is clear that the *Avg_betweenness* feature is less correlated to other features (value below .5). In contrast, the *#Edges* feature is highly correlated



(a) ELF

(b) SDSS

(c) EnerNOC

Figure 4.7 – Number of structural changes in ELF, SDSS and EnerNOC data sets.

to the features *#Nodes*, *Avg_in_degree* and *Force_attraction*, so that any one of the latter features could have been selected in place of the one chosen in the paper. In the SDSS dataset (Fig 4.6(*b*)), more features were selected: *Avg_out_degree*, *Density* and *Force_attraction*. Here, we can see that the selected features do have a low correlation (less than .6 in general) to other features. The same observations are made for EnerNOC data set, however, for this last case, we note more features have been selected due to the fact that they all less correlated.

Structural changes

Having selected suitable topological features, we associate with these the structural changes describing the evolution of each evolving cluster. Figure 4.7 shows the overall number of customers joining, leaving and staying in evolving clusters and the overall number of structural changes (expand, shrink, merge, split and stable). It can be seen

that, the number of nodes staying in clusters is higher than the number of nodes joining or quitting clusters in all data sets. Such observations reveal that most customers tend to maintain their electricity consumption behavior in ELF, SDSS and EnerNOC. With a high number of nodes staying in their respective cluster, we could expect having a high number of clusters that remain stable. However, in contrary, in all data sets, we instead note that the merge and split features are the most exhibited. This can be due to the fact that, all clusters merging into one cluster use to preserve the majority of their nodes.

From the results reported in Figure 4.7, it can be seen that many customers tend not to change their electricity consumption behavior, whereas some others leave their respective evolving clusters to join others and later keep on exhibiting this latest behavior. From this it can be understood that some clusters are splitting to get merge to other clusters. This explains why cluster merging and splitting are the most frequent changes in Figure 4.7, meaning that many customers tend to adopt some particular consumption behavior over time. From the plots, we can see that as some clusters are growing larger, others are growing smaller. This may explain why the other phenomena are relatively rare.

4.3.4 Lifespan probability

Recall that the probability of a cluster's persisting over time, given in Eq.(4.17), is time-dependent. To allow the use of this equation, we need the general hazard function $\lambda(d)$, the topological features $F_d^\bullet$ and the Cox parameters Γ_d. In the following discussion, we assume, in all data sets, that all days are known except the last 30 days (1 month). Based on these intervals, we start by presenting the estimation of the hazard function, and then describe how we evaluate the Cox parameters. The whole is ultimately used to calculate the lifespan probability within the selected interval.

Estimation of the general hazard

In order to estimate the general hazard, we proceed by a counting process which consists of enumerating the number of events observed per day. For instance, Figure 4.8 gives the cumulative number of evolving clusters disappearing, plotted against

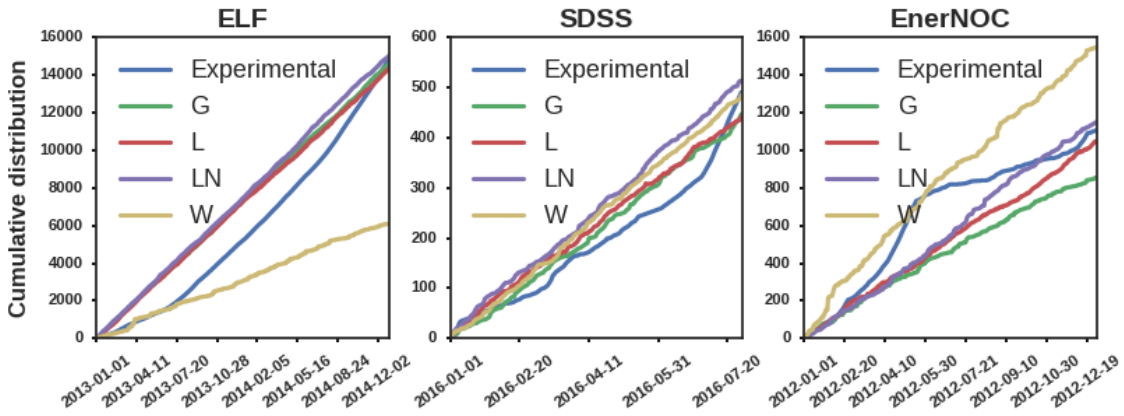


Figure 4.8 – Experimental against theoretical Cumulative distribution functions of the number of clusters disappearing in ELF, SDSS and EnerNOC.

days.

Considering the number of clusters disappearing over time, we assume that the corresponding hazard function can be modeled by a known statistical distribution. We therefore fit the cumulative time variation (empirical distribution) of the number of clusters disappearing to several theoretical cumulative density functions (CDFs): Weibull (W), log-normal (LN), gamma (G) and logistic (L). Then, to select the best distribution, we compare each of the theoretical distributions to the empirical distributions using the Kolmogorov-Smirnov (KS) test [70].

As depicted in Figure 4.8, the experimental cumulative distributions obtained show an increasing trend. Such an evolution suggests that each empirical distribution can be uniquely decomposed into two distributions [62], one being a uniform distribution and the other a martingale. Hence, we can assume that the uniform distribution corresponds to the theoretical distribution that best fits the cumulative empirical distribution from which the hazard will be deduced. Plotting theoretical cumulative density functions with the empirical distributions as in Figure 4.8, it can be clearly seen that most of the theoretical distributions fit the experimental distributions well. However, calculating the goodness of fit gives a more detailed picture. Table 4.3(a) reports fitting parameters used for each of the theoretical distribution in ELF, SDSS and EnerNOC. Table 4.3(b) reports the results for goodness of fit obtained in ELF, SDSS and EnerNOC. Results highlighted in gray correspond to the optimal theoretical

Table 4.3 – Baseline selection.

(a) Estimated parameters $\boldsymbol{\Theta}$.

	ELF	SDSS	EnerNOC
W	$\boldsymbol{\Theta} = \{0.5, -2.63e-26\}$	$\boldsymbol{\Theta} = \{1.35, -0.07\}$	$\boldsymbol{\Theta} = \{0.68, -2.18e-29\}$
LN	$\boldsymbol{\Theta} = \{0.025, -319.42\}$	$\boldsymbol{\Theta} = \{0.56, -0.726\}$	$\boldsymbol{\Theta} = \{0.75, -0.78\}$
G	$\boldsymbol{\Theta} = \{4159.05, -538.65\}$	$\boldsymbol{\Theta} = \{2.04, -0.19\}$	$\boldsymbol{\Theta} = \{0.73, -3.75e-29\}$
L	$\boldsymbol{\Theta} = \{20.35, 5.08\}$	$\boldsymbol{\Theta} = \{2.01, 0.94\}$	$\boldsymbol{\Theta} = \{2.67, 1.49\}$

(b) Goodness of fit. Boldface indicates the optimal fitting score.

	W	LN	G	L
ELF	0.67	0.05	**0.04**	0.05
SDSS	0.23	**0.21**	0.23	0.25
EnerNOC	0.23	**0.12**	0.28	0.15

distribution, indicating that the log-normal (LN) distributions should be used as the general hazard function ($\lambda(d)$ in Eq.(4.16)) in SDSS and EnerNOC respectively whereas the gamma (G) distribution should be used as baseline function in ELF data set.

Cox parameters & survival probability

Having presented the topological features and structural changes extracted from evolving clusters, we can now use this information to identify the time-dependent Cox regression parameters Γ_d. In Figure 4.9 we have an illustration of the values taken by Γ_d at 5 first consecutive days in ELF, SDSS and EnerNOC. As can be seen, the value taken by each component $P_1 - P_{13}$ of Γ_d differ from one day to another in all data sets. This variation in the estimation of the parameter Γ_d corroborate with the assumption we made on the changing behavior exhibited by different customers in their electricity consumption. It should be noted that, the number of components varies from one data set to another. In ELF data set for example $\Gamma_d = \{P_1, ..., P_{10}\}$, in SDSS $\Gamma_d = \{P_1, ..., P_{11}\}$ and in EnerNOC $\Gamma_d = \{P_1, ..., P_{13}\}$.

Based on the identified Cox coefficients in each of the data sets (ELF, SDSS and EnerNOC), we use Eq. (4.17) to calculate the time-varying probability of an evolving

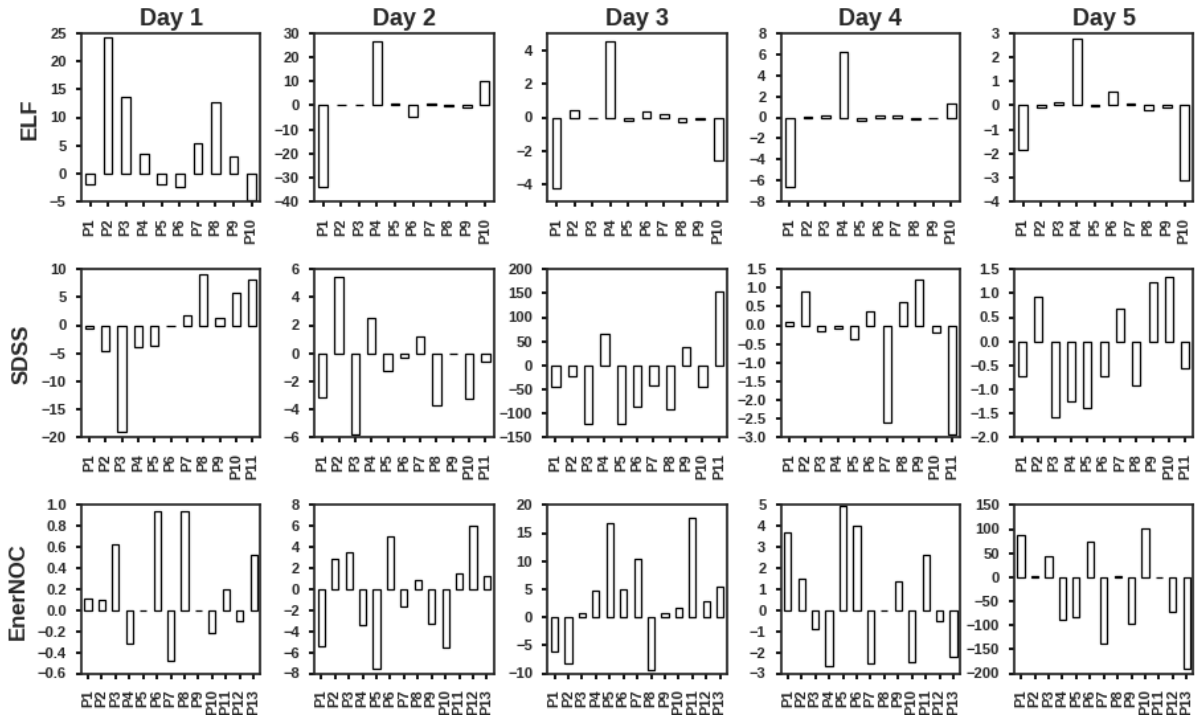


Figure 4.9 – Cox coefficients evaluated at 5 consecutive days in the ELF, SDSS and EnerNOC datasets, respectively.

cluster to still be observed. Note that, for interval where we have no observation for a given evolving cluster, we fix the parameter and the feature values to zero. Hence, for a day where we have no observation, the intensity process given in Eq. (4.16) is calculated only based on the baseline function $\lambda(d)$. In Figure 4.10, we have the average probabilities of evolving clusters' persisting in ELF, SDSS and EnerNOC. In all cases, we can see that the we have in general a survival curve that decreases with time. In ELF case, we note that the survival curve decreases less rapidly than in SDSS and EnerNOC data sets. This means that electricity consumption behaviors exhibited by customers persist the most in ELF than in SDSS and EnerNOC. Such observation corroborate with the fact that, in ELF data set, evolving clusters are lasting more than the ones in SDSS and EnerNOC data sets.

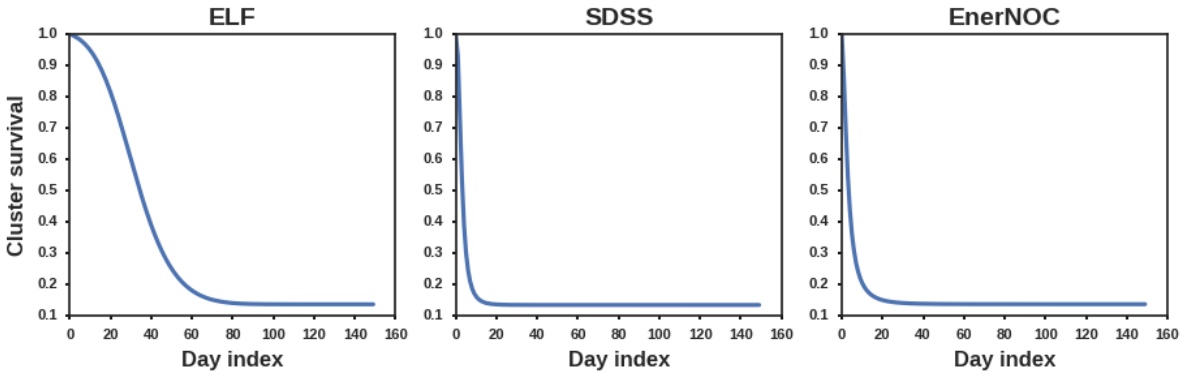


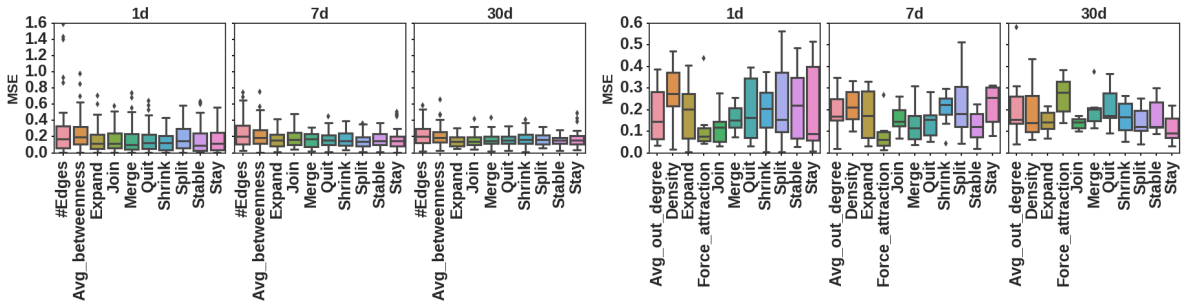
Figure 4.10 – Average of evolving cluster survival in ELF, SDSS and EnerNOC. Each index on the X-axis corresponds to half a day (12 hours).

4.3.5 Forecasting

As discussed in section 4.3.4, we assume that all days of customer electricity consumption are known except the last month in all data sets. In order to predict customer consumption at subsequent days we need to apply Eq.(4.22). For that purpose, however, feature values at the subsequent days must be estimated. Below, we present the use of the LSTM to forecast features and thus the electricity load consumption.

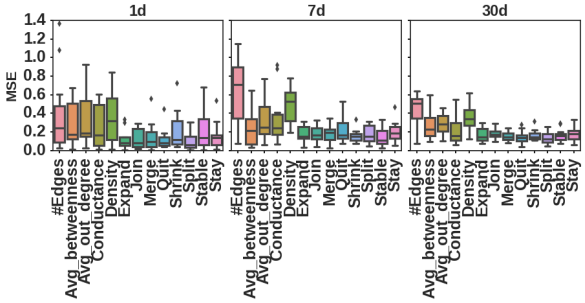
Features forecasting

In our experiments, a test was done to evaluate the accuracy of our LSTM in forecasting feature values at subsequent days. The predicted values were compared with the truth ones. The boxplots in Figure 4.11 present the accuracy of the model in predicting features values at 1, 7 and 30 days ahead in ELF, SDSS and EnerNOC datasets, respectively. Overall, with respect to the mean square error (MSE), the low error rate depicted in Figure 4.11 shows that, the LSTM used in the proposed approach can capture the non-linearity that exists between the selected features.



(a) ELF

(b) SDSS



(c) EnerNOC

Figure 4.11 – Forecasting accuracy for topological features and structural changes, in ELF, SDSS and EnerNOC.

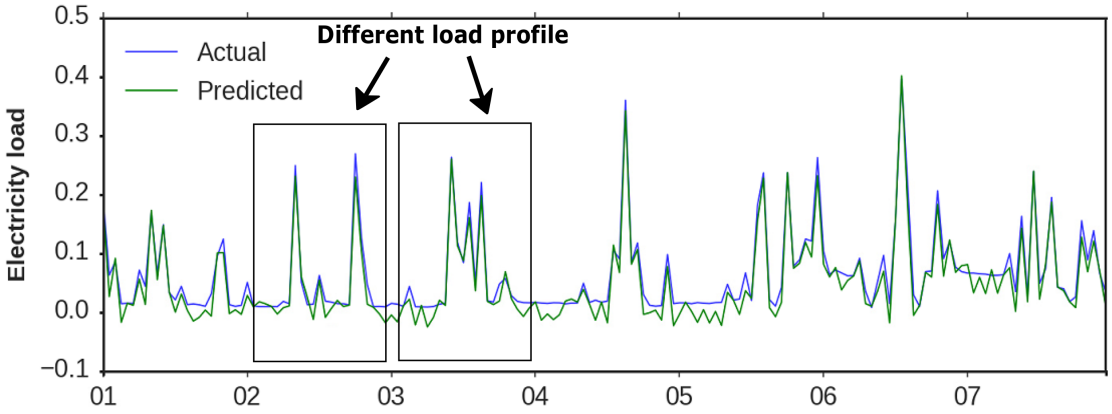


Figure 4.12 – Example of electricity load forecasting in SDSS data set. We selected one customer electricity consumption among the set of customer electricity consumption in SDSS.

Table 4.4 – Average mean square error when forecasting 1 day ($1d$), 7 days ($7d$) and 30 days ($30d$) ahead in ELF, SDSS and EnerNOC data sets. Reported values are in percentage.

	ELF			SDSS			EnerNOC		
Algorithms	$1d$	$7d$	$30d$	$1d$	$7d$	$30d$	$1d$	$7d$	$30d$
SARIMA	18.37	37.98	57.14	34.56	41.2	57.8	38.73	43.4	61
LSTM	18.56	21.34	45.94	39.08	47.06	49.53	37.04	50.26	54.37
RNN	19.34	25.21	32.54	37.61	48.23	56.03	39.13	61.24	66.17
SVR	34.28	52.94	63.15	41.01	46.73	51.80	39.03	44.58	54.29
FAG-F	19.67	31.69	42.63	23.54	28.64	43.97	22.37	34.56	41.37
NBAG-F	12.18	20.02	39.8	16.61	22.37	31.54	17.75	19.38	31.86
Proposed appraoch	19.16	20.15	**26.45**	**16.61**	**19.25**	**21.45**	**17.75**	**19.38**	**20.24**

Electricity load forecasting

To demonstrate the importance of tracking customer electricity consumption behavior, we evaluate our model in load forecasting . Figure 4.12 for instance, depicts the forecasting of one customer electricity consumption is SDSS data set. As can be seen, the customer electricity consumption does not exhibit a behavior that persists over day. From the reported predicted values, we can see that the proposed approach can well trace the future electricity consumption of this customer. Though no further information (such as the weather, expenditure, etc.) than the customer electricity consumption are given, our model can capture the changing behavior of the customer.

To further show the capability of the proposed approach in forecasting customer electricity consumption, we conduct a comparative analysis of our proposed model against well known approaches already used in time series forecasting in general and for electricity load forecasting in particular. The compared algorithms are divided in two main groups: non-aggregated and aggregated algorithms. In the non-aggregated algorithms camp, one time series is treated at a time whereas in the aggregated algorithms, several time series are taken at once. For the non-aggregated time series, we use the Seasonal AutoRegressive Integrated Moving Average (SARIMA) model [117] which can capture the repeated behavior that might be exhibited in time series. The SARIMA model was manually parameterized as follows: For the non-seasonal component we applied an $ARIMA(6, 0, 2)$. For the seasonal component we applied an $ARIMA(0, 1, 2)_{24}$. The 24 corresponds to a period of one day (24 hours). However, due to the non-linearity aspect of time series that are not always linked to seasonality parameter, the SARIMA model may less accurately forecast customer electricity consumption. For this reason, we use also Support Vector Regression (SVR) [118] which can capture hidden non-linearity. It is worth noting that, for the SVR we choose a Gaussian kernel. Still in the non-aggregated models, we considered LSTM and Recurrent Neural Network (RNN). For both LSTM and RNN we used as input layer 120 nodes and two layers having 30 and 15 neurons respectively. The last layer has one node which corresponds to the predicted series values.

Regarding the aggregated algorithms, the main idea is to split the set of time series under investigation into clusters of homogeneous time series [101], [76]. Then, from each cluster, extract a representative pattern from which a forecasting model

will be trained to predict subsequent series values of the selected cluster. In this paper, we start by using a non-incremental aggregating principle for forecasting customer. In this setting, we cluster the overall customer electricity consumption using the FINCH [5] algorithm [119] then use the SARIMA (set as the one previously defined) to predict the load of each cluster [101] (we will use the term FAG-F for FINCH aggregation forecasting). Later on, we are using an incremental aggregating model where clusters are gradually identified as time evolves. For this case, we use the network-based approach that we have proposed in [76] (we will use the term NBAG-F for network-based aggregation forecasting) where the SVR (with the Gaussian kernel) is used for the forecasting.

In Table 4.4 we have the accuracy with respect to the mean square error when forecasting the customer electricity consumption one day, one week and one month ahead. The boldface corresponds to the lowest error made (that is , the best results). As can be seen, for all algorithms, as the number of days ahead to forecast is increasing, the error made is increasing as well. However, we note that, for our proposed approach, the error made is the lowest despite the fact that we want to predict 1 day, one week or one month ahead in SDSS and EnerNOC data sets. In the ELF data set in contrary, we note that our previous model in [76] has competitive results with the one proposed in this paper when is to forecast one day and one week ahead. This might due to the fact that, the supplemented information extracted from evolving clusters have helped in better capturing various electricity consumption behaviors. In a nutshell, results reported in Table 4.4 suggest that, the proposed approach achieves consistent results compared to other algorithms where we note that they encounter difficulties when having several days ahead to forecast.

4.4 Conclusion

In this paper, we have proposed an approach for electricity load forecasting based on survival analysis and the LSTM recurrent neural network. Our survival analysis-based model makes use of the Cox regression function. In contrast to the conventional Cox

5. We used FINCH because it is the most recent and efficient parameter-free algorithm which outperforms modern state-of-the-art clustering algorithms.

model in which the input features and parameters are time-independent, our approach considers this information as changing over time. Our hypothesis is that customers are likely to change their electricity consumption from one day to another. Hence we identify clusters of customers exhibiting similar electricity consumption day to day and track the evolution of these clusters. We make use of a network-based approach to identify clusters and thus discover topological features summarizing the overall interrelation between customers. To demonstrate the suitability of our approach, we compared our model to non-aggregated forecasting models including the SARIMA, LSTM, RNN, and SVR. We also compare the approach to aggregated forecasting models including the FAG-F and the NBAG-F models. Obtained results demonstrate that the propose approach outperforms compared algorithms. While the proposed approach achieves good results, some aspects still require further work in order to improve the model. For instance, investigating more criteria for better tracking and handling customer electricity consumption behavior. The actual tracking process relies on the correlation between representative patterns. However, the representative patterns may also drastically change with time and thus cause several cases of evolving clusters that disappear. Investigating further aspects that will strengthen the tracking of customer electricity consumption behaviors are the perspective of the current work.

Conclusion

Dans la majorité des travaux où l'on exploite les structures de graphes pour modéliser les interactions entre les différentes observations, on note deux grandes orientations prises par les chercheurs. Certains chercheurs, adoptent des approches globales où les changements sont exclusivement portés au niveau de noeuds et des liens entre ces derniers. D'autres utilisent une approche locale pour suivre les phénomènes de changements observés sur l'évolution des sous-structures de graphes. Dans notre cas, tout au moins dans cette thèse, nous nous sommes exclusivement appesantis sur une approche locale. La pertinence du travail théorique élaborée dans cette thèse s'est faite au travers des cas d'application liés à l'ARS et l'ASM. Dans les deux cas de figure, nous avons développé une nouvelle approche pour détecter un intervalle de temps pour mieux percevoir les phénomènes de changements dont les sous-structures de graphes pourraient être sujettes lors de leurs évolutions. À partir de là, nous avons modélisé les différents phénomènes de changements en nous basant sur une méthode statistique connue sous le nom d'analyse de survie. La modélisation faite sous le prisme de l'analyse de survie nous a permis de prédire les phénomènes de changements plusieurs instants futurs contrairement aux approches existantes où les phénomènes de changements sont prédits juste pour un seul intervalle de temps futur. Dépendamment du domaine ou du champ d'application, les sous-structures de graphes même s'ils sont détectés suivant le même principe (le même algorithme) présentent des interprétations distinctes et des analyses différentes. En d'autres termes, bien que les structures de graphes ont été employées pour modéliser l'information traitée dans les domaines de l'ARS et l'ASM, la modélisation et l'interprétation sont toutes divergentes.

Concernant l'ARS, les sous-structures de graphes appelées communautés sont suivies à travers le temps. Nous revisitons la notion d'intervalle de temps en définis-

sant une nouvelle notion de fenêtre coulissante qui nous permet de mieux apprécier l'évolution du réseau social en investigation. La taille de la fenêtre est ici détectée de manière automatique en se basant sur les phénomènes de base observés au niveau des noeuds. Grâce à ce principe de fenêtre coulissante, nous avons pu percevoir, de manière consistante les phénomènes de changements que pourraient subir les communautés qui évoluent dans le temps. Pour chacun des phénomènes de changements observés au niveau des communautés, en particulier les phénomènes d'*rétrécissement*, de *élargissement*, de *fusion*, de *division* et de *stabilité*, nous définissons une fonction de survie prenant en compte les régressions de Cox et linéaire et multivariée. Ainsi, pour chacun de ces phénomènes de changements, il devient possible de prédire à un plusieurs instants futurs les risques auxquels s'expose une communauté qui évolue dans le temps.

Il est important de noter que la régression linéaire ici ajoutée au modèle de Cox a pour but de générer des caractéristiques des communautés aux instants futurs pour lesquels on voudrait faire de la prédiction. Sur le plan expérimental, nous démontrons effectivement que l'apport de la régression linéaire nous a permis d'améliorer la qualité de prédiction comparativement au modèle de Cox conventionnel. Il vaut la peine de rappeler que dans le modèle de Cox conventionnel, on a connaissance des caractéristiques jusqu'à un moment connu. Pour les dates futures pour lesquelles on voudrait faire de la prédiction, uniquement la fonction de base est exploitée pour faire le calcul de probabilité. Ainsi, dans le cas où le choix de cette fonction de base est mal fait, la qualité de la prédiction sera par conséquent peu précise. Bien qu'ayant combiné les régressions de Cox et linéaire multivarié pour modéliser chacun des phénomènes de changements, les caractéristiques exploitées ne sont pas toujours linéaires. En d'autres termes, une fonction linéaire ne définirait pas toujours le principe suivant lequel les caractéristiques extraites des communautés évoluent dans le temps. Par conséquent, la qualité de la précision lors du calcul de probabilité pourrait elle aussi être affectée.

Concernant l'ASM, particulièrement pour les séries exhibant plusieurs régimes à plusieurs instants distincts, les sous-structures de graphes représentent les différents régimes. Comme dans le cas de l'ARS, nous déterminons automatiquement un intervalle de temps régulier pour lequel on pourrait exprimer les différents régimes exhibés par les différentes séries. Ainsi, grâce à une matrice d'appartenance nous identifions toutes

les séries exhibant chacun des régimes à différents intervalles de temps redéfinis. Ces appartenances nous informent pour chacun des régimes comment est-ce que les séries les ont exprimées tout au long du temps. C'est ainsi que, en se basant sur le modèle d'analyse de survie, nous avons pu modéliser le risque qu'un régime soit exhibé ou pas. Par la force de cette matrice d'appartenance, nous construisons également à chaque intervalles de temps consécutifs une matrice de transition qui permet de calculer la probabilité de transition d'un régime à un autre contrairement aux modèles de Markov où une seule matrice fait office de calcul de probabilité de passage d'un régime à un autre. Sachant que les caractéristiques extraites des séquences de sous-structures de graphes pourraient ne pas être linéaires, nous jumelons à la régression de Cox la regression des K plus proches voisins (KNNR) pour générer les caractéristiques aux intervalles de temps futurs pour lesquels on aimerait faire de la prévision.

L'usage de l'analyse de survie et des graphes dynamiques est également exploité dans cette thèse pour modéliser le comportement des clients dans leurs consommations électriques. Les sous-structures de graphes sont exploitées pour représenter un groupe de clients dont la consommation électrique suit le même patron. Les différents comportements sont par la suite suivis de manière quotidienne dans le temps. Ceci nous permet de détecter les changements de comportements que présentent les clients dans leurs consommations électriques respectives à travers le temps. Comme dans l'ARS, dans ce cas de figure de l'ASM, on extrait également les phénomènes de changements liés aux noeuds et aux sous-structures de graphes comme informations pertinentes auxquelles on lie les caractéristiques topologiques pour calculer le risque qu'un comportement se produise à des instants futurs. À partir de ce risque, nous déduisons donc les valeurs de consommation électrique qu'auront les clients dans les futurs jours. Il faut noter que nous générons les valeurs des différentes caractéristiques dans le futur par le biais d'un réseau récurrent LSTM dont la mémoire est itérativement recalculée.

En résumé, nous avons présenté dans cette thèse des méthodes de régressions linéaire et non linéaire et réseau de neurones qui ont contribué au modèle de survie exploitée dans cette thèse. De par les résultats obtenus, on peut justifier que les modèles théoriques développés dans différents cas de figure nous ont permis de prédire les phénomènes de changements auxquels s'exposeraient les sous-structures de graphes qui évoluent dans le temps. Il est important de rappeler que, de toutes les

problématiques concrètement adressées dans cette thèse, nous avons d'une part utilisé les caractéristiques topologiques sur lesquels nous avons appliqué une méthode de sélection des caractéristiques. Le choix des caractéristiques ici fait soit par selection de celles qui sont moins correlées entre elles soit en projetant dans un espace latent par le biais de l'ACP. D'autre part, nous avons fait usage des graphes autoencodeurs pour automatiquement projeter la structure du graphe dans un espace latent et extraite les caractéristiques qui permettraient de mieux représenter les différentes observations. Bien que ces différentes approches de sélections de caractéristiques ont bel et bien permis d'obtenir des résultats pertinents, un travail supplémentaire est requis quant à la généralisation du choix des caractéristiques lorsque nous avons à faire à l'analyse d'une structure de graphe qui évolue dans le temps.

Bien que le problème de généralisation de caractéristiques est important dans la modélisation de l'évolution des structures de graphes, on note également une pertinence quant aux choix des fonctions de base des différents modèles de Cox exploités. Dans tous les cas (que soit dans les ARS ou les ASM) on peut constater que la fonction de base était toujours choisie parmi un certain nombre de densités de probabilités prédéfinies. Or l'on pourrait utiliser une approche bayésienne pour automatiquement déterminer une densité de type exponentielle satisfaisante. La définition de la fonction de survie y compris la recherche des caractéristiques générales qui pourraient expliquer l'évolution des graphes en général et en particulier des sous-structures de graphes sont des points importants à investiguer non pas seulement dans les domaines respectifs de l'ARS et de l'ASM.

Il est important de noter que, bien que le travail élaboré dans cette thèse soit entièrement appesanti sur les cas spécifiques des ARS et ASM, les modèles théoriques développés vont au-delà des cas pratiques présentés. L'on pourrait par exemple exploiter les modèles théoriques développés dans les études cliniques pour étudier l'espérance de vie des patients ou des bêtes dans le cas de l'élevage. On pourrait aussi exploiter le modèle théorique pour modéliser la dégradation du langage communicationnel exposé par des internautes dans les réseaux sociaux.

Bibliographie

[1] Shanshan Feng, Gao Cong, Arijit Khan, Xiucheng Li, Yong Liu, and Yeow Meng Chee. Inf2vec : Latent representation model for social influence embedding. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 941–952. IEEE, 2018.

[2] Haiquan Chen, Wei-Shinn Ku, Haixun Wang, Liang Tang, and Min-Te Sun. Linkprobe : Probabilistic inference on large-scale social networks. In *2013 IEEE 29th international conference on data engineering (ICDE)*, pages 290–301. IEEE, 2013.

[3] Jianxin Li, Chengfei Liu, and Md Saiful Islam. Keyword-based correlated network computation over large social media. In *2014 IEEE 30th International Conference on Data Engineering*, pages 268–279. IEEE, 2014.

[4] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. Efficiently reinforcing social networks over user engagement and tie strength. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 557–568. IEEE, 2018.

[5] Philippe Fournier-Viger, Chao Cheng, Zhi Cheng, Jerry Chun-Wei Lin, and Nazha Selmaoui-Folcher. Mining significant trend sequences in dynamic attributed graphs. *Knowledge-Based Systems*, 2019.

[6] Jacob Van Der Woude, Christian Commault, and Taha Boukhobza. A dynamic graph characterisation of the fixed part of the controllable subspace of a linear structured system. *Systems & Control Letters*, 129 :17–25, 2019.

[7] Dan Shi, Lei Zhu, Zhiyong Cheng, Zhihui Li, and Huaxiang Zhang. Unsupervised multi-view feature extraction with dynamic graph learning. *Journal of Visual Communication and Image Representation*, 56 :256–264, 2018.

[8] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time : densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th International Conference on Knowledge Discovery in Data mining (SIGKDD)*, pages 177–187. ACM, 2005.

[9] Reda Alhajj and Jon Rokne. *Encyclopedia of social network analysis and mining*. Springer Publishing Company, Incorporated, 2014.

[10] Jaideep Srivastava. Data mining for social network analysis. In *2008 IEEE International Conference on Intelligence and Security Informatics*, pages xxxiii–xxxiv. IEEE, 2008.

[11] Charu C Aggarwal. An introduction to social network data analytics. In *Social network data analytics*, pages 1–15. Springer, 2011.

[12] Giulio Rossetti, Luca Pappalardo, Dino Pedreschi, and Fosca Giannotti. Tiles : an online algorithm for community discovery in dynamic social networks. *Machine Learning*, 106(8) :1213–1241, 2017.

[13] Francesco Bonchi, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Social network analysis and mining for business applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3) :22, 2011.

[14] Ashish Sureka, Atul Goyal, and Ayushi Rastogi. Using social network analysis for mining collaboration data in a defect tracking system for risk and vulnerability analysis. In *Proceedings of the 4th india software engineering conference*, pages 195–204. ACM, 2011.

[15] Jung-hwan Bae, Ji-eun Son, and Min Song. Analysis of twitter for 2012 south korea presidential election by text mining techniques. *Journal of Intelligence and Information Systems*, 19(3) :141–156, 2013.

[16] Ravneet Kaur and Sarbjeet Singh. A survey of data mining and social network analysis based anomaly detection techniques. *Egyptian informatics journal*, 17(2) :199–216, 2016.

[17] Eytan Adar and Lada A Adamic. Tracking information epidemics in blogspace. In *Proceedings of the 2005 IEEE/WIC/ACM international conference on web intelligence*, pages 207–214. IEEE Computer Society, 2005.

[18] Naoki Masuda and Petter Holme. Predicting and controlling infectious disease epidemics using temporal networks. *F1000prime reports*, 5, 2013.

[19] Mayank Lahiri and Tanya Y Berger-Wolf. Structure prediction in temporal networks using frequent subgraphs. In *2007 IEEE Symposium on Computational Intelligence and Data Mining*, pages 35–42. IEEE, 2007.

[20] Etienne Gael Tajeuna, Mohamed Bouguessa, and Shengrui Wang. Tracking the evolution of community structures in time-evolving social networks. In *IEEE International Conference on Data Science and Advanced Analytics*, pages 1–10. IEEE, 2015.

[21] Mansoureh Takaffoli, Reihaneh Rabbany, and Osmar R Zaïane. Community evolution prediction in dynamic social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 9–16. IEEE, 2014.

[22] Derek Greene, Donal Doyle, and Padraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *International Conference on Advances in social networks analysis and mining (ASONAM)*, pages 176–183. IEEE, 2010.

[23] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23) :8577–8582, 2006.

[24] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6) :066111, 2004.

[25] Martin Rosvall et al. Maps of random walks on complex networks reveal community structure. *Proc. of the National Academy of Sciences*, 105(4) :1118–1123, 2008.

[26] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043) :814–818, 2005.

[27] Etienne Gael Tajeuna, Mohamed Bouguessa, and Shengrui Wang. Tracking communities over time in dynamic social network. In *Machine Learning and Data Mining in Pattern Recognition*, pages 341–345. Springer, 2016.

[28] Mansoureh Takaffoli, Justin Fagnan, Farzad Sangi, and Osmar R Zaïane. Tracking changes in dynamic information networks. In *International Conference on Computational Aspects of Social Networks (CASoN)*, pages 94–101. IEEE, 2011.

[29] Piotr Bródka, Stanislaw Saganowski, and Przemyslaw Kazienko. Group evolution discovery in social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 247–253. IEEE, 2011.

[30] Leonardo N Ferreira and Liang Zhao. A time series clustering technique based on community detection in networks. *Procedia Computer Science*, 53 :183–190, 2015.

[31] Leonardo N Ferreira and Liang Zhao. Time series clustering via community detection in networks. *Information Sciences*, 326 :227–242, 2016.

[32] Piotr Bródka, Przemysław Kazienko, and Bartosz Kołoszczyk. Predicting group evolution in the social network. In *International Conference on Social Informatics*, pages 54–67. Springer, 2012.

[33] Etienne Gael Tajeuna, Mohamed Bouguessa, and Shengrui Wang. Modeling and predicting community structure changes in time evolving social networks. *TKDE*, 2018.

[34] Xishun Wang, Minjie Zhang, and Fenghui Ren. Learning customer behavior for effective load forecasting. *IEEE Transaction on Knowledge and Data Engineering*, 2018.

[35] Yasuko Matsubara et al. Dynamic modeling and forecasting of time-evolving data streams. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 458–468. ACM, 2019.

[36] Etienne Gael Tajeuna, Mohamed Bouguessa, and Shengrui Wang. Survival analysis for modeling critical events that communities may undergo in dynamic social networks. In *Proceedings of the Symposium on Applied Computing*, pages 1068–1075. ACM, 2017.

[37] Axel Hochstein, Hyung-Il Ahn, Ying Tat Leung, and Matthew Denesuk. Switching vector autoregressive models with higher-order regime dynamics application to prognostics and health management. In *Proceedings of the International conference on PHM*, pages 1–10. IEEE, 2014.

[38] David Hallac et al. Network inference via the time-varying graphical lasso. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 205–213. ACM, 2017.

[39] Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282) :457–481, 1958.

[40] David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society : Series B (Methodological)*, 34(2) :187–202, 1972.

[41] Ville H Tuulos and Henry Tirri. Combining topic models and social networks for chat data mining. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, pages 206–213. IEEE, 2004.

[42] Tanja Falkowski, Jorg Bartelheimer, and Myra Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 52–58. IEEE, 2006.

[43] Quazi Marufur Rahman, Anna Fariha, Amit Mandal, Chowdhury Farhan Ahmed, and Carson K Leung. A sliding window-based algorithm for detecting leaders from social network action streams. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 1, pages 133–136. IEEE, 2015.

[44] Jiang Yang and Scott Counts. Predicting the speed, scale, and range of information diffusion in twitter. In *Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[45] Manuel Gomez-Rodriguez, Jure Leskovec, and Bernhard Schölkopf. Modeling information propagation with survival theory. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 666–674, 2013.

[46] Govind S Mudholkar, Deo Kumar Srivastava, and Georgia D Kollia. A generalization of the weibull distribution with application to the analysis of survival data. *Journal of the American Statistical Association*, 91(436) :1575–1583, 1996.

[47] Carl Lee, Felix Famoye, and Olugbenga Olumolade. Beta-weibull distribution : some properties and applications to censored data. *Journal of modern applied statistical methods*, 6(1) :17, 2007.

[48] Pei Lee, Laks VS Lakshmanan, and Evangelos E Milios. Incremental cluster evolution tracking from highly dynamic network data. In *Proceedings of the 30th International Conference on Data Engineering (ICDE)*, pages 3–14. IEEE, 2014.

[49] Nan Du, Xiaowei Jia, Jing Gao, Vishrawas Gopalakrishnan, and Aidong Zhang. Tracking temporal community strength in dynamic networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(11) :3125–3137, 2015.

[50] Sajid Yousuf Bhat and Muhammad Abulaish. Hoctracker : Tracking the evolution of hierarchical and overlapping communities in dynamic social networks. *IEEE Transactions on Knowledge and Data engineering*, 27(4) :1019–1013, 2015.

[51] Sadamori Koujaku, Mineichi Kudo, Ichigaku Takigawa, and Hideyuki Imai. Community change detection in dynamic networks in noisy environment. In *Proceedings of the 24th International Conference on World Wide Web*, pages 793–798. ACM, 2015.

[52] Georgios Diakidis, Despoina Karna, Dimitris Fasarakis-Hilliard, Dimitrios Vogiatzis, and George Paliouras. Predicting the evolution of communities in social networks. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics (WIMS)*, page 1. ACM, 2015.

[53] Bogdan Gliwa, Piotr Bródka, Anna Zygmunt, Stanislaw Saganowski, Przemyslaw Kazienko, and Jaroslaw Kozlak. Different approaches to community evolution prediction in blogosphere. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1291–1298. IEEE, 2013.

[54] Nagehan İlhan and Şule Gündüz Öğüdücü. Predicting community evolution based on time series modeling. In *Proceedings of the International Conference on*

Advances in Social Networks Analysis and Mining (ASONAM), pages 1509–1516. ACM, 2015.

[55] Nagehan İlhan and Şule Gündüz Öğüdücü. Feature identification for predicting community evolution in dynamic social networks. *Engineering Applications of Artificial Intelligence*, 55 :202–218, 2016.

[56] Ryan A Rossi, Brian Gallagher, Jennifer Neville, and Keith Henderson. Modeling dynamic behavior in large evolving graphs. In *Proceedings of the 6th International Conference on Web Search and Data Mining (WSDM)*, pages 667–676. ACM, 2013.

[57] David B Skillicorn, Quan Zheng, and Carlo Morselli. Spectral embedding for dynamic social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 316–323. ACM, 2013.

[58] Zhifeng Bao, Yong Zeng, and YC Tay. sonlp : social network link prediction by principal component regression. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 364–371. IEEE, 2013.

[59] Antoni Calvó-Armengol and Yves Zenou. Social networks and crime decisions : The role of social structure in facilitating delinquent behavior. *International Economic Review*, 45(3) :939–958, 2004.

[60] Douglas A Luke and Jenine K Harris. Network analysis in public health : history, methods, and applications. *Annu. Rev. Public Health*, 28 :69–93, 2007.

[61] DY Lin. Cox regression analysis of multivariate failure time data : the marginal approach. *Statistics in medicine*, 13(21) :2233–2247, 1994.

[62] Odd Aalen, Ornulf Borgan, and Hakon Gjessing. *Survival and event history analysis : a process point of view*. Springer Science & Business Media, 2008.

[63] Wes McKinney, Josef Perktold, and Skipper Seabold. Time series analysis in python with statsmodels. *Jarrodmillman. Com*, pages 96–102, 2011.

[64] Paul D Allison. *Event history and survival analysis : Regression for longitudinal event data*, volume 46. SAGE publications, 2014.

[65] Dimitrios Michalopoulos and Ioannis Mavridis. Utilizing survival analysis for modeling child hazards of social networking. In *HAISA*, pages 195–204, 2012.

[66] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer : extraction and mining of academic social networks. In *Proceedings of the 14th International Conference on Knowledge Discovery and Data mining)(SIGKDD)*, pages 990–998. ACM, 2008.

[67] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms : a comparative analysis. *Physical review E*, 80(5) :056117, 2009.

[68] Harald Cramér. *On the composition of elementary errors*. Almqvist & Wiksells, 1928.

[69] Theodore W Anderson and Donald A Darling. Asymptotic theory of certain" goodness of fit" criteria based on stochastic processes. *The annals of mathematical statistics*, pages 193–212, 1952.

[70] Nickolay Smirnov. Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics*, 19(2) :279–281, 1948.

[71] Jure Leskovec et al. Statistical properties of community structure in large social and information networks. In *Proc. of the International Conference on World Wide Web*, pages 695–704. ACM, 2008.

[72] Zhenqing Ye, Songnian Hu, and Jun Yu. Adaptive clustering algorithm for community detection in complex networks. *physical review E*, 78(4) :046115, 2008.

[73] Riwal Lefort and François Fleuret. treekl : A distance between high dimension empirical distributions. *Pattern Recognition Letters*, 34(2) :140–145, 2013.

[74] Matthieu Sanquer, Florent Chatelain, Mabrouka El-Guedri, and Nadine Martin. A smooth transition model for multiple-regime time series. *IEEE Transactions on Signal Processing*, 61(7) :1835–1847, 2013.

[75] Robert Dürichen et al. Multitask gaussian processes for multivariate physiological time-series analysis. *IEEE Trans. on Biomedical Engineering*, 62(1) :314–322, 2015.

[76] Etienne Gael Tajeuna, Mohamed Bouguessa, and Shengrui Wang. A network-based approach to enhance electricity load forecasting. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 266–275. IEEE, 2018.

[77] Zhongfu Tan, Jinliang Zhang, Jianhui Wang, and Jun Xu. Day-ahead electricity price forecasting using wavelet transform combined with arima and garch models. *Applied Energy*, 87(11) :3606–3610, 2010.

[78] Razvan-Gabriel Cirstea et al. Correlated time series forecasting using multi-task deep neural networks. In *Proc. of the International Conference on Information and Knowledge Management*, pages 1527–1530. ACM, 2018.

[79] Yi Zhao et al. Forecasting wavelet transformed time series with attentive neural networks. In *Proc. of the International Conference on Data Mining*, pages 1452–1457. IEEE, 2018.

[80] Yasuko Matsubara et al. Regime shifts in streams : Real-time forecasting of co-evolving time sequences. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 1045–1054. ACM, 2016.

[81] Yasuko Matsubara et al. Non-linear mining of competing local activities. In *Proc. of the International Conference on World Wide Web*, pages 737–747. International World Wide Web Conferences Steering Committee, 2016.

[82] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the International conference on ICML*, pages 1067–1075, 2013.

[83] Sybren Drijfhout et al. Catalogue of abrupt shifts in intergovernmental panel on climate change climate models. *Proc. of the National Academy of Sciences*, 112(43) :E5777–E5786, 2015.

[84] Ming-Hsiang Chen. State dependence in the influence of monetary policy regime shifts on hospitality index returns. *International Journal of Hospitality Management*, 31(4) :1203–1212, 2012.

[85] Dick van Dijk et al. Smooth transition autoregressive models—a survey of recent developments. *Econometric reviews*, 21(1) :1–47, 2002.

[86] Yongjie Cai, Hanghang Tong, Wei Fan, and Ping Ji. Fast mining of a network of coevolving time series. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 298–306. SIAM, 2015.

[87] Michel Lubrano. Bayesian analysis of nonlinear time series models with a threshold. In *Proc. of the International Symposium in Economic Theory*, volume 11, page 79. Cambridge University Press, 2000.

[88] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

[89] Lei Li, James McCann, Nancy S Pollard, and Christos Faloutsos. Dynammo : Mining and summarization of coevolving sequences with missing values. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 507–516, 2009.

[90] Yan Li, Jie Wang, Jieping Ye, and Chandan K Reddy. A multi-task learning formulation for survival analysis. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1715–1724, 2016.

[91] Huayu Li, Yong Ge, Hengshu Zhu, Hui Xiong, and Hongke Zhao. Prospecting the career development of talents : A survival analysis perspective. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 917–925, 2017.

[92] Jianfei Zhang, Shengrui Wang, Lifei Chen, Gongde Guo, Rongbo Chen, and Alain Vanasse. Time-dependent survival neural network for remaining useful life prediction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 441–452. Springer, 2019.

[93] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering : A deep attentional embedding approach. *arXiv preprint arXiv :1906.06532*, 2019.

[94] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax : a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2) :107–144, 2007.

[95] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105, 1998.

[96] Mohamed Bouguessa and Shengrui Wang. Mining projected clusters in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering*, 21(4) :507–522, 2008.

[97] Cunchao Tu, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Transnet : Translation-based network representation learning for social relation extraction. In *IJCAI*, pages 2864–2870, 2017.

[98] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs : Methods and applications. *arXiv preprint arXiv :1709.05584*, 2017.

[99] Yunsheng Song, Jiye Liang, Jing Lu, and Xingwang Zhao. An efficient instance selection algorithm for k nearest neighbor regression. *Neurocomputing*, 251 :26–34, 2017.

[100] Ruidi Chen and Ioannis Paschalidis. Selecting optimal decisions via distributionally robust nearest-neighbor regression. In *Advances in Neural Information Processing Systems*, pages 748–758, 2019.

[101] Carlos Alzate and Mathieu Sinn. Improved electricity load forecasting via kernel spectral clustering of smart meters. In *2013 IEEE 13th International Conference on Data Mining*, pages 943–948. IEEE, 2013.

[102] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1) :37–45, 2018.

[103] Liangda Li and Hongyuan Zha. Energy usage behavior modeling in energy disaggregation via hawkes processes. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(3) :36, 2018.

[104] Douglas C Montgomery, Lynwood A Johnson, and John S Gardiner. *Forecasting and time series analysis*. McGraw-Hill Companies, 1990.

[105] Joanna Nowicka-Zagrajek and Rafał Weron. Modeling electricity loads in california : Arma models with hyperbolic noise. *Signal Processing*, 82(12) :1903–1915, 2002.

[106] Jan Kostrzewa. Time series forecasting using clustering with periodic pattern. In *7th International Joint Conference on Computational Intelligence (IJCCI)*, volume 3, pages 85–92. IEEE, 2015.

[107] Peter Laurinec, Marek Lóderer, Petra Vrablecová, Mária Lucká, Viera Rozinajová, and Anna Bou Ezzeddine. Adaptive time series forecasting of energy consumption using optimized cluster analysis. In *16th IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 398–405. IEEE, 2016.

[108] Yogesh Simmhan and Muhammad Usman Noor. Scalable prediction of energy consumption using incremental time series clustering. In *Proceedings of the International Conference on Big Data*, pages 29–36. IEEE, 2013.

[109] Etienne Gael Tajeuna, Mohamed Bouguessa, and Shengrui Wang. A network-based approach to enhance electricity load forecasting. In *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018.

[110] Junjing Yang, Chao Ning, Chirag Deb, Fan Zhang, David Cheong, Siew Eang Lee, Chandra Sekhar, and Kwok Wai Tham. k-shape clustering algorithm for building energy usage patterns analysis and forecasting model accuracy improvement. *Energy and Buildings*, 146 :27–37, 2017.

[111] Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity : uses and interpretations. *Neuroimage*, 52(3) :1059–1069, 2010.

[112] Juan David Cruz, Cécile Bothorel, and François Poulet. Community detection and visualization in social networks : Integrating structural and semantic information. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1) :11, 2013.

[113] Xuning Tang and Christopher C Yang. Detecting social media hidden communities using dynamic stochastic blockmodel with temporal dirichlet process. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(2) :36, 2014.

[114] Zhonggang Wu, Zhao Lu, and Shan-Yuan Ho. Community detection with topological structure and attributes in information networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2) :33, 2017.

[115] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6) :066133, 2004.

[116] Zhenxing Wang and Laiwan Chan. Learning causal relations in multivariate time series data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4) :76, 2012.

[117] Tingting Fang and Risto Lahdelma. Evaluation of a multiple linear regression model and sarima model in forecasting heat demand for district heating system. *Applied energy*, 179 :544–552, 2016.

[118] Yongbao Chen, Peng Xu, Yiyi Chu, Weilin Li, Yuntao Wu, Lizhou Ni, Yi Bao, and Kun Wang. Short-term electrical load forecasting using the support vector regression (svr) model to calculate the demand response baseline for office buildings. *Applied Energy*, 195 :659–670, 2017.

[119] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2019.