UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

« Maël GUILLEMÉ »

« Extraction de connaissances interprétables dans des séries temporelles »

Thèse présentée et soutenue à Rennes, le 16 décembre 2019
Unité de recherche : UMR 6074 IRISA

Rapporteurs avant soutenance :

Anne LAURENT	Professeur, Université de Montpellier
Bruno CRÉMILLEUX	Professeur, Université de Caen-Normandie

Composition du Jury :

Président :	Jacques NICOLAS	Directeur de Recherche INRIA, INRIA RBA
Examinateurs :	Benoît FRENAY	Professeur associé, Université de Namur
	Benjamin NÉGREVERGNE	Maître de conférences, Université de Paris-Dauphine
	Véronique MASSON	Maître de conférences, Université de Rennes 1
	Laurence ROZÉ	Maître de conférences, INSA Rennes
Dir. de thèse :	Alexandre TERMIER	Professeur, Université de Rennes 1

Invité(s) :

Aymeric HERVIEU	Directeur Technique en charge de la R& D, Energiency

REMERCIEMENTS

Je remercie d'abord Arnaud et Energiency sans qui cette thèse n'aurait jamais pu exister. Je remercie Arnaud pour la confiance qu'il m'a accordé dans ce projet et de son écoute tout au long de ces 3 années. Je remercie tous les collègues que j'ai côtoyé à Energiency au fil de cette thèse.

Je remercie l'ensemble de mes encadrant.e.s de Energiency. Erwan qui m'a guidé durant ma première année de thèse. Cérès dont le soutien et les conseils durant mes deux premières années m'ont été d'une aide incommensurable. Enfin, Aymeric qui m'a suivi sur l'ensemble de cette thèse et dont je remercie la bienveillance et l'écoute.

Je remercie les rapporteurs et les membres de mon jury d'avoir pris de leur temps pour évaluer ma thèse. Je remercie aussi l'ensemble des chercheurs extérieurs avec qui j'ai collaboré durant cette thèse. En particulier, Romain et Simon qui m'ont intégrer et accompagner dans leur travaux, et Esther qui a toujours été de très bon conseil sur ma recherche.

Je remercie l'ensemble des membres de l'équipe de recherche LACODAM, qui m'ont accueilli au cours de ces trois ans (et plus si je compte les multiples stages). Les permanents qui ont toujours été là pour répondre à mes questions. Les doctorants avec qui j'ai passé de très bons moments.

Je remercie en particulier les membres dont j'ai côtoyé le bureau tout au long de cette thèse. Yann qui m'a initié au végétarisme. Clément qui a toujours su m'aider quand j'avais une question. Grégory avec qui j'ai adoré discuter de politique et dont je dédie le titre de mon dernier travail. Anne-Isabelle dont la bonne humeur a toujours été rafraichissante. Enfin, Johanne dont le sourire, l'écoute et les histoires m'ont permis de tenir ma dernière année de thèse.

Je remercie mes encadrant.e.s de l'équipe. René dont les connaissances bibliographiques m'ont bien aidé durant ma première année. Mon directeur Alexandre, pour son écoute et ses encouragements tout au long de ma thèse.

Je remercie particulièrement Véronique et Laurence, le meilleure duo d'encadrantes que j'aurais pu souhaiter avoir pour cette aventure. Elles ont toujours été là pour moi. Elles ont su me rassurer qu'en j'étais au plus bas et me pousser quand j'en avais

besoin. Je me souviendrai toujours de leur gentillesse et de leur rire.

Je remercie l'ensemble de mes amis qui m'ont supporté et soutenu tout au long de cette thèse.

Enfin, je souhaite remercier ma famille, ma mère Marie-Odile, mon petit frère Corentin et mon père Michel, qui ont été à mes côtés tout au long de cette thèse.

TABLE DES MATIÈRES

1 Introduction **9**
1.1 Contributions 10
1.2 Axes de la thèse 12

I État de l'art **15**

2 Séries temporelles et fouille de données **16**
2.1 Les séries temporelles 16
2.1.1 Définitions d'une série temporelle 17
2.1.2 Représentation des séries temporelles 18
2.1.3 Mesure de distance entre des séries temporelles 20
2.1.4 Tâches d'apprentissage automatique dans les séries temporelles 22
2.2 La découverte de régularités 24
2.2.1 Découverte de motifs 25
2.2.2 Découverte de règles temporelles 27
2.3 La classification de séries temporelles 30
2.3.1 Classification par similarité 32
2.3.2 Classification par changement de représentation 34
2.3.3 Classification par ensemble de classifieurs 36
2.3.4 Classification par réseaux de neurones profonds 37

3 Interprétabilité **39**
3.1 Interprétabilité dans la classification 40
3.1.1 Modèle interprétable 41
3.1.2 Méthode d'explication par proxy 42
3.2 Interprétabilité et classification de séries temporelles 44
3.2.1 Classifieurs interprétables de séries temporelles 45
3.2.2 Classifieurs non-interprétables de séries temporelles 47

3.2.3 Méthode d'explication par proxy de classifieur de séries temporelles 49
3.3 Évaluation de l'interprétabilité 50

II Recherche d'occurrences de règles temporelles dans des séries temporelles 51

4 Recherche d'occurrences élastiques de règles temporelles 52
4.1 Motivations 52
4.2 Recherche d'occurrences élastiques de règles temporelles 55
4.2.1 Mesures de distance élastiques 55
4.2.2 Algorithme de recherche d'occurrences de règles temporelles . 59
4.3 Expérimentations 65
4.3.1 Protocole expérimental 65
4.3.2 Résultats 66
4.4 Conclusion 69

III Classification de séries temporelles et interprétabilité 71

5 Classification de séries temporelles par shapelets aléatoires et localisées 72
5.1 Motivations 72
5.2 Modèle LRS 73
5.2.1 Modèle LRS 74
5.2.2 Apprentissage du classifieur 75
5.3 Du modèle LRS à l'explication 79
5.4 Expérimentations 84
5.4.1 Protocole expérimental 84
5.4.2 Performance contre les méthodes de l'état de l'art 86
5.4.3 Analyse de l'impact de la position des shapelets dans la classification 87
5.5 Conclusion 91

6 Explication locale et agnostique pour la classification de séries temporelles 93

6.1 Motivations 93
6.2 Proxy interprétable et agnostique pour explication locale 95
6.2.1 Fonctionnement général 95
6.2.2 Parties communes de LIME et SHAP 100
6.2.3 Parties spécifiques à LIME 102
6.2.4 Parties spécifiques à SHAP 103
6.3 Méthode LEFTIST 105
6.3.1 Extraction de composants interprétables dans une série temporelle 106
6.3.2 Transformation des voisins en séries temporelles 107
6.4 Évaluation quantitative 110
6.4.1 Protocole expérimental global 110
6.4.2 Fidélité avec des classifieurs interprétables 112
6.4.3 Fidélité avec des classifieurs non-interprétables 116
6.5 Évaluation utilisateur 125
6.5.1 Construction du questionnaire 125
6.5.2 Résultats 129
6.6 Conclusion 130

7 Conclusion **133**
7.1 Perspectives 135
7.1.1 Seuil de distance guidé par l'utilisateur 135
7.1.2 Composants interprétables et séries temporelles 136
7.1.3 Évaluation de l'interprétabilité des explications 137

Bibliography **139**

CHAPITRE 1

INTRODUCTION

Ces dernières années ont vu l'émergence d'une nouvelle révolution dans l'industrie avec l'arrivée des objets connectés dans les systèmes de production [HPO16]. En effet, les usines s'équipent progressivement de capteurs pour enregistrer en temps réel des informations sur le fonctionnement de leurs machines et de leurs infrastructures. Les informations récoltées couvrent un large panel de variables (ex : température, consommation d'électricité, état de fonctionnement, etc) et peuvent être numériques ou catégoriques (ex : "en marche"/"en panne"). Si les variables stockées sont numériques (comme c'est souvent le cas), alors elles sont représentées sous la forme de *séries temporelles*, c'est-à-dire une liste de valeurs horodatées. Toutes les données considérées dans cette thèse sont de ce type.

Le simple accès en temps réel à la valeur d'une variable et à son historique peut être suffisant pour certains usages. Cependant, les données contiennent de très nombreuses informations qui peuvent être exploitées à différentes fins. Par exemple, si un capteur enregistre une anomalie dans un système (ex : un pic de température élevée dans une machine), alors l'historique des données peut avoir enregistré des signes précurseurs de cette anomalie. Cette information peut être utilisée pour construire un outil informatique capable, à partir des valeurs de ce capteur, de prédire l'anomalie. En informatique, le domaine de recherche qui s'intéresse à l'extraction de telles connaissances s'appelle l'apprentissage automatique et consiste en l'apprentissage de modèles informatiques à partir d'un ensemble de données pour réaliser une tâche définie. De très nombreuses applications dans d'autres domaines que l'industrie existent pour extraire des connaissances à partir des séries temporelles issues de capteurs. Il peut s'agir de la détection d'une anomalie dans un rythme cardiaque, la prédiction de la consommation d'un moteur, etc.

Energiency est une entreprise qui propose à des clients industriels une plate-forme accessible par internet qui récupère, met en forme, présente et analyse leur données industrielles issues de capteurs placés dans leurs infrastructures et leurs machines.

Ces données sont majoritairement des données d'énergie (eau, gaz, électricité) mais il peut aussi s'agir de données de production (ex : la vapeur produite par une chaudière). L'objectif d'Energiency est de proposer des solutions à leurs clients pour optimiser leur consommation d'énergie ou leur productivité à partir de connaissances extraites à partir de ces données. Par exemple, des machines défectueuses sont identifiées ou bien un nouveau planning est proposé pour améliorer la production. La plate-forme développée par Energiency s'adapte aux besoins des clients. Ainsi Energiency a déjà intégré dans sa plate-forme des modèles d'apprentissage automatique pour prédire la consommation d'énergie ou détecter des anomalies.

En général, l'application de modèles d'apprentissage automatique sur des systèmes réels, utilisés par des utilisateurs non-experts en apprentissage automatique, fait face à deux problèmes importants :

— l'adaptation du modèle au cas d'application. En effet, un modèle utilisé pour prédire la consommation électrique d'une machine ne donnera pas les mêmes performances de prédiction s'il est utilisé pour prédire la consommation d'une autre machine ou la température dans une usine. Bien que la tâche soit la même et consiste à prédire une valeur numérique, le modèle doit prendre en compte les caractéristiques de la série temporelle considérée.

— l'interprétabilité du modèle, c'est-à-dire la capacité d'un modèle à expliquer ses résultats. En effet, les utilisateurs finaux peuvent prendre des décisions importantes à partir des résultats du modèle. Sans explication, l'utilisateur est obligé d'accorder une confiance aveugle dans le modèle. Ce problème peut freiner son utilisation.

Dans cette thèse, nous explorons séparément ces deux problèmes sur deux tâches différentes d'apprentissage automatique sur des séries temporelles : la première sur la recherche d'occurrences de règles temporelles, et la seconde sur la classification.

1.1 Contributions

La premier problème qui nous intéresse est l'adaptation de la recherche d'occurrences de règles temporelles dans des séries temporelles de consommation d'énergie d'infrastructures industrielles. Une règle temporelle est une connaissance qui permet de capturer des relations de succession entre des comportements répétés dans les séries temporelles (ex : plusieurs observations de la même suite de valeurs dans une

série). Il s'agit en général de comportements réguliers de la source des séries temporelles (ex : l'observation des cycles de fonctionnement d'une machine dans sa courbe de consommation électrique). Le fait de découvrir des relations de succession entre ces comportements réguliers peut mettre en évidence des relations inconnues ou être utilisé pour prédire un comportement. Une des étapes essentielles à la découverte de règles temporelles est la recherche d'occurrences de règles temporelles.

Un système industriel (et réel en général) et ses capteurs sont affectés par de nombreux facteurs environnementaux (température, humidité, etc), matériels (défaut de conception, usure, etc) ou bien humains (mauvaise manipulation, etc) qui peuvent introduire des variations dans les valeurs capturées dans les séries temporelles. Ainsi, deux occurrences du même comportement produisent deux suites de valeurs légèrement différentes. Par conséquent, la recherche des occurrences d'un comportement régulier dans les séries temporelles nécessite de la flexibilité pour capturer cette variabilité. La première contribution de cette thèse est la proposition d'une méthode de recherche d'occurrences de règles temporelles capable de résister à cette variabilité dans des séries temporelles industrielles.

La second problème concerne l'interprétabilité des modèles de classification de séries temporelles. Cette tâche implique que les séries temporelles tirées d'une source soient catégorisées, par exemple, les séries de consommation électrique d'une machine peuvent être catégorisée selon les états de fonctionnement de cette machine. La classification est la tâche consistant à associer à une nouvelle série temporelle sa catégorie. Notons que la problématique de l'interprétabilité de la classification de séries temporelles s'étend au delà de l'application à des séries temporelles industrielles et concernent toutes les sources de séries temporelles (ex : électrocardiogramme, spectrogramme, capteur de position, etc).

Deux approches existent pour expliquer les résultats d'un classifieur de séries temporelles : les explications peuvent être extraites directement du classifieur (celui-ci est considéré comme interprétable), ou une méthode a posteriori est utilisée pour expliquer les résultats du classifieur. Cependant, chaque approche fait face à une difficulté. Les classifieurs interprétables de séries temporelles existants ne font pas partie des meilleurs classifieurs. Les méthodes d'explication a posteriori de classifieurs de séries temporelles sont peu nombreuses et spécifiques à un type particulier de classifieur (ex : les réseaux de neurones). Cela limite leur utilisation car les types de classifieurs les plus performants ne sont pas les mêmes selon les types de jeux de séries tempo-

relles.

La seconde contribution de cette thèse est un classifieur interprétable LRS (*Localized Random Shapelet*) dont l'objectif est d'être plus efficace que les classifieurs interprétables de l'état de l'art. Notre méthode repose sur la présence ou l'absence de sous-séries discriminantes dans les séries temporelles pour les classifier. Cette approche est utilisée dans de nombreux classifieurs, mais jusqu'à présent celle-ci s'était contentée d'utiliser seulement la présence comme attribut de classification. Dans LRS, nous proposons d'utiliser la position des sous-séries discriminantes comme nouvel attribut de classification.

Enfin, le dernier travail de cette thèse comprend deux contributions. La troisième contribution est un processus pour expliquer a posteriori le résultat de n'importe quel classifieur pour une donnée spécifique. Ce processus, appelé PIAEL (*Proxy Interprétable et Agnostique pour Explication Locale*), repose sur l'apprentissage d'un classifieur interprétable, nommé proxy, dont l'objectif est de retourner les mêmes résultats que le classifieur à expliquer. Les explications des résultats sont ensuite extraites à partir du proxy. Ce processus a été extrait à partir de deux méthodes d'explications a posteriori existantes LIME [RSG16] et SHAP [LL17]. La quatrième contribution de cette thèse définit, grâce au processus PIAEL, la première méthode capable d'expliquer, a posteriori, les résultats de n'importe quel classifieur de séries temporelles. Nous avons appelé cette méthode, LEFTIST (*agnostic Local Explanation For TIme Series classificaTion*).

Ces travaux ont donné lieu à trois publications [Gui+17 ; Gui+19b ; Gui+19c].

1.2 Axes de la thèse

La suite de ce manuscrit est organisée de la façon suivante. La première partie présente l'état de l'art en deux chapitres. Le **chapitre 2** est dédié à la présentation des séries temporelles, à la tâche de découverte de régularités (dont les règles temporelles) et à celle de classification de séries temporelles. Le **chapitre 3** est consacré à la présentation de l'interprétabilité, en particulier dans le cadre de la classification de données.

Puis, la seconde partie composée du **chapitre 4** présente notre méthode de recherche d'occurrences de règles temporelles dans des séries industrielles.

À la suite, la troisième partie s'intéresse à l'interprétabilité dans les modèles de

classification de séries temporelles. Le **chapitre 5** présente notre classifieur de séries temporelles interprétable LRS, et le **chapitre 6** présente le processus PIAEL et notre méthode d'explication de résultats LEFTIST applicable sur n'importe quel classifieur de séries temporelles.

Enfin, le **chapitre 7** conclut la thèse et discute des perspectives de ce travail.

PREMIÈRE PARTIE

État de l'art

CHAPITRE 2

SÉRIES TEMPORELLES ET FOUILLE DE DONNÉES

Les travaux présentés dans cette thèse traitent de l'apprentissage automatique dans des séries temporelles. Ce chapitre est consacré à la présentation des définitions concernant les séries temporelles et à l'état de l'art des méthodes d'apprentissage automatique explorées dans cette thèse. La section 2.1 introduit les définitions et leurs utilisations dans la fouille de données. Puis la section 2.2 présente l'état de l'art des méthodes de découverte de régularités. Enfin, la section 2.3 est consacrée à l'état de l'art des méthodes de classification de séries temporelles.

2.1 Les séries temporelles

Une série temporelle se définit comme une séquence de valeurs continues ordonnées, qui représente l'évolution d'une variable numérique dans le temps. Elle est la mesure d'un système évoluant dans le temps avec des attributs numériques : par exemple, les fonctions vitales d'un être humain ou animal (ECG, EEG, etc), les déplacements d'un véhicule ou bien la valeur de l'action d'une entreprise. Une série temporelle est porteuse de nombreuses informations sur le système mesuré. Ces informations peuvent être utilisées pour extraire des connaissances utiles à un utilisateur. La première partie de ce chapitre présente les définitions de base et les outils essentiels à la manipulation des séries temporelles.

En informatique, l'apprentissage automatique est la tâche qui consiste à extraire et à exploiter des connaissances à partir d'un ensemble de données (des séries temporelles dans notre cas). Cette tâche peut être divisée en différentes catégories selon la nature et l'utilisation des connaissances extraites. La seconde partie de cette section est consacrée à la présentation de ces catégories.

2.1.1 Définitions d'une série temporelle

Il existe de nombreuses références dans la littérature qui définissent les notions de bases des séries temporelles. Dans notre cas, nous utilisons les définitions données dans [EA12] et dans lesquelles une *série temporelle* est définie de la manière suivante.

Definition 2.1.1. Série temporelle
Une *série temporelle* T est une séquence ordonnée de n valeurs réelles. $T = (t_1, \ldots, t_n)$, avec $t_i \in \mathbb{R}$ pour $1 \leq i \leq n$.

Les valeurs des séries temporelles sont ordonnées selon une unité. Dans la grande majorité des cas, l'unité est le temps (les valeurs sont associées à des dates qui sont espacées uniformément dans le temps), mais il en existe d'autres (ex : la longueur d'onde dans les spectrogrammes). Une série temporelle peut être *univariée* (Définition 6.3.1) ou *multivariée* quand nous disposons de plusieurs séries temporelles qui se déroulent sur le même intervalle de temps, chacune associée à une variable. Dans cette thèse, nous nous intéressons uniquement aux séries temporelles univariées.

Dans certaines tâches d'apprentissage automatique, il est nécessaire de considérer seulement une partie de la série temporelle, c'est-à-dire une *sous-série*.

Definition 2.1.2. Sous-série temporelle
Soit une série temporelle $T = (t_1, \ldots, t_n)$ de taille n. Une *sous-série* temporelle S de T est une série temporelle de taille $m \leq n$ composée de m valeurs consécutives de T

$$S = (t_i, t_{i+1}, \ldots, t_{i+m-1})$$

avec $1 \leq i \leq n - m + 1$. Nous notons S_T^m, l'ensemble de toutes les sous-séries de taille m issues de T, ces sous-séries peuvent se superposer.

Les méthodes d'apprentissage automatique dans des séries temporelles sont généralement confrontées à deux obstacles : la haute dimension des séries temporelles (une série peut être composée de millions de valeurs) et la mesure de similarité entre les séries temporelles. La solution pour le premier consiste à simplifier la représentation des séries temporelles sans perdre trop d'informations. Dans le second cas, il s'agit de trouver une mesure de distance capable d'approximer la similarité entre des séries temporelles souhaitée par l'utilisateur.

2.1.2 Représentation des séries temporelles

Nous appelons séries temporelles brutes, les séries issues directement des capteurs sans traitement. La haute dimension de séries temporelles brutes a des conséquences négatives pour les méthodes d'apprentissage automatique : un temps de calcul prohibitif, une baisse des performances et une impossibilité d'expliquer les résultats. La solution pour résoudre ce problème consiste à réduire la dimensionalité des séries temporelles, en changeant la *représentation* des séries temporelles.

Nous distinguons trois types de changement de représentation d'une série temporelle (illustrés dans la Figure 2.1) : changement en une autre série temporelle, en une séquence symbolique, et en un vecteur numérique.

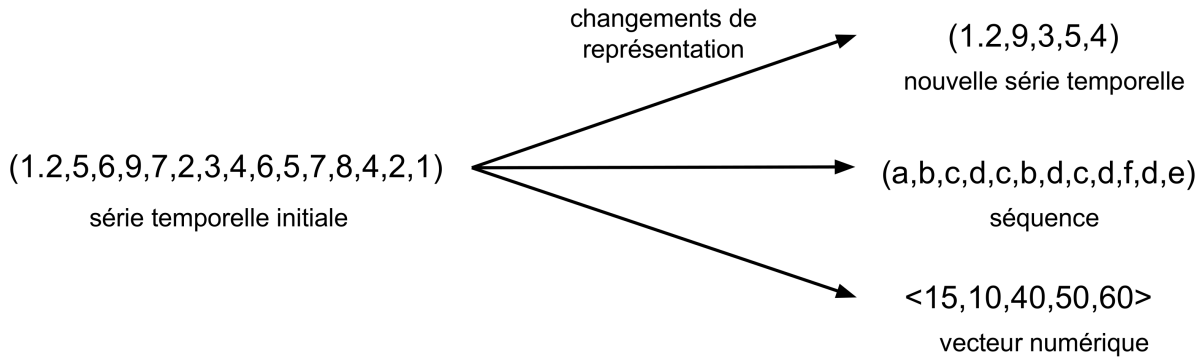


FIGURE 2.1 – Les différents types de représentation de séries temporelles.

Représentation par une série temporelle

La représentation par une nouvelle série temporelle consiste à sélectionner ou calculer un sous-ensemble de points tout en conservant l'ordre des valeurs dans la nouvelle série.

La méthode la plus simple est sans doute l'échantillonage [Åst69] qui consiste à sélectionner les valeurs tous les n points dans la série temporelle. Il peut s'agir aussi de sélectionner seulement les points importants dans la série temporelle tels que les valeurs extrêmes [FG11] ou bien des valeurs pivots [Per+00].

Une autre approche consiste à découper les séries temporelles en intervalles consécutifs et à encoder chaque intervalle par la moyenne des valeurs de l'intervalle. Cette méthode a été introduite en même temps dans deux papiers sous le nom de *Piecewise Aggregate Approximation* (PAA) [KP00] et *Segmented Means* [YF00].

Représentation par une séquence symbolique

Dans cette famille de changement de représentation, il s'agit de projeter le domaine des valeurs de la série temporelle dans un domaine discret, c'est la raison pour laquelle elles sont nommées *discrétisation*. Une série temporelle discrétisée est une *séquence*.

Definition 2.1.3. Séquence
Soit $A = \{a_1, .., a_k\}$ un alphabet de k symboles. Une *séquence* Seq est une suite ordonnée de m valeurs symboliques $Seq = (seq_1, ..., seq_m)$, avec $seq_i \in A$ pour $1 \leq i \leq m$.

Une discrétisation peut être réalisée point par point, comme par exemple dans [Bag+06] où chaque point est encodé par zéro ou par un selon si sa valeur est supérieure ou inférieure à la moyenne de la série.

Dans [Das+98], l'ensemble S_T^m de toutes les sous-séries possibles (cf. Définition 2.1.2) de taille m est extrait de la série temporelle T. Les sous-séries sont ensuite regroupées dans K groupes de façon à ce que la similarité calculée entre les éléments d'un groupe soit faible et que celle calculée entre les éléments de groupes différents soit élevée. Chaque groupe est associé à un symbole qui est utilisé pour encoder l'ensemble des sous-séries présentes dans le groupe.

Une autre approche appelée SAX (introduite la première fois dans [Lin+02] puis formalisée dans [Lin+07]), découpe la série temporelle en intervalles successifs de même taille et non-superposés. La moyenne des valeurs de chaque intervalle est calculée (méthode PAA). La distribution des moyennes est divisée en K quantiles, chacun associé à un symbole. Les intervalles sont encodés par le symbole du quantile auquel appartient leur moyenne. Cette méthode suppose que la distribution des moyennes est gaussienne.

Enfin, dans [HS05 ; PC01], le domaine des valeurs à représenter est divisé en K intervalles successifs (ex : groupe 1 les valeurs de 0 à 3, groupe 2 les valeurs de 4 à 6, etc). Chaque intervalle est associé à un symbole qui encode les valeurs présentes dans l'intervalle. Cette représentation n'assure pas que les intervalles soient équiprobablement répartis dans les groupes (à la différence de SAX).

Représentation par un vecteur numérique

Dans cette représentation, un ensemble de valeurs numériques est calculé à partir de la série temporelle pour la représenter. Ces valeurs ne sont pas obligatoirement

corrélées dans le temps. Différentes approches ont été proposées, certaines se basent sur la série temporelle initiale, d'autres sur une transformation préalable de la série temporelle.

Une approche issue du traitement du signal, consiste à passer les séries temporelles dans le domaine fréquentiel et à représenter les séries temporelles selon les coefficients issus de la transformation. Nous pouvons citer dans ce domaine la transformation de Fourier discrète [FRM94], la transformation en ondelettes [CF99 ; PM02 ; CFY03] ou la transformation en cosinus discrète [KJF97].

Une description plus complète de ces méthodes de changement de représentation, ainsi que d'autres propositions figurent dans [EA12 ; Fu11]. Nous présentons d'autres approches de représentation dans la section 2.3 et la section 2.2.

Dans cette thèse, les méthodes que nous proposons dans le chapitre 5 et le chapitre 6 utilisent des représentations en vecteurs numériques pour extraire à partir des séries temporelles des attributs compréhensibles pour l'utilisateur. Ces attributs sont ensuite utilisés pour fournir des explications sur les résultats de classifieurs de séries temporelles. Ces méthodes de changement de représentation sont présentées plus en détail dans la section 3.2 consacrée à l'explication des résultats de classifieurs de séries temporelles.

2.1.3 Mesure de distance entre des séries temporelles

La *similarité* entre des séries temporelles est une des propriétés les plus utilisées en apprentissage automatique sur des séries temporelles. Pour évaluer cette similarité, les méthodes utilisent une mesure de distance.

Definition 2.1.4. Mesure de distance
Soit E_T un ensemble non-ordonné de séries temporelles. Une *mesure de distance* $\mathcal{D}$ est une fonction $\mathcal{D} : E_T \times E_T \rightarrow \mathbb{R}^+$ qui calcule la *distance* D entre deux séries temporelles. Une distance vérifie trois propriétés :

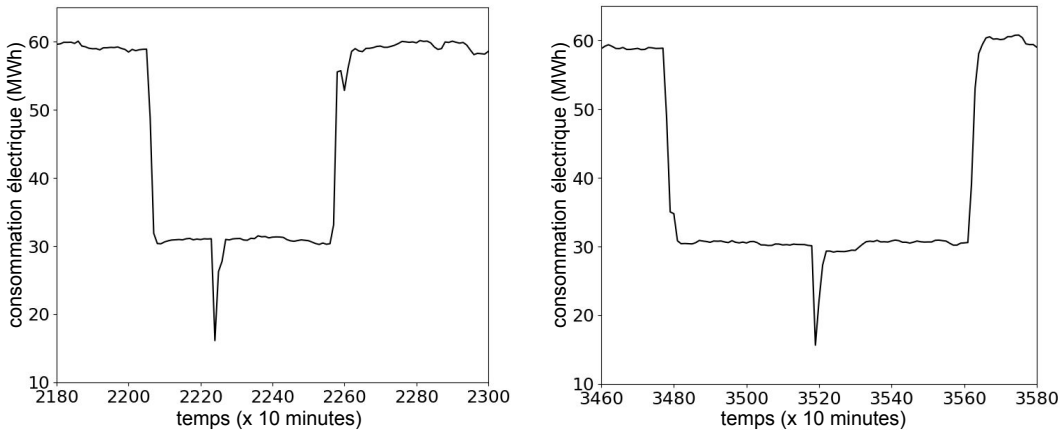
- l'identité : $\forall U, T$ des séries temporelles, $\mathcal{D}(U, T) = 0 \Leftrightarrow U = T$
- la symétrie : $\forall U, T$ des séries temporelles, $\mathcal{D}(U, T) = \mathcal{D}(T, U)$
- l'inégalité triangulaire : $\forall U, T, Q$ des séries temporelles, $\mathcal{D}(U, Q) \leq \mathcal{D}(U, T) + \mathcal{D}(T, Q)$

La mesure de distance la plus connue est la distance euclidienne, dite de norme 2.

Definition 2.1.5. Distance euclidienne
Soit T et U deux séries temporelles de même taille n, la distance euclidienne se calcule de la manière suivante.

$$\mathcal{D}_{euclidienne}(T, U) = \sqrt{\sum_{i=1}^{n}(T_i - U_i)^2}$$

Avec T_i et U_i les valeurs respectives de T et de U à la date i.



(a) première série temporelle (b) seconde série temporelle

FIGURE 2.2 – Deux séries temporelles qui se ressemblent fortement mais avec des légères déformations.

Certaines mesures de distance peuvent être trop rigides pour évaluer la similarité entre deux séries temporelles. Prenons pour exemple les deux séries temporelles illustrées dans la Figure 2.2. Nous pouvons considérer que ces séries temporelles sont similaires car la variable évolue de la même manière mais avec de légers décalages temporels. De nombreux facteurs peuvent être la cause de ces décalages : une action de l'utilisateur sur la source des séries temporelles, l'environnement qui perturbe le capteur dont est extrait la série temporelle, etc. Par conséquent, il est rare que deux séries représentant le même phénomène soient parfaitement identiques. Ce sont ces perturbations qui pénalisent en particulier les mesures qui somment l'écart des valeurs entre des points à la même position dans les séries temporelles. En effet, il suffit de décaler, entre deux séries similaires, toutes les valeurs d'une des séries de deux ou trois

positions pour que la distance calculée soit élevée alors que les séries se ressemblent mais avec un décalage.

Pour règler ce problème, des mesures de distance *élastiques* ont été créées, dont la plus réputée est la déformation temporelle dynamique ($Dynamic\ Time\ Warping$ ou DTW). Introduite dans le domaine de la fouille de donnée par Berndt et Clifford [BC94], cette mesure est issue de travaux plus anciens en traitement du langage [Ita75 ; SC78]. DTW calcule l'alignement optimal, entre deux séries temporelles, qui minimise la distance euclidienne calculée entre les points alignés. En réalité, DTW n'est pas réellement une distance car elle ne respecte pas les propriétés de symétrie et d'inégalité triangulaire.

Des variations de DTW ont été proposées telle WDTW [JJO11] qui pondère les alignements de DTW pour éviter des alignements anormaux (ex : aligner tous les points d'une série sur un unique point de l'autre série), ou DDTW [GL13] qui applique DTW sur les séries temporelles différenciées (c'est-à-dire des séries temporelles où chaque point est remplacé par sa dérivée locale). Enfin des approches alternatives de mesures de distance élastiques ont été proposées, en particulier celles basées sur la distance d'édition largement utilisée dans la comparaison de séquences génétiques. Cette mesure introduit des décalages (des points sans valeurs) dans les séries temporelles pour décaler les points et aligner les séries temporelles de façon plus favorable. Comme mesures de distance à base de distance d'édition nous pouvons citer LCSS [Hir77], ERP [CN04], TWED [Mar07] ou MSM [SAD13].

Une description plus complète de ces mesures de distance ainsi que la présentation d'autres mesures de distance sont présentées dans [EA12 ; LB15].

2.1.4 Tâches d'apprentissage automatique dans les séries temporelles

La partie précédente a introduit les définitions et concepts essentiels à la tâche d'apprentissage automatique dans des séries temporelles. Selon la nature et l'objectif de la tâche effectuée, nous la rangeons dans une des sept catégories suivantes :

— **Description de séries temporelles**
Cette tâche consiste à trouver une représentation des séries temporelles qui simplifie les séries temporelles tout en minimisant l'erreur de reconstruction de la représentation vers la série initiale.

— **Recherche de séries temporelles similaires à une série donnée**
Cette tâche consiste à identifier les k séries temporelles les plus similaires à une série temporelle fournie par l'utilisateur. La similarité est calculée via une mesure de distance (cf. section 2.1.3) appliquée sur les données brutes ou bien sur la représentation des données (cf. section 2.1.2). Les k séries temporelles dont la distance est la plus faible avec la série temporelle fournie sont retournées.

— **Regroupement de séries temporelles**
Le regroupement de séries temporelles (ou *clustering* en anglais) consiste à identifier K sous-ensembles dans un ensemble de séries temporelles. En général, un sous-ensemble est un groupe de séries temporelles dont les éléments sont similaires les uns avec les autres et dissimilaires avec les éléments des autres groupes.

— **La prédiction**
La prédiction consiste à prévoir les k futures valeurs d'une série temporelle à partir d'un instant t. Pour cela, les méthodes de régression s'appuient sur les valeurs passées de la série temporelle, ainsi que sur ses comportements réguliers et périodiques tels que la saisonnalité ou la tendance.

— **La détection d'anomalie**
La détection d'anomalie consiste à identifier dans une série temporelle des sous-séries ou des points dont le comportement est différent du reste de la série temporelle. Il peut s'agir, par exemple, de valeurs aberrantes ou d'un décalage des valeurs sur un des axes.

— **Découverte de régularités**
La découverte de régularités consiste à découvrir des sous-séries qui sont redondantes dans les séries temporelles ou des relations redondantes entre ces sous-séries.

— **La classification**
Un ensemble de séries temporelles peut être divisé en catégories, ou classes, connues. La classification d'une série temporelle consiste à associer une catégorie à une nouvelle série temporelle.

Les travaux dans cette thèse s'inscrivent dans les tâches de découverte de régularités et de classification dans des séries temporelles. La suite de ce chapitre, présente plus en détail ces deux tâches ainsi que leur état de l'art. La section 2.2 est dédiée à la tâche de découverte de régularités dont fait partie la découverte de règles tempo-

relles, et dans laquelle s'inscrit notre méthode de recherche d'occurrences élastiques de règles temporelles présentée de la Partie II. La section 2.3 est consacrée à la classification de séries temporelles qui est le sujet de la Partie III.

2.2 La découverte de régularités

Cette section est consacrée à la présentation de l'état de l'art de la tâche de découverte de régularités dans des séries temporelles. Les régularités dans les séries temporelles sont en général représentatives de comportements intéressants pour l'utilisateur.

Différentes régularités peuvent être découvertes dans une série temporelle. La première prend la forme d'une sous-série qui apparaît plusieurs fois dans la série temporelle (de façon périodique ou non). Nous la désignons par le terme *motif*. La seconde régularité, se construit à partir de motifs et représente la relation temporelle régulière entre les occurrences de deux motifs. Nous la désignons par le terme *règle temporelle*.

Pour illustrer ces deux types de connaissances, imaginons que nous ayons à notre disposition la courbe de consommation électrique d'un foyer. Dans ce foyer, un four est régulièrement utilisé. En général, l'utilisation d'un appareil produit une sous-série particulière dans la courbe de consommation électrique. Et l'utilisation régulière fait que cette sous-série particulière apparait plusieurs fois et à différents moments dans la courbe de consommation globale. Ce comportement peut être capturé par un motif.

Dans le cas des règles temporelles, imaginons cette fois que le foyer de notre exemple dispose d'une machine à laver et d'un séchoir électrique. Tout comme le four, ces deux appareils sont utilisés régulièrement et leurs fonctionnements peuvent être capturés par des motifs. Cependant, une connaissance supplémentaire peut être découverte. En effet, le séchoir est en principe toujours utilisé après la machine à laver dans un laps de temps relativement court. Cette relation temporelle de succession des deux motifs est capturée par une règle temporelle.

La découverte de règles temporelles est donc indissociable de la découverte de motifs. La section 2.2.1 présente les méthodes pour découvrir des motifs dans des séries temporelles. La section 2.2.2 quant à elle a pour but de présenter les méthodes de découverte de règles temporelles dans des séries temporelles.

2.2.1 Découverte de motifs

La définition d'un motif nécessite de définir deux éléments : l'*occurrence* d'une sous-série et son *support*.

Definition 2.2.1. Occurrence d'une sous-série
Soit une série temporelle T, une sous-série S issue de T et une mesure de distance $\mathcal{D}$. Une *occurrence* O de S dans T est un triplet $O = (S_{occ}, d, f)$. S_{occ} est une sous-série de T telle que la distance $\mathcal{D}(S, S_{occ}) \leq th_{occ}$ avec un seuil fixé $th_{occ} \in \mathbb{R}$. $d, f \in \mathbb{N}$ désignent respectivement la date de début et la date de fin de l'occurrence S_{occ} dans T.

Definition 2.2.2. Support
Soit E_T^S l'ensemble des occurrences de S dans T. Le *support* de S est $supp_S = |E_T^S|$.

A présent, nous pouvons définir ce qu'est un *motif*.

Definition 2.2.3. Motif
Un *motif* de T est une sous-série S de T telle que le $supp_S \geq th_{supp}$ où $supp_S$ est le support des occurrences de S dans T, et $th_{supp} \in \mathbb{N}$ est un seuil donné.

En général, la taille des motifs à découvrir est fixée. Soit n la taille des motifs recherchés, la méthode naïve pour découvrir des motifs consiste à extraire toutes les sous-séries possibles de taille n dans la série temporelle, puis à calculer la matrice des distances entre toutes les paires possibles de sous-séries. Les colonnes et les lignes de la matrice sont les sous-séries, et les éléments de la matrice, les distances calculées entre deux sous-séries. Pour une sous-série (une ligne de la matrice), le nombre de sous-séries dont la distance est inférieure au seuil donné th_{occ} est déterminé (support de la sous-série). Les sous-séries dont le support est supérieur au seuil donné th_{supp} sont conservées. Cependant, cette méthode n'est pas réaliste car l'espace de recherche devient conséquent quand le nombre de sous-séries est grand.

Une première approche [Mue+09] restreint le problème en cherchant seulement les K paires dont les sous-séries sont les plus similaires. La méthode commence par sélectionner un sous-ensemble de sous-séries E_{str}, dites référentes. Pour toute sous-série str appartenant à E_{str}, les distances entre str et toutes les autres sous-séries sont calculées (calcul d'une ligne de la matrice). Si deux sous-séries st_1 et st_2 sont similaires alors pour toute série de référence str, la distance entre st_1 et str est proche de celle entre st_2 et str. L'heuristique utilisée dans [Mue+09] consiste à trouver les K paires qui vérifient au mieux cette propriété.

D'autres méthodes [Lin+02 ; CKL03 ; RT04 ; TIU05] découpent la série temporelle en sous-séries de taille fixe, représentées par des séquences à l'aide de la méthode SAX [Lin+02] (cf. section 2.1.2). Nous nommons cet ensemble de séquences E_{seq}. La recherche est limitée aux K motifs avec les plus grands supports. Les méthodes se différencient ensuite sur l'heuristique pour extraire les motifs à partir de E_{seq}.

— Dans [Lin+02], une heuristique est utilisée pour calculer uniquement des parties de la matrice des distances entre séquences de E_{seq} ;

— Dans [TIU05], la recherche des motifs les plus fréquents repose sur le principe MDL [BRY98] et consiste à trouver les séquences dans E_{seq} dont l'encodage en bits n'est pas trop conséquent et permet de compresser au mieux la série temporelle ;

— Dans [RT04], un symbole supplémentaire X est ajouté à l'alphabet utilisé pour la transformation par SAX. Ce symbole a pour particularité de pouvoir remplacer n'importe quel autre symbole (une sorte de joker) dans une séquence de E_{seq}. Un arbre est construit pour compter les occurrences des séquences de E_{seq}. L'algorithme étend l'espace des séquences en s'autorisant le remplacement de tout symbole par le symbole joker X (cf. Figure 2.3). Les K séquences les plus fréquentes sont retournées comme résultat.

— Dans [CKL03], les séquences de E_{seq} sont d'abord comparées entre elles sur certaines positions de symboles. La distance entre séquences est calculée uniquement pour les paires de séquences qui ont statistiquement le plus de symboles en commun.

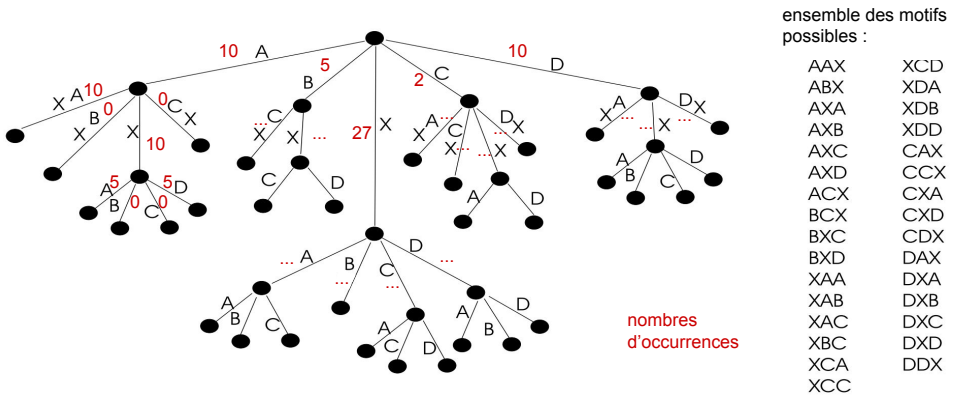


FIGURE 2.3 – Arbre utilisé pour compter toutes les sous-séries possibles de taille 3 à partir de l'alphabet $\mathcal{A} = \{A, B, C, D\} \cup \{X\}$ où X est le symbole joker, et A, B, C et D les symboles utilisés pour la représentation par SAX. Image extraite de [CKL03]

Récemment, Zhu et al. [Zhu+17] déterminent pour toute sous-série st de la série temporelle, la sous-série st_p la plus similaire précédant st et la sous-série st_s la plus similaire suivant st. Deux paires de valeurs sont associées à st :

— la position et la distance de la sous-série, parmi toutes celles qui précédent st dont la distance avec st est minimum ;
— parmi toutes les sous-séries qui suivent une sous-série, la position et la distance de la sous-série dont la distance est minimum avec la sous-série ;

Cette connaissance offre la possibilité de découvrir des ensembles de sous-séries similaires les unes des autres. En effet, en partant d'une sous-série nous connaissons la sous-série suivante la plus similaire, dont elle même nous connaissons la sous-série la plus similaire qui la suit, etc. Si les différences de distances entre les sous-séries sont petites, alors nous pouvons estimer que la première sous-série est assez similaire de la dernière sous-série de la chaîne. Cet enchaînement est nommé *Time Series Chain* et les auteurs étendent la définition d'un motif de cette sorte. À la différence de notre définition, une sous-série candidate en tant que motif n'est pas comparée à un ensemble de sous-séries dans le but d'y trouver ses occurrences, à la place les occurrences sont retrouvées en remontant la chaîne dans laquelle la sous-série candidate est présente. Comme la similarité est seulement évalué paire à paire, nous n'avons pas l'assurance que les sous-séries en bouts de chaîne soient similaires à la sous-série candidate.

Cette section a présenté la découverte de motifs dans des séries temporelles. La section suivante présente la découverte de règles temporelles à partir de la notion de motifs.

2.2.2 Découverte de règles temporelles

Une règle temporelle exprime une relation de succession, bornée dans le temps, entre deux motifs. Elle permet de décrire une nouvelle connaissance dans une série temporelle.

Initialement, la découverte de règles a été appliquée sur des *bases de transactions* [AIS93]. Une base de transactions est un ensemble de *transactions*, où chacune est un ensemble d'objets associé à un identifiant. L'exemple le plus connu est un ensemble de tickets de caisse de supermarché, où chaque transaction correspond à un ticket de caisse et à la liste d'achats associée. Les règles dites d'association dé-

couvrent des achats corrélés entre des ensembles de produits, mais ne prennent pas en compte l'aspect temporel. Les premières méthodes n'ont considéré cet aspect que sur des séquences [AS95 ; MTV97]. La découverte de règles temporelles dans des séries temporelles est apparue ensuite [Das+98].

Nous définissons une règle temporelle dans une série temporelle de la manière suivante.

Definition 2.2.4. Règle temporelle

Soit une série temporelle T. Une *règle temporelle* R est un triplet $R = (A, C, p)$ où :

- A est un motif, avec $O_A^i = (S_A^i, d^i, f^i)$ une occurrence de A dans T ;
- C est un motif, avec $O_C^j = (S_C^j, d^j, f^j)$ une occurrence de C dans T ;
- p est une contrainte de temps.

R est une règle temporelle si :

$$\exists E_T^R = \{(O_A^i, O_C^j) \mid (d^j - f^i)\ v\acute{e}rifie\ p\} \text{ tel que } |E_T^R| \geq th_R \text{ où un } th_R \in \mathbb{N}$$

Le motif A est appelé l'antécédent, le motif C le conséquent et E_T^R l'ensemble des occurrences de la règle temporelle R. Une règle se note de la façon suivante :

$$A \xrightarrow{p} C$$

Par conséquent, deux paramètres sont nécessaires pour trouver un ensemble de règles temporelles : p, une contrainte de temps entre l'antécédent et le conséquent, et th_R, le seuil minimum d'occurrences des règles. La contrainte de temps peut être la durée minimum ou bien la durée maximum entre l'antécédent et le conséquent.

En général, l'utilisation de ces deux paramètres seuls retourne un très grand nombre de règles, sans fournir d'information sur leur pertinence. Cela a pour conséquence de laisser à l'utilisateur la tâche de trouver les règles qui sont intéressantes. Pour résoudre ce problème, les méthodes de découverte de règles temporelles ont introduit des *mesures d'intérêt* pour trier et restituer les règles les plus intéressantes à l'utilisateur. La mesure d'intérêt la plus connue est la *confiance* [AIS93].

Definition 2.2.5. Confiance

Soit une règle temporelle $R = (A, C, p)$ de T, l'ensemble E_T^R de ses occurrences dans T et E_T^A l'ensemble des occurrences du motif A dans T. La *confiance* de R se calcule de la manière suivante :

$$confiance_R = \frac{|E_T^R|}{|E_T^A|}$$

La confiance permet d'éliminer les règles dont la découverte est due à un motif antécédent surreprésenté dans la série temporelle (qui a donc plus de chance d'être associé avec les autres motifs). Elle est utilisée dans de nombreuses méthodes de découverte de règles temporelles [Höp01 ; JLS02 ; CS02].

Dans [Das+98], la mesure d'intérêt est la J-mesure [SG91] qui se base sur les probabilités d'apparition de l'antécédent et du conséquent. Un ensemble de mesures d'intérêt est présenté dans [GH06].

Différentes stratégies ont été proposées pour la découverte de règles temporelles dans une série temporelle. La toute première, définie dans [Das+98], extrait des règles temporelles à partir de la série temporelle discrétisée en une séquence. Dans cette discrétisation, l'ensemble des sous-séries de taille n sont extraites à partir de la série temporelle. Chaque sous-série est ensuite associée à un symbole via une méthode de regroupement (cf. section 2.1.2). La recherche de règles temporelles consiste à énumérer toutes les paires de symboles possibles dans la série discrétisée. Si le nombre de leurs occurrences vérifiant la contrainte de temps p est supérieur au seuil th_R, alors la paire est considérée comme une règle temporelle. Les paramètres sont fixés par l'utilisateur.

Une autre stratégie de découverte de règles, repose sur la construction d'un arbre à partir de la série temporelle discrétisée. En effet, un arbre est une connaissance qui peut représenter un ensemble de règles (un chemin dans un arbre peut être écrit sous la forme d'une règle). Dans [PC01], le domaine des valeurs de la série temporelle est divisé en un nombre fini d'intervalles, chaque intervalle étant associé à un symbole. La série temporelle est discrétisée en remplaçant chaque valeur par le symbole de l'intervalle auquel elle appartient. La séquence est ensuite compressée en remplaçant les sous-séquences de même symbole (ex : une suite de A) par une seule occurrence du symbole (ex : (A, A, A, B, B, C) devient (A, B, C)). À partir de cette séquence un arbre est appris pour stocker et compter toutes les sous-séquences de taille n (de la même façon que la méthode illustrée dans la Figure 2.3). Pour chaque nœud de l'arbre, les règles associées ont toutes pour antécédent la suite de symboles de la racine au nœud. Le conséquent de chaque règle correspond à la suite de symboles du nœud à l'une des feuilles. L'arbre est utilisé pour calculer la confiance de chaque règle.

Dans [LKK01], la série temporelle est segmentée en intervalles selon ses tendances successives. Pour chaque intervalle, un ensemble de variables est calculé (pente, fluctuation et durée). Un modèle est ensuite appris pour prédire la durée d'un

intervalle à partir de l'intervalle précédent. Pour finir, un ensemble de règles temporelles est tiré du modèle.

Une dernière approche de découverte de règles temporelles consiste à générer un ensemble de règles et à les évaluer a posteriori. Dans [HS05], la série temporelle est discrétisée puis un algorithme génétique est utilisé pour générer un ensemble de règles. Les occurrences des règles sont ensuite cherchées dans la série discrétisée pour les évaluer. Dans [Sho+15], un motif est extrait de la série temporelle par une préroutine (*MK algorithm* [Mue+09]) et est utilisé pour générer un ensemble de règles candidates. La génération d'une règle temporelle consiste à séparer le motif en deux parties, à une position donnée. Le sous-motif gauche devient l'antécédent et le sous-motif droit le conséquent. Un ensemble de règles temporelles est généré en découpant le motif à différentes positions uniformément réparties. L'évaluation d'une règle consiste à rechercher les occurrences de la règle dans la série temporelle et à calculer un score inspiré de MDL [BRY98] à partir des occurrences. La règle avec le score le plus élevé est retournée. Il s'agit d'une des rares méthodes qui découvre des règles temporelles sans changer la représentation des séries temporelles.

Le chapitre 4 présente une amélioration de la recherche d'occurrences de règles dans cette dernière méthode.

2.3 La classification de séries temporelles

La classification de séries temporelles consiste à associer à une série temporelle une *classe*.

Definition 2.3.1. Ensemble de classes
Un ensemble de classes correspond à une partition d'un ensemble de données selon une propriété cible. Chaque sous-ensemble de données est associé à une valeur particulière de cette propriété cible, appelée *classe*. Le nombre de classes est fini.

L'association d'une classe à une nouvelle donnée est confiée à un *classifieur* qui a été appris sur un ensemble de séries temporelles donné, appelé aussi ensemble d'apprentissage.

Definition 2.3.2. Classifieur
Soit X un ensemble de données, Y un ensemble de classes. Un classifieur est une

fonction $f : X \rightarrow [0,1]^{|Y|}$ qui associe à une donnée les probabilités d'appartenir à chaque classe. La somme des probabilités doit être égale à 1. Selon les modèles, le résultat retourné est le vecteur des probabilités d'appartenance à chaque classe, ou bien la classe majoritaire.

Un classifieur est généralement évalué sur sa *précision*.

Definition 2.3.3. Précision
Soit f un classifieur déjà appris et D un ensemble de données dont nous connaissons la classe. La précision de f sur T se calcule de la manière suivante :

$$précision(f, D) = \frac{TP}{|D|}$$

avec TP le nombre de données dans D dont la classe retournée par f est la même que celle associée la donnée.

Il existe deux approches pour apprendre un classifieur. La première approche, dite supervisée, apprend le classifieur sur un ensemble de données dont la classe de chaque élément est connue. La seconde, dite non-supervisée, raisonne sur un ensemble de données de classe inconnue. L'apprentissage a la tâche supplémentaire de définir lui-même l'ensemble des classes. Dans le cadre de cette thèse, nous nous intéressons uniquement aux méthodes supervisées de classification de séries temporelles.

La classification de séries temporelles est un domaine de recherche vaste et diversifié. Nous identifions quatre grandes familles de classification de séries temporelles :

— la classification par similarité, qui repose sur le calcul d'une distance entre des séries temporelles ;
— la classification par changement de représentation, qui repose sur la projection des séries temporelles dans un nouvel espace de représentation où la classification est réalisée ;
— la classification par ensemble, qui repose sur l'apprentissage d'un ensemble de classifieurs dont les résultats sont combinés pour obtenir une classe ;
— la classification par réseaux profonds, qui utilise les dernières avancées dans l'apprentissage par réseaux de neurones.

La suite est consacrée à la présentation de chaque famille de méthodes de classification de séries temporelles.

2.3.1 Classification par similarité

Dans certains jeux de données, les séries temporelles brutes sont suffisamment caractéristiques pour permettre d'identifier leur classe. Ainsi, deux séries temporelles qui se ressemblent ont de grande chance de partager la même classe. Cette information peut être utilisée pour classifier une nouvelle série temporelle en lui associant la classe de la série temporelle qui lui ressemble le plus. Nous identifions deux approches dans cette famille selon la portée de la similarité : globale (les séries temporelles sont comparées entièrement) ou locale (la ressemblance concerne uniquement une ou plusieurs sous-séries dans les séries temporelles).

Classification par similarité globale

Les méthodes de classification par similarité globale reposent sur la méthode des plus proches voisins. L'algorithme consiste à utiliser une mesure de distance pour calculer les distances entre la nouvelle série à classifier et l'ensemble des séries d'apprentissage. Les k séries dont les distances sont les plus faibles avec la nouvelle série sont sélectionnées. La classe retournée pour la nouvelle série est la classe majoritaire associée aux k séries. Comme l'algorithme est le même pour toutes, les méthodes de classification se distinguent par la mesure de distance utilisée. Toutes les mesures de distances que nous avons présentées dans la section 2.1.3 ont été proposées dans le contexte de la classification par la méthode des plus proches voisins.

Classification par similarité locale

Dans certains jeux de données l'information pour discriminer une série temporelle peut se réduire à la présence ou l'absence d'une ou plusieurs sous-séries caractéristiques. Par exemple, une machine peut être classée défectueuse quand nous observons la présence de pics de consommation électrique anormaux dans sa courbe de consommation. Ces perturbations peuvent passer inaperçues pour les méthodes de classification par similarité globale, car leur impact sur les distances calculées avec la série temporelle entière peut être minime. Cependant si nous prenons en compte uniquement la présence ou l'absence de ces sous-séries caractéristiques, il est alors possible de les classifier. Ces sous-séries caractéristiques ont été nommées *shapelet* et introduites pour la première fois dans [YK09].

Une shapelet $S = (s_1, \ldots, s_l)$ est une sous-série de taille l qui est extraite (ou apprise [Gra+14]) à partir d'un ensemble de séries temporelles. Soit une série temporelle $T = (t_1, \ldots, t_L)$ de taille L, la distance entre S et T se calcule de la manière suivante :

$$d_T^S = \min_{1 \leq j \leq L-l+1} \sqrt{\sum_{i=1}^{l} (s_i - t_{i+j-1})^2}. \tag{2.1}$$

Il s'agit de calculer la distance euclidienne entre S et toutes les sous-séries (de taille l) dans T et de conserver la distance minimale (qui correspond à la sous-série qui ressemble le plus à S).

Il existe deux approches de classification de séries temporelles à l'aide de shapelets. La première consiste à construire un arbre de décision en partitionant itérativement les séries temporelles d'apprentissage selon leur distance avec un ensemble de shapelets. À chaque nouvelle partition, l'algorithme énumère toutes les sous-séries possibles dans les séries d'apprentissage et conserve la sous-série qui discrimine le mieux les séries d'apprentissage en utilisant l'entropie [YK09]. Dans [KR13], ce processus est accéléré en changeant la représentation des sous-séries explorées en mots via la méthode SAX [Lin+07]. La sous-série trouvée est enregistrée en tant que shapelet ainsi que le seuil de distance avec cette shapelet qui permet de discriminer les séries temporelles. Ces informations sont utilisées pour permettre la classification des futures séries temporelles. Le partitionnement continue jusqu'à ce que toutes les séries temporelles d'une partition soient de la même classe. Cette classe est associée à cette partition et à toutes les futures séries temporelles classifiées dans cette partition.

La seconde approche consiste à représenter les séries temporelles via leurs distances avec K shapelets. Soit un ensemble $\mathcal{D} = \{S_1, \ldots, S_K\}$ de K shapelets, la *transformation à base de shapelets* a été introduite pour la première fois dans [Lin+12]. Cette technique consiste à représenter T en un vecteur $\mathbf{v_T} = (d_T^{S_1}, \ldots, d_T^{S_K})$ avec $d_T^{S_i}$ la distance entre la shapelet S_i et T , pour tout $1 \leq i \leq K$. Cette représentation est ensuite utilisée pour apprendre des classifieurs classiques telles que une machine à vecteur de supports ou une forêt d'arbre de décisions aléatoire [Lin+12 ; Gra+14 ; BB15 ; KPB16]. Plusieurs stratégies ont été proposées pour identifier les K meilleures shapelets :

— énumérer toutes les sous-séries possibles dans les séries d'apprentissage, et conserver les shapelets qui discriminent le mieux les séries d'apprentissage selon leur distance (celles avec la meilleure entropie, par exemple [Lin+12]) ;

— créer les shapelets optimales pendant l'apprentissage du classifieur [Gra+14] ;
— tirer un grand nombre aléatoire de shapelets dans toutes les sous-séries possibles issues des séries de l'ensemble d'apprentissage. Un classifieur qui intègre une phase de sélection d'attributs est utilisé pour conserver les shapelets les plus utiles [Ren+15 ; KPB16].

L'intérêt de cette dernière stratégie est d'être moins coûteuse en ressources (en particulier sur les gros jeux de données) car la sélection des shapelets est intégrée dans l'apprentissage du classifieur. Dans [Ren+15], il a été montré que quelques milliers de shapelets suffisent pour atteindre les performances de classification des méthodes de l'état de l'art sur les jeux de données standards.

2.3.2 Classification par changement de représentation

Dans cette partie, nous présentons les classifieurs de séries temporelles qui reposent principalement sur le changement de représentation des séries temporelles pour les classifier. Nous avons vu dans la partie précédente que la transformation à base de shapelets est une approche parmi ces méthodes. Nous identifions deux autres approches parmi les classifieurs avec changement de représentation. La première utilise un dictionnaire de sous-séries pour encoder les séries temporelles. La seconde représente les séries temporelles par des attributs calculés sur un ensemble d'intervalles extraits des séries temporelles.

Changement de représentation par dictionnaire

La classification à base de dictionnaire s'inspire d'une méthode utilisée en traitement de texte, appelée sac de mots. Elle consiste à représenter un texte selon le nombre d'occurrences des mots présents dans un dictionnaire. Cette représentation prend la forme d'un histogramme qui est ensuite utilisé pour indexer, regrouper ou classer les documents.

L'application de cette technique sur les séries temporelles repose sur la transformation des séries temporelles en ensembles de mots. Pour cela les séries temporelles sont découpées en intervalles réguliers et consécutifs. Chaque intervalle est ensuite encodé en un mot. Dans toutes les méthodes, les séries temporelles sont découpées en intervalles via une fenêtre glissante dont la taille est fixée par l'utilisateur. Les méthodes se différencient ensuite sur la façon de représenter les inter-

valles en mots et la façon de classifier les histogrammes. Dans [LKL12 ; SM13], les intervalles sont transformés en chaînes de caractères via la méthode de discrétisation SAX [Lin+02]. Dans [Sch15], les auteurs ont développé leur propre méthode de discrétisation reposant sur la transformation de Fourier discrète. Enfin dans [Bai+15], la méthode SIFT [Low99] utilisée pour la description d'images est adaptée aux séries temporelles pour décrire les intervalles.

Une fois les séries temporelles transformées en séquences de mots, un dictionnaire est extrait en listant l'ensemble des mots utilisés dans les séries temporelles. Chaque série temporelle est ensuite représentée en histogramme du nombre d'occurrences de chaque mot du dictionnaire dans la série temporelle. Cette représentation est utilisée pour la classification des séries temporelles. Dans [LKL12 ; Sch15], une série temporelle est classifiée par la classe de la série temporelle dont l'histogramme est le plus similaire à celui de la série classifiée. Dans [SM13], un histogramme est calculé par classe, une série est classfiée par la classe dont l'histogramme est le plus similaire au sien. Les mesures de distance utilisées sont la distance Euclidienne [LKL12], la distance cosinus [SM13] ou une mesure de distance personnalisée [Sch15]. À la différence des autres méthodes, un classifieur SVM est appris sur les histogrammes dans [Bai+15].

Changement de représentation par intervalles de temps fixe

Dans certains jeux de données, l'information pour discriminer les séries temporelles peut s'extraire dans un intervalle de temps fixe dans les séries temporelles. Par exemple, nous pouvons différencier les courbes de consommation électrique journalière d'une machine à café avec celle d'un four en observant uniquement la consommation le matin. En effet, la machine à café est régulièrement utilisée le matin alors que le four l'est beaucoup moins. Les méthodes présentées dans cette partie représentent les séries temporelles via des attributs extraits à partir d'intervalles de temps fixes (ex : la sous-série du matin dans notre exemple précédent). Ces représentations sont ensuite utilisées pour classifier les séries temporelles.

Le premier obstacle dans cette technique est d'identifier les intervalles de temps pertinents pour représenter les séries temporelles. Pour résoudre ce problème, les méthodes présentées [Den+13 ; BRT13 ; BR16] sélectionnent aléatoirement les bornes de k d'intervalles de temps. Puis, un classifieur intègre une phase de sélection d'attributs pour conserver les intervalles de temps les plus utiles.

Une fois les bornes des k intervalles de temps extraits, le changement de représentation et la classification des séries temporelles peut commencer.

Dans [Den+13], chaque série temporelle est représentée par la concaténation d'attributs calculés sur ses valeurs présentes dans chaque intervalle de temps. Les attributs calculés sont la moyenne, l'écart type et la pente. Un classifieur est appris pour chaque attribut de chaque intervalle sur les représentations des séries temporelles de l'apprentissage. Lors de la classification d'une nouvelle série temporelle, cette dernière est transformée de la même façon que précédemment. Pour chaque valeur d'attribut une classe est retournée et la classe majoritaire est retournée pour la série temporelle.

D'autres propositions [BRT13 ; BR16], extraient toutes les sous-séries dans les séries d'apprentissage correspondant aux k intervalles de temps sélectionnés. Pour chaque intervalle de temps, un arbre de décision [BRT13] ou un arbre de régression [BR16] est appris sur les sous-séries de l'intervalle. Ces arbres sont utilisés pour définir un nouvel ensemble d'attributs pour chaque intervalle. Chaque série est ensuite représentée en concaténant les attributs de ses sous-séries. Enfin, une forêt aléatoire d'arbres décisionnels est apprise sur les représentations des séries d'apprentissage.

2.3.3 Classification par ensemble de classifieurs

Les méthodes de classification par ensemble reposent sur l'idée de combiner plusieurs classifieurs pour créer un unique classifieur plus performant. Selon les méthodes, les classifieurs utilisés sont les mêmes mais appris sur différents sous-ensembles des données d'apprentissage, ou bien sont des classifieurs différents appris sur l'ensemble des données d'apprentissage. La combinaison des classifieurs repose sur la mise en commun des résultats de chaque classifieur pour obtenir un unique résultat. Différentes approches sont proposées telles que retourner la classe majoritaire ou apprendre une pondération des résultats de chaque classifieur.

Il existe des méthodes ensemblistes dans toutes les grandes familles de classification de séries temporelles présentées précédemment. La méthode proposée dans [LB15] combine 11 classifieurs par similarité globale qui sont tous des classifieurs du type le plus proche voisin avec une mesure de distance différente. Dans [BB15], une méthode transformant les séries temporelles par des shapelets combine les résultats de 7 classifieurs différents appris sur la représentation des séries d'apprentissage. Dans [Sch15], plusieurs classifieurs sont appris sur différentes représentations par dic-

tionnaire des séries d'apprentissage. Les dictionnaires se différencient selon la taille des intervalles utilisée pour segmenter les séries temporelles.

D'autres propositions de méthodes ensemblistes combinent les classifieurs issus de différentes familles de classifieurs. Dans [LB15], les auteurs combinent un classifieur par similarité globale utilisant DTW et un classifieur à base de dictionnaire. Dans [Bag+15] une méthode ensembliste par similarité globale est combinée avec des classifieurs qui utilisent différentes représentations des séries temporelles. Cette méthode est étendue dans [LTB18] par l'ajout de méthodes de classification avec changement de représentation par dictionnaire et par intervalle.

2.3.4 Classification par réseaux de neurones profonds

Un réseau de neurones est un modèle qui s'inspire du fonctionnement du cerveau. Il s'agit de l'interconnexion en couches successives de plusieurs fonctions mathématiques simples (symbolisant les neurones). La première couche reçoit en entrée les données et la dernière retourne le résultat. Chaque connexion entre les neurones est pondérée et l'apprentissage du modèle consiste à optimiser les poids sur l'ensemble du réseau. Ce qui différencie un réseau de neurones d'un autre est la structure du réseau, la méthode d'apprentissage des poids et les fonctions utilisées dans les neurones. En classification de séries temporelles, le réseau prend en entrées des séries temporelles entières et retourne des probabilités de classification.

Récemment, Fawaz et al. [Faw+19] ont montré que différentes architectures de réseaux neurones pouvaient atteindre la même performance que les classifieurs de séries temporelles de l'état de l'art (par exemple, RESNET [WYO17]).

CHAPITRE 3

INTERPRÉTABILITÉ

En apprentissage automatique, l'interprétabilité est la capacité d'un modèle à pouvoir retourner une explication à ses décisions en des termes compréhensibles pour un humain. Cette notion existe depuis de très nombreuses années [ADT95] cependant son objectif, et par conséquent sa définition, restent encore débattus. En effet, l'interprétabilité a été utilisée pour répondre à de très nombreux objectifs :

- la confiance de l'utilisateur dans le système, car l'explication permet à l'utilisateur de valider le fonctionnement du modèle [ADT95 ; JNK04 ; RSG16]
- le respect de la vie privée, car l'explication permet de vérifier si des informations sensibles peuvent être extraites à partir du résultat [Tou+10]
- l'équité, car l'explication peut mettre en évidence l'utilisation d'un biais dans les données [HPS16]
- la causalité, car l'explication peut mettre en évidence des liens entre certains attributs et le résultat [Gui+19a ; DK17]
- la robustesse/fiabilité, car l'explication permet de s'assurer que les valeurs bruitées ne sont pas prises en compte dans le résultat [Gui+19a ; DK17]

Récemment, des travaux [Gui+19a ; DK17] ont considéré que l'interprétabilité se définissait par l'ensemble des objectifs qu'un modèle de fouille de données devait atteindre. Par exemple, si une méthode est proposée comme étant équitable et inspirant la confiance, alors cette méthode est interprétable si l'explication des résultats fournit des réponses à ces deux critères.

L'intérêt pour l'interprétabilité n'a cessé de prendre de l'ampleur ces dernières années. Nous pensons que cette effervescence est due à la démocratisation de l'utilisation de méthodes de fouilles de données dans des d'applications critiques (ex : outil d'autorisation de prêt, outil de diagnostic médical, etc), et dont nous sommes incapables d'expliquer les décisions. En effet, cette absence d'explication suggère que des décisions importantes sont prises sur la base d'une confiance aveugle dans ces systèmes, ce qui soulève de nombreuses craintes. Les travaux sur l'interprétabilité dans

des systèmes de décision sont une réponse à ces inquiétudes.

Dans le cadre de cette thèse, nous nous intéressons seulement à l'interprétabilité de modèles de classification. Dans la suite, nous présentons dans la section 3.1 les méthodes proposées pour expliquer les résultats de classifieurs. La section 3.2 est consacrée en particulier à la présentation de l'interprétabilité dans la classification de séries temporelles. Enfin, un des défis de l'interprétabilité est son évaluation et la section 3.3 est dédiée aux propositions pour évaluer l'interprétabilité.

3.1 Interprétabilité dans la classification

Récemment, [Gui+19a ; DK17 ; FK18] ont défini un ensemble de dimensions à prendre en considération dans l'interprétabilité d'un modèle :

- la portée de l'interprétabilité, qui distingue si l'explication d'un modèle est spécifique au résultat d'une unique instance dans les données (portée **locale**) ou bien à l'ensemble des résultats possibles (portée **globale**) ;
- le temps : selon le cadre d'utilisation des explications, il est nécessaire d'adapter l'explication au temps alloué à l'utilisateur. Si l'analyse doit être rapide, une explication simple et concise sera préférée. À l'inverse, si l'utilisateur a le temps d'analyser le résultat, il peut être préférable de retourner une explication plus longue avec plus d'informations ;
- l'expertise de l'utilisateur : si l'utilisateur a une connaissance des données et de l'apprentissage automatique, il est raisonnable de proposer une explication adaptée ;
- les biais cognitifs : il s'agit d'une distortion dans le traitement cognitif d'une information. Récemment, [FK18] a montré que l'ignorance de ces biais lors de la présentation de résultats, pouvait conduire à interpréter des résultats à l'inverse de ce qui était escompté (ex : portée d'une explication incomprise).

Il existe deux approches pour expliquer les résultats d'un classifieur. Dans la première, le classifieur est intrinsèquement interprétable et les explications peuvent être extraites directement à partir du classifieur. Dans la seconde, un classifieur interprétable (nommé proxy) est appris pour approximer les résultats du classifieur à expliquer. Les explications des résultats sont alors extraites du proxy. La suite de cette section est consacrée à la présentation de ces deux approches..

3.1.1 Modèle interprétable

Pour qu'un modèle soit considéré comme intrinsèquement interprétable, il doit vérifier deux propriétés :

— **propriété 1** : les attributs utilisés par le modèle doivent être compréhensibles pour l'utilisateur ;
— **propriété 2** : l'utilisation de ces attributs dans le calcul du résultat du classifieur doit être compréhensible.

Pour la première propriété, les attributs peuvent être tirés des données brutes (ex : les variables d'un tableau) ou extraites à partir d'un prétraitement (ex : les tendances d'une série temporelle). En général, la compréhension des attributs dépend de l'expertise de l'utilisateur sur les données.

La seconde propriété dépend du type de classifieur utilisé dans le modèle. Dans la littérature, les types de modèles régulièrement cités comme interprétables sont les modèles linéaires simples, les arbres de décision et les règles [RSG16 ; RSG18 ; LL17 ; Fre13 ; Lip18]. Ces classifieurs permettent de déduire facilement les contributions des différents attributs dans la classification. Dans [Fre13], les classifieurs de type les plus proches voisins sont également considérés comme interprétables. Une analyse plus approfondie sur l'interprétabilité de ces modèles est réalisée dans [Fre13]. Notons que Lipton [Lip18], considère que ces types de modèles ne sont pas intrinsèquement interprétables car selon leur taille (ex : le nombre de coefficients dans les modèles linéaires, la profondeur dans les arbres de décision ou le nombre de conditions dans une règle) ces modèles peuvent être considérés comme trop complexes pour être interprétables.

Enfin, il existe de nombreux classifieurs qui ne respectent pas au moins l'une des propriétés. En ce qui concerne la première propriété, de nombreuses méthodes reposent sur le changement de représentation des données (ex : discrétisation des séries temporelles, transformation de texte en histogramme de fréquence de mots, représentation d'images dans le domaine fréquentiel). Les nouveaux attributs générés peuvent alors n'avoir aucun sens pour l'utilisateur. Dans le cas de la seconde propriété, il existe de nombreux classifieurs dans lesquels il est impossible de tirer la contribution des attributs dans la classification d'une donnée, par exemple les réseaux de neurones ou les méthodes par ensemble de classifieurs.

3.1.2 Méthode d'explication par proxy

Comme tous les classifieurs ne sont pas interprétables, une technique a été proposée pour pouvoir les expliquer. Cette technique repose sur l'apprentissage d'un modèle interprétable dont l'objectif est de fournir des résultats similaires au classifieur non-interprétable. Le modèle interprétable utilisé est appelé proxy, et doit vérifier deux propriétés supplémentaires en plus de celles énoncées précédemment :

— **propriété 3** : le proxy doit être **fidèle**, c'est-à-dire fournir les mêmes classifications que le classifieur visé ;
— **propriété 4** : le proxy doit être **précis**, c'est-à-dire capable de bien classer de nouvelles données ;

Les méthodes d'explication par proxy peuvent être divisées en deux catégories selon la portée de l'approximation et donc de l'interprétabilité : globale ou locale. Cette différence n'est pas réalisée pour les modèles interprétables car ils sont considérés comme globalement et localement interprétables.

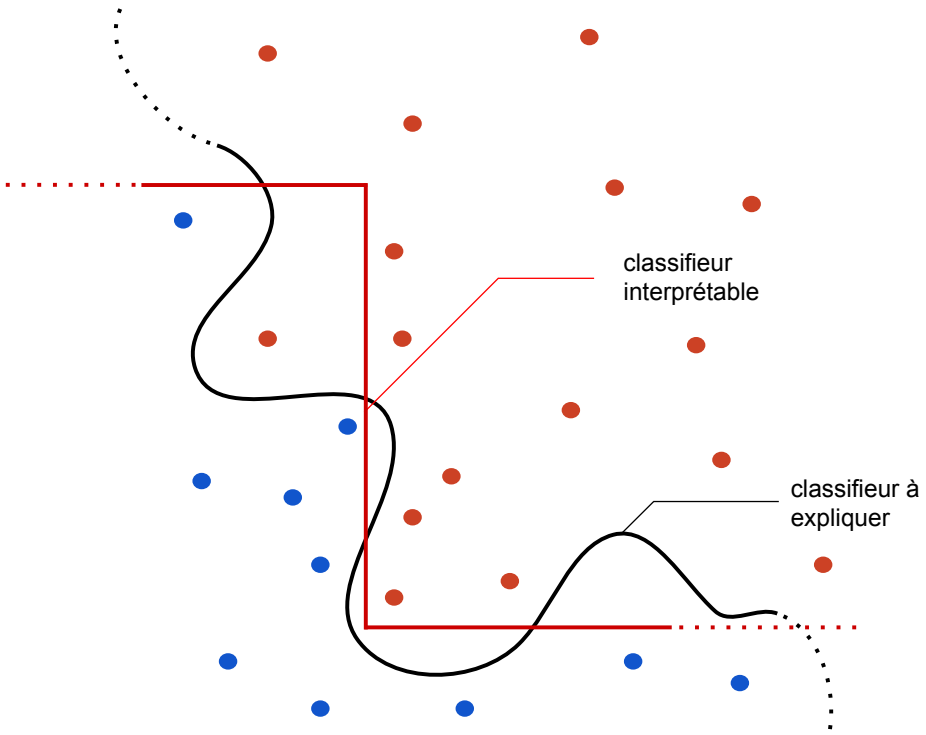


FIGURE 3.1 – Méthode d'explication globale par proxy.

Explication globale

Une méthode d'explication globale vise à expliquer n'importe quel résultat du classifieur visé. Dans cette approche le proxy à apprendre doit approximer le modèle sur l'ensemble des données tel que illustré dans la Figure 3.1. En général, une méthode d'explication globale s'applique à un type particulier de classifieur et un type précis de données. Dans [AK12 ; JKN04 ; ADT95] par exemple, un ensemble de règles est obtenu par ingénierie inverse à partir d'un modèle de type réseau de neurones.

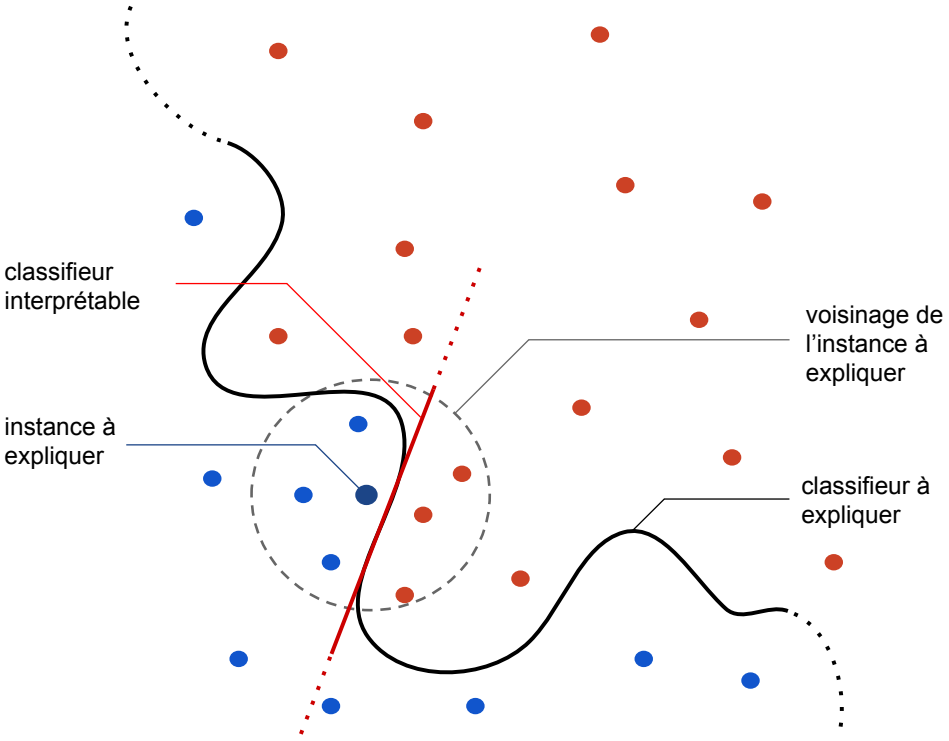


FIGURE 3.2 – Méthode d'explication locale par proxy.

Explication locale

Une méthode d'explication locale vise à expliquer un seul résultat du classifieur. À la différence des méthodes globales, le proxy est appris pour approximer le modèle dans le voisinage du résultat à expliquer. En se concentrant sur une instance et son voisinage, le modèle interprétable peut être plus simple tout en restant fidèle au modèle à expliquer. La figure 3.2 illustre un modèle explicatif local.

Dans [Zho+16 ; Xu+15], des méthodes pour expliquer localement la classification d'images ont été proposées. Il s'agit de méthodes spécifiques à certaines architectures

de réseaux de neurones qui retournent comme explication, les zones de l'images qui ont contribué à la classification.

À l'inverse, des méthodes agnostiques ont été proposées pour expliquer les résultats de n'importe quel type de classifieur. LIME [RSG16] propose d'apprendre le proxy sur un ensemble de variables interprétables pour faciliter l'explication aux utilisateurs. L'intuition principale derrière LIME est qu'une explication peut être générée localement en calculant un modèle interprétable à partir d'instances de données générées aléatoirement autour de l'instance à expliquer. Les contributions des voisins dans l'apprentissage du proxy sont pondérées selon leur proximité avec l'instance sélectionnée. Dans leurs expérimentations, les auteurs adoptent un modèle linéaire comme modèle interprétable local. Avec ANCHOR [RSG18], une variation de LIME, Ribeiro et al. proposent de décrire par une règle, l'espace autour de l'instance dont nous cherchons à expliquer le résultat. L'objectif consiste à décrire l'espace le plus grand et dont les données doivent être classifiées de la même façon que l'instance à expliquer.

SHAP [LL17] a le même objectif que LIME. Cependant, SHAP s'appuie sur la théorie des jeux coopératifs, pour présenter l'apprentissage du proxy sous un autre angle. Ici, le résultat du classifieur est considéré comme une récompense à partager entre les attributs des données en fonction de leur contribution à ce résultat. La solution à ce problème se présente sous la forme d'une pondération des attributs dont la somme donne le résultat du classifieur. Le proxy est vu comme un modèle linéaire dont les coefficients correspondent aux poids des attributs. Présenté de cette façon, les auteurs montrent que l'apprentissage du proxy a une solution unique et correspond à un modèle linéaire dont les coefficients sont nommés valeurs de *Shapley* [Sha53].

Dans le Chapitre 6, nous présentons une généralisation des méthodes d'explication LIME et SHAP. À partir de cette généralisation nous avons créé une méthode d'explication locale par proxy capable d'expliquer les résultats de n'importe quel classifieur de séries temporelles.

3.2 Interprétabilité et classification de séries temporelles

Dans cette section, nous présentons les classifieurs de séries temporelles selon leur interprétabilité ainsi que les méthodes d'explication qui ont été proposées pour

rendre interprétable les classifieurs qui ne le sont pas. La section 3.2.1 présente les classifieurs de séries temporelles considérés comme interprétables, tandis que la section 3.2.2 est consacrée aux classifieurs non-interprétables. Enfin, la section 3.2.3 présente les quelques méthodes d'explications qui ont été proposées pour rendre interprétables les classifieurs qui ne le sont pas.

3.2.1 Classifieurs interprétables de séries temporelles

Dans cette section, nous présentons les classifieurs de séries temporelles considérés comme interprétables. Pour rappel, un classifieur est interprétable s'il vérifie les deux propriétés suivantes :

— **propriété 1** : les attributs utilisés par le modèle doivent être compréhensibles pour l'utilisateur ;
— **propriété 2** : l'utilisation des ces attributs dans le calcul du résultat du classifieur doit être compréhensible.

À notre connaissance, il existe très peu de classifieurs interprétables de séries temporelles. Ces derniers appartiennent à la catégorie des classifieurs par similarité vue dans la section 2.3.1.

Ces classifieurs peuvent être séparés en deux catégories selon la portée de la similarité. Dans la première, la similarité est globale et concerne la série temporelle entière ; dans la seconde, elle est locale et porte sur des sous-séries discriminantes.

La suite est consacrée à la présentation de chacune des catégories.

Classification interprétable par similarité globale

Dans cette catégorie, les classifieurs proposés reposent sur la technique des plus proches voisins. Pour rappel, cette technique de classification consiste à associer à une nouvelle série temporelle ts, la classe majoritaire parmi les k séries temporelles les plus similaires à ts. La similarité est calculée via une mesure de distance (cf. section 2.1.3). Cette technique est considérée comme interprétable [Fre13] car la notion d'associer deux objets, qui partagent des traits communs, à la même classe est intuitive pour un être humain. Par exemple, si nous sommes confrontés à un objet qui possède des roues, nous supposons que cet objet est mobile car d'autres objets possédant des roues sont mobiles. Ce même raisonnement peut être appliqué aux séries temporelles : si deux séries temporelles ont des suites de valeurs très proches

alors ces dernières peuvent appartenir à la même classe. Cependant certaines limites doivent être considérées :

— la mesure de distance calcule une similarité que l'utilisateur peut comprendre, comme par exemple la distance euclidienne ou DTW [BC94] ;
— la taille des séries temporelles n'est pas trop grande, car observer les similarités entre deux séries temporelles très longues peut devenir fastidieux pour un utilisateur.

En général, tous les travaux qui ont présenté une nouvelle mesure de distance [KR05 ; Hir77 ; CN04 ; Mar07 ; SAD13] pour la classification de séries temporelles, utilisent la technique des plus proches voisins pour la classification et peuvent donc être considérés comme interprétables si les limites définies plus tôt sont respectées.

Classification interprétable par similarité locale

Dans certains jeux de séries temporelles, l'information discriminante entre les séries temporelles est la présence ou l'absence de sous-séries spécifiques. Par exemple, la présence d'un pic de valeurs dans la courbe de consommation électrique d'un type de machines peut être discriminante lorsqu'une machine est défectueuse. Cependant, pour que ces sous-séries soient compréhensibles pour l'utilisateur, ce dernier doit être capable d'associer des comportements de la source des séries temporelles à la sous-série. Dans l'exemple précédent, l'utilisateur doit comprendre que le pic de valeur dans la série temporelle est associé une surconsommation électrique de la machine.

Ces sous-séries ont été introduites sous le terme shapelet dans la section 2.3.1. La distance entre une shapelet et une série temporelle est un attribut compréhensible pour l'utilisateur car elle peut être comprise comme la présence ou l'absence de la shapelet dans la série temporelle (si la distance est très faible alors la shapelet est présente et inversement si la distance est très élevée). La distance avec les shapelets est un attribut utilisé dans de nombreux classifieurs [YK09 ; KR13 ; Lin+12 ; Gra+14 ; LTB18 ; Bag+15 ; Hil+14] dont certains sont interprétables car ils utilisent des arbres de décisions simples [YK09 ; KR13]. Un point est à noter quant à la profondeur de l'arbre de décision créé. Si cette dernière est trop élevée, le classifieur cesse d'être interprétable à cause de la complexité de l'arbre.

Actuellement, les classifieurs interprétables à base de shapelets n'ont pas une très bonne précision par rapport aux méthodes de l'état de l'art. Dans le Chapitre 5, nous proposons un nouveau classifieur interprétable à base de shapelets, qui a une

meilleure précision que ceux existants.

3.2.2 Classifieurs non-interprétables de séries temporelles

Il existe deux raisons importantes pour qu'un classifieur ne soit pas interprétable : les attributs utilisés sont incompréhensibles pour l'utilisateur, et/ou la contribution des attributs dans la classification est incompréhensible. Cette présentation aborde les classifieurs de séries temporelles non-interprétables selon ces deux raisons.

Classification avec des attributs non-compréhensibles

Toutes les méthodes de classification qui prennent en entrée des vecteurs numériques peuvent être utilisées pour classifier des séries temporelles. Dans ce cas, chaque valeur des séries temporelles devient un attribut indépendant des autres valeurs. Comme chaque valeur est considérée indépendamment, les explications sur cette représentation des séries temporelles consistent à expliquer la contribution individuelle de chaque valeur de la série temporelle dans la classification. Il s'agit dans la majorité des cas d'une granularité trop fine pour être utile à un utilisateur.

Dans la section 2.1.2 nous avons vu que le changement de représentation des séries temporelles est un processus souvent utilisé pour permettre la réalisation de tâches de fouille de données dans les séries temporelles. Dans certains cas, la représentation utilisée extrait des attributs non-compréhensibles pour l'utilisateur. La suite est consacrée à la présentation de ces représentations.

Dans la section précédente, nous avons vu que les shapelets pouvaient être un attribut compréhensible extrait des séries temporelles. Une shapelet est compréhensible si elle est tirée d'une série temporelle réelle, ce qui permet à un utilisateur d'y associer un comportement de la source des séries temporelles. Or dans certains classifieurs [Gra+14], les shapelets sont transformées pour être optimales dans leur tâche de classification. Cela conduit à l'utilisation de shapelets qui ne ressemblent plus du tout aux sous-séries dans les séries temporelles et par conséquent ne peuvent plus être associées à des comportements de la source.

Les changements de représentation des séries temporelles par dictionnaire ou par intervalle de temps fixe (cf. section 2.3.2) représentent respectivement les séries temporelles par des histogrammes de fréquences de mots, ou par des concaténations d'attributs divers calculés sur des intervalles de temps. Ces nouveaux attributs ne sont

pas compréhensibles pour un humain.

Les attributs non-compréhensibles ne sont pas l'unique raison pour qu'un modèle ne soit pas interprétable, le type de classification a aussi un impact.

Type de classification non-compréhensible

Il existe des types de classifieurs qui sont intrinsèquement non-interprétables. Cette partie est consacrée à leur présentation dans la classification de séries temporelles.

Les machines à vecteurs de support [BGV92] utilisées dans [Cut11 ; Bag+17 ; RAM05 ; Kat16] sont considérées comme non-interprétables. Ce sont des classifieurs linéaires dont l'objectif peut être considéré comme la génération d'une frontière qui cherche à séparer les données selon leur classe. La frontière doit maximiser la distance entre elle et les échantillons proches d'elle (appelés *vecteurs de support*). Dans le cas où les données ne sont pas linéairement séparables, les machines à vecteurs de support peuvent changer l'espace de représentation des séries dans un espace de plus haute dimension, où peut exister une séparation linéaire. Cette transformation est réalisée via une fonction *noyau*. Ce changement de dimension rend les attributs non-compréhensibles pour l'utilisateur, par conséquent le classifieur n'est pas interprétable.

Un autre type de classification repose sur les réseaux de neurones (cf. section 2.3.4) utilisés dans les méthodes proposées dans [Bag+17 ; WYO17 ; Faw+19]. Dans ce type de classification, la contribution d'un attribut dans le résultat se perd dans l'interconnexion des couches du modèle, qui peuvent être nombreuses.

La classification par ensemble de classifieurs (cf. section 2.3.3) repose sur la combinaison en un unique modèle de plusieurs classifieurs. Dans ce type de méthodes, la contribution d'un attribut est fragmentée entre les différents classifieurs dont certains peuvent être non-interprétables, c'est la raison pour laquelle ce type de classifieurs est considéré comme non-interprétable.

Le développement de telles méthodes non-interprétables tient en particulier au fait que ces méthodes sont très performantes pour classifier. La proposition qui a été faite pour réintroduire de l'interprétabilité dans ces modèles consiste à apprendre un second classifieur interprétable qui approxime les modèles non-interprétables. Les explications du classifieur non-interprétable sont ensuite tirées de ce second classifieur.

3.2.3 Méthode d'explication par proxy de classifieur de séries temporelles

Cette section présente les solutions proposées pour expliquer les classifieurs de séries temporelles non-interprétables. Nous avons vu dans la section 3.1.2 que l'approche utilisée pour résoudre ce problème consiste à apprendre un classifieur qui est interprétable (appelé proxy) et qui approxime les résultats du classifieur non-interprétable. Les explications du classifieur sont tirées du proxy.

La littérature comprend peu de méthodes d'explication par proxy pour classifieur de séries temporelles. Dans une étude récente [Gui+19a] qui réalise un état de l'art des méthodes d'explication par proxy pour classifieur, nous voyons que les données considérées sont les images, le texte et avec une très grande majorité les données tabulaires. Ces méthodes peuvent être séparées en deux catégories selon la portée de l'approximation et donc de l'interprétabilité : globale ou locale.

À notre connaissance, il n'existe pas de méthode d'explication par proxy qui permette d'expliquer globalement un classifieur de séries temporelles.

À l'inverse, des propositions ont été faites pour expliquer localement les résultats de classifieurs non-interprétables [WYO17 ; Faw+19]. Cependant, ces méthodes permettent d'expliquer seulement les classifieurs de séries temporelles à base de réseaux de neurones possédant une couche de *Global Average Pooling*. Wang et al. ont exploré l'utilisation de *Class Activation Maps* (CAM) pour fournir une visualisation interprétable mettant en valeur les sous-séquences de séries temporelles qui ont contribué le plus à sa classification. Fawaz et al. propose une autre technique de visualisation, basée sur *Multi Dimensional Scaling*, pour comprendre la représentation latente apprise par un réseau de neurone profond. L'objectif est d'obtenir des connaissances sur la distribution spatiale des séries temporelles appartenant aux différentes classes dans le jeu de données.

Dans le dernier travail de cette thèse, présenté dans le Chapitre 6, nous proposons une nouvelle méthode d'explication locale par proxy de résultats de classifieurs de séries temporelles. Cette méthode à pour particularité d'être modèle-agnostique, c'est-à-dire qu'elle est capable de fournir une explication pour les résultats de n'importe quel classifieur.

3.3 Évaluation de l'interprétabilité

En pratique, la grande majorité des travaux évaluent l'interprétabilité selon la taille de l'explication [ADT95 ; Lak+17 ; AK12], c'est-à-dire la taille du modèle interprétable. Pour les règles, il s'agit du nombre de règles et de leur nombre de conditions dans l'antécédent. Dans les modèles linéaires, l'interprétabilité est donnée à partir du nombre de coefficients non-nuls, et dans les arbres de décision il s'agit de la profondeur de l'arbre et du nombre de nœuds.

D'autres travaux [RSG16 ; LL17 ; RSG18 ; Kul+15], utilisent des études utilisateur (ou *user studies*) qui consistent à interroger un panel d'utilisateurs sur les explications retournées. Par exemple, dans [RSG16], les questions données cherchent à évaluer la pertinence des mots utilisés pour de la classification de textes. En général, les utilisateurs sont soit des collègues ou des étudiants [RSG16 ; LL17 ; RSG18 ; Kul+15], soit des travailleurs de plate-formes de production participative [RSG16 ; LL17] (comme *Amazon Mechanical Turk*[1] ou *Foule Factory*[2]).

Ces deux types d'évaluation ont été catégorisés dans [DK17] selon trois niveaux d'évaluation :

— applicatif : une étude utilisateur où les utilisateurs sont des experts et la tâche est réelle, il s'agit d'une évaluation en situation réelle ;
— étude utilisateur : une étude utilisateur où les utilisateurs sont non-experts sur une tâche simple comme dans [RSG16 ; LL17 ; RSG18 ; Kul+15] ;
— fonctionnel : l'évaluation est réalisée via une métrique sur les explications et sans interrogation d'utilisateur, ce niveau couvre l'évaluation selon la taille des modèles par exemple.

À notre connaissance, aucune méthode n'a évalué l'interprétabilité dans le cadre de la classification de séries temporelles. Dans la section 6.5, nous présentons une étude utilisateur réalisée pour évaluer l'interprétabilité des explications de classification de séries temporelles.

1. `https://www.mturk.com/`
2. `https://www.foulefactory.com/en/`

DEUXIÈME PARTIE

Recherche d'occurrences de règles temporelles dans des séries temporelles

CHAPITRE 4

RECHERCHE D'OCCURRENCES ÉLASTIQUES DE RÈGLES TEMPORELLES

4.1 Motivations

Dans ce chapitre, nous nous intéressons à la découverte d'un type de régularité en particulier : les règles temporelles (Définition 2.2.4). Soient deux motifs A et C (Définition 2.2.3) présents dans une série temporelle, la règle temporelle $R = (A, C, p)$ exprime la relation suivante :

$$A \xrightarrow{p} C$$

où les occurrences du motif A (nommé antécédent) sont suivies, avec une contrainte de temps p, par les occurrences du motif C (nommé conséquent). Nous nous limitons à une contrainte de temps usuelle de délai maximum entre A et C. L'approche générale pour découvrir des règles temporelles dans des séries temporelles s'appuie sur la discrétisation des séries temporelles. Cette technique consiste à simplifier la représentation des séries temporelles dans le but de faciliter une tâche de fouille de données (cf. section 2.1.2). Cependant, un des défauts de cette technique est d'accompagner la simplification des données par une perte d'information qui peut être pénalisante pour l'utilisateur.

Pour éviter ce problème, une méthode récente [Sho+15] a proposé une méthode de découverte de règles temporelles qui conserve les séries brutes. Cette méthode commence par découvrir un motif fréquent par une préroutine (*MK algorithm* [Mue+09]). Le motif (qui est une sous-série) est séparé en deux à différentes positions choisies arbitrairement pour former un ensemble de paires de sous-motifs. Ces paires de sous-motifs sont ensuite évaluées individuellement comme règles temporelles. Le premier sous-motif est l'antécédent et le second est le conséquent. Pour évaluer une règle can-

didate, l'algorithme calcule un score inspiré de *MDL* [BRY98] sur les occurrences de la règle candidate dans la série temporelle. Enfin, la règle avec le score le plus élevé est retournée à l'utilisateur.

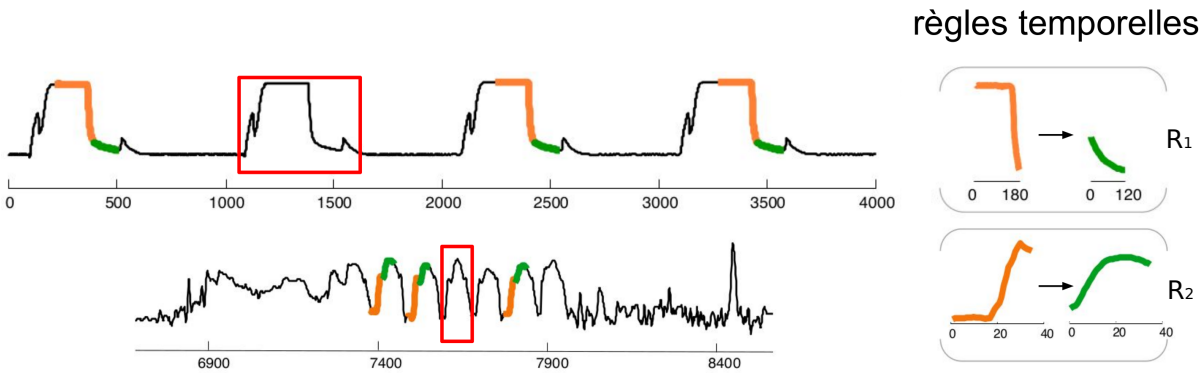
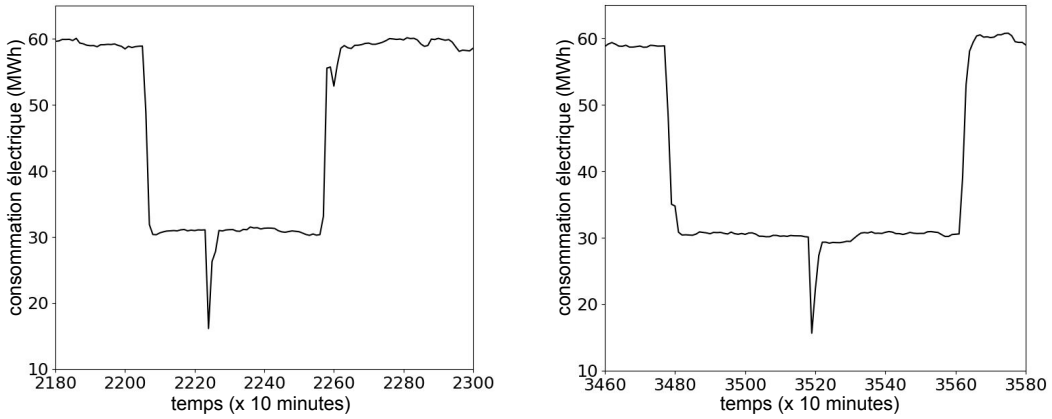


FIGURE 4.1 – Occurrences de deux règles temporelles R_1 et R_2 dans leur séries respectives (figure extraite de [Sho+15]). Les rectangles rouges indiquent des occurrences de règles qui semblent avoir été manquées.

La Figure 4.1 présente deux règles découvertes dans [Sho+15] et leur occurrences dans leur série temporelle respective. Nous voyons que la plupart des occurrences des règles ont été trouvées, cependant quelques occurrences semblent avoir été manquées (rectangles rouges). Sans être des experts de ces données, nous jugeons que les occurrences ratées semblent similaires aux règles à de légères différences près. Il s'agit là d'une situation qui est assez commune dans les tâches de fouilles de données : implémenter une similarité entre les séries temporelles satisfaisant l'expert.

De nombreuses méthodes s'autorisent à considérer similaires des séries temporelles dont la valeur évolue de la même façon malgré des décalages dans le temps ; comme illustré dans la Figure 4.2. Dans la réalité, de nombreux facteurs peuvent agir sur les capteurs ou la source des séries temporelles, et par conséquent créer des perturbations dans les séries temporelles. Dans ces conditions, deux enregistrements d'un comportement redondant d'un système a de grande chance de produire deux sous-séries légèrement différentes (comme illustré dans la Figure 4.2), mais jugées similaires par l'utilisateur car représentant le même comportement.

Dans [Sho+15], l'évaluation de la similarité entre des séries intervient lors de la recherche des occurrences des règles candidates. Cette étape consiste à trouver dans la série temporelle les sous-séries qui ressemblent à l'antécédent et au conséquent d'une règle et qui respectent la règle $A \xrightarrow{p} C$. Dans la pratique, la similarité entre



(a) première série temporelle (b) seconde série temporelle

FIGURE 4.2 – Deux séries temporelles qui se ressemblent fortement mais avec des légères déformations.

une sous-série et un motif (antécédent ou conséquent) est calculée via une mesure de distance. Si la distance calculée est inférieure à un seuil de "similarité" alors la sous-série est considérée comme une occurrence du motif. La mesure de distance utilisée dans [Sho+15] est la distance Euclidienne qui est une distance classique et rapide à calculer. Cependant, elle est connue pour être inadaptée pour l'évaluation de la similarité entre des séries jugées similaires mais avec des valeurs décalées dans le temps. comme illustré dans la Figure 4.3. En effet, la distance Euclidienne calcule la distance en sommant une distance entre les points qui sont à la même position dans les séries. Par conséquent, un léger décalage dans le temps entre des valeurs normalement alignées a un impact important sur la distance qui est calculée. Pour résoudre ce problème, un ensemble de mesures de distance dites "élastiques" a vu le jour (cf. section 2.1.3). Il s'agit de mesures de distance qui s'autorisent à modifier l'alignement entre les séries dans le but de minimiser la distance finale, et dont la plus réputée est la déformation temporelle dynamique (DTW) [BC94].

Le travail présenté dans ce chapitre est l'intégration de mesures de distance élastiques dans la recherche d'occurrences de règles temporelles dans une série temporelle [Gui+17], telle que réalisée dans [Sho+15]. Notre objectif est d'améliorer la recherche de règles temporelles dont les occurrences de motifs peuvent être légèrement déformées (très fréquentes dans les système réels tels que ceux liés à la consommation d'énergie). À notre connaissance, aucune méthode de ce genre n'a été proposée,

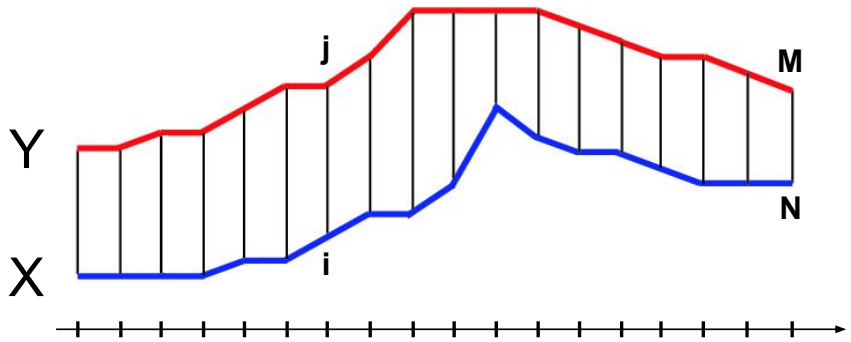


FIGURE 4.3 – Calcul de la distance euclidienne entre deux séries temporelles.

bien que cette idée ait été évoquée dans les perspectives d'amélioration de [Sho+15]. La section 4.2 est séparée en deux parties. La première partie présente les mesures de distance élastiques utilisées pour la recherche des occurrences de règles dans les séries temporelles. La seconde partie présente les modifications de l'algorithme de recherche d'occurrences pour intégrer ces mesures de distance. La section 4.3 est consacrée aux expérimentations.

4.2 Recherche d'occurrences élastiques de règles temporelles

Dans cette section, nous présentons tout d'abord les mesures de distance élastiques que nous avons utilisées pour la recherche d'occurrences de règles temporelles. La partie suivante est consacrée à la présentation de l'algorithme de recherche d'occurrences de règles temporelles dans une série temporelle.

4.2.1 Mesures de distance élastiques

Dans la section 2.1.3, nous avons présenté un certain nombre de mesures de distance élastiques. Dans cette section nous présentons les deux mesures élastiques que nous allons utiliser pour la recherche d'occurrences de règles temporelles dans des séries temporelles. Le premier choix est la DTW [BC94]. En effet, il s'agit d'une mesure de distance élastique simple et robuste, dont les performances sont meilleures que la plupart des mesures élastiques qui ont été proposées [Wan+13]. La section 4.2.1

présente plus en détail comment cette distance est calculée. En plus de DTW, nous avons sélectionné une de ses variantes appelé *sous-séries DTW* (ssDTW) [Mül07] qui permet de calculer efficacement la mesure DTW entre une série temporelle et toutes les sous-séries possibles d'une autre série temporelle. La section 4.2.1 est consacrée à sa présentation.

DTW

Soient deux séries temporelles $X = (x_1, \ldots, x_N)$, de taille $N \in \mathbb{N}$, et $Y = (y_1, \ldots, y_M)$, de taille $M \in \mathbb{N}$. Un alignement Al est une séquence de paires d'indices $Al = (w_1, \ldots, w_K)$ de taille K avec $max(N, M) \leq K \leq M + N - 1$. Pour $1 \leq k \leq K$, $w_k = (i, j)$ la paire d'indices qui représente l'alignement entre la $i^{ème}$ valeur de Y et la $j^{ème}$ valeur de X.

Un alignement par la mesure DTW, tel que illustré dans la Figure 4.4, est sujet à deux contraintes :

— **la bordure :** $w_1 = (1, 1)$ et $w_K = (N, M)$. L'alignement doit ainsi respectivement aligner entre elles les premières valeurs et les dernières valeurs de X et de Y.

— **la monotonie :** soit $w_k = (i, j)$ alors $w_{k+1} = \begin{cases} & (i+1, j) \\ ou & (i, j+1) \\ ou & (i+1, j+1) \end{cases}$
Cette contrainte permet de s'assurer que tout point de X est aligné avec au moins un point dans Y et inversement. De plus, l'ordre des valeurs des deux séries est préservé.

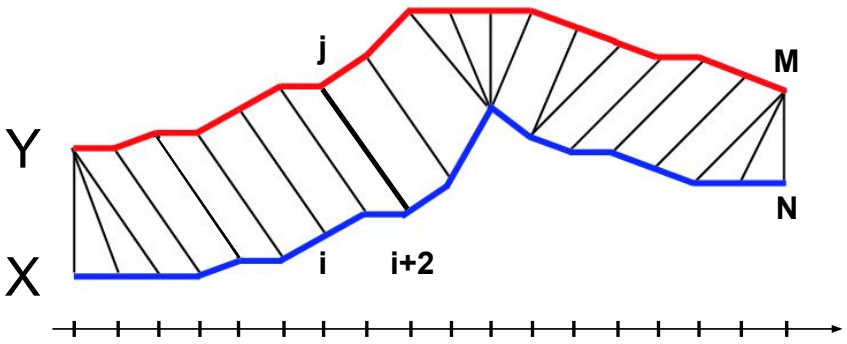


FIGURE 4.4 – Exemple d'un alignement $A = ((1, 1), (2, 1), (3, 1), (4, 2), \ldots, (N, M))$ entre deux séries temporelles par la mesure DTW.

La distance d'un alignement Al entre X et Y se calcule de la manière suivante :

$$d_A l = \sqrt{\sum_{(i,j)\in Al}^{K} (x_i - y_j)^2}$$

Soit $\mathcal{AL}$ l'ensemble des alignements possibles selon les contraintes énoncées précédemment, l'objectif de la distance DTW est de retourner la distance de l'alignement $A \in \mathcal{AL}$ entre X et Y qui est minimale, c'est-à-dire :

$$DTW(X,Y) = \min_{Al\in\mathcal{AL}}\{d_A l\}$$

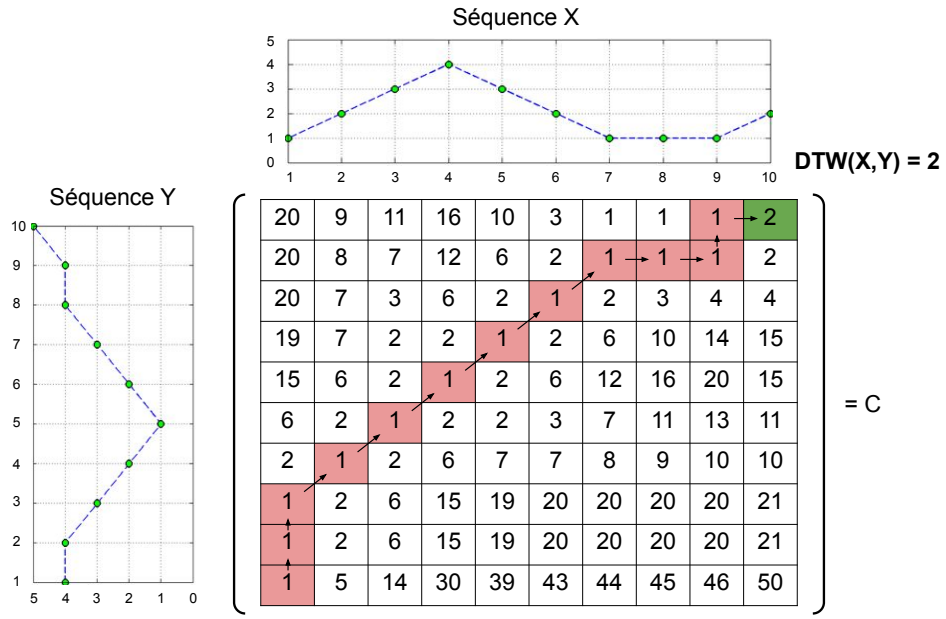


FIGURE 4.5 – Matrice pour calculer la distance DTW.

La méthode naïve pour calculer cette distance est de calculer la distance de tous les alignements présents dans $\mathcal{A}$ et de retourner le minimum. Dans la mesure DTW, le calcul est réalisé par programmation dynamique tel que illustré dans la Figure 4.5. Il s'agit de remplir une matrice $C \in \mathbb{R}^{M\times N}$ où chaque cellule $c_{i,j}$ est calculée de la façon

suivante :

$$c_{i,j} = \begin{cases} (x_i - y_j)^2 + min\{c_{i-1,j}, c_{i,j-1}, c_{i-1,j-1}\} & si\ i > 1\ et\ j > 1 \\ (x_i - y_j)^2 + c_{i-1,j} & si\ i > 1\ et\ j = 1 \\ (x_i - y_j)^2 + c_{i,j-1} & si\ i = 1\ et\ j > 1 \\ (x_i - y_j)^2 & si\ i = 1\ et\ j = 1 \end{cases}$$

La matrice est calculée ligne par ligne ou colonne par colonne en commençant par la cellule $c_{1,1}$. Une fois la matrice calculée, la distance DTW entre X et Y est :

$$DTW(X, Y) = \sqrt{c_{N,M}}$$

et l'alignement correspond aux couples (i, j) du chemin minimal menant de $c_{1,1}$ à $c_{N,M}$ (chemin en rouge dans la Figure 4.5).

ssDTW

DTW oblige à aligner tous les points de X avc Y. Or dans certains cas, seule une sous partie de X est intéressante. ssDTW permet de pallier à ce problème en permettent d'aligner entièrement Y avec une sous-série de X. Pour se faire, la contrainte de bordure est modifiée :

— **bordure sous-série :** $w_1 = (1, e)$ et $w_K = (M, f)$ avec $1 \leq e \leq f \leq N$. Y peut être aligné avec une sous-série de X (possiblement un unique point dans X).

ssDTW se calcule de la même façon que DTW, par programmation dynamique comme illustré dans la Figure 4.6. Il s'agit là aussi de remplir une matrice $C \in \mathbb{R}^{M \times N}$ où chaque cellule $c_{i,j}$ est :

$$c_{i,j} = \begin{cases} (x_i - y_j)^2 + min\{c_{i-1,j}, c_{i,j-1}, c_{i-1,j-1}\} & si\ i > 1\ et\ j > 1 \\ \mathbf{(x_i - y_j)^2 + min\{c_{i-1,j}, 0\}} & \mathbf{si\ i > 1\ et\ j = 1} \\ (x_i - y_j)^2 + c_{i,j-1} & si\ i = 1\ et\ j > 1 \\ (x_i - y_j)^2 & si\ i = 1\ et\ j = 1 \end{cases}$$

La seule différence est le calcul de la première ligne de la matrice (indiqué en gras) qui permet à chaque position dans X de calculer un nouvel alignement DTW avec Y.

Comme la série temporelle X peut ne pas être entièrement prise en compte, il faut considérer que la distance calculée à n'importe quelle position de X peut être la

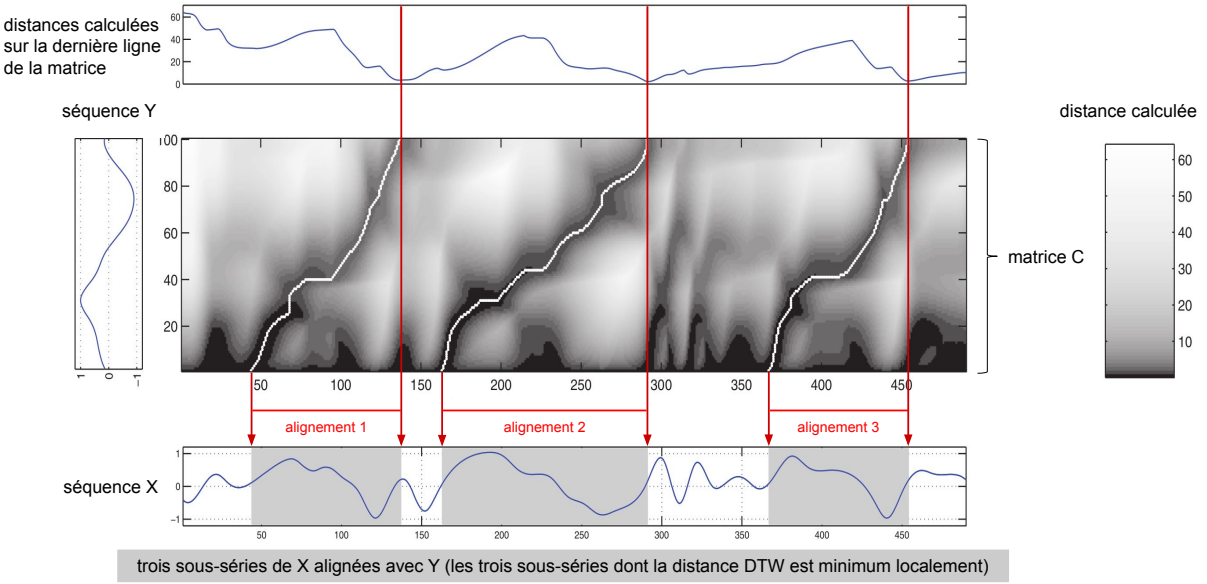


FIGURE 4.6 – Mesure ssDTW entre deux séries temporelles et extraction des sous-séries les plus similaires à Y dans X.

meilleure. Soit $Edist_{X,Y}$ l'ensemble de ces distances tel que :

$$Edist_{X,Y} = \{c(k, M)\} \in C \; telle \; que \; 1 \leq k \leq N$$

Il s'agit des valeurs dans la dernière ligne de la matrice C. Enfin, la distance ssDTW n'est donc pas calculée à partir $c_{M,N}$ comme dans DTW, mais à partir du minimum de $Edist_{X,Y}$ soit :

$$ssDTW(X, Y) = \min_{c \in Edist_{X,Y}}^{N} \sqrt{c}$$

De plus, ssDTW peut être utilisée pour effectuer du multi-alignement. Au lieu d'extraire l'alignement minimum, un ensemble d'alignements minimaux localement sont extraits, comme illustré dans la Figure 4.6.

4.2.2 Algorithme de recherche d'occurrences de règles temporelles

Nous utilisons la définition des règles temporelles donnée dans la section 2.2.2. La recherche des occurrences d'une règle temporelle $R = (A, C, p)$ dans une série

temporelle T se découpe en deux étapes :

- étape 1 : la recherche de l'ensemble des occurrences de l'antécédent E_T^A dans T ;
- étape 2 : la recherche de l'ensemble des occurrences du conséquent E_T^C dans T qui sont consécutives, avec la contrainte p, aux occurrences de l'antécédent trouvées.

La recherche des occurrences de l'antécédent et du conséquent repose sur le même fonctionnement. Une fenêtre est glissée de le long la série temporelle et à chaque position, nous calculons la distance entre la sous-série présente dans la fenêtre et le motif comparé (antécédent ou conséquent). Si cette distance est inférieure au seuil de "similarité" (une valeur fixée préalablement) alors la sous-série est conservée comme occurrence du motif comparé. Dans le cas des occurrences du conséquent, ces dernières sont cherchées après les occurrences de l'antécédent et dans une période de temps qui est contrainte. Les paramètres nécessaires sont donc :

- une valeur $maxlag$ qui spécifie le délai où chercher les occurrences du conséquent après celles de l'antécédent ;
- une mesure de distance $dist$;
- deux tailles de fenêtre w_a (respectivement w_c) pour découper la série temporelle en sous-séries qui seront ensuite comparées à l'antécédent (respectivement avec le conséquent) ;
- deux seuils de "similarité" th_a (respectivement th_c) pour décider si une sous-série est une occurrence de l'antécédent (respectivement du conséquent).

Nous ne souhaitons pas que l'utilisateur ait à définir lui même la taille des fenêtres et la valeur des seuils, car il s'agit de valeurs abstraites pour lui. À la place nous avons défini deux paramètres interprétables pour l'utilisateur : p_{window} et p_{th}, qui permettent au travers de deux fonctions de calculer les paramètres de taille de fenêtre et les paramètres de seuil.

Le paramètre p_{window} permet de définir w_a et w_c de la manière suivante :

$$w_a = p_{window} * |A|$$

$$w_c = p_{window} * |C|$$

Ce paramètre permet de fixer la taille des fenêtres à partir de la taille du motif selon un facteur simple (ex : la fenêtre fait deux fois la taille du motif ou sa moitié). Dans le cas

de la distance euclidienne, p_{window} est fixé à 1 car la distance euclidienne ne peut être calculée qu'entre des séries temporelles de même taille.

Le paramètre p_{th} permet de fixer th_a et th_c. Soit $\mathcal{D}_{ant}$ l'ensemble des distances calculées entre l'antécédent et toutes les sous-séries de taille w_a dans T. Le seuil de similarité th_a est la distance dans $\mathcal{D}_{ant}$ qui permet de séparer les p_{th}% plus petites distances dans $\mathcal{D}_{ant}$ des autres. Pour le calculer, nous extrayons un second ensemble $\mathcal{D}'_{ant}$ tel que :

$$\mathcal{D}'_{ant} = \{d_i \in \mathcal{D}_{ant} | d_i < p_{th} * |\mathcal{D}_{ant}|\}$$

Nous utilisons la distance maximale de cet ensemble comme seuil $th_a = max(\mathcal{D}'_{ant})$. Le calcul est le même pour th_c en remplaçant w_a et l'antécédent, par w_c et le conséquent. Pour calculer th_a et th_c à partir de p_{th}, il est donc nécessaire de calculer au préalable toutes les distances entre le motif et les sous-séries présentes dans T. Cette approche à l'avantage de permettre à l'utilisateur de sélectionner un seuil de distance à partir de la distribution des distances entre les sous-séries et le motif. Par exemple, si la distribution est bimodale est que le premier mode est composé de distances très faibles, il peut s'agir des distances d'un sous-ensemble de sous-séries que l'utilisateur peut considérer comme similaire. Il reste à l'utilisateur de sélectionner le quantile adéquate pour considérer uniquement ces sous-séries comme similaires au motif. Cependant, notre méthode nécessite que l'utilisateur sache analyser une distribution de distances entre un motif et des sous-séries, ce qui n'est pas toujours le cas.

L'Algorithme 1 présente le fonctionnement de la recherche d'occurrences de règles temporelles dans une série temporelle. De la ligne 3 à 6, nous avons l'étape de recherche des occurrences de l'antécédent dans T. Il s'agit de calculer la distance entre l'antécédent et un ensemble de sous-séries dans T. À partir de ces distances, nous pouvons calculer le seuil th_a qui est utilisé pour conserver seulement les occurrences de l'antécédent parmi toutes les sous-séries de T comparées avec l'antécédent (ligne 10). Généralement, lorsqu'une occurrence est détectée à une position alors les sous-séries à proximité sont elles aussi considérées comme des occurrences. En effet, les sous-séries qui sont à proximité d'une occurrence ont une distance avec l'antécédent qui est proche de celle de l'occurrence, et par conséquent qui peut être inférieure au seuil. Ces occurrences sont dites triviales [Lin+02] et ne doivent pas être prises en compte dans les occurrences de l'antécédent. Pour résoudre ce problème, lorsqu'une plage de sous-séries successives sont toutes considérées comme des occurrences, nous conservons l'occurrence ayant la distance la plus petite avec l'antécédent. Si plu-

Algorithme 1 Recherche d'occurrences d'une règle temporelle dans une série temporelle

Entrées : $dist$ une mesure de distance ,
T une série temporelle,
$R = (A, C, \leq maxlag)$ une règle,
$maxlag$ qui correspond à une contrainte de temps de durée max,
$p_{fenetre}$ et p_{th} deux valeurs réelles
Sortie : E_T^R l'ensemble des occurrences de R dans T

$E_T^R, E_T^A, E_T^C \leftarrow \emptyset$ // *E_T^A est l'ensemble des occurrences de l'antécédent de R dans T, respectivement E_T^C est l'ensemble des occurrences du conséquent de R dans T*
$w_a, w_c \leftarrow calcul_de_la_taille_des_fenetres(R, p_{window})$
$D_A, E_T^A \leftarrow calcule_distance_et_sous_serie(w_a, R.antecedent, T)$ // *l'ensemble des distances calculées avec l'antécédent*
$th_a \leftarrow calcul_seuil_similarite(D_A, p_{th})$
$E_T^A \leftarrow supprimer_occurrences_pas_similaires(E_T^A, th_a)$
$E_T^A \leftarrow supprimer_occurrences_triviales(E_T^A)$
$D_C \leftarrow \emptyset$ // *l'ensemble des distances calculées avec le conséquent*
pour tout $(ss_a, d_a, f_a) \in E_T^A$ **faire**
    $Dtemp_C, Etemp_T^C \leftarrow calcule_distance_et_sous_serie(w_c, R.consequent, T[f_a{:}f_a{+}maxlag])$
    $D_C \leftarrow D_C \cup Dtemp_C$
    $E_T^C \leftarrow E_T^C \cup Etemp_T^C$
fin pour
$th_c \leftarrow calcul_seuil_similarite(D_C, p_{th})$
pour tout $(ss_a, d_a, f_a) \in E_T^A$ **faire**
    $SE_T^C \leftarrow \{(dist_x, (ss_c, d_c, f_c)) \in E_T^C \mid dist \leq w_c \; et \; f_a \leq d_c \leq f_a + maxlag\}$
    si $SE_T^C \neq \emptyset$ **alors**
        $c \leftarrow \underset{x \in SE_T^C}{argmin}\ dist_x$
        $E_T^R.ajoute(((ss_a, d_a, f_a), (ss_c, d_c, f_c)))$
    fin si
fin pour
retourne E_T^R

sieurs occurrences ont la même distance, on extrait la première. Cette opération est réalisée dans la fonction à la ligne 6.

À présent que les occurrences de l'antécédent sont trouvées, la recherche des conséquents peut commencer. De la ligne 11 à 19, l'algorithme calcule le seuil de similarité pour les occurrences du conséquent. Puis de la ligne 20 à 25, l'algorithme énumère les occurrences de l'antécédent à la recherche de la meilleure occurrence du conséquent pour créer une occurrence de la règle R. Cette recherche locale consiste à calculer la distance entre le conséquent et les sous-séries de taille w_c présentes dans la période définie par le $maxlag$. Si aucune occurrence du conséquent n'est trouvée, alors l'occurrence de la règle n'est pas créée. Sinon une occurrence de la règle est créée avec l'occurrence du conséquent dont la distance est la plus petite avec C. L'ensemble des occurrences de la règle est ensuite retourné.

Lors de la recherche des occurrences de l'antécédent et du conséquent dans la série temporelle, nous utilisons une fonction $calcule_distance_et_sous_serie$ pour extraire l'ensemble des sous-séries E_T^A et E_T^C, dont les distances sont calculées avec A et C, et enregistrées dans D_A et D_C.

et calculer leur distance avec ces derniers. Cette fonction est différente selon les deux mesures de distance élastiques que nous avons sélectionnées (DTW et ssDTW). L'Algorithme 2 présente la méthode dédiée à la mesure DTW et réciproquement l'Algorithme 3, pour la mesure ssDTW. Dans la mesure DTW, l'apport par rapport aux mesures classiques consiste à pouvoir calculer des distances entre un motif (antécédent ou conséquent) et des sous-séries de taille différente de celle du motif (ce que ne peut pas faire la distance euclidienne par exemple). Dans la mesure ssDTW, il n'y a plus besoin de définir de taille de fenêtre pour l'antécédent et le conséquent grâce au multi-alignement (tel illustré dans la Figure 4.6). Le choix entre ces deux mesures élastiques dépend de la taille des occurrences recherchées Si nous n'avons pas de contraintes sur la taille des occurrences à trouver, alors ssDTW permet de calculer avec une seule matrice l'ensemble des alignements grâce aux multi-alignements. A l'inverse, si nous souhaitons contraindre la taille des occurrences, alors il faut utiliser DTW avec la taille de fenêtre souhaitée. Par contre, cela nécessite de relancer l'Algorithme 2 pour chaque taille de fenêtre testée.

Algorithme 2 Calcul des distances et des sous-séries avec DTW

Entrées : w_m une taille de fenêtre,
M un motif,
T une série temporelle
Sortie : D_M l'ensemble des distances entre le motif M les sous-séries dans T,
E_T^M l'ensemble des sous-séries dans T comparées avec M

$E_T^M \leftarrow \emptyset$ // *E_T^M est l'ensemble des sous-séries dans T comparées avec M*
$D_M \leftarrow \emptyset$ // *l'ensemble des distances calculé avec le motif*
pour tout position p entre 0 et $|T| - w_m$ **faire**
    $ss_m \leftarrow$ la sous-série de taille w_m dans T à la position p
    $dist \leftarrow DTW(ss_m, M)$
    $D_M.ajoute(dist)$
    $E_T^M.ajoute((dist, (ss_m, p, p + w_m)))$
fin pour
retourne D_M, E_T^M

Algorithme 3 Calcul des distances et des sous-séries avec ssDTW

Entrées : M un motif,
T une série temporelle
Sortie : D_M l'ensemble des distances entre le motif M les sous-séries dans T,
E_T^M l'ensemble des sous-séries dans T comparées avec M

$E_T^M \leftarrow \emptyset$ // *E_T^M est l'ensemble des sous-séries dans T comparées avec M*
$\mathcal{M}_M \leftarrow ssDTW(M, T).recuperer_matrice()$ // *la matrice calculée par ssDTW*
$D_M \leftarrow \mathcal{M}_M.recuperer_derniere_ligne()$ // *récupérer l'ensemble de distance $Edist$*
$E_T^M \leftarrow extraire_sous_serie(\mathcal{M}_M, T)$
retourne D_M, E_T^M

4.3 Expérimentations

Cette section est consacrée à l'évaluation de notre méthode de recherche d'occurrences élastiques de règles temporelles dans des séries temporelles issues de systèmes en condition réel. Notre objectif est de montrer que l'utilisation de mesures de distance élastiques améliore la recherche d'occurrences de règles temporelles dans ce type de séries temporelles. Pour cette évaluation, nous utilisons des séries temporelles issues issues de systèmes industriels.

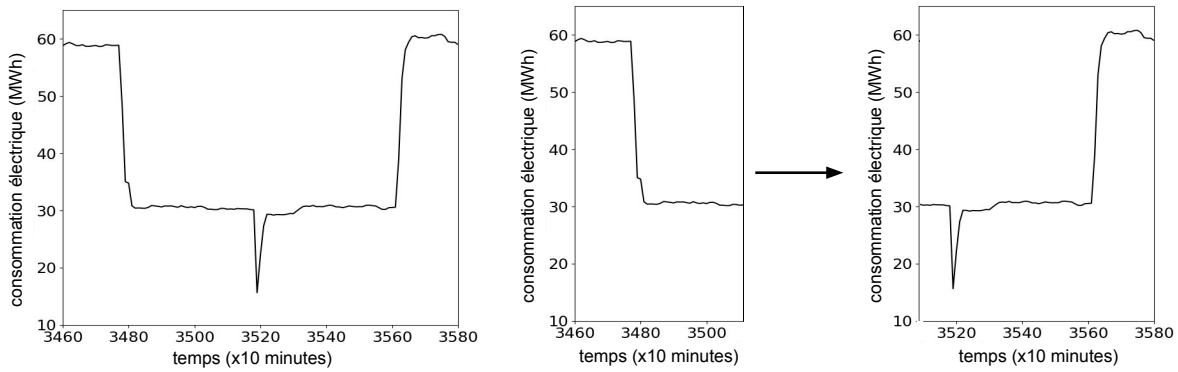
4.3.1 Protocole expérimental

Dans cette expérimentation, nous utilisons une série temporelle fournie par Energiency qui représente l'évolution de la consommation électrique d'un ensemble de machines dans une usine. Il s'agit d'un cas réel, soumis à de nombreux facteurs environnementaux et humains, et qui présente des comportements réguliers. La série temporelle fait 26.253 points de données avec une fréquence d'acquisition de 10 minutes, ce qui représente 6 mois d'acquisition.

Nous comparons trois mesures de distance sur leur capacité à retrouver les occurrences d'un ensemble de règles temporelles dans la série temporelle : la distance Euclidienne, la distance DTW et la distance ssDTW. Pour réaliser cette expérimentation nous avons besoin de définir un ensemble de règles et de marquer leur occurrences dans la série temporelle.

Avec un expert de Energiency, nous avons sélectionné un motif présent dans la série temporelle. La génération d'une règle temporelle consiste a séparer le motif en deux à une position donnée, le sous-motif gauche devient l'antécédent et le sous-motif droit le conséquent. Dans cette expérimentation, nous générons un ensemble de 20 règles temporelles en découpant le motif à 19 positions uniformément réparties sur le motif. Nous générons plusieurs règles pour évaluer nos mesures sur différentes tailles d'antécédent et de conséquent. Il s'agit de la procédure utilisée dans [Sho+15] pour générer des règles candidates. La Figure 4.7 présente le motif que nous utilisons et une des règles temporelles générée à partir de celui-ci.

Une fois les règles générées, nous devons marquer leurs occurrences dans les séries temporelles. Comme cette étape est à réaliser manuellement, il semble difficile de pouvoir estimer la position exacte de début de l'occurrence d'une règle. À la place,



(a) un motif (b) une règle

FIGURE 4.7 – Une règle (b), générée à partir du $8^{ième}$ point de séparation dans le motif (a).

un intervalle de temps pour le début de l'occurrence est défini comme illustré dans la Figure 4.8. Nous jugeons qu'une occurrence de règle temporelle est valide si son l'occurrence de son antécédent commence dans cet intervalle.

Pour les règles générées à partir des points de séparation 1, 2, 3, 18 et 19, nous avons annoté dans la série temporelle 48 intervalles, et 65 intervalles pour les règles restantes.

Nous testons les valeurs suivantes de paramètres pour la recherche :

— $maxlag \in \{0, 120, 240, 720\}$

— $p_{th} \in \{5, 10, 15, 20\}$

— selon la mesure de distance, différentes valeurs de p_{window} sont testées : une $p_{fenetre} = 1$ pour la distance euclidienne (une sous-série doit forcément avoir la taille du motif recherché), $p_{fenetre} \in \{50, 100, 125\}$ pour DTW et aucune valeur pour ssDTW car cette dernière ne nécessite pas de fenêtre glissante pour la recherche des sous-séries (cf. Section 4.2.2).

4.3.2 Résultats

Cette section est dédiée à la présentation des résultats des expérimentations. Nous comparons dans un premier temps les résultats des trois mesures de distances. Puis nous évaluons l'impact de la taille de la fenêtre glissante sur la recherche d'occurrences avec la mesure DTW par rapport à ssDTW.

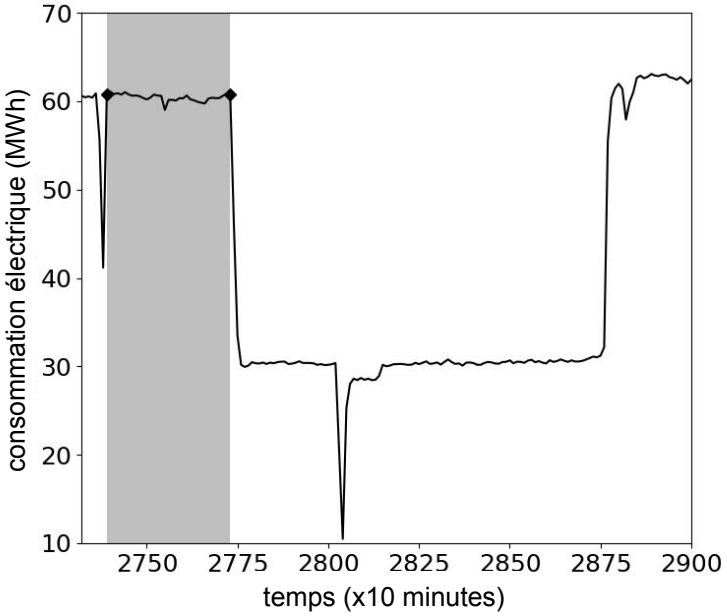


FIGURE 4.8 – Un intervalle (en gris) signifiant où l'occurrence de l'antécédent d'une règle temporelle doit commencer pour être valide.

Performances selon la mesure de distance

Dans cette expérimentation, nous comparons les performances des trois mesures de distance proposées. Nous calculons la précision et le rappel des occurrences capturées selon chaque règle temporelle et chaque mesure de distance. La Figure 4.9 présente la précision opposée au rappel pour les trois mesures de distances évaluées. Nous avons entouré d'une enveloppe convexe les résultats des 20 règles pour chaque mesure de distance, pour bien percevoir la dispersion des performances.

Sur quelques règles, nous voyons que DTW se trompe moins souvent que ssDTW (précision plus élevée pour 4 règles temporelles) mais en général, DTW et la distance euclidienne capturent moins d'occurrences annotées que ssDTW (un rappel de ssDTW sur toutes les règles).

Dans les résultats présentés, la taille de fenêtre pour la distance euclidienne et pour DTW est la même ($p_{fenetre} = 1$, c'est-à-dire que les occurrences doivent avoir la même taille que les motifs). Or, nous voyons que les résultats sont meilleurs pour la distance DTW que pour la distance euclidienne. Comme la taille de fenêtre est la même, la seule hypothèse qui reste pour expliquer cette amélioration est le fait que la distance DTW aligne les sous-séries et les motifs recherchés. Cela met en évidence

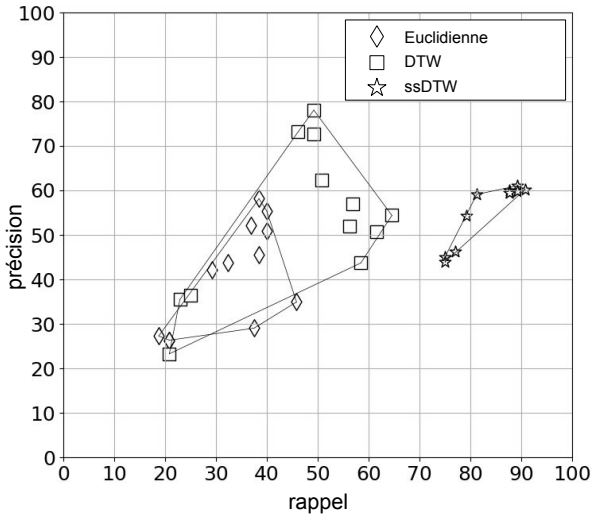


FIGURE 4.9 – Précision et rappel pour chaque règle temporelle et chaque mesure de distance.

que la prise en compte de la variabilité des occurrences peut être nécessaire dans des séries temporelles issues de systèmes réels.

Dans la section 4.9, nous avons vu que la mesure DTW pouvait être utilisée dans la recherche d'occurrence de règles avec différentes tailles de fenêtre. Comme ici, nous n'avons testé qu'une seule taille de fenêtre, la question est de savoir si la prise en compte d'autres tailles de fenêtres permettrait à DTW d'égaler ssDTW voir de le dépasser. La partie suivante est dédiée à cette analyse.

Performance de DTW selon différentes tailles de fenêtre

La figure 4.10 présente les résultats de ssDTW et de DTW en faisant varier la valeur de $p_{fenetre}$. Nous voyons que la taille de la fenêtre à un impact important sur les performances de la distance DTW. Si la taille de fenêtre est trop grande, nous ratons beaucoup plus d'occurrences annotées (le rappel est plus faible pour $p_{fenetre} = 1.25$). Pour certaines règles, une taille de fenêtre permet de capturer plus d'occurrences mais au prix de plus d'erreurs (sur l'ensemble des règles, la précision est plus faible pour $p_{fenetre} = 0.5$ mais le rappel est meilleur). La grande dispersion des résultats pour la distance DTW indique que selon la taille de la fenêtre, les occurrences de certaines règles sont mieux capturées que d'autres. Or, chaque règle est différente selon la taille de son antécédent et de son conséquent. Par conséquent, malgré la capacité d'aligne-

ment, cela nous dit qu'il n'existe pas une taille de fenêtre universelle pour toutes les règles et que cette dernière doit être paramétrée au cas par cas. Cette hypothèse est confirmée par les performances de ssDTW qui sont moins dispersées et meilleures. La seule différence entre DTW et ssDTW est la capacité de cette dernière à n'avoir aucune contrainte de taille d'occurrences. Cela semble indiquer que pour être performant dans la recherche d'occurrences de règles dans des séries temporelles réelles, il est préférable que la mesure de distance soit capable de capturer des occurrences de tailles différentes.

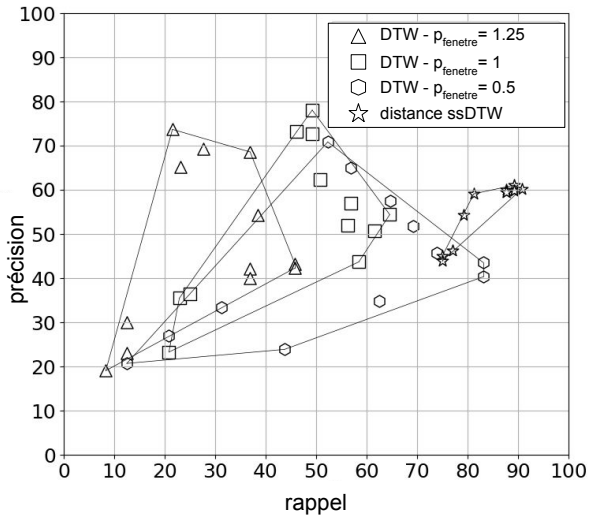


FIGURE 4.10 – Précision et rappel pour chaque règle temporelle avec la mesure de distance ssDTW et différentes versions de DTW avec différentes tailles de fenêtre.

4.4 Conclusion

Nous avons vu que le comportement régulier d'un système réel peut produire des sous-séries jugées similaires mais légèrement déformées. La prise en compte de cette variabilité est déterminante pour la découverte de connaissances dans les séries temporelles tirées de systèmes réels. Jusqu'à présent aucune méthode de découverte de règles temporelles, reposant uniquement sur la représentation brute de séries temporelles, a été développée. Notre proposition consiste à utiliser une mesure de distance élastique connue pour sa capacité à pouvoir identifier des sous-séries similaires malgré de légères déformations. Nous avons présenté deux mesures de distance élastiques

existantes pour cette tâche : la distance DTW et la distance ssDTW, ainsi que leur intégration dans un algorithme de recherche d'occurrences de règles temporelles.

Il ressort que la distance ssDTW est plus adaptée pour la recherche d'occurrences de règles temporelles dans des séries temporelles réelles car elle permet d'aligner des sous-séries entre elles et de capturer des occurrences de tailles différentes.

TROISIÈME PARTIE

Classification de séries temporelles et interprétabilité

CHAPITRE 5

CLASSIFICATION DE SÉRIES TEMPORELLES PAR SHAPELETS ALÉATOIRES ET LOCALISÉES

5.1 Motivations

r

Dans la section 3.2.1, nous avons présenté un ensemble de classifieurs de séries temporelles considérés comme interprétables. Ces classifieurs peuvent être regroupés en deux approches. Les premiers [KR05 ; Hir77 ; CN04 ; Mar07 ; SAD13] utilisent la similarité entre les séries temporelles entières. Les seconds [YK09 ; KR13] reposent sur l'utilisation de shapelets. Pour rappel, les shapelets sont des sous-séries permettant de discriminer la classe de séries temporelles selon leur présence ou leur absence dans ces dernières.

Récemment, Bagnall et al. [Bag+17] ont montré que les classifieurs interprétables existants ont une précision moins bonne que les classifieurs non-interprétables. Cependant, comme nous l'avons dit en introduction, l'absence d'interprétabilité dans un classifieur peut freiner les utilisateurs finaux car ils ne savent pas sur quoi se basent les décisions du classifieur. Le compromis entre la précision et l'interprétabilité est un problème connu [BB94 ; JNK04 ; Gui+19a].

Pour résoudre ce problème, nous proposons un classifieur à base de shapelets qui reste interprétable tout en ayant une précision comparable aux méthodes de l'état de l'art. Parmi les classifieurs les plus performants, ceux à base de shapelets ont perdu en interprétabilité (cf. section 3.2.2). Nous nous sommes donc interrogés sur l'existence d'un autre attribut qui puisse être extrait des shapelets, qui soit discriminant et compréhensible pour un utilisateur. Parmi les attributs possibles, un en particulier a

attiré notre attention, la position des shapelets, et ce pour trois raisons :

1. c'est un attribut qui peut être facilement extrait d'une série temporelle ;
2. la position d'une shapelet a du sens pour l'utilisateur ;
3. un travail récent [Tav+17] a montré que la position de caractéristiques extraites des séries temporelles peut améliorer leur classification.

Le modèle que nous proposons intègre la prise en compte de la position des shapelets dans la classification de séries temporelles. Cependant, nous pensons que la prise en compte des positions des shapelets ne doit pas être systématique. En particulier, nous souhaitons que les deux conditions suivantes soient respectées :

— **condition 1** : notre modèle prend en compte la position d'une shapelet seulement si sa présence a un impact sur la classification ;
— **condition 2** : notre modèle peut ignorer la position d'une shapelet si sa seule présence est suffisante pour la classification.

Notre modèle *Localized Random Shapelet* [Gui+19b] (LRS), apprend un classifieur interprétable qui est une régression logistique minimisant une fonction de perte dont la régularisation respecte les deux conditions énoncées. La section 5.2 présente notre modèle de classification. La section 5.3 est consacrée à l'extraction d'une explication de classification à partir de notre méthode. Enfin la section 5.4 compare et analyse la précision de notre modèle par rapport aux méthodes de l'état de l'art, et l'influence des positions des shapelets dans la classification.

Ce travail a été mené en collaboration avec Romain Tavenard [1] et Simon Malinowski [2].

5.2 Modèle LRS

Dans la section 2.3.1, nous avons présenté l'approche de classification de séries temporelles à partir de shapelets. Pour rappel, une shapelet $S = (s_1, \ldots, s_l)$ est une sous-série de taille l qui permet de discriminer la classe de séries temporelles selon sa présence ou son absence des séries temporelles. La classification à base de shapelets d'une série temporelle $T = (t_1, \ldots, t_L)$ de taille L, repose sur la distance entre S et T

1. équipe OBELIX, IRISA, Rennes
2. équipe LINKMEDIA, IRISA, Rennes

qui se calcule de la manière suivante :

$$d_T^S = \min_{1 \leq j \leq L-l+1} \sqrt{\sum_{i=1}^{l} (s_i - t_{i+j-1})^2}. \tag{5.1}$$

Enfin, soit un ensemble $\mathcal{D} = \{S_1, \ldots, S_K\}$ de K shapelets, la transformation à base de shapelets consiste a représenter T en un vecteur $\mathbf{v_T} = (d_T^{S_1}, \ldots, d_T^{S_K})$ avec $d_T^{S_k}$ la distance entre la shapelet S_k et T , pour tout $1 \leq k \leq K$.

5.2.1 Modèle LRS

La représentation par transformation à base de shapelets prend en compte uniquement les distances entre les shapelets et les séries temporelles, mais jamais les positions où les shapelets correspondent le mieux dans les séries temporelles (où la distance est minimum entre une shapelet et une série temporelle). Nous considérons que la position de la shapelet peut être utilisée pour améliorer la précision de classification de séries temporelles. Dans notre modèle *Localized Random Shapelet* (LRS), nous proposons d'étendre la transformation à base de shapelets avec la position des shapelets en plus de la distance. Dans LRS, nous tirons K shapelets uniformément et aléatoirement à partir de toutes les sous-séries possibles dans les séries d'apprentissage. La valeur K est fixée par l'utilisateur. Chaque shapelet est associée à deux attributs. Le premier est le même que celui utilisé dans la représentation à base de shapelets, c'est-à-dire la distance d_T^S entre T et S comme défini dans l'Équation 5.1. Le second attribut correspond à la position où cette distance est atteinte. La position se calcule de la manière suivante :

$$p_T^S = \operatorname*{argmin}_{1 \leq j \leq L-l+1} \sqrt{\sum_{i=1}^{l} (s_i - t_{i+j-1})^2}. \tag{5.2}$$

Avec ce nouvel attribut, nous pouvons étendre la représentation de T par $\mathcal{S}$ comme le vecteur $mathbfv_T = (d_T^{S_1}, p_T^{S_1}, \ldots, d_T^{S_K}, p_T^{S_K})$ avec $p_T^{S_h}$ la position de la shapelet d'indice h dans T pour tout $1 \leq h \leq K$.

Soit $D = \{T_1, \ldots, T_N\}$, un ensemble de N séries temporelles. Cet ensemble est représenté par notre modèle sous la forme d'une matrice $\mathbf{X}$ dont chaque ligne est la représentation d'une série temporelle :

$$\mathbf{X} = \begin{pmatrix} d_{T_1}^{S_1} & p_{T_1}^{S_1} & \cdots & d_{T_1}^{S_K} & p_{T_1}^{S_K} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{T_N}^{S_1} & p_{T_N}^{S_1} & \cdots & d_{T_N}^{S_1} & p_{T_N}^{S_K} \end{pmatrix}$$

Une fois cette matrice calculée, elle est utilisée pour apprendre un classifieur standard de type régression logistique comme illustré dans la Figure 5.1. Ce modèle prend en entrée un vecteur de dimension $2 \times K$ (notre représentation d'une série temporelle selon K shapelets) et retourne les probabilités de classification pour chaque classe du jeu de données

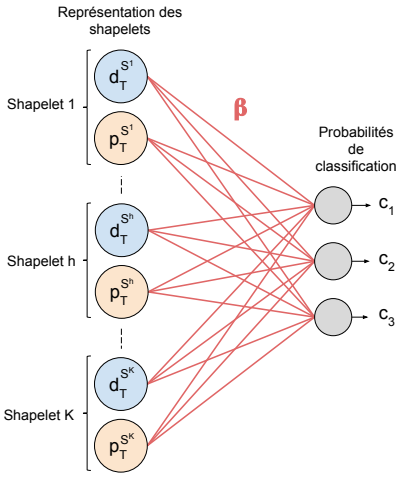


FIGURE 5.1 – Le modèle LRS. Les disques bleus indiquent les attributs de distance, tandis que les orange correspondent à ceux des positions.

5.2.2 Apprentissage du classifieur

Nous désignons l'ensemble des coefficients de notre régression logistique par β. Dans la pratique, l'approche classique pour apprendre un modèle de classification à base de régression logistique consiste à chercher β qui minimise la fonction de perte $\mathcal{L}$ suivante :

$$\mathcal{L}(y, \mathbf{X}, \beta) = \frac{1}{m} * (-y^T log(\mathbf{X}\beta) - (1-y)^T log(1 - \mathbf{X}\beta))$$

avec la matrice $\mathbf{X}$ définie précédemment et un vecteur cible $\mathbf{y}$ de m classes. Dans l'introduction du chapitre, nous avons énoncé deux conditions sur la prise en compte de

la position des shapelets dans la classification des séries temporelles. Ces conditions se traduisent par les propriétés suivantes attendues dans le classifieur final :

— **condition 1** : les coefficients associés à l'attribut de position β_p^h d'une shapelet h sont non-nuls seulement si ses coefficients associés à son attribut de distance β_d^h sont non-nuls ;
— **condition 2** : les coefficients β_p^h peuvent être non-nuls même si les coefficients β_d^h sont nuls.

La régularisation permet d'avoir un modèle avec de bonne capacité de généralisation. La suite est consacrée à la présentation de la méthode de régularisation que nous utilisons dans notre modèle LRS.

Régularisation de LRS

L'approche standard pour obtenir une solution parcimonieuse est de dériver une fonction de perte régularisée, comme proposée dans la régression *Lasso* [Tib96] :

$$arg\min_{\beta}(\mathcal{L}(y, \mathbf{X}, \beta) + \lambda\|\beta\|_1)$$

qui est le compromis entre la fonction de perte et une pénalisation des coefficients (élevée), avec $\lambda \in [0, 1]$ un paramètre pour ajuster la prise en compte de la pénalisation.

Cette première régularisation vérifie la condition 2 mais enfreint la condition 1 car il est possible pour une shapelet d'avoir son coefficient de position non-nul alors que celui de distance est nul. Une solution à ce problème est la régularisation nommée *Group-Lasso* [YL06] qui propose une solution basée sur des groupes d'attributs. Dans notre cas, un groupe représente les deux attributs associés à une shapelet comme illustré dans la Figure 5.2. Nous avons donc K groupes. La régularisation est la suivante :

$$arg\min_{\beta}(\mathcal{L}(y, \mathbf{X}, \beta) + \lambda \sum_{k=1}^{K} \sqrt{p_h}\|\beta^h\|_2)$$

avec X^h la sous-matrice de X dont les colonnes sont les attributs du groupe h, β^h les coefficients associés à ce groupe et p_h la taille de ce groupe.

Cette régularisation force les coefficients des attributs d'un groupe à être tous les deux à zéro ou bien non-nuls. Cette régularisation vérifie la condition 1, mais enfreint la

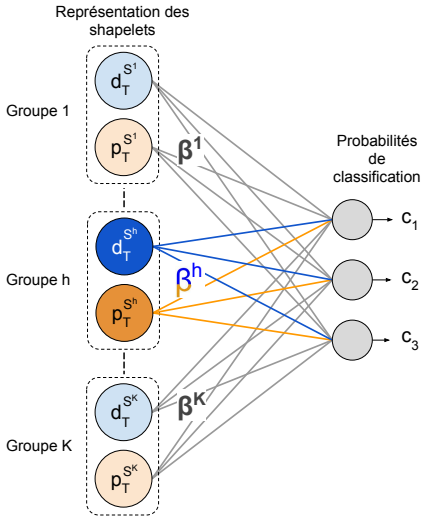


FIGURE 5.2 – Découpage du vecteur d'entrée en K groupes. Les disques bleus indiquent les attributs de distance, tandis que les orange correspondent à ceux des positions. Pour chaque shapelet, un groupe est formé dont les coefficients sont notés β^h (où h est l'indice de la shapelet).

condition 2. Notre intérêt est d'introduire de la parcimonie dans les groupes. La régularisation nommée *Sparse-Group Lasso* (SGL) [Sim+13] a proposé de combiner Lasso et Group-Lasso en une seule régularisation dans le but de combiner la parcimonie au niveau des attributs et celle au niveau des groupes. Cette régularisation est définie de la manière suivante :

$$arg\min_{\beta}(\mathcal{L}(y, \mathbf{X}, \beta) + (1-\alpha)\lambda \sum_{h=1}^{K} \sqrt{p_k}\|\beta^h\|_2) + \alpha\lambda\|\beta\|_1)$$

avec $\alpha \in [0, 1]$ un paramètre qui ajuste le compromis entre les deux régularisations. Cette régularisation vérifie la condition 2, mais enfreint elle aussi la condition 1, car elle autorise les coefficients des attributs de position à être non-nuls, peu importe les valeurs des coefficients de distance. Pour résoudre ce problème, nous avons donc proposé notre propre fonction de perte *Semi Sparse-Group Lasso* (SSGL) basée sur Sparse-Group Lasso et qui se définit de la manière suivante :

$$arg\min_{\beta}(\mathcal{L}(y, \mathbf{X}, \beta) + (1-\alpha)\lambda \sum_{k=1}^{K} \sqrt{p_k}\|\beta^k\|_2) + \alpha\lambda\|M_{ind}\beta\|_1)$$

où M_{ind} est une matrice diagonale indicatrice qui alterne des zéros et des uns. Les zéros sont associés aux coefficients de position des shapelets et les uns aux

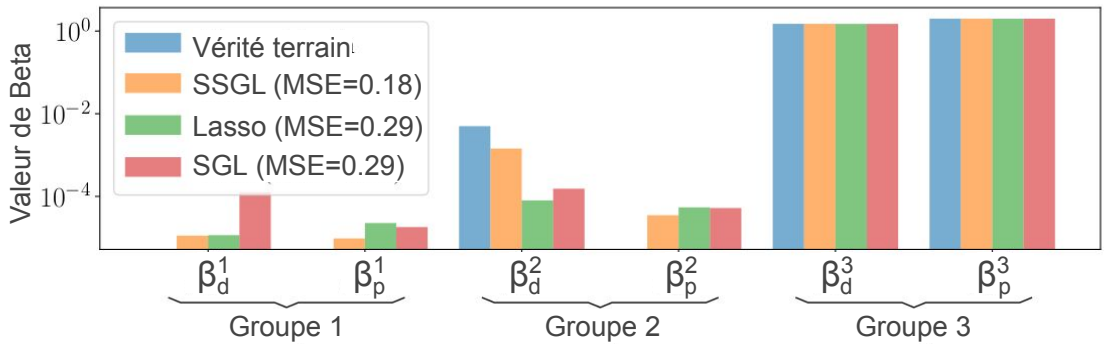


FIGURE 5.3 – Les coefficients appris en utilisant différents stratégies pour un problème de régression linéaire. Les coefficients à retrouver sont en bleus.

coefficients de distance. Ainsi nous nous assurons que si les coefficients associés à un groupe sont non-nuls alors seulement les coefficients associés à l'attribut de position peuvent être mis à zéro. Cette régularisation respecte les deux conditions que nous avons énoncées.

Exemple jouet : la régression linéaire structurée

Nous utilisons le modèle suivant pour générer un ensemble d'instances :

$$\mathbf{y} = \mathbf{X} \cdot \boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{5.3}$$

où $\mathbf{X}$ et ε sont tirées d'une loi normale centrée avec pour écart type 1 et 0.01 respectivement.

Pour cet exemple, nous créons un scénario avec trois shapelets en fixant à l'avance les coefficients $\boldsymbol{\beta} = \left[\beta_d^1, \beta_p^1, \beta_d^2, \beta_p^2, \beta_d^3, \beta_p^3\right]$ et seulement trois composants qui ne sont pas à zéro : β_d^2, β_d^3 et β_p^3 (voir Figure 5.3). Ces coefficients forment le scénario où la shapelet 1 ne contribue pas à la classification, la shapelet 2 contribue seulement par sa distance et la shapelet 3 contribue à la fois par sa distance et sa position. Une fois les coefficients fixés, nous calculons le vecteur cible y des instances.

Nous comparons trois stratégies de régularisation décrites précédemment : Lasso, SGL et notre régularisation SSGL. Cette comparaison consiste à leur faire apprendre β à partir de la matrice X et la cible y, puis à comparer l'erreur quadratique moyenne (MSE) entre les coefficients trouvés par les régularisations et ceux fixés. Pour apprendre β, nous utilisons la descente de gradient globale pour minimiser la fonction de

perte avec régularisation. Les résultats sont illustrés dans la Figure 5.3.

Nous voyons que SSGL surpasse SGL et Lasso en terme d'erreur quadratique moyenne (les valeurs de coefficients de SSGL en jaune sont les plus proches de ceux de la vérité terrain en bleu), montrant les bénéfices de prendre en compte la relation entre les attributs de position et de distance, dans le modèle.

5.3 Du modèle LRS à l'explication

Dans cette section nous présentons la façon d'extraire une explication des classifications de notre modèle LRS. La classification d'une série temporelle à partir de LRS, se découpe en deux étapes. Dans un premier temps, un vecteur de représentation de la série temporelle est calculé à partir des shapelets $\mathcal{S}$ extraites par LRS (cf. section 5.2.1). Ce vecteur est fourni au modèle LRS pour calculer la probabilité d'appartenance de la série temporelle à chaque classe du jeu de données. Dans ce calcul, les attributs calculés entre les shapelets et la série temporelle sont pondérés par l'ensemble de coefficients $\boldsymbol{\beta}$. Plus un coefficient est élevé et plus l'attribut associé est influent dans la probabilité qu'il permet de calculer. Ces coefficients peuvent donc être utilisés pour expliquer l'influence des shapelets dans les classifications du modèle LRS.

Plus précisement, deux explications peuvent être données à partir des coefficients $\boldsymbol{\beta}$:

— Quelles shapelets sont les plus influentes pour la classification ?

— Pour chacune de ces shapelets, est ce que sa position est décisive ?

Ces explications se basent sur le classement des shapelets selon leur coefficients dans $\boldsymbol{\beta}$. Pour cela, nous extrayons pour chaque shapelet $S_k \in \mathcal{S}$ les coefficients $\beta^k = \beta^k_d \cup \beta^k_p$ qui lui sont associés, avec β^k_d l'ensemble des coefficients associés à l'attribut de distance de la shapelet, et respectivement β^k_p ceux associés à l'attribut de position. À partir de ces coefficients, nous calculons trois valeurs par shapelet :

— $influence_{globale} = \|\beta^k\|_2$, la norme ℓ_2 des coefficients β^k, qui synthétise l'influence globale de la shapelet.

— $influence_{distance} = \|\beta^k_d\|_2$, la norme ℓ_2 des coefficients β^k_d, qui synthétise l'influence de la distance de la shapelet.

— $influence_{position} = \|\beta^k_p\|_2$, la norme ℓ_2 des coefficients β^k_p, qui synthétise l'influence de la position de la shapelet.

Une fois ces valeurs calculées, nous pouvons établir trois classements sur chacune des influences calculées. C'est à partir de ces classements que nous pouvons tirer des explications sur l'influence des shapelets dans la classification de séries temporelles par le modèle LRS.

La suite de cette section est consacrée à une mise en application de cette extraction d'explications du modèle LRS sur le jeu de données TWOPATTERNS issu de l'archive UCR & UEA [AK]. TWOPATTERNS est un jeu de séries temporelles synthétiques dans lequel chaque série temporelle est la succession de deux motifs séparés par du bruit. Ces deux motifs, nommés A et B, sont illustrés dans la Figure 5.4. Le problème de classification est constitué de quatre classes, correspondant à toutes les permutations possibles des motifs A et B pour les deux positions, comme montré dans la Figure 5.5. Pour l'apprentissage de LRS, nous tirons $K = 2000$ shapelets à partir des séries temporelles d'apprentissage, et nous utilisons une descente de gradient globale pour minimiser la fonction de perte avec SSGL.

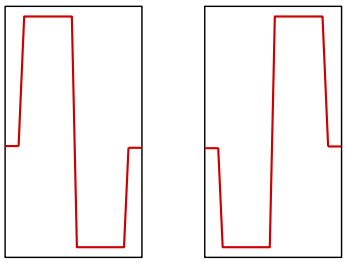
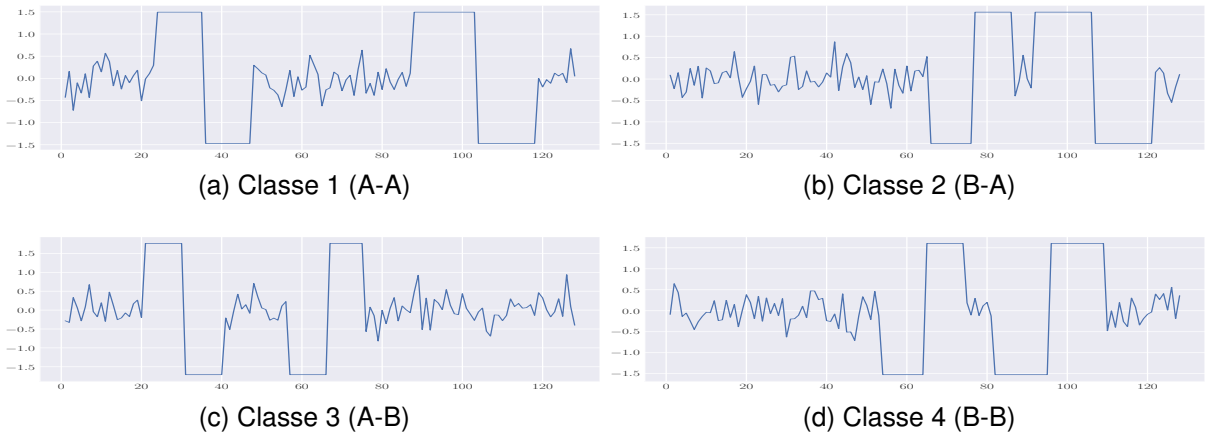


FIGURE 5.4 – Les motifs A et B.

L'extraction d'explications commence par le calcul des trois classements présentés précédemment. À partir du classement de l'influence globale des shapelets, nous extrayons les deux shapelets les plus influentes globalement, S^{n1} et S^{n2}. Ces shapelets sont illustrées dans la Figure 5.6. Nous présentons seulement les deux premières shapelets pour donner une idée de comment analyser l'influence.

Analyse de la première shapelet

Nous estimons qu'une shapelet est influente dans un des classements si elle est dans le top 1%. Dans notre cas, il s'agit donc des 20 premières shapelets puisque $K = 2000$. La première shapelet S^{n1} au classement global (Figure 5.7) est classée $4^{ème}$ selon l'influence de la position et $14^{ème}$ selon l'influence de la distance. Nous



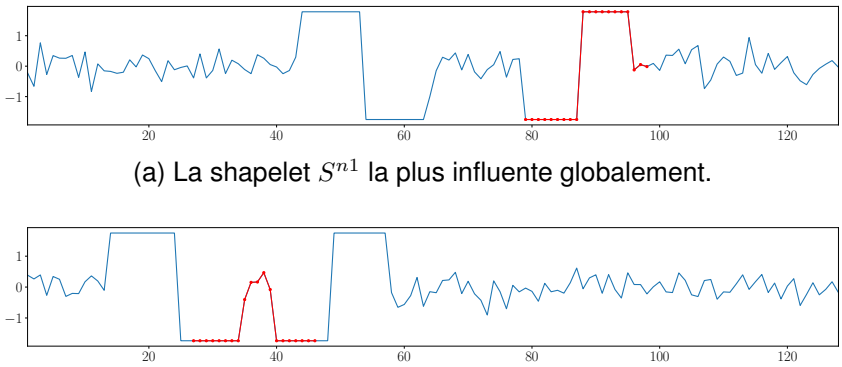
(a) Classe 1 (A-A)

(b) Classe 2 (B-A)

(c) Classe 3 (A-B)

(d) Classe 4 (B-B)

FIGURE 5.5 – Un exemple de chaque classe pour le jeu de données TWOPATTERNS.



(a) La shapelet S^{n1} la plus influente globalement.

(b) La shapelet S^{n2} la seconde plus influente globalement.

FIGURE 5.6 – Les deux shapelets les plus influentes surlignées dans leur série respectives.

pensons que la distance et la position de S^{n1} est influente dans la classification de séries temporelles par LRS.

Une shapelet influente dans LRS, est une shapelet qui a permis de discriminer les séries d'apprentissage. Par conséquent, nous nous attendons à ce que la shapelet S^{n1} soit discriminante avec les séries d'apprentissage selon sa position et sa distance.

Pour vérifier l'influence de la distance avec S^{n1} dans LRS, nous comparons les distributions des distances entre la shapelet S^{n1} et les séries d'apprentissage de chaque classe (Figure 5.8). Nous voyons que la distribution des valeurs de distances entre S^{n1} et les séries temporelles de la classe 1 (A-A) se distingue des autres distributions. Plus précisément, les valeurs de distance avec les séries de cette classe sont beaucoup plus élevées. Par conséquent, si la distance entre S^{n1} et une série temporelle

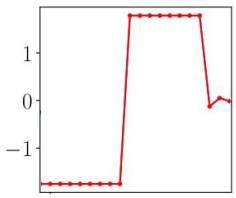


FIGURE 5.7 – La shapelet S^{n1}.

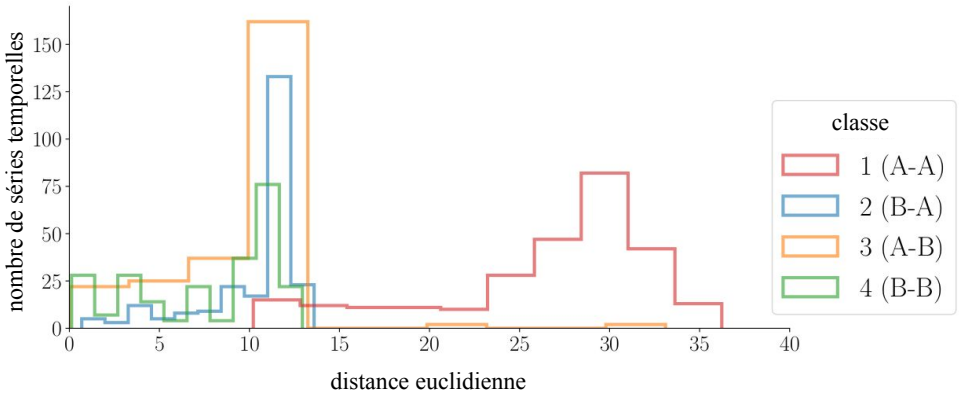


FIGURE 5.8 – Distribution, selon chaque classe, des distances entre la shapelet S^{n1} avec les séries d'apprentissage de jeu de données TWOPATTERNS.

est élevée alors elle a de grande chance d'appartenir à la classe 1. Il s'agit là d'une information discriminante. Cette discriminance peut s'expliquer à l'oeil nu. En effet, la shapelet S^{n1} ressemble au motif B et son absence dans une série (équivalent à une distance élevée avec la série) est possible seulement dans les séries où le motif B n'est pas présent, c'est-à-dire les séries de la classe 1 (A-A).

Pour vérifier l'influence de la position de S^{n1} dans LRS, nous comparons cette fois les distributions des positions de la shapelet S^{n1} dans les séries d'apprentissage de chaque classe (Figure 5.9). Nous observons que la distribution des positions pour la classe 3 (A-B) est la seule qui commence autour de la position 30. Par conséquent, si la shapelet S^{n1} est positionnée en début d'une série temporelle (avant la position 30) alors elle a de grande chance de ne pas appartenir à la classe 3 (A-B). Il s'agit là d'une information qui est discriminante. Le même raisonnement peut être réalisé pour la classe 2 (B-A). En effet, nous observons qu'il s'agit de la seule distribution qui s'arrête avant la position 90. Par conséquent, si la shapelet S^{n1} est positionnée en fin d'une série temporelle (au delà de la position 90) alors elle a de grande chance de ne pas appartenir à la classe 2 (B-A). Là aussi nous pouvons l'expliquer en observant les

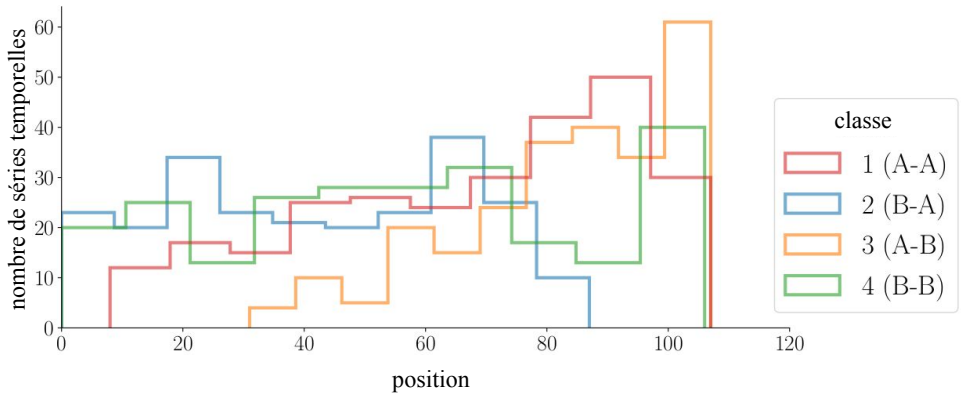


FIGURE 5.9 – Distribution, selon chaque classe, des positions de la shapelet S^{n1} dans les séries d'apprentissage de jeu de données TWOPATTERNS.

séries temporelles. En effet la shapelet S^{n1} qui ressemble au motif B ne peut pas être présente en fin de séries temporelle dans les séries de classe 2 (B-A) car cette position est occupée par le motif A. Inversement, la shapelet S^{n1} ne peut pas être présente en début de série temporelle dans les séries de la classe 3 (A-B) car elles commencent par le motif A.

Cependant, la shapelet S^{n1} utilisée seule est insuffisante pour trouver la classe de n'importe quelle série temporelle (à l'exception de la classe 1 (A-A) si la distance avec la shapelet est élevée). C'est la raison pour laquelle les modèles de classification à base de shapelets reposent sur un ensemble de shapelets pour mettre en commun les caractéristiques discriminants de chaque shapelet.

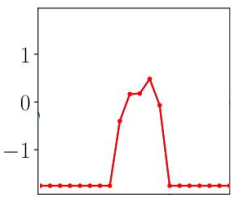


FIGURE 5.10 – La seconde shapelet dans le classement global S^{n2}.

Analyse de la seconde shapelet

La seconde shapelet S^{n2} au classement global (Figure 5.10) est classé 2^{nde} selon l'influence de la distance et $217^{ème}$ selon l'influence de sa position (soit au delà de 10% des shapelets utilisées). Cette shapelet est donc influente seulement par sa distance.

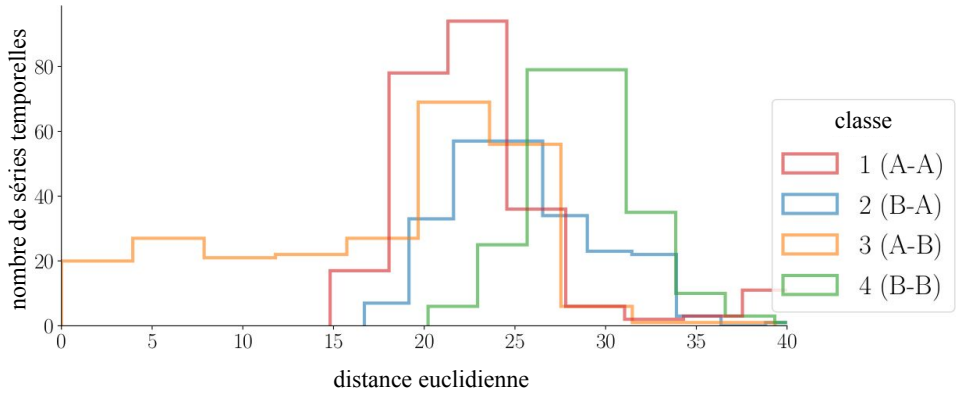


FIGURE 5.11 – Distribution, selon chaque classe, des distances entre la shapelet S^{n2} avec les séries d'apprentissage de jeu de données TWOPATTERNS.

Comme pour la shapelet précédente, nous allons vérifier l'influence de la distance avec S^{n2} dans LRS, en comparant les distributions des distances entre la shapelet S^{n2} et les séries d'apprentissage de chaque classe (Figure 5.11).

En regardant de plus près, nous remarquons que cette sous-série peut être présente uniquement entre un motif A suivi d'un motif B, c'est à dire dans une série temporelle de la classe 3 (A-B). Par conséquent, une distance faible entre la shapelet S^{n2} et une série temporelle indique qu'elle appartient à la classe 3 (A-B). Cette hypothèse est vérifiée dans la Figure 5.11 car la distribution des distances entre cette shapelet et les séries de classe 3 est la seule qui soit inférieure à la distance 15.

5.4 Expérimentations

Cette section est consacrée aux expérimentations de notre méthode LRS. La section 5.4.1 introduit le protocole expérimental. La section 5.4.2 présente les performances de classification de notre modèle contre les classifieurs à base de shapelets de l'état de l'art. Enfin la section 5.4.3 analyse l'influence de la position des shapelets dans la classification des séries temporelles.

5.4.1 Protocole expérimental

Le modèle LRS proposé est comparé à trois modèles de référence dans la classification de séries temporelles à base de shapelets : *Shapelet Transform* (ST) [Lin+12],

Learning Shapelet (LS) [Gra+14] et *Fast Shapelet* (FS) [KR13]. Nous nous sommes restreints aux modèles à base de shapelets pour voir dans un premier temps si notre modèle pouvait rivaliser avec les modèles basés sur la même stratégie de classification. Par ailleurs, certains de ces modèles sont classés parmi les meilleures méthodes de classification de séries temporelles, en particulier ST [Bag+17].

Parmi ces trois modèles nous considérons seulement FS comme étant interprétable (cf. section 3.2.1). En effet, ST est un classifieur ensembliste appris sur la représentation des séries temporelles par les shapelets. Comme nous l'avons présenté dans la section 3.2.2, un classifieur ensembliste est non-interprétable. Par ailleurs, nous estimons que LS est non-interprétable, car ce modèle (qui utilise aussi la représentation des séries temporelles par des shapelets) génère des shapelets synthétiques optimales pour la classification. Mais ces shapelets perdent leur interprétabilité pour l'utilisateur car elles ne sont plus associées à un comportement réel (ce qui est toujours le cas quand les shapelets sont extraites dans les séries temporelles d'apprentissage).

Les expérimentations sont réalisées sur 69 jeux de données du répertoire UCR & UEA de classification de séries temporelles [AK], un ensemble de jeux de données de référence dans le domaine de la classification de séries temporelles. Une description de ces datasets peut être trouvée dans [Bag+17]. Les performances de classification des modèles de référence ST, LS et FS ont été récupérées à partir du site web associé au répertoire des jeux de données.

Pour chaque jeu de données, nous tirons cinq ensembles de $K = 2000$ shapelets de manière aléatoire et uniforme à partir des séries temporelles d'apprentissage. Pour le choix de la taille des shapelets, nous suivons la procédure décrite dans [Gra+14] qui consiste à définir un intervalle de taille de shapelet dans lequel les shapelets tirées doivent se répartir équitablement. Les séries temporelles d'apprentissage sont ensuite représentées par les shapelets selon la procédure présentée dans la section 5.2.1.

Dans toutes les expérimentations présentées dans cette partie, nous utilisons une descente de gradient globale sur la fonction de perte avec SSGL. Cette stratégie a été appliquée avec succès dans [Sca+17]. Nous utilisons l'optimiseur RMSPROP [TH12] avec un taux d'apprentissage de 0.001. La régularisation SSGL utilise deux hyper-paramètres : λ qui contrôle la force de la régularisation et α qui contrôle le compromis entre les termes du Lasso et du Group Lasso. Nous ajoutons à ces hyper-paramètres, un paramètre de décrochage (*dropout* en anglais) pour éviter les optima locaux. Ce paramètre s'assure que tous les attributs du modèles soient explorés en sélection-

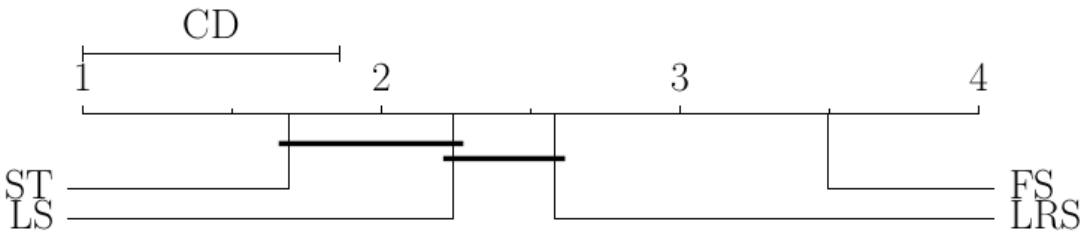


FIGURE 5.12 – Le diagramme des différences critiques.

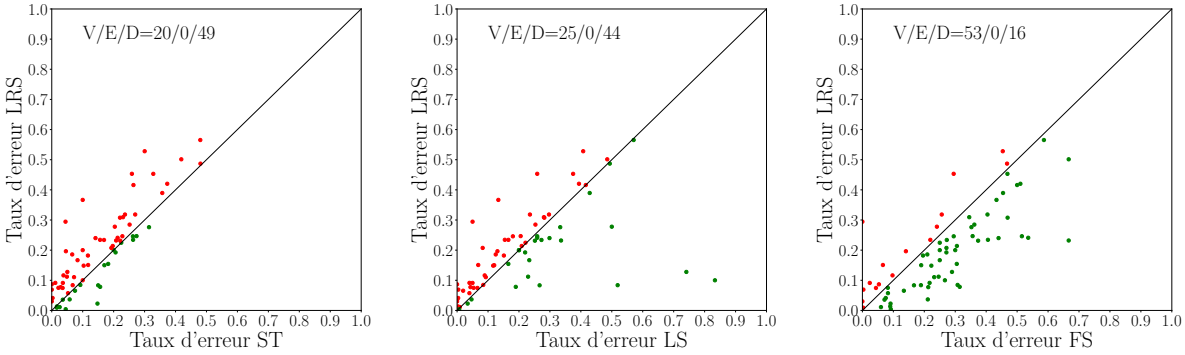
nant aléatoirement à chaque itération de l'apprentissage, un pourcentage des attributs à ne pas utiliser. Nous réalisons une validation croisée pour λ sur l'ensemble $\{10^{-1}, 10^{-2}, \ldots, 10^{-7}\}$, pour α sur l'ensemble $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ et pour le décrochage sur $\{0., 0.3, 0.5, 0.7\}$.

5.4.2 Performance contre les méthodes de l'état de l'art

La première étape de cette évaluation consiste à classer les modèles selon leur précision sur chaque jeu de données. Ce classement est utilisé pour calculer le diagramme des différences critiques de la Figure 5.12, qui présente le rang moyen des classifieurs sur les 69 jeux de données. La différence critique (la valeur CD) représente la différence de rang moyen qui doit séparer deux classifieurs pour que ceux-ci soient considérés comme significativement différents selon leur rang sur les 69 jeux de données. Les barres noires horizontales représentent les ensembles de classifieurs dont leur différence de rang n'est pas significative (inférieure à la valeur CD).

Nous voyons que parmi les classifieurs interprétables notre méthode LRS est significativement meilleure que FS, alors que son classement moyen n'est pas significativement différent de celui de LS. Cependant, nous constatons que les modèles non-interprétables ST et LS, sont mieux classés en moyenne que les modèles interprétables LRS et FS.

Pour creuser ces premières conclusions, nous avons comparé les taux d'erreur de LRS avec chaque classifieur de l'état de l'art (Figure 5.13). Les résultats confirment que les modèles non-interprétables sont plus précis en moyenne que notre modèle LRS, et que LRS est plus précis en moyenne que FS. Cependant, si nous regardons plus en détail le classement des modèles par jeu de données, LRS est classé premier sur 9 jeux, ST sur 39, LS sur 14 et FS sur 4. Il n'existe donc pas, parmi ces quatre modèles, un qui soit universellement plus précis que les autres. Cette observation



(a) comparaison avec ST (b) comparaison avec LS (c) comparaison avec FS

FIGURE 5.13 – Comparaison des taux d'erreurs entre LRS et chaque classifieur de l'état de l'art sur les 69 jeux de données. Les valeurs entre les barres obliques sont respectivement les nombres de jeux de données où LRS a une meilleure précision que le classifieur comparé (V), où la précision est égale (E), et où la précision est moins bonne (D).

a déjà été faite dans d'autres travaux [Bag+17 ; Faw+19] suggérant qu'un classifieur serait adapté pour seulement un type de séries temporelles.

5.4.3 Analyse de l'impact de la position des shapelets dans la classification

Cette partie a pour but d'analyser l'influence de la prise en compte des positions des shapelets dans la classification de séries temporelles. Pour cela, nous entraînons un nouveau classifieur identique à notre modèle LRS mais sans la prise en compte des positions des shapelets. Il s'agit d'une régression logistique simple qui prend en entrée la représentation des séries temporelles par la distance avec les shapelets, et utilise un Lasso comme régularisation (là où LRS utilise la régularisation SSGL). Par conséquent, la différence entre les deux modèles concerne uniquement la prise en compte des positions des shapelets.

La Figure 5.14 compare le taux d'erreur entre les deux modèles sur les 69 jeux de données. Nous voyons que la prise en compte des positions des shapelets n'améliore pas significativement la précision. Néanmoins, pour certains jeux de données, la position est utile et améliore la précision. Enfin, pour d'autres jeux, le fait que le résultat soit meilleur pour la régression simple par rapport à LRS suggère que notre

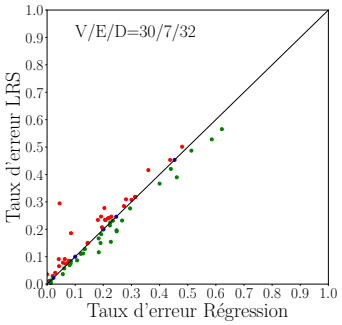


FIGURE 5.14 – Comparaison du taux d'erreur entre LRS et une régression logistique.

régularisation n'est pas optimale. En effet, LRS est censé ignorer les positions des shapelets si celles-ci n'aident pas à la classification. Par conséquent, nous nous attendons à ce que sa précision soit au minimum équivalente à la régression car elle aussi ne prend pas en compte la position des shapelets. Comme ce n'est pas le cas, nous concluons que la position a été prise en compte quand ce n'était pas nécessaire et a pénalisé la classification. Bien que cette conclusion soit décevante pour notre modèle, cela suppose aussi que la prise en compte des positions des shapelets peut pénaliser la classification et permettre de caractériser certains jeux de séries temporelles.

Nous creusons cette analyse en comparant la différence de précision entre la méthode LRS et la régression simple, selon le domaine des jeux de données. L'objectif est de trouver les domaines où les positions des shapelets peuvent être utiles à la classification des séries temporelles. Nous comparons l'influence de la position seulement par rapport au domaine des séries, car corréler les positions des shapelets avec d'autres caractéristiques telles que la taille des séries temporelles, le nombre de classes ou le nombre d'instances dans le jeu de données n'a aucun sens. Dans le répertoire UCR & UEA, il existe 7 domaines différents :

— les électrocardiogrammes (ECG), qui représentent l'activité cardiaque d'un patient ;
— les spectrographes d'aliments (SPECTRO) : il s'agit d'une donnée utilisée en chimiométrie qui permet l'analyse de la qualité des aliments et de leur origine ;
— les capteurs de mouvements (MOUVT) : il s'agit de la position en x, en y et en z de capteurs placés en général sur des êtres humains ;
— la consommation électrique d'appareils du quotidien telle que des télévisions, des fours, etc (APPAREIL) ;

— les images (IMAGE) : il s'agit d'images transformées en séries temporelles ;
— les capteurs divers, qui regroupent les séries temporelles d'applications très variées telle que des enregistrements sonores, des courbes de production électrique ou bien la luminosité d'étoiles ;
— les synthétiques (SYNTH), qui regroupent l'ensemble des séries temporelles générées artificiellement.

Les résultats sont présentés dans le Tableau 5.1. La différence est exprimée en pourcentage et séparée dans différents intervalles. Si la différence est positive, alors LRS est meilleur que la régression simple. Enfin, nous ignorons les 14 jeux de données des capteurs divers car la variabilité des applications est trop importante pour pouvoir en tirer une conclusion utile.

TABLE 5.1 – Différence de précision entre LRS et la régression simple par domaine applicatif.

domaine	entre -10% et 0%	0%	entre 0% et 10%	entre 10% et 25%	total
ECG	**5**	0	1	0	6
SPECTRO	3	2	2	0	7
MOUVT	3	0	**8**	0	11
APPAREIL	2	0	3	0	5
IMAGE	9	3	8	0	20
SYNTH	3	0	2	1	6
total	25	5	24	1	55

Nous voyons dans ce tableau qu'à l'exception des domaines MOUVT et ECG, la distribution des différences de précision en faveur des deux modèles est équilibrée.

Dans le domaine MOUVT, nous voyons que les différences de précision penchent en faveur de LRS (8 sur 11). La position de la shapelet apporte donc une information utile pour la classification. En général, il s'agit de jeux de données dont l'objectif est d'identifier le mouvement réalisé par une personne grâce aux enregistrements de capteurs de mouvements placés sur lui. Un mouvement correspond à une séquence de gestes, chaque geste produit un déplacement du capteur qui est enregistré dans la série temporelle sous la forme d'une sous-série. Un geste peut être présent dans différents mouvements mais à différents moments dans le mouvement. Par conséquent, la détection de ce geste ne permet pas d'identifier le mouvement, mais sa position dans le temps le permet. Un autre exemple, consiste à distinguer différents acteurs qui réalisent le même mouvement. Ici, le mouvement enregistré est le même mais avec des

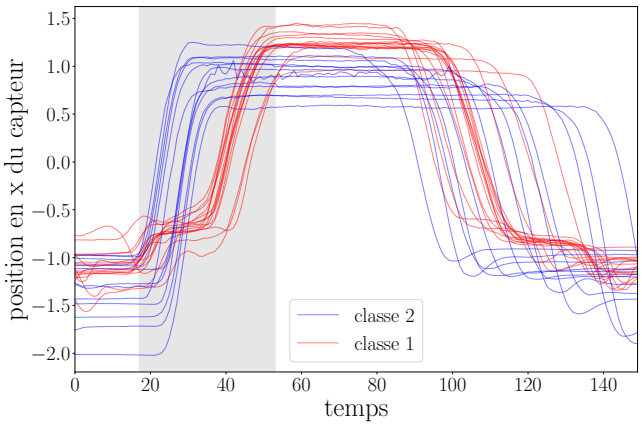


FIGURE 5.15 – Ensemble des séries temporelles d'apprentissage du jeu de données *GunPoint* d'un des deux acteurs.

amplitudes différentes et possiblement avec des décalages si certains acteurs sont plus lents. Cette information peut alors être capturée par la position de la sous-série associée à l'éxécution du mouvement.

Dans la Figure 5.15, nous avons superposé les séries temporelles du jeu de données *GunPoint* qui appartient au domaine MOUVT. Un acteur et une actrice ont été utilisés pour réaliser un mouvement qui consiste à dégainer une arme et à la ranger. Ce mouvement est réalisé avec et sans arme, et l'objectif consiste à pouvoir distinguer les deux situations. Le capteur enregistre uniquement les mouvements de la main sur l'axe avant/arrière par rapport au corps. Nous présentons ici les mouvements d'un des deux acteurs. La zone grise couvre, dans les séries temporelles des deux mouvements (avec et sans arme), le geste où l'arme est pointée en avant (la valeur augmente quand la main avance). Ce geste est le même dans les deux cas et produit la même forme de sous-série : une pente croissante jusqu'à un palier. Nous remarquons que dans la classe 2 ce geste est réalisé plus rapidement, et cette sous-série apparaît plus tôt dans les séries temporelles qui lui sont associées. Il s'agit de la même sous-série dans deux classes différentes mais la position de cette dernière permet de discriminer la classe.

À l'inverse, dans le domaine ECG, nous voyons que les différences de précision penchent en faveur de la régression simple (5 sur 6 jeux), c'est-à-dire que la prise en compte des positions des shapelets pénalise la précision. Les électrocardiogrammes sont des séries temporelles extrêmement régulières, dont la position d'une sous-série ne peut fournir une explication à l'utilisateur. Par exemple, dans la Figure 5.16 nous

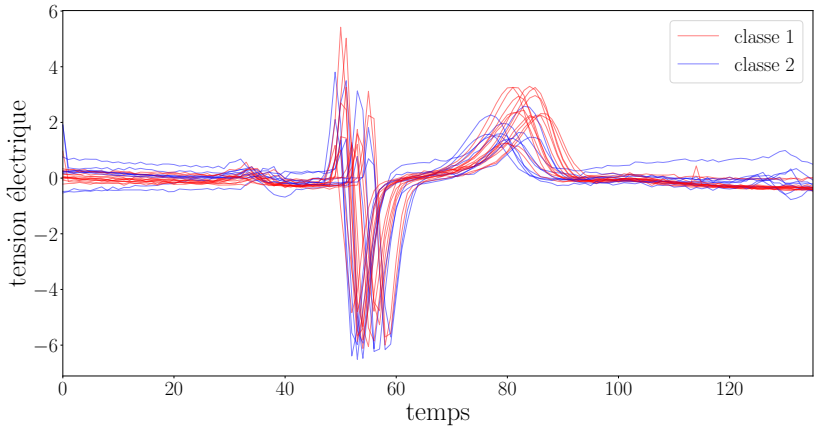


FIGURE 5.16 – Ensemble des séries temporelles d'apprentissage du jeu de données *ECGFiveDays*.

avons superposé les séries temporelles d'apprentissage du jeu de données *ECGFiveDays*. Le seul élément qui différencie les deux classes est l'amplitude des valeurs d'une sous-série (ici, la hauteur de la "bosse" permet de discriminer les deux classes). La position d'une shapelet n'est pas une information qui ait du sens.

5.5 Conclusion

Dans ce chapitre, nous avons exploré la piste d'une méthode de classification de séries temporelles qui est par nature interprétable. Nous avons choisi de proposer un classifieur à base de shapelets car les explications retournées permettent de mettre en évidence des connaissances locales (des sous-séries discriminantes) dans les séries temporelles. Jusqu'à présent, seulement les distances entre les shapelets et les séries temporelles ont été utilisées comme attribut pour la classification. La distance d'une shapelet avec une série temporelle nous informe uniquement si la présence ou l'absence de la shapelet a été déterminante dans la classification d'une série temporelle.

Notre proposition a étendu l'utilisation des shapelets en prenant aussi en compte la position des shapelets dans les séries temporelles comme nouvel attribut pour la classification. La position d'une shapelet dans une série temporelle est la position où la shapelet correspond le mieux à une sous-série dans la série temporelle (là où la distance de la shapelet a été calculée). Notre modèle est une régression logistique

combinée avec une méthode de régularisation capable de prendre en compte la position d'une shapelet si sa présence contribue à la classification. Nous avons présenté la façon d'extraire les explications globales de notre modèle. Enfin, nous avons comparé les performances de classification de notre modèle avec les classifieurs à base de shapelets de l'état de l'art. Nous en tirons trois conclusions :

— les méthodes de classification à base de shapelets non-interprétables sont en général plus performantes que celles interprétables ;
— il n'existe pas de classifieur à base de shapelets qui soit universellement meilleur que tous les autres. Cela suppose que selon le type de jeu de séries temporelles, un classifieur peut être plus adapté qu'un autre ;
— dans la continuité de la conclusion précédente, la prise en compte de la position des shapelets dans certains cas permet effectivement d'améliorer leur classification.

CHAPITRE 6

EXPLICATION LOCALE ET AGNOSTIQUE POUR LA CLASSIFICATION DE SÉRIES TEMPORELLES

6.1 Motivations

Dans le chapitre précédent nous avons conclu que les méthodes de classification à base de shapelets non-interprétables sont en général plus performantes que celles interprétables. Par ailleurs, il n'existe pas de classifieur à base de shapelets qui soit universellement meilleur que tous les autres. Cela suppose que selon le type de jeu de séries temporelles, un classifieur peut être plus adapté qu'un autre.

Ces conclusions peuvent être étendues sur l'ensemble des types de classifieurs de séries temporelles. Nous avons analysé deux études [Bag+17 ; Faw+19] comparant les performances des classifieurs de l'état de l'art, sur les mêmes 85 jeux de données du répertoire UCR&UEA [AK]. Dans [Bag+17], les auteurs présentent et comparent 18 modèles couvrant l'ensemble des stratégies de classification de séries temporelles. Dans [Faw+19], les auteurs présentent et comparent l'application de différentes architectures de réseaux de neurones pour de la classification de séries temporelles. La meilleure des architectures a ensuite été comparée avec les quatre classifieurs les plus précis selon l'étude de [Bag+17].

La première conclusion est que les modèles non-interprétables ont de meilleures performances que ceux interprétables. En effet, les quatre meilleurs classifieurs dans [Bag+17] sont des modèles ensemblistes, c'est à dire des modèles qui combinent les résultats de plusieurs classifieurs différents. Par ailleurs, dans [Faw+19], la meilleure architecture de réseaux de neurones trouvée a des performances équivalentes au meilleur modèle dans [Bag+17]. Ces deux types de modèles (modèle ensembliste et à

base de réseau de neurones) sont non-interprétables car il est difficile de retrouver la contribution d'un quelconque attribut dans une prédiction (cf. section 3.2.2).

La seconde conclusion est qu'il n'existe pas de classifieur de séries temporelles très précis sur tous les jeux de séries temporelles. La meilleure architecture de réseaux de neurones (*RESNET* [WYO17]) trouvée dans [Faw+19] est meilleure en général que les autre architectures sans pour autant être la meilleure sur tous les jeux. Dans [Bag+17], le meilleur classifieur *COTE* [Bag+15] est premier sur 36 des 85 jeux de séries temporelles, le second meilleur classifieur *Shapelet Transform* [Lin+12] est premier sur 17 jeux, et le troisième meilleur classifieur *BOSS* [Sch15] est premier sur 17 jeux. Par conséquent, aucun modèle n'est meilleur que tous les autres sur tous les jeux de séries temporelles.

Une explication proposée par les deux études suggère qu'un type de classifieurs serait performant seulement sur un type de séries temporelles. Les deux études ont réalisé une analyse pour catégoriser les meilleurs classifieurs selon les caractéristiques des jeux de séries temporelles. Les auteurs y analysent les performances des types de classification selon le nombre de classes, le nombre d'instances d'apprentissage, la taille des séries temporelles et la source des séries temporelles (capteur de mouvements, consommation électrique, ECG, etc). Ces premiers résultats expérimentaux montrent que certaines familles de classifieurs sont plus performantes pour certains types de séries temporelles (ex : les classifieurs à base de vecteurs fonctionnent mieux sur les jeux de données de spectrogrammes).

En résumé, actuellement il n'existe pas de classifieur de séries temporelles universellement précis, mais plutôt un ensemble de classifieurs, majoritairement non-interprétables, se partageant les meilleures performances sur l'ensemble des jeux de séries temporelles. Par conséquent, si nous souhaitons expliquer les prédictions du classifieur avec la meilleure précision dans toutes les situations, alors nous avons besoin d'une **méthode d'explication a posteriori et agnostique**, c'est-à-dire, capable d'expliquer les résultats de n'importe quel classifieur (et en particulier ceux non-interprétables).

Dans la section 3.2.3, nous avons présenté succintement deux méthodes LIME [RSG16] et SHAP [LL17] capables d'expliquer les résultats locaux de n'importe quel classifieur en approximant les prédictions des classifieurs par un modèle interprétable (appelé proxy). Cependant et jusqu'à présent, ces méthodes ont été seulement appliquées sur des données textuelles, des données tabulaires et des images, mais jamais

sur des séries temporelles.

À partir de ces deux méthodes nous avons extrait un processus d'explication de modèle, que nous nommons PIAEL, pour Proxy Interprétable et Agnostique pour Explication Locale. En effet, bien que ce dernier ait été introduit dans LIME, et repris dans SHAP, aucune définition générique et globale n'avait été donnée jusqu'à présent. Il s'agit là d'une première contribution, car de nombreuses méthodes d'explications par proxy ont été proposées, mais il y a peu d'effort fourni pour prendre du recul, et tenter d'identifier les éléments en communs et les différences entre ces méthodes. La section 6.2 est consacrée à cette définition.

Ce processus a ensuite été utilisé pour notre seconde contribution : la définition de la première méthode agnostique d'explication de prédiction locale de classifieur de séries temporelles, nommée *LEFTIST* [Gui+19c] (*agnostic Local Explanation For TIme Series classificaTion*), dont la présentation est réalisée dans la section 6.3. La section 6.4 présente les premières expérimentations pour évaluer l'approximation de classifieurs de séries temporelles par des proxys. Enfin, la section 6.5 est dédiée à l'évaluation des réponses d'une étude utilisateur sur l'interprétabilité de nos explications de prédictions par des humains.

6.2 Proxy interprétable et agnostique pour explication locale

Cette section est consacrée à la présentation du processus d'explication PIAEL (pour proxy interprétable et agnostique pour explication locale). La première sous-section présente le fonctionnement général du processus qui est ensuite détaillé dans les deux sous-sections suivantes pour les deux méthodes basées sur ce processus : *LIME* [RSG16] et *SHAP* [LL17].

6.2.1 Fonctionnement général

Nous considérons que nous avons à disposition un ensemble de données X, qui peuvent être des textes, des images, des données tabulaires, des séries temporelles, etc. Ces données sont associées à un ensemble de classes Y. Soit $f : X \mapsto [0,1]^{|Y|}$ un classifieur qui retourne un vecteur de distribution des probabilités d'appartenance

d'une donnée de X aux différentes classes de Y.

L'objectif du processus PIAEL est de fournir une explication à la prédiction retournée par f pour une donnée $x \in X$. Cette explication doit permettre à un humain de comprendre pourquoi le classifieur f a choisi la classe c pour la donnée x (où c est la classe la plus probable de $f(x)$). Pour expliquer une prédiction, le processus PIAEL apprend un **proxy** g_x qui est un classifieur interprétable et dont l'objectif est d'approximer le mieux possible les résultats de f dans le voisinage de x. L'explication de la prédiction est ensuite extraite à partir de g_x.

En général, les méthodes d'explication fournissent une explication basée sur les attributs manipulés par le classifieur. Cependant, dans certains classifieurs ces attributs n'ont pas de sens pour un utilisateur (ex : un pixel dans une image ou la fréquence d'une lettre dans un texte). L'idée fondamentale du processus PIAEL pour résoudre ce problème et permettre ainsi l'explication de n'importe quel classifieur, consiste à fournir des explications basées sur des composants de la donnée qui sont interprétables pour l'utilisateur. Par exemple, dans des images, les **composants interprétables** peuvent être des régions de l'image. Dans des textes, il peut s'agir de mots où d'ensembles de mots enrichis avec des informations grammaticales. Ces composants interprétables sont utilisés pour représenter la donnée x dans un nouvel espace dans lequel g_x est ensuite appris. Nous appelons cet espace, l'espace de représentation en composants interprétables.

Le processus d'explication PIAEL, se définit par la présence des quatres étapes suivantes :

— l'extraction des composants interprétables de x ;
— la génération d'un voisinage autour de la donnée, dans l'espace de représentation en composants interprétables ;
— la classification par f des voisins générés ;
— l'apprentissage d'un proxy interprétable g_x à partir des voisins de x, dans l'espace de représentation en composants interprétables, de façon à ce que g_x approxime la classification par f.

Chacune des étapes est présentée plus en détails dans la suite de cette partie.

Extraction des composants interprétables

L'extraction des composants interprétables d'une donnée x consiste à extraire l'ensemble des composants $I_x = \{I_x^1, \ldots, I_x^m\}$ (avec $|I_x| = m$ et $m \in \mathbb{N}$) à partir de x. Cette

étape est spécifique aux types des données expliquées. En effet, les composants interprétables d'un texte ne sont pas les mêmes que ceux d'une image. Ces composants doivent être individuellement interprétables pour l'utilisateur et peuvent être extraits par un algorithme (ex : un algorithme de segmentation d'images) ou par l'utilisateur lui même. La Figure 6.1 illustre l'extraction de composants interprétables dans le domaine des images et du texte comme introduit dans LIME [RSG16]. L'image est segmentée en "super pixels" via une méthode standard comme par exemple *SLIC* [Ach+12].

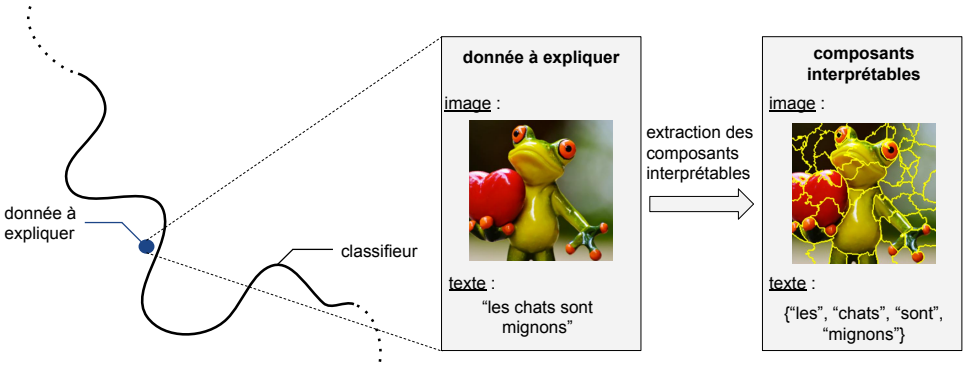


FIGURE 6.1 – L'étape d'extraction des composants interprétables.

Génération des voisins

Notre objectif est d'apprendre un classifieur interprétable qui approxime f autour de x sur les composants interprétables. L'intérêt est de savoir quels composants ont contribué à la classification de x par f. Soit le domaine $IF = \{0,1\}^m$, où le $i^{ème}$ attribut indique la présence ou l'absence du composant interprétable I_x^i. Dans ce domaine, x équivaut au vecteur $m_x = (1,1,\ldots,1,1)$, vecteur composé uniquement de 1 car tous les composants interprétables sont présents. Pour mettre en évidence les composants qui ont servi à la classification de x, nous tirons des voisins de x en lui enlevant des composants interprétables. Nous générons des vecteurs dans IF avec un ou plusieurs attributs à 0. Par exemple, le voisin de x dont seulement le premier composant est enlevé correspond au vecteur $(0,1,\ldots,1,1)$. La génération des voisins consiste à générer un ensemble de voisins $V = \{z_{x_j}\}_{j=1}^N$ avec $z_{x_j} \in IF$ le $j^{ème}$ voisin de x et $N \in \mathbb{N}$, comme illustré dans la Figure 6.2. Le nombre de voisins N est un paramètre qui doit être fixé.

Classification des voisins

Pour que le classifieur interprétable approxime f dans l'espace de représentation

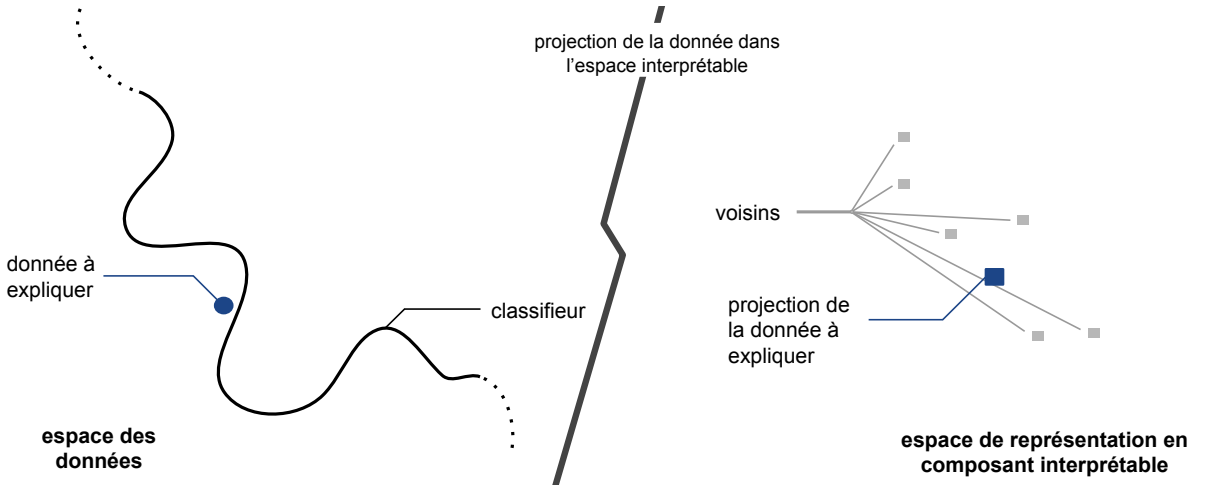


FIGURE 6.2 – L'étape de génération des voisins.

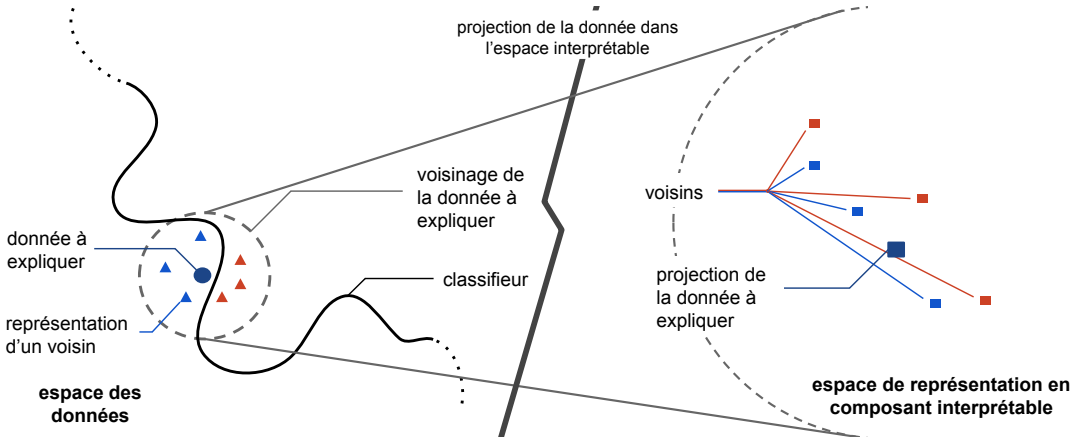


FIGURE 6.3 – L'étape de classification des voisins.

en composants interprétables, nous avons besoin de classifier l'ensemble des voisins V par f. Cependant, le classifieur f prend comme entrée des éléments de l'espace initial de la donnée x (nommé $domaine(X)$). Par conséquent, il nous faut introduire une fonction $h_x : IF \rightarrow domaine(X)$ pour pouvoir associer à un voisin dans l'espace IF, un élément dans l'espace initial. Par exemple, pour une image, la fonction h_x peut remplacer les composants absents par une zone grisée, comme le montre la Figure 6.4. Attention, cette fonction est spécifique à x car elle utilise les composants interprétables tirés de x. Grâce à la fonction h_x, il est maintenant possible d'associer à un voisin z_x sa classification par f : $f(h_z(z_x))$. Ce qui nous permet à partir de l'ensemble des voisins $V = \{z_{x_j}\}_{j=1}^{N}$ de construire l'ensemble $A = \{(z_{x_j}, f(h_z(z_{x_j}))\}_{j=1}^{N}$ nécessaire à l'apprentissage du classifieur interprétable.

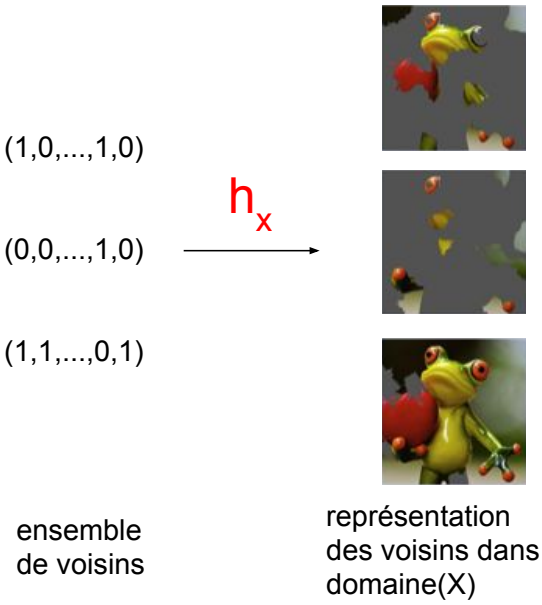


FIGURE 6.4 – Exemple de fonction h_x.

Apprentissage du proxy

La dernière étape du processus PIAEL consiste à apprendre le proxy interprétable g_x à partir de l'ensemble d'apprentissage A comme illustré dans la Figure 6.5. Pour rappel, un proxy est interprétable si il vérifie ces quatre propriétés :

- **propriété 1** : les attributs utilisés par le modèle doivent être compréhensibles pour l'utilisateur ;
- **propriété 2** : l'utilisation de ces attributs dans le calcul du résultat du classifieur doit être compréhensible.
- **propriété 3** : le proxy doit être **fidèle**, c'est-à-dire qu'il doit fournir les mêmes classifications que le classifieur visé ;
- **propriété 4** : le proxy doit être **précis**, c'est-à-dire qu'il doit être capable de bien classer de nouvelles séries dans le voisinage de x ;

La première propriété est vérifiée car le proxy est appris dans l'espace de représentation en composants interprétables. La seconde propriété nécessite l'utilisation d'un type de classifieur considéré interprétable, pour le proxy. Par exemple dans LIME et SHAP, le proxy est un modèle linéaire simple. Pour s'assurer que le proxy respecte l'ensemble de ces propriétés, ce dernier est calculé de la manière suivante :

$$PIAEL(x) = \underset{g_x \in G}{argmin} \mathcal{L}(f, g_x, \Pi_x) + \Omega(g_x)$$

où G est l'ensemble des proxys possibles et $\mathcal{L}$ une fonction de fidélité qui évalue à quel point g_x approxime f sur les voisins de x. Dans cette fonction $\mathcal{L}$, les voisins sont pondérés selon leur distance avec la représentation de x dans l'espace de représentation en composants interprétables (le vecteur $m_x = (1, 1, \ldots, 1, 1)$). La distance est calculée par la mesure de distance Π_x. Enfin, Ω est une fonction pour évaluer la complexité de g_x car, comme nous l'avons évoqué dans la section 6.5, même les modèles les plus simples peuvent devenir non-interprétables quand leur taille devient trop conséquente (ex : un modèle linéaire avec trop de coefficients ou une règle avec trop de conditions).

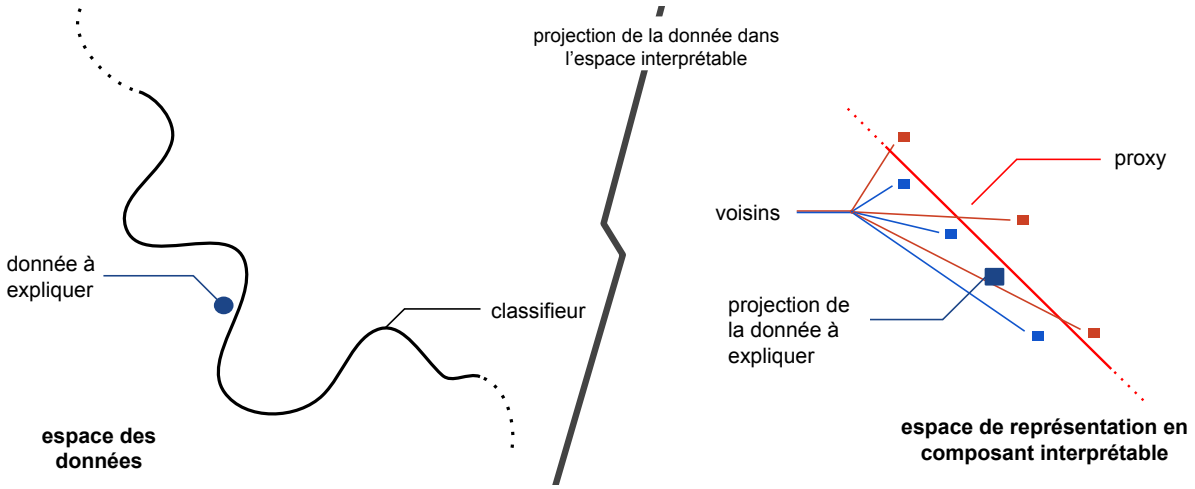


FIGURE 6.5 – L'étape d'apprentissage du proxy.

Dans cette partie, nous avons défini le fonctionnement du processus PIAEL qui est présenté dans l'Algorithme 4. Ce processus est employé dans deux méthodes d'explication : LIME [RSG16] et SHAP [LL17]. Dans la suite, nous présentons quelles étapes du processus sont communes à ces deux méthodes avant de présenter les spécificités de chaque méthode.

6.2.2 Parties communes de LIME et SHAP

Notons que SHAP a mis en place un formalisme pour unifier différentes méthodes (dont LIME) d'explication de résultat de classifieur. SHAP a de plus introduit trois pro-

Algorithme 4 PIAEL, Méthode d'explication agnostique et locale de classifieur

Entrées : N le nombre de voisins à tirer,
h_x la fonction de transformation des voisins dans l'espace initial,
Π_x la mesure de proximité,
Ω la fonction qui évalue la complexité du proxy
Sortie : g_x le proxy local qui explique la classification de x par f
1: $I_x \leftarrow extraction_composants_interpretables(x)$
2: $V \leftarrow generation_des_voisins(I_x, N)$
3: $A \leftarrow classification_des_voisins(V, f, h_x)$
4: $g_x \leftarrow apprentissage_du_proxy(A, \Pi_x, \Omega)$
5: **retourne** g_x

priétés qui, si elles étaient vérifiées, garantiraient que le proxy appris était unique. Le calcul de la solution exacte est combinatoirement irréalisable. Différents procédés sont donc mis en place pour approximer le proxy optimal.

Après avoir testé la méthode de Strumbelj et al. [SK14], les auteurs de SHAP sont finalement repartis de l'algorithme présenté dans LIME en adaptant la fonction de fidélité $\mathcal{L}$ et la mesure de proximité Π_x pour vérifier les trois propriétés. C'est la raison pour laquelle SHAP et LIME partagent de nombreuses similitudes dont les étapes d'extraction des composants interprétables et celle de classification des voisins.

Extraction des composants interprétables

Dans SHAP et LIME, nous retrouvons une méthode d'extraction de composants interprétables pour des données textuelles et une pour les images. Dans le cas des images, il s'agit de segmenter l'image en "super pixels" tel que illustré dans la Figure 6.1. Chaque "super pixel" correspond à un composant interprétable. Pour le texte, un document est représenté en sac de mots, c'est-à-dire en un dictionnaire qui associe à chaque mot le nombre de ses occurrences dans le document. Dans cette représentation, chaque mot est un composant interprétable.

Classification des voisins

Les méthodes SHAP et LIME définissent toutes les deux une façon de classifier les voisins en s'appuyant sur la relation h_x qui est utilisée pour associer à un voisin, un élément dans l'espace initial de x. Les fonctions h_x sont définies pour les images et les données textuelles. Pour les images, la fonction h_x conserve les super-pixels de x dont la valeur est à 1 dans le voisin et remplace les super-pixels à 0 par des zones grises.

En ce qui concerne les textes, la fonction conserve uniquement les mots dont la valeur est à 1 dans le voisin.

6.2.3 Parties spécifiques à LIME

LIME [RSG16] a été le premier algorithme à poser les bases du processus PIAEL. LIME se distingue de SHAP sur les étapes de génération des voisins et d'apprentissage du proxy.

Génération des voisins

Dans LIME, la génération des voisins consiste à tirer aléatoirement avec remise les voisins z_{x_i} dans $IF = \{0,1\}^m$. Il est à noter que plusieurs voisins peuvent être identiques, c'est-à-dire des voisins qui ont le même vecteur de composants interprétables.

Méthode d'apprentissage du proxy

Pour rappel, la formule pour apprendre g_x est la suivante :

$$PIAEL(x) = \underset{g_x \in G}{argmin} \left(\mathcal{L}(f, g_x, \Pi_x) + \Omega(g_x)\right)$$

Dans LIME, les opérateurs de l'apprentissage du proxy sont définis de la manière suivante :

— la fonction de distance Π_x entre un voisin z_x et x est définie par l'équation suivante :

$$\Pi_x(z_x) = e^{-\frac{D(m_x, h_x(z_x))^2}{\sigma^2}}$$

il s'agit d'un kernel exponentiel défini sur une mesure de distance D entre les voisins de x et la représentation $m_x = (1, 1, \ldots, 1, 1)$ de x dans l'espace en composants interprétables ;

— la fonction de fidélité $\mathcal{L}$ est définie par l'équation suivante :

$$\mathcal{L}(f, g_x, \Pi_x) = \sum_{z_x \in IF} \Pi_x(z_x)(f(h_x(z_x)) - g_x(z_x))^2 \tag{6.1}$$

avec h_x la fonction de transformation dans l'espace initial. La pondération de l'erreur des voisins par leur distance avec la représentation de x dans l'espace

de représentation en composants interprétables permet de donner plus d'importance aux voisins qui sont proches de x pour s'assurer que g_x approxime localement f.

— $\Omega(g_x) = K$ le nombre de coefficients ϕ_i non-nuls pour limiter la taille du proxy appris.

Le proxy g_x appris par LIME est le modèle linéaire suivant :

$$g_x(z_x) = \sum_{i=1}^{m} \phi_i p_i$$

avec $z_x \in IF$ ($IF = \{0,1\}^m$), p_i la valeur associée au $i^{ième}$ composant de z_x, m le nombre de composants interprétables extraits de x et $\phi_i \in \mathbb{R}$ les coefficients du modèle linéaire.

Pour rappel, un modèle linéaire est interprétable si le nombre de coefficients n'est pas trop élevé (cf. section 3.1.1). Dans LIME, un proxy est préalablement appris avec une régularisation de type Lasso [Efr+04] pour sélectionner les K meilleures composants interprétables. L'apprentissage du proxy final est basé sur les attributs associés à ces K composants interprétables.

6.2.4 Parties spécifiques à SHAP

Dans [LL17], l'approche initiale pour construire le proxy interprétable dans SHAP est totalement différente de celle présentée dans LIME. Elle s'appuie sur la théorie des jeux coopératifs, et présente le problème sous un autre angle. Dans ce problème, $f(x)$ est considérée comme une récompense que nous cherchons à partager entre les composants interprétables de x en fonction de leur contribution dans $f(x)$. Selon les auteurs une solution raisonnable doit vérifier les trois propriétés :

— **précision locale** : soit $f(x) = g_x(m_x)$ avec m_x la représentation de x dans l'espace de représentation en composants interprétables. Cette propriété assure que la classification de f et de g_x pour x est la même (cette propriété n'est pas assurée dans le cas de LIME) ;

— **absence** : si un composant interprétable n'est pas présent dans un voisin alors sa contribution est nulle ;

— **cohérence** : soit deux classifieurs f_1 et f_2 appris sur le même ensemble de donnée X, et leurs proxys respectifs g_1 et g_2 expliquant la même donnée $x \in X$. Si f_1 donne plus d'importance à un composant interprétable de x que f_2, alors

son proxy g_1 doit donner une contribution égale ou supérieure à ce composant que le proxy g_2.

[LL17] montre qu'un proxy g_x vérifiant ces trois propriétés est unique et que les coefficients vérifient :

$$\phi_i(f,x) = \sum_{z_x \in IF} \frac{|z_x|!(m-|z_x|-1)!}{m!}[f(h_x(z_x)) - f(h_x(z_x \setminus i)]$$

avec $|z_x|$ le nombre de composants à 1 dans le voisin z_x et $z_x \setminus i$ dénote le voisin z_x dont le $i^{ème}$ composant est fixé à zéro. Ce théorème vient de la théorie des jeux coopératifs, et ces coefficients sont connus sous le nom de valeurs de *Shapley* [Sha53]. Dans [You85], Young a démontré que les valeurs de Shapley satisfont des axiomes équivalents aux propriétés de **précision locale** et de **cohérence**. Tandis que la propriété d'**absence** est nécessaire pour pouvoir adapter les preuves de Shapley aux modèles linéaires. Dans la pratique, calculer les valeurs de Shapley pour un proxy est insoluble pour un ordinateur si m (le nombre de composants interprétables) est trop grand car le calcul des coefficients est en factorielle de m.

Pour résoudre ce problème, les auteurs ont proposé SHAP, une méthode pour approximer les valeurs de Shapley à partir de la méthode LIME. C'est la raison pour laquelle cette méthode peut être intégrée dans le processus PIAEL. SHAP se différencie de LIME sur trois étapes du processus : l'apprentissage du proxy et la génération des voisins.

Méthode d'apprentissage du proxy

Le proxy g_x à apprendre est un modèle linéaire interprétable identique à celui de LIME. Pour rappel, l'apprentissage du proxy g_x pour une donnée x est défini par la formule suivante :

$$PIAEL(x) = \underset{g_x \in G}{argmin}\,(\mathcal{L}(f, g_x, \Pi_x) + \Omega(g_x))$$

Dans SHAP, les opérateurs de l'apprentissage du proxy sont définis de la manière suivante :

— la fonction de distance Π_x entre un voisin z_x et la représentation de x dans l'espace de représentation en composants interprétables est définie par l'équation

suivante :

$$\Pi_x(z_x) = \frac{m-1}{\mathcal{C}^m_{|z_x|}|z_x|(m-|z_x|)}$$

avec $|z_x|$ le nombre de composants interprétables de x gardés dans le voisin z_x, si $|z_x| \in \{0, m\}$ alors $\Pi_x(z_x) = \infty$. Dans le code de SHAP, ces cas n'apparaissent pas car les voisins en question ne sont jamais générés ;

— la fonction de fidélité $\mathcal{L}$ est la même que celle dans LIME (Equation 6.1) ;

— $\Omega(g_x) = 0$. Il peut être surprenant de ne plus prendre en compte la complexité du proxy, mais cette dernière est prise en compte par une heuristique lors de l'apprentissage du modèle linéaire.

Génération des voisins À la différence de LIME, SHAP génère des voisins selon la distance Π_x en sélectionnant en priorité les voisins dans IF dont le poids selon Π_x sont les plus importants. Il s'agit des voisins dont le nombre de composants masqués est aux extrêmes de ses valeurs possibles, c'est-à-dire ceux dont très peu de composants sont masqués (voisin avec beaucoup de valeur à 1 et quelques 0) ou bien ceux où tous les composants sont pratiquement masqués (voisins avec beaucoup de zéros et quelques 1). Plus le nombre de composants masqués et celui de non-masqués s'équilibrent, plus le poids de Π_x est faible.

Dans cette partie, nous avons présenté le processus PIAEL qui synthétise les méthodes d'explication LIME et SHAP. Ce processus permet de créer une méthode d'explication locale par proxy qui fonctionne sur n'importe quel type de classifieur pour un type de données en particulier. Dans la suite, nous utilisons ce processus pour créer la première méthode d'explication locale par proxy et modèle agnostique à destination des classifieurs de séries temporelles. Il s'agit d'un type de données qui n'a pas a été traité par LIME ou SHAP.

6.3 Méthode LEFTIST

Cette section présente la contribution principale de ce chapitre : l'approche LEFTIST (*agnostic Local Explanation For TIme Series classificaTion*), une implémentation du processus PIAEL dédiée à l'explication de classification locale de séries temporelles.

Soit un ensemble de séries temporelles D, un ensemble de classes Y et $f : D \rightarrow$

$[0, 1]^{|Y|}$ un classifieur qui retourne un vecteur de distribution des probabilités sur les classes Y. Soit la série temporelle $ts \in D$, une séquence ordonnée $ts = (t_1, \ldots, t_n)$ de taille n, avec $t_i \in \mathbb{R}$ et $1 \leq i \leq n$. Intuitivement, t_i correspond à une valeur à la date i. La classification de ts par f nous retourne la classe c (où c est la classe la plus probable de $f(ts)$).

L'objectif de LEFTIST est de fournir une explication pour la classe c retournée par f pour ts. Pour expliquer la classification, LEFTIST apprend un proxy g_{ts} qui est un classifieur interprétable dont l'objectif est d'approximer le mieux possible les résultats de f dans le voisinage de ts. L'explication de la classification est ensuite extraite à partir de g_{ts}.

LEFTIST reprend exactement le fonctionnement de l'Algorithme 4 présenté dans la section précédente, il s'agit des quatre étapes suivantes :

— l'extraction de composants interprétables pour ts ;
— la génération de voisins à ts dans l'espace de représentation en composants interprétables ;
— la classification de ces voisins par le classifieur f ;
— l'apprentissage d'un proxy g_{ts} dans l'espace de représentation en composants interprétables qui approxime la classification des voisins de ts par f.

L'exercice d'avoir défini le processus d'explication PIAEL nous permet de pouvoir facilement combiner des blocs de différentes méthodes pour créer une méthode d'explication agnostique et locale sur les séries temporelles. Nous utilisons les solutions proposées dans LIME et SHAP pour la génération des voisins et l'apprentissage du proxy. Cependant en ce qui concerne les étapes d'extraction des composants interprétables et de classification des séries, nous avons besoin de les adapter aux séries temporelles. La première consiste à proposer une nouvelle méthode d'extraction, tandis que la seconde consiste à proposer une fonction h_x dédiée au séries temporelles.

6.3.1 Extraction de composants interprétables dans une série temporelle

Un composant interprétable est une connaissance extraite des données et qui a du sens pour l'utilisateur. Dans le cas des séries temporelles, nous avons vu à de nombreuses reprises que les sous-séries sont très largement utilisées à cette fin. C'est le cas pour décrire des connaissances dans une série temporelle (cf. Chapitre 4) ou

bien pour classifier de façon interprétable des séries temporelles (cf Chapitre 5).

Nous définissons les composants interprétables d'une série temporelle ts comme les éléments d'une segmentation de ts. Nous notons cette segmentation par S_{ts}, où $S_{ts} = \{S_{ts}^1, \ldots, S_{ts}^m\}$, avec : $S_{ts}^1 = (t_1, \ldots, t_{d^1}), \ldots, S_{ts}^m = (t_{d^{m-1}+1}, \ldots, t_n)$ où d^i est la date de fin du $i^{ème}$ segment. Cette segmentation peut provenir d'une méthode connue (ex : le découpage par tendances de la séries temporelles) ou bien être fournie par l'utilisateur.

6.3.2 Transformation des voisins en séries temporelles

Pour rappel, l'apprentissage du proxy g_{ts} nécessite la classification des voisins générés autour de ts. Ces voisins sont générés dans l'espace de représentation en composants interprétables de domaine $IF = \{0, 1\}^m$ (avec m le nombre de composants interprétables). Par exemple, dans cet espace notre série temporelle ts est représentée par $z_{ts} = (1, \ldots, 1)$, c'est à dire que ts présente tous les segments de S. Pour classifier les voisins par f, nous avons besoin de les représenter dans l'espace initial des séries temporelles (de domaine $domaine(D)$). Cette tâche est confiée à la fonction de transformation $h_ts : IF \rightarrow domaine(D)$.

Definition 6.3.1. Concaténation de séries temporelles
L'opérateur de concaténation de segments $\bigoplus_{i=1}^{k}$ est défini de la manière suivante pour k segments de la série temporelle ts :

$$\bigoplus_{i=1}^{k} S_t s^i = (t_1, \ldots, t_{d^1}, \ldots, t_j, \ldots, t_{d^j}, \ldots, t_{d^{k-1}+1}, \ldots, t_d^k)$$

où t_j et t_{d_j} sont respectivement la première et la dernière valeur du $j^i ème$ segment $S_t s^j$ de ts, avec $1 \leq j \leq k$.

Par exemple, selon cet opérateur $ts = \bigoplus_{i=1}^{m} S_t s^i$.

Soit z_{ts}^j le $j^{ème}$ voisin de ts alors ts_j la série temporelle reconstruite à partir de ts et du voisin $z_{ts}^j \in \{0, 1\}^m$ est est obtenue par :

$$ts^j = h_t s(z_{ts}^j) = \bigoplus_{i=1}^{m} \begin{cases} S_{ts}^i \text{ if } z_{ts}^{j,i} = 1 \\ transform_{ts}(S_{ts}^i) \text{ if } z_{ts}^{j,i} = 0 \end{cases}$$

où $z_{ts}^{j,i}$ est la valeur associé au $i^i ème$ composant interprétable du $j^i ème$ voisin z_{ts}^j. Cela

veut dire que si $z_{ts}^{j,i}$ est égal à 1 alors le segment S_{ts}^{i} de ts est conservé dans ts_j, sinon ce dernier est transformé par la fonction $transform_{ts}$.

Nous proposons plusieurs méthodes de transformation de segment, chacune est illustrée sur la série temporelle ts donnée dans la Figure 6.6. Chaque segment est représenté par une couleur différente et chaque fonction $transform_{ts}$ est appliquée sur le voisin $z_{ts}^{j} = (0, 0, 1, 1, 1)$.

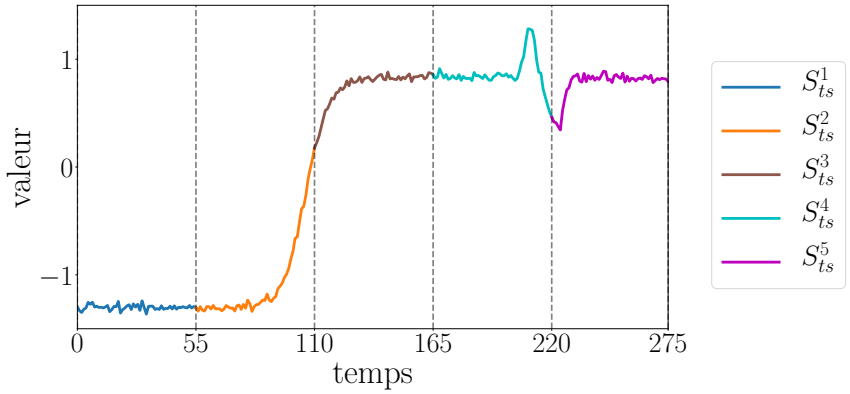


FIGURE 6.6 – Série temporelle ts et l'ensemble de ses composants interprétables S.

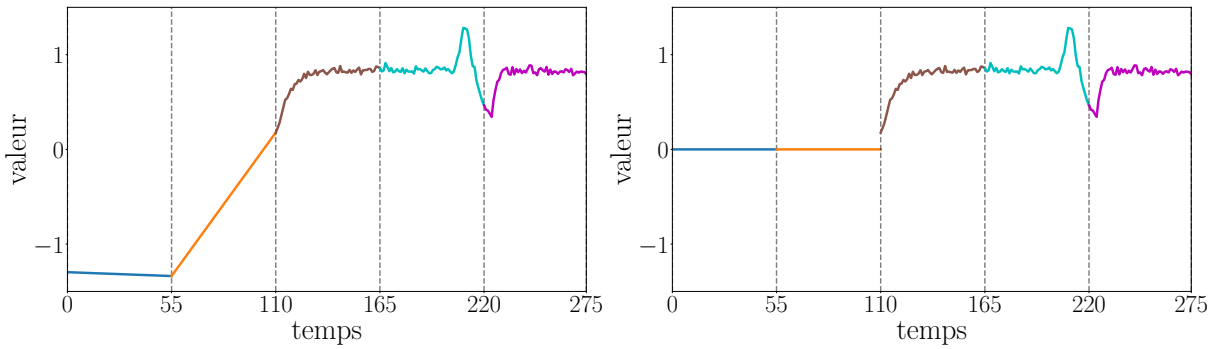
— **Interpolation linéaire** Soit $l_i(x) = a_i x + b_i$ l'équation de la droite allant de $(t_{d^{i-1}})$ à (t_{d^i+1}), où d^i est la dernière position du $i^ième$ segment S_{ts}^i. Il s'agit de la droite rejoignant le dernier point précédant $S_t s^i$ et le premier point suivant $S_t s^i$. Nous utilisons cette équation pour transformer les segments masqués selon la formule :

$$transform_{ts}(S_{ts}^i) = (l_i(t_i), \ldots, l_i(t_{t^i})) \tag{6.2}$$

Cette transformation est illustrée dans la Figure 6.7a.

— **Constante** Soit $l_i(x) = c$ avec c une constante qui est fixée ou bien calculée à partir de la série temporelle (par exemple en prenant la moyenne des valeurs de la série temporelle). $transform_{ts}(S_{ts}^i)$ est aussi définie par l'équation (6.2). Cette transformation est illustrée dans la Figure 6.7b. Une approche similaire a été proposée dans le domaine des images [RSG16] où des pixels gris remplacent les composants interprétables masqués.

— **Arrière-plan aléatoire** Ici, un ensemble de séries temporelles E_{ap} de même taille que ts est supposée disponible. Une série temporelle $tr \in E_{ap}$ est sélectionnée aléatoirement ($tr = \bigoplus_{i=1}^{m} S_{tr}^i$) et est utilisée pour calculer $transform_{ts}$



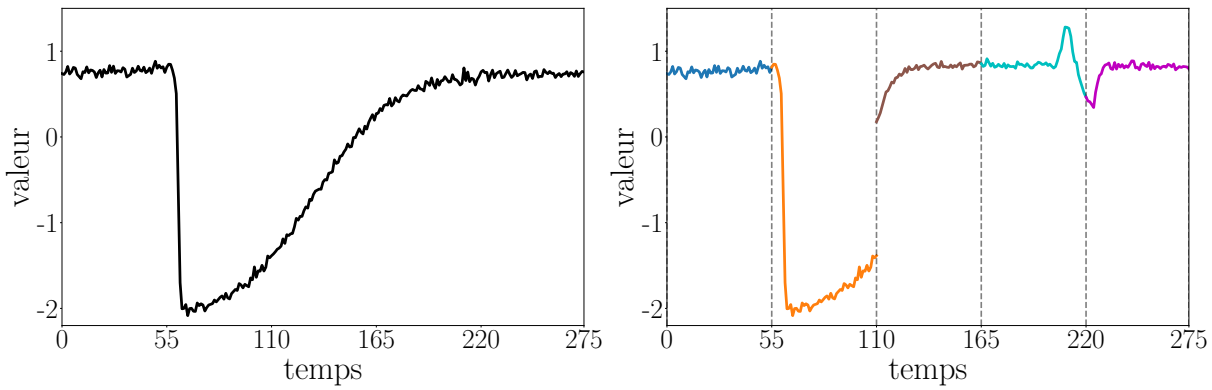
(a) transformation **Interpolation linéaire** (b) transformation **Constante** où $c = 0$

FIGURE 6.7 – Exemples de transformation avec l'équation (6.2).

de la manière suivante :

$$transform_{ts}(S_t s^i) = S^i_{tr} \tag{6.3}$$

où le $i^i ème$ segment S^i_{ts} de ts est masqué et est remplacé par S^i_{tr} (le $i^{ème}$ segment de tr). La Figure 6.8b illustre cette transformation. Une telle méthode est utilisée dans [RSG18] dans le domaine des images, où les composants interprétables masqués sont remplacés par des zones tirées dans d'autres images. Par exemple, si la série temporelle aléatoire tr est donnée par la Figure 6.8a alors la Figure 6.8b montre $h_x(0, 0, 1, 1, 1)$ pour la série temporelle ts représentée dans la Figure 6.6.



(a) Série temporelle tr aléatoirement choisie (b) transformation **Arrière-plan aléatoire**

FIGURE 6.8 – Exemple de transformation avec l'équation (6.3).

6.4 Évaluation quantitative

Dans LEFTIST, l'explication de la classification d'une donnée consiste à apprendre un proxy interprétable qui approxime le classifieur dans le voisinage de la donnée. C'est à partir de ce proxy qu'est ensuite tirée l'explication. Pour rappel, une des propriétés essentielle du proxy est sa fidélité avec le classifieur. En effet, si cette fidélité est désastreuse alors aucun utilisateur ne peut croire l'explication.

Nous avons proposé deux expérimentations pour évaluer la fidélité. La première compare les explications retournées par LEFTIST avec celles tirées d'un classifieur considéré comme interprétable. La fidélité est mesurée en prenant comme vérité terrain les explications du classifieur interprétable. La Section 6.4.2 est consacrée à cette évaluation.

La seconde évaluation cherche à mesurer la fidélité de LEFTIST avec un classifieur non-interprétable. Comme nous n'avons pas accès à une quelconque explication du classifieur, nous allons utiliser les résultats du classifieur. L'objectif est de montrer que les proxys appris par LEFTIST sont localement fidèles au classifieur. Par conséquent, nous allons mesurer la correspondance des classifications obtenues par le classifieur et celles des proxys dans le voisinage des données expliquées. L'évaluation de la fidélité avec des classifieurs non-interprétables est présentée dans la Section 6.4.3.

Avant cela, il nous faut commencer par présenter le protocole expérimental utilisé dans l'ensemble des évaluations réalisées.

6.4.1 Protocole expérimental global

Nous avons sélectionné 25 jeux de données parmi les 128 jeux de données de l'archive UCR& UEA [AK], une base de données connue et libre d'accès pour de la classification de séries temporelles.

Les jeux de données dans l'archive y sont très diversifiés : par le nombre de classes, par la taille des séries temporelles, par le nombre d'instances d'apprentissage et par leur domaine (capteur de mouvement, spectrographe, courbe de consommation électrique, ECG, etc). Nous avons sélectionné 25 jeux représentatifs de cette diversité. Leur nombre de classes varient entre 2 et 7, leur taille entre 20 et 900 points, et ces jeux couvrent un vaste nombre de domaines (ECG, spectrographe, capteur de mouvement, représentation d'images en séries temporelles, jeu synthétique).

L'ensemble des évaluations repose sur la comparaison de proxys avec des classifieurs dont les résultats doivent être expliqués. Pour chaque jeu de données et chaque classifieur testé, nous commençons par apprendre le classifieur sur le jeu d'apprentissage du jeu de données. Puis, le classifieur est utilisé pour apprendre un proxy sur chaque série temporelle du jeu de test. Enfin, nous évaluons la fidélité entre les proxys et le classifieur à expliquer.

Pour l'apprentissage du proxy, nous pouvons utiliser une fonction $transform_{ts}$ (Arrière-plan aléatoire) qui nécessite un ensemble E_{ap} de séries temporelles. Par conséquent, dans tous nos jeux de données, nous réservons en une fois, 25% des séries temporelles d'apprentissage pour E_{ap}. Les 75% restants sont utilisés pour entraîner les classifieurs.

Les paramètres de LEFTIST sont les suivants :

— le nombre m de composants interprétables à extraire ;
— la méthode pour extraire m composants interprétables des séries temporelles ;
— le nombre N de voisins à générer pour apprendre le proxy ;
— la fonction $transform_{ts}$;
— le processus d'apprentissage du proxy ;
— le nombre de composants interprétables k que le proxy doit utiliser, $1 \leq k \leq m$.

Les valeurs testées de N, de m et de k sont spécifiques à chaque expérimentation. En ce qui concerne la méthode d'extraction des composants interprétables de la série temporelle, nous avons choisi une segmentation qui découpe les séries temporelles en m segments consécutifs de taille équivalente. Pour la fonction $transform_{ts}$, nous testons les fonctions présentées dans la Section 6.3 soit **Interpolation linéaire**, **Constante** (avec pour constante, la moyenne de la série temporelle expliquée) et **Arrière-plan aléatoire**. Enfin, pour le processus d'apprentissage du proxy, nous testons les solutions proposées dans LIME et SHAP, respectivement appelées app_{LIME} et app_{SHAP}.

Notons que l'apprentissage d'un seul proxy pour expliquer la classification d'une série temporelle nécessite de classifier les N voisins générés par le classifieur expliqué. En étendant cette opération sur l'ensemble d'un jeu de test, cela veut dire que nous appelons le classifieur N fois la taille du jeu de test. Par conséquent, le temps pour évaluer un jeu de donné sur un classifieur en particulier est très variable. Sur les plus petit jeux avec un classifieur rapide, une expérimentation prend moins d'une heure, à l'inverse un classifieur lent sur un jeu de données conséquent peut prendre

une cinquantaine d'heure. C'est la raison pour laquelle nous avons mis à profit une grille de calcul pour réaliser nos expérimentations.

6.4.2 Fidélité avec des classifieurs interprétables

Dans cette expérimentation, nous évaluons la fidélité entre un classifieur déjà considéré comme interprétable et son proxy appris par LEFTIST. Pour rappel, nous considérons qu'un classifieur est interprétable s'il est possible d'en extraire une explication compréhensible pour un utilisateur pour n'importe quelle donnée classifiée. Comme le classifieur utilisé et le proxy sont tous les deux interprétables nous allons évaluer leur fidélité en comparant leurs explications pour la classification de mêmes données. Si le proxy est fidèle au classifieur alors les explications extraites du proxy doivent être similaires à celles fournies par le classifieur.

Notre objectif est de savoir si les explications retournées par le classifieur correspondent aux explications données par notre proxy. Dans notre cas, un proxy est un modèle linéaire (nous utilisons les proxys appris par LIME et SHAP). Les attributs qui ont contribué à la classification sont ceux dont le coefficient associé est positif. Les attributs manipulés par le proxy sont les composants interprétables extraits de la série expliquée, dans notre cas des sous-séries de la série temporelle. Par conséquent, l'explication tirée de notre proxy a pour attribut des sous-séries. Comme nous comparons l'explication de notre proxy avec l'explication du classifieur, ce dernier doit aussi utiliser comme attributs des sous-séries. À notre connaissance, un seul type de modèle de classification utilise des sous-séries : les classifieurs à base de shapelets. Pour rappel, une shapelet est une sous-série dont la distance avec les séries temporelles peut être utilisée pour discriminer leur classe.

Parmi les classifieurs de séries temporelles à base de shapelet de l'état de l'art, seulement deux peuvent être considérés comme interprétables : treeShapelet [YK09] et *Fast Shapelet* [KR13]. Le dernier est une variante du premier plus précis et plus rapide, c'est la raison pour laquelle nous avons sélectionné Fast Shapelet comme classifieur interprétable pour notre évaluation. Ce modèle est interprétable car il utilise un arbre de décision qui partitionne l'espace des séries temporelles selon leurs distances avec un ensemble de shapelets.

La suite de cette partie présente comment nous comparons les explications du proxy avec les explications extraites de Fast Shapelet, et analyse les résultats.

Protocole expérimental

Soit une série temporelle ts donnée et le classifieur Fast Shapelet f. Notre méthode LEFTIST apprend le proxy g_{ts} pour expliquer la classification de ts par f. À partir de g_{ts}, nous extrayons l'ensemble des composants interprétables CIC qui ont contribués positivement dans la classification $f(ts)$ (les composants dont les coefficients sont positifs dans g_{ts}).

Avec Fast Shapelet, nous obtenons non seulement des shapelets qui ont été utilisées pour classifier la série temporelle ts mais aussi les fenêtres temporelles dans ts où ces shapelets sont présentes. Ces informations vont nous servir de vérité terrain pour évaluer notre proxy. En effet, il nous suffit maintenant de comparer les fenêtres temporelles utilisées par Fast Shapelet aux fenêtres temporelles retournées par le proxy (les fenêtres des composants de CIC). Si la présence d'une shapelet a participé à la classification de ts alors la zone de cette shapelet doit être couverte par les composants CIC de l'explication retournée par le proxy.

Pour évaluer l'adéquation entre les deux ensembles de fenêtres temporelles, nous définissons une fonction de couverture. Pour une shapelet qui a participé positivement à la classification dans Fast Shapelet, la fonction calcule le pourcentage de la fenêtre associée à la shapelet qui est couvert par les fenêtres des composants interprétables de CIC. Plus cette valeur est élevée et plus nous estimons que l'explication de notre proxy est valide avec f.

Notre méthode pour évaluer la couverture des shapelets sur un jeu de données consiste à récupérer l'ensemble des shapelets apprises par le classifieur sur le jeu. Pour chaque shapelet nous énumérons les séries du test, si la shapelet a contribué à la classification d'une série par Fast Shapelet alors sa couverture est calculée puis enregistrée dans une liste. Une fois toutes les shapelets parcourues, la médiane des couvertures pour chaque shapelet est calculée, puis la médiane des résultats sur l'ensemble des shapelets est retournée. Ce calcul est réalisé pour tous les jeux de données, et la médiane des couvertures sur l'ensemble des jeux de données est retournée. Ce processus est présenté dans l'Algorithme 5.

Dans cette expérimentation, le nombre de composants interprétables m est fixé à 10 , le nombre N de voisins à générer à 1000 et nous faisons varier k entre 1 et 10. Nous avons lancé cette expérimentation sur toutes les combinaisons possibles entre les $transform_{ts}$ testées (**Interpolation linéaire**, **Constante** et **Arrière-plan aléatoire**) et les méthodes d'apprentissage du proxy (app_{LIME} et app_{SHAP}).

Algorithme 5 Evaluation de LEFTIST avec FS

Entrées : app méthode d'apprentissage du proxy,
m le nombre de composants interprétables à extraire,
N le nombre de voisins à générer,
D l'ensemble des jeux de séries temporelles.

Sortie : $mediane_k$ la médiane des médianes de couvertures de shapelets dans tous les jeux de D pour chaque taille d'explication k du proxy.

$mediane_k \leftarrow \emptyset$
pour tout $k\ entre\ 1\ et\ m$ **faire**
  $couv_k \leftarrow \emptyset$
  pour tout $d \in D$ **faire**
    $f \leftarrow apprentissage_FS(d.train)$// *apprentissage du classifieur* f
    $couv_jeu_k \leftarrow \emptyset$ // *initialisation d'un dictionnaire pour stocker les couvertures des shapelets dans les séries du jeu* d *pour une taille d'explication du proxy* k
    pour tout $ts \in d.test$ **faire**
      $c, shapelets_presente \leftarrow f(ts)$// *lors de la classification par* f *nous enregistrons la position des shapelets dont la présence a contribué à la classification de* ts
      $proxy \leftarrow LEFTIST(ts, c, f, app, m, N, k)$
      $S \leftarrow extraire_segments_contributifs(proxy)$
      pour tout $sh \in shapelets_presente$ **faire**
        $couv \leftarrow couverture_shapelet_par_segment_contributif(sh.zone, S)$
        $couv_jeu_k[sh.id].ajoute(couv)$
      fin pour
    fin pour
    $couv_k.ajoute(mediane(couv_jeu_k))$
  fin pour
  $mediane_k.ajoute(mediane(couv_k))$
fin pour
retourne $mediane_k$

Résultats

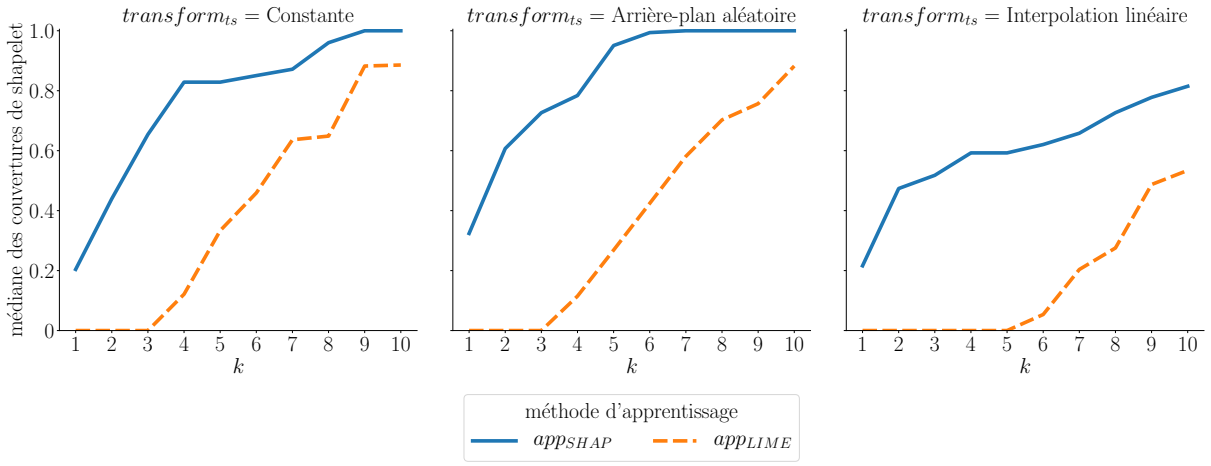


FIGURE 6.9 – Médiane de médiane des couvertures des shapelets de FS par les segments contributifs du proxy.

La Figure 6.9 montre la médiane des couvertures des shapelets sur les 25 jeux de données selon le nombre de composants interprétables utilisés pour l'apprentissage du proxy. Il y a un graphe par fonction $transform_{ts}$, et deux courbes par graphe, une pour chaque processus d'apprentissage. La conclusion la plus importante est qu'une bonne couverture peut être obtenue avec un nombre limité de composants interprétables dans les explications (≈ 4 composants). Les explications apprises avec **Arrière-plan aléatoire** sont celles qui capturent le mieux les shapelets de FS, tandis que celles apprises avec **Interpolation linéaire** sont les moins bonnes.

Notre interprétation est que les voisins générés par la fonction $transform_{ts}$ **Interpolation linéaire** sont beaucoup trop proches de la série temporelle ts (en terme de similarité). Par conséquent le proxy approxime le classifieur dans un voisinage trop près de ts et n'est pas assez générique. Cette hypothèse est confirmée avec la Figure 6.10 qui présente la distribution des distances euclidiennes calculées entre ts et ses voisins représentés dans l'espace des séries temporelles initial selon chaque fonction $transform_{ts}$. Ces valeurs ont été calculées pour une génération de 1000 voisins pour chaque série temporelle du jeu de test de *GunPoint*. Nous pouvons voir que la transformation **Interpolation linéaire** est celle qui génère le plus de voisins similaires à ts (distance proche de la valeur 0).

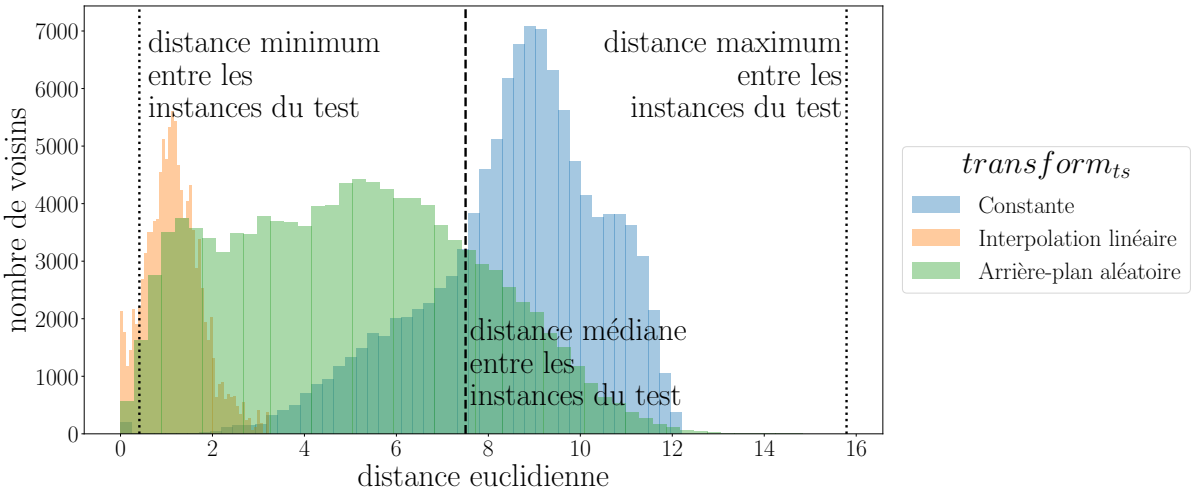


FIGURE 6.10 – Distribution des distances entre les séries du jeu de test de *GunPoint* et leurs 1000 voisins respectifs par fonction $transform_{ts}$.

6.4.3 Fidélité avec des classifieurs non-interprétables

À la différence des classifieurs interprétables, nous ne pouvons pas extraire une explication de classification compréhensible pour un utilisateur à partir d'un classifieur non-interprétable. La seule information qui peut être utilisée pour évaluer la fidélité avec un classifieur non-interprétable est le résultat d'une classification. Un classifieur et son proxy sont fidèles, s'il retournent les mêmes classifications pour les mêmes données. Cependant, comme dans notre cas le proxy approxime le classifieur seulement autour d'une donnée à expliquer, la fidélité de classification est recherchée seulement dans le voisinage de cette donnée.

C'est sur ce principe que repose l'évaluation de fidélité présentée dans cette partie. Elle consiste, pour un classifieur non-interprétable et son proxy, à générer un ensemble de séries temporelles dans le voisinage de la donnée expliquée, à les classifier par le classifieur et son proxy, et enfin à calculer la précision de la fidélité entre les classifications des deux méthodes.

Definition 6.4.1. Précision de la fidélité
Soit f un classifieur déjà appris, g un proxy déjà appris, et V un ensemble de séries temporelles. La précision de fidélité entre f et g sur V se calcule de la manière suivante :

$$précision(f, g, V) = \frac{TP}{|V|}$$

avec TP le nombre de séries temporelles dans V dont la classe retournée par f est la même que celle retournée par g.

Dans la suite de cette partie, nous présentons en détail la méthode d'évaluation, ainsi que l'analyse des résultats.

Protocole expérimental

Soit un classifieur f non-interprétable, un ensemble de séries temporelles D et une série temporelle $ts \in D$. Nous apprenons un proxy g_{ts} qui explique le classification $f(ts)$. L'évaluation de la fidélité entre g_{ts} et f consiste à calculer la précision de la fidélité, c'est-à-dire le pourcentage de concordance entre les classifications de f et g_{ts} dans le voisinage de ts.

Pour cette expérimentation, nous avons besoin de générer des séries temporelles ts' dans le voisinage de ts. Notre proposition consiste à copier la série ts et à perturber les valeurs de certains points selon deux paramètres :

- $ratio_{nbpoint} \in [0, 1]$ correspondant au pourcentage de points à modifier dans ts,
- $ratio_{amplitude} \in [0, 1]$ correspondant au maximum d'amplitude de modification des valeurs des points perturbés.

En pratique, $(ratio_{nbpoint} * size(ts))$ points de données sont modifiés dans ts. Chacun de ces points se voit ajouter une valeur tirée uniformément à partir de l'intervalle de valeur suivant :

$$[-((max(ts) - min(ts)) * ratio_{amplitude}), +(max(ts) - min(ts)) * ratio_{amplitude}]$$

où $max(ts)$ est la valeur maximum dans ts et $min(ts)$ la valeur minimum. Nous nous assurons ainsi que les copies générées restent dans le voisinage de ts.

Dispersion des copies perturbées selon $ratio_{nbpoint}$ et $ratio_{amplitude}$

Dans cette sous-partie, nous analysons la dispersion des copies perturbées autour de leur série originale en fonction de $ratio_{nbpoint}$ et $ratio_{amplitude}$. La Figure 6.11 présente la distribution des distances euclidiennes entre 100 copies perturbées et leur

série originale, pour l'ensemble des séries temporelles du test du jeu de données *GunPoint*. Plus une distance est faible et plus la copie perturbée est similaire à sa série originale. Nous avons testé les quatre combinaisons possibles entre $ratio_{nbpoint} \in \{0.1, 0.9\}$ et $ratio_{amplitude} \in \{0.1, 0.9\}$, leur distribution de distances étant identifiée par une couleur. Nous voyons que quand les paramètres de modification sont très faibles ($ratio_{nbpoint} = ratio_{amplitude} = 0.1$) les copies créées sont toutes très similaires à leur série originale. À l'inverse avec des paramètres très élevés ($ratio_{nbpoint} = ratio_{amplitude} = 0.9$) les copies peuvent être loin de leur série originale. Pour s'assurer que les copies perturbées soient dans le voisinage de leur série originale, les paramètres de modification doivent donc être faibles ou bien de valeurs opposées.

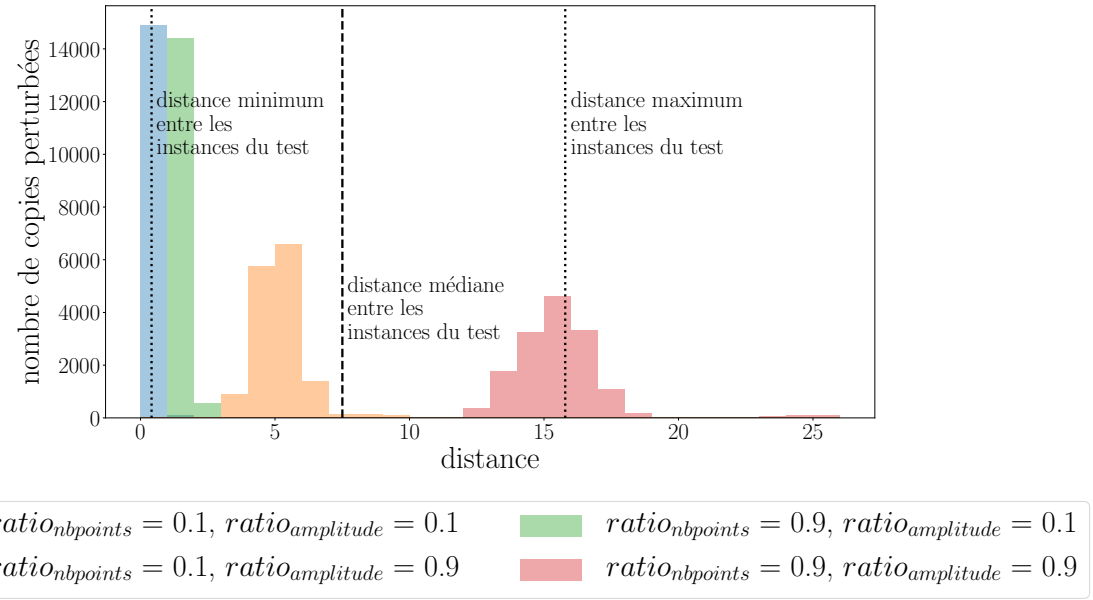


FIGURE 6.11 – Distribution des distances euclidiennes des copies de séries temporelles avec leur original selon différentes valeurs de $ratio_{nbpoint}$ et $ratio_{amplitude}$.

Une fois les copies perturbées ts' de ts générées. Nous avons besoin de les représenter dans l'espace de représentation en composants interprétables IF pour que nous puissions les classifier par g_{ts}. Pour cela, nous étendons l'espace $IF = \{0, 1\}^m$ en l'espace continu $CIF = [0, 1]^m$. Pour représenter ts' dans CIF, nous considérons chaque segment de ts' issu de la segmentation en composants interprétables, et nous donnons pour chaque composant une valeur dans $[0, 1]$ qui correspond à la proportion de points dans ts qui n'ont pas été modifiés pour générer ts'. Il peut être noté que si aucun point du segment n'est modifié alors l'attribut associé est à 1 et si tous les points

dans le segment sont modifiés alors l'attribut est égal à 0.

Par définition, g_{ts} est défini par :

$$\begin{aligned} g_{ts} : \quad & \{0,1\}^m \to \mathbb{R} \\ & \vec{v} = (v_1, \ldots, v_m) \mapsto \phi_0 + \sum_{i=1}^{m} \phi_i * v_i \end{aligned}$$

Or les voisins à notre disposition sont dans l'espace continu $CIF = [0,1]^m$. Nous devons donc étendre le proxy g_{ts} sur l'espace continu CIF pour pouvoir classifier les copies perturbées. Nous définissons ce proxy g^e_{ts} de la manière suivante :

$$\begin{aligned} g^e_{ts} : \quad & [0,1]^m \to \mathbb{R} \\ & \vec{v} = (v_1, \ldots, v_m) \mapsto \phi_0 + \sum_{i=1}^{m} \phi_i * v_i \end{aligned}$$

puisque le proxy g_{ts} est une combinaison linéaire des éléments de IF, il peut être utilisé aussi sur des éléments de CIF.

Maintenant que nous avons introduit la méthode pour générer un voisinage et la façon de le classifier par g_{ts}, nous pouvons calculer la fidélité entre un classifieur f et son proxy g_{ts} sur un jeu de données. Pour chaque série du jeu de test, nous générons 100 copies perturbées dont la classification par f et g_{ts} (en utilisant g^e_{ts}) est utilisée pour calculer la précision de la fidélité entre f et g_{ts} pour cette série. La précision de fidélité retournée pour un jeu de données est la médiane des précisions calculées sur les séries du jeu. Enfin, la précision de fidélité sur l'ensemble des jeux de données est la médiane des précisions de tous les jeux de données. L'Algorithme 6 présente cette évaluation.

Dans cette expérimentation, nous évaluons l'évolution de la précision de la fidélité dans le voisinage des séries temporelles pour différents paramétrages de LEFTIST.

Pour la génération du voisinage, nous fixons le nombres de copies v à 100 et le $ratio_{amplitude}$ à 20. La variation sur le voisinage est confiée au paramètre $ratio_{nbpoints}$ dans $\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. Nous nous assurons ainsi d'avoir des copies qui couvrent un voisinage proche des séries temporelles.

Nous avons sélectionné trois classifieurs non-interprétables de l'état de l'art : une machine à vecteurs de support (SVM) implémentée dans [TFV17], le classifieur à base de shapelets Learning Shapelet [Gra+14] (LS) et un réseau de neurones profond [WYO17] basé sur l'architecture ResNet [Efr+04] (RESNET). Les paramétrages de LEFTIST testés comprennent toutes les combinaisons possibles entre :

— les $transform_{ts}$ testées (Interpolation linéaire, Constante et Arrière-plan aléa-

Algorithme 6 Evaluation de LEFTIST avec un classifieur non-interprétable

Entrées : app méthode d'apprentissage du proxy,
k le nombre de composants interprétables à utiliser par le proxy,
m le nombre de composants interprétables à extraire,
N le nombre de voisins à générer pour l'apprentissage du proxy,
D l'ensemble des jeux de séries temporelles,
f le classifieur non-interprétable,
$ratio_{nbpoint}$ et $ratio_{amplitude}$ les ratios pour générer les copies perturbées,
v le nombres de copies perturbées a générer par série.

Sortie : $mediane_precision_globale$ la médiane de précision de fidélité calculée pour tous les jeux de D entre les proxys de LEFTIST et le classifieur f.

$mediane_precision_globale \leftarrow 0$
$precision_globale \leftarrow \emptyset$
pour tout $d \in D$ **faire**
  $precision_jeu \leftarrow \emptyset$
  $f \leftarrow f.apprendre(d.ens_apprentissage)$
  pour tout $ts \in d.ens_test$ **faire**
    $T' \leftarrow generer_copies_perturbees(ts, ratio_{nbpoint}, ratio_{amplitude}, v)$
    $c \leftarrow f(ts)$
    $g_{ts} \leftarrow LEFTIST(ts, c, f, app, m, N, k)$
    $prediction_concordance \leftarrow \emptyset$
    pour tout $ts' \in T'$ **faire**
      si $f(ts') == g_{ts}(ts')$ **alors**
        $prediction_concordance.ajoute(1)$
      sinon
        $prediction_concordance.ajoute(0)$
      fin si
    fin pour
    $precision \leftarrow calculer_precision(prediction_concordance)$
    $precision_jeu.ajoute(precision)$
  fin pour
  $precision_globale.ajoute(precision_jeu)$
fin pour
$mediane_precision_globale \leftarrow mediance(precision_globale)$
retourne $mediane_precision_globale$

toire) ;
— les méthodes d'apprentissage du proxy (app_{LIME} et app_{SHAP}) ;
— les trois classifieurs donnés ;
— les valeur $m \in \{5, 10, 20\}$.

Enfin, le nombre de composants à utiliser pour l'apprentissage des proxys est fixé à $k = m$.

Évaluation du nombre de voisins N sur la fidélité

Dans cette sous-partie, nous analysons l'influence du nombre de voisins N sur la fidélité entre un classifieur et son proxy appris par LEFTIST. Nous avons lancé notre évaluation de la fidélité avec m fixé à 20, $ratio_{nbpoints}$ à 50 et $transform_{ts} =$ Arrière-plan aléatoire. Les autres paramètres ont été testés avec les valeurs présentées précédemment. La Figure 6.12 présente la distribution des précisions de fidélité sur l'ensemble des jeux de données, selon différents nombres de voisins $N \in \{40, 60, 80, 100, 200, 300, 500, 700, 1000, 3000\}$. Pour app_{LIME} le nombre de voisins n'affecte pas drastiquement la médiane des précisions pour l'ensemble des classifieurs. À l'inverse, pour app_{SHAP} la précision est la meilleure puis se stabilise à partir de 300 voisins. Cette évaluation valide notre nombre de voisins $N = 1000$, sachant qu'à partir de 300 voisins la précision reste stable sur toutes les configurations de LEFTIST.

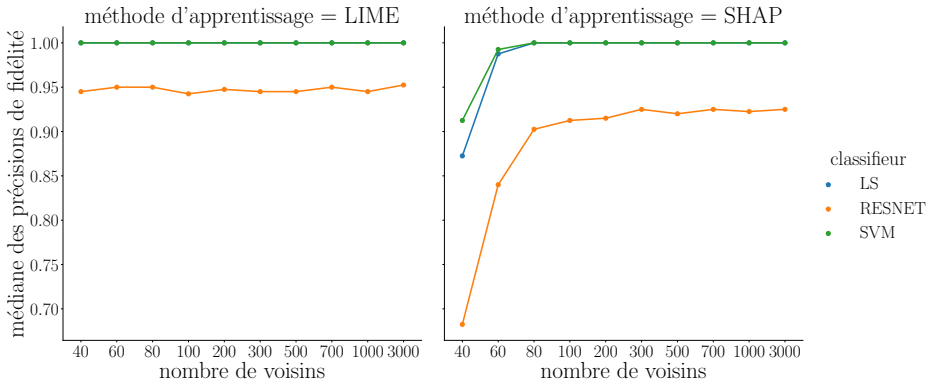


FIGURE 6.12 – Médiane des précisions de fidélité sur les 25 jeux de données selon la méthode d'apprentissage du proxy, le classifieur et le nombre de voisins (pour LIME les résultats de LS et SVM sont superposés).

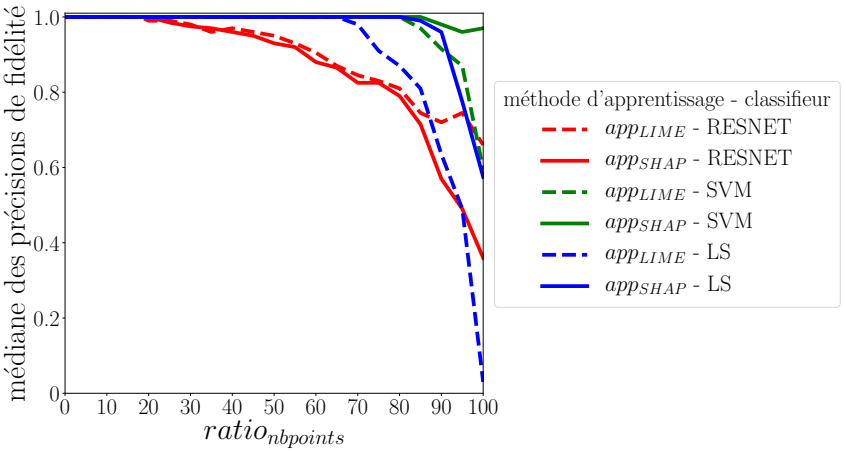


FIGURE 6.13 – Médiane des précisions de la fidélité selon $ratio_{nbpoints}$, les classifieurs et les méthodes d'apprentissage.

Résultats

La figure 6.13 montre l'évolution de la médiane, sur tous les jeux de données, de la précision de la fidélité selon $ratio_{nbpoints}$, les classifieurs et la méthode d'apprentissage. Pour cette évaluation, nous avons fixé $m = 10$ et $transform_{ts} =$ Arrière-plan aléatoire. Nous pouvons observer que pour tous les classifieurs, la précision reste très élevée jusqu'à 50% de points modifiés dans les séries temporelles : cela veut dire que LEFTIST génère des explications qui approximent fidèlement les classifieurs dans un voisinage assez large autour de l'instance à expliquer. La Figure 6.14 présente deux distribution des distances euclidiennes entre 100 copies perturbées et leur série originale, pour l'ensemble des séries temporelles de test du jeu de données *GunPoint*. Les distributions sont différentes selon si les copies sont classées de la même façon ou différemment, par f et par g_{ts}. Nous pouvons observer que le nombre de copies dont la classification est différente entre f et g_{ts} augmente avec la distance entre la copie et son original. Cela confirme que le proxy approxime f uniquement dans le voisinage des séries expliquées.

Parmi les classifieurs, la précision de RESNET descend en premier. Il s'agit d'un résultat attendu, puisque c'est le classifieur avec la surface de décision la plus complexe et donc la plus difficile à approximer. Nous fixons le classifieur à RESNET (le plus difficle à approximer) pour évaluer l'influence des fonctions $transform_{ts}$ sur la fidélité. Les résultats sont affichés dans la Figure 6.15.

Ici, il est intéressant de noter que les différentes fonctions $transform_{ts}$ se différen-

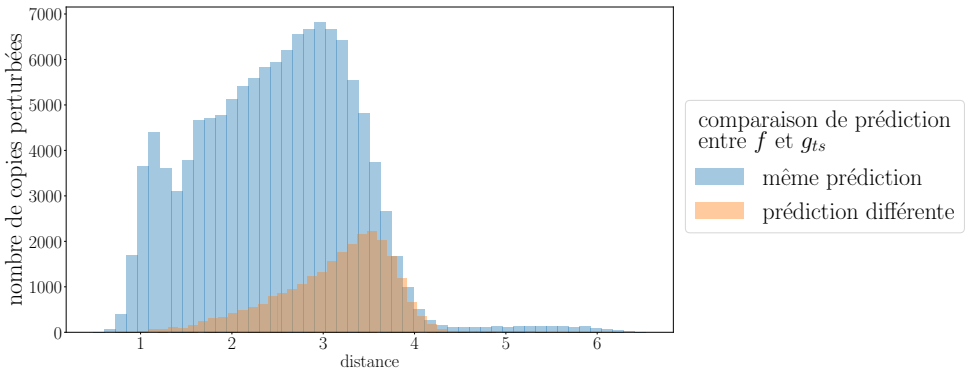


FIGURE 6.14 – Distribution des distances euclidienne des copies de séries temporelles avec leur original, ainsi que la comparaison de leur classification par f et g_{ts}.

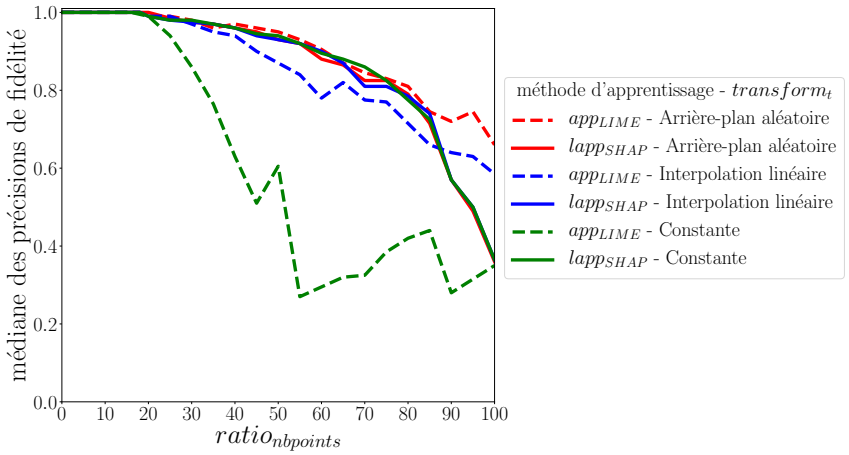


FIGURE 6.15 – Médiane des précisions de la fidélité selon $ratio_{nbpoints}$, les fonctions $transform_{ts}$ et les méthodes d'apprentissage.

cient très peu les unes par rapport aux autres avec app_{SHAP}. Cela peut s'expliquer par la façon dont SHAP initialise l'apprentissage du proxy. Dans ce dernier, le ϕ_0 de g_{ts} est appris à partir de l'ensemble d'apprentissage. Cet apprentissage est le même pour toutes les configurations de LEFTIST testées avec app_{SHAP}. Par conséquent ϕ_0 est le même pour toutes ces configurations. Or, quand $ratio_{nbpoints}$ tend vers 100% alors la classification de g_{ts} tend vers ϕ_0 (qui est commun peu importe la série ou la configuration évaluée). Cela explique la convergence des résultats des fonctions $transform_{ts}$ avec la méthode d'apprentissage du proxy de SHAP.

Pour LIME, la fonction $transform$ la moins performante est la fonction **Constante**. Nos expérimentations montrent que cette transformation basique peut générer des voisins très éloignés des instances à expliquer, ne permettant pas à g_{ts} de bien approximer la surface de décision de RESNET. Sur la Figure 6.10 nous observons que c'est la fonction **Constante** qui créée les représentations des voisins dans l'espace initial les plus éloignées des séries à expliquer, ce qui rend plus difficile l'approximation du voisinage des séries à expliquer.

Enfin, nous étudions l'influence du nombre de composants interprétables m sur la fidélité. Les résultats sont présentés dans la Figure 6.16. Pour cela, nous fixons la fonction $transform_{ts} =$ *Arrière-plan aléatoire* car c'est elle qui fournit les meilleurs résultats.

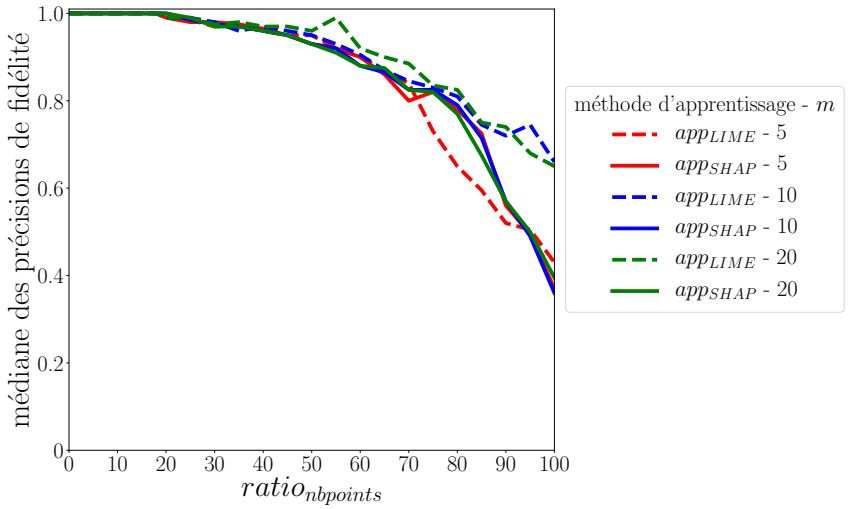


FIGURE 6.16 – Médiane des précisions de la fidélité selon $ratio_{nbpoints}$, le nombre de composants interprétables m et les méthodes d'apprentissage.

Le nombre de composants interprétables a un impact limité sur le processus d'ap-

prentissage de SHAP. Ceci est dû une nouvelle fois à la convergence vers ϕ_0, qui ne dépend pas non plus de la valeur de m. Pour la méthode d'apprentissage du proxy de LIME, nous pouvons seulement émettre l'hypothèse que LIME a du mal à approximer le classifieur quand le nombre de composants interprétables est trop faible, c'est-à-dire quand la représentation interprétable choisie est trop simple.

6.5 Évaluation utilisateur

Dans la Section 3.3, nous avons présenté un ensemble de méthodes pour évaluer l'interprétabilité d'explications de modèles. Parmi ces méthodes d'évaluation, de plus en plus d'entre elles se basent sur des études utilisateur [RSG16 ; LL17 ; RSG18]. Ces études consistent à interroger via un questionnaire des utilisateurs humains sur leur compréhension d'explications de résultats de modèles.

Dans cette section, nous évaluons l'interprétabilité des explications extraites des proxys appris par LEFTIST au travers d'une étude utilisateur. Cette section est séparée en deux parties. La première présente le protocole expérimental. La seconde est consacrée à l'analyse des résultats de l'étude

6.5.1 Construction du questionnaire

La question qui nous intéresse est "Est-ce qu'une explication *aide* un expert des séries temporelles concernées à comprendre la prédiction retournée par un classifieur boîte noire ?". Comme la notion "d'aide" est difficile à définir, nous traduisons la question par "Est-ce que l'explication seule permet à un expert des séries temporelles de retrouver la classe retournée par un classifieur ?".

La construction du questionnaire s'organise autour de deux points : le public adressé et la nature des questions. Interroger de vrais experts sur leur données est difficile et coûteux à mettre en place. Pour remédier à ce problème, nous avons décidé d'interroger des personnes non-expertes. Cependant, cela nécessite la formation des utilisateurs pour répondre aux questions. Nous avons mis en place un tutoriel introduisant à la fois les séries temporelles, la classification et la lecture des explications. Le public interrogé est majoritairement composé de personnes travaillant ou étudiant dans un domaine scientifique, si possible en informatique/électronique. En effet, les séries

temporelles sont un type de données plus complexe que les images ou le texte et qui est rencontré plus souvent dans des parcours scientifiques.

Dans la suite nous présentons d'abord le tutoriel, puis nous montrons comment les questions ont été choisies.

Tutoriel

Le tutoriel a pour but d'initier l'utilisateur à la tâche de classification sur les séries temporelles, et à la lecture des explications extraites des proxys de LEFTIST.

Le tutoriel se présente sous la forme de trois questions :

— Qu'est ce qu'une série temporelle ? (cf. Figure 6.17)
— Comment classifier une série temporelle ?
— Comment lire une explication ?

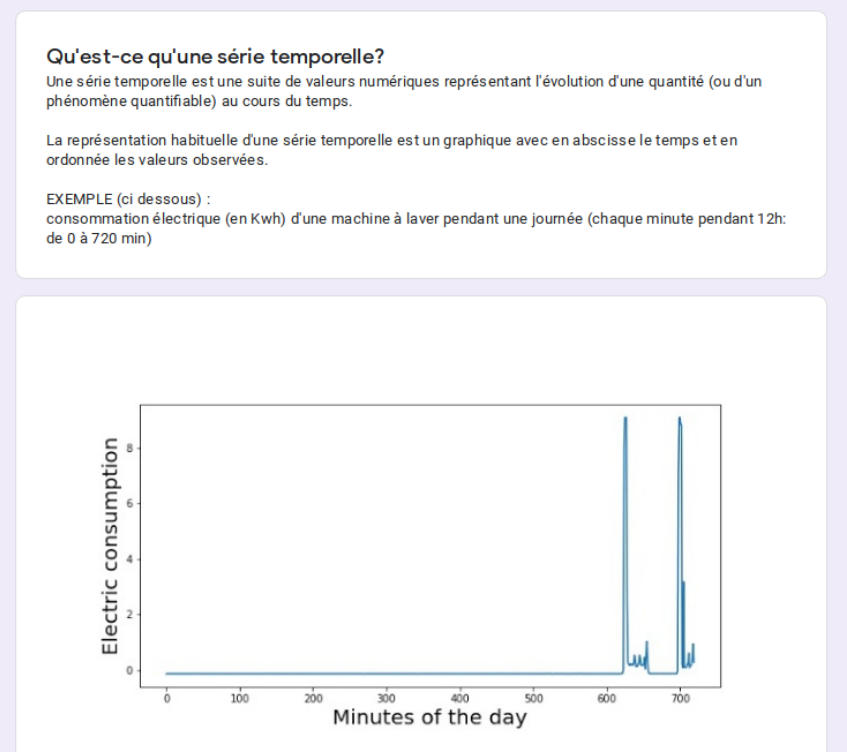


FIGURE 6.17 – Tutoriel : Qu'est ce qu'une série temporelle ?

Enfin, un test composé de 5 questions (cf. Figure 6.18) est réalisé pour vérifier l'acquisition de ces connaissances. Les réponses des utilisateurs, qui n'arrivent à pas à passer le test parfaitement, ne sont pas prises en compte dans l'analyse des réponses au questionnaire.

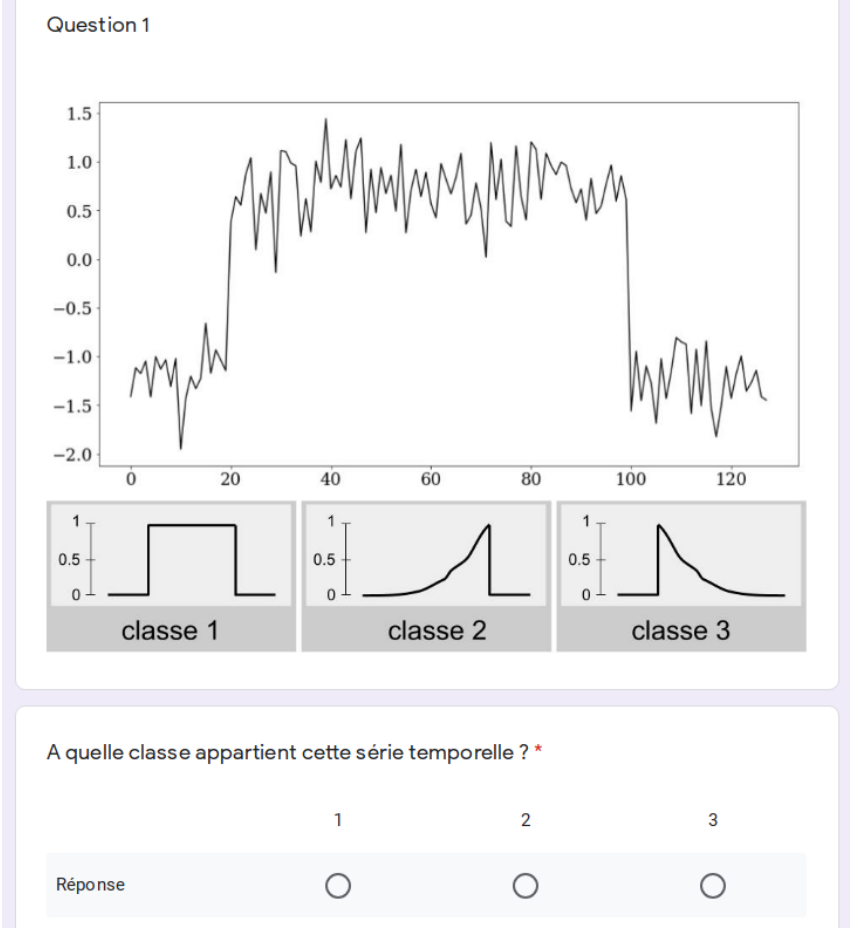


FIGURE 6.18 – Tutoriel : question d'évaluation de l'utilisateur

Questions

L'étude utilisateur est un ensemble de questions qui demande à l'utilisateur de déduire la classe retournée par le classifieur à partir de l'explication extraite du proxy appris par LEFTIST. Nous comparons les classes assignées par les utilisateurs et celles retournées par le classifieur pour calculer la précision.

Nous nous sommes concentrés sur des jeux de données simples de l'archive UCR : CBF (simulé, taille des séries temporelles = 128 points, 3 classes), Trace (simulé, taille des séries temporelles = 125 points, 4 classes) et GunPoint (réel, taille des séries temporelles 150 points, 2 classes). Dans ces jeux de données, des motifs caractéristiques permettent de facilement différencier les classes à l'œil nu. Ces motifs sont connus préalablement et doivent permettent à des utilisateurs lambda de devenir rapidement des "experts" de ces jeux de données de séries temporelles.

Nous construisons une question de la manière suivante. Nous commençons par sélectionner une série temporelle ts. Puis, un proxy est appris par LEFTIST pour expliquer la classification retournée par un classifieur. Nous utilisons les explications générées pendant les expérimentations de la section 6.4. Le classifieur choisi est RESNET et la fonction $transform_{ts}$ est Arrière-plan aléatoire.

Le proxy appris est un modèle linéaire simple. L'explication extraite est constituée des coefficients associés aux composants interprétables. Ces coefficients sont interprétés comme la contribution des composants interprétables dans la classification de ts. À partir de cette explication, nous extrayons uniquement les segments dont la contribution est positive dans la classification de ts. Les segments sont coloriés en vert plus ou moins prononcé, selon le poids de la contribution (vert sombre correspond à une contribution élevée, vert clair à une contribution faible). Enfin, les segments sont présentés sous leur forme graphique à l'utilisateur. Nous lui demandons d'identifier la classe retournée par le classifieur. Nous permettons à l'utilisateur de répondre "je ne sais pas". Une question est illustrée dans la Figure 6.19.

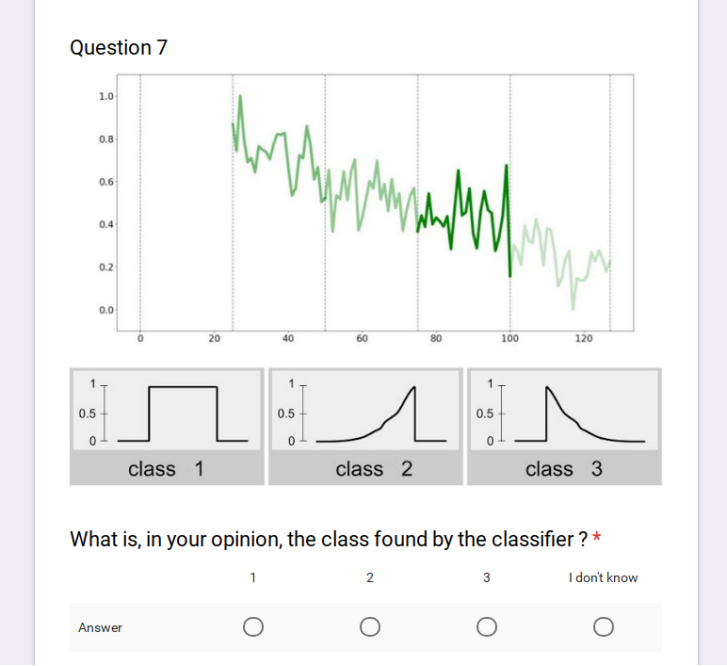


FIGURE 6.19 – Un exemple de question dans l'étude utilisateur.

Le temps nécessaire pour répondre au questionnaire est un paramètre important. Une étude trop longue mènera à des réponses erronées à cause d'une perte de concentration de la part des interrogés. Un temps raisonnable nous a semblé être 15 minutes. Nous avons réalisé un test préliminaire sur des collègues pour déterminer

le nombre de questions qui peuvent êtres répondues dans ce délai. Nous avons abouti à 33 questions.

Nous avons sélectionné empiriquement des explications qui nous semblaient faciles, moyennement faciles et difficiles pour retrouver la classe retournée par le classifieur. Nous avons aussi souhaité avoir des séries temporelles mal classifiées par RESNET pour évaluer la robustesse des explications. Comme le classifieur a de très bonnes performances, nous avons appris une version dégradée de RESNET (en réduisant le nombre d'optimisation du réseau de neurones durant l'apprentissage) sur les 3 jeux de données pour créer 3 questions parmi les 33 où la classification de RESNET n'est pas en accord avec la classe réelle.

Le questionnaire a été réalisé en anglais et en français, sur internet via une application en ligne pour créer des questionnaires gratuitement. Ce questionnaire est anonyme et a été principalement envoyé à des collègues et des étudiants en informatique/électronique.

6.5.2 Résultats

Nous avons collecté les réponses de 214 personnes interrogées. Après vérification des résultats au test du tutoriel, seulement les réponses de 194 utilisateurs ont été conservées.

Analyse quantitative

Les utilisateurs interrogés peuvent être regroupées dans trois catégories : ceux qui travaillent ou étudient en informatique/électronique (75%), ceux qui travaillent ou étudient dans un autre domaine scientifique (15%), ceux qui travaillent et étudient dans un domaine non-scientifique (10%).

La moyenne du pourcentage de bonnes réponses est de 0.77 quand on considère les 33 questions. La précision monte à 0.82 si on considère uniquement les 30 questions où le classifieur ne s'est pas trompé sur la classification. Nous pouvons en conclure que les explications ont aidé l'utilisateur à identifier la classe retourner par le classifieur.

La répartition des pourcentages de bonnes réponses selon la catégorie de l'utilisateur est présenté dans la Figure 6.20. Nous pouvons noter qu'il n'y a pas de différence flagrante entre les pourcentages de bonnes réponses selon les catégories. Cela laisse

supposer que le tutoriel a permis aux utilisateurs de s'approprier ce type de données et la tâche demandée.

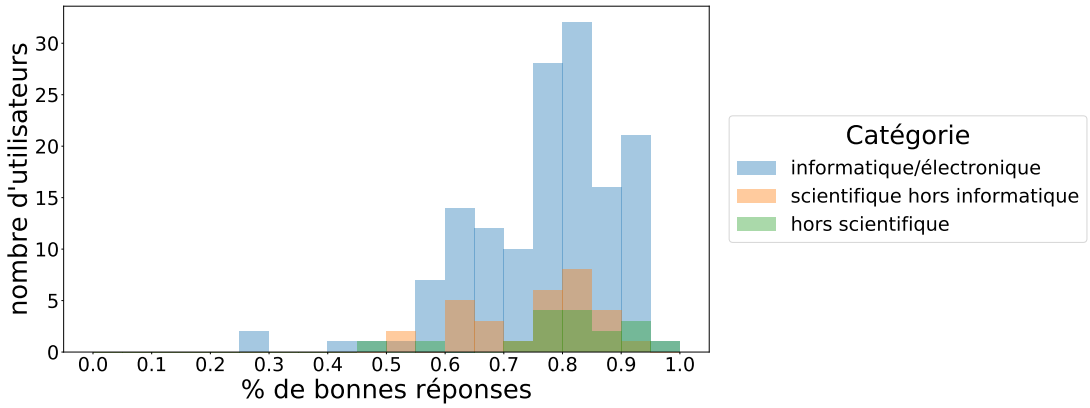


FIGURE 6.20 – Distribution du pourcentage de bonnes réponses par nombre d'utilisateurs et par catégorie.

Analyse qualitative

Suite au questionnaire, nous avons pu discuter avec une quinzaine de collègues/étudiants sur leur ressenti et les réponses données.

Une première remarque concerne la longueur du questionnaire. Pour certains le nombre de questions était trop important. La lassitude engendrée a induit alors des stratégies de réponses inadaptées (ex : toujours répondre "je ne sais pas").

Un second point concerne la perception des explications. Certaines personnes ont utilisé le degré de contribution des segments (dégradé de couleur) pour répondre. D'autres ne se sont préoccupés que de la forme générale de l'explication (concaténation de tous les segments).

Enfin, certaines personnes ont pris en compte l'amplitude et la forme de la courbe alors que d'autres se sont concentrés sur la forme de la courbe. Ces deux derniers points montrent la difficulté de présenter des explications concernant des séries temporelles à des utilisateurs.

6.6 Conclusion

Dans ce chapitre avons présenté LEFTIST, une méthode pour expliquer localement la prédiction de n'importe quel classifieur de séries temporelles. Nous nous sommes

inspirés du fonctionnement d'une famille de méthodes qui utilisent un proxy pour expliquer un résultat de n'importe quel classifieur. À partir de cette famille, nous avons défini le processus PIAEL, qui peut être employé pour créer des méthodes d'explication a posteriori qui soient agnostiques et locales. Notons que ce processus peut être adapté à tout type de données, et pas seulement aux séries temporelles. Nous avons utilisé ce processus à été utilisé pour définir LEFTIST, la première méthode d'explication locale et agnostique de classification de séries temporelles.

Nous avons évalué la fidélité des proxys appris par LEFTIST à partir d'un ensemble de classifieurs interprétables (arbre de décision à base de shapelets) et non-interprétables (un SVM, un réseau de neurones et un classifieur à base de shapelets non-interprétable). Les explications extraites des proxys concordent avec celles tirées de l'arbre de décision à base de shapelets. Les résultats montrent que notre méthode apprend un proxy capable d'approximer des classifieurs non-interprétables avec une grande précision.

Une étude utilisateur a été réalisée pour évaluer l'interprétabilité des explications extraites des proxys. L'analyse des résultats montre que les explications fournies sont compréhensibles par un utilisateur humain et l'aide à identifier la classe retournée par le classifieur.

CHAPITRE 7

CONCLUSION

Lors de la mise en application d'un modèle d'apprentissage automatique pour résoudre une tâche concrète, deux problématiques sont généralement rencontrées : l'adaptation du modèle à l'application sur laquelle il est utilisé et son interprétabilité. La première est essentielle pour s'assurer que le modèle soit performant dans la tâche qu'il doit résoudre. La seconde permet de convaincre les utilisateurs que le "raisonnement" du modèle pour un résultat est correct. En effet, les utilisateurs peuvent prendre des décisions importantes à partir des résultats d'un modèle et l'absence d'explication revient à faire confiance aveuglément au modèle, ce qui peut être un frein à son utilisation. Nous avons exploré séparément ces problématiques sur deux tâches différentes d'apprentissage automatique sur des séries temporelles.

Dans la première problématique, nous nous sommes confrontés à l'adaptation de la recherche d'occurrences de règles temporelles dans des séries temporelles industrielles. Nous avons proposé d'intégrer l'utilisation de deux mesures élastiques pour la recherche d'occurrences de règles, car elles sont connues pour leur capacité à mieux évaluer la similarité en présence de légères déformations. En effet, les séries temporelles industrielles sont issues de systèmes qui subissent l'influence de nombreux facteurs pouvant perturber les occurrences de motifs présentes dans les séries temporelles. Au travers d'expérimentations sur des données réelles industrielles, nous avons montré que l'utilisation des mesures de distance élastiques a permis d'améliorer la recherche d'occurrences de règles d'intérêt.

La seconde problématique concernait l'interprétabilité dans les méthodes de classification de séries temporelles. Nous avons vu que deux approches existaient pour expliquer les résultats d'un classifieur. Dans la première approche, le classifieur est basé sur un modèle "naturellement interprétable" dont nous pouvons extraire directement une explication d'un résultat. Dans la deuxième approche, le classifieur est basé sur un modèle trop complexe pour expliquer un résultat, et dans ce cas il faut utiliser une approche a posteriori pour l'expliquer les classifications de ce modèle. Cependant,

chaque approche a un inconvénient. Les classifieurs interprétables de séries temporelles ont actuellement une moins bonne précision que les meilleures méthodes de classification. Tandis que les méthodes d'explication a posteriori de classifieurs de séries temporelles sont spécifiques à un type de classifieur ce qui réduit la portée de leur utilisation aux jeux de données où ces classifieurs fonctionnent bien.

Nous avons proposé LRS, un classifieur interprétable de séries temporelles capable de rivaliser en précision avec les méthodes de l'état de l'art. LRS est un classifieur à base de shapelets, qui en plus de la distance avec les shapelets, utilise la position des shapelets dans les séries temporelles pour mieux les discriminer. Nous avons comparé LRS avec trois classifieurs à base de shapelets de l'état de l'art, un qui est interprétable, et les deux autres non. Les résultats ont montré que LRS avait dans l'ensemble une meilleure précision que la méthode interprétable mais n'arrivait pas à égaler les classifieurs non-interprétables. Cependant, notre modèle était le meilleur sur certains jeux de données. Nous en avons tiré deux conclusions, qui sont corroborées par des études récentes [Bag+17 ; Faw+19] : les méthodes non-interprétables sont plus performantes que les méthodes interprétables, et certains types de classifieurs sont seulement performants sur un certain type de jeu de séries temporelles.

Ces conclusions nous ont mené à nous intéresser aux méthodes a posteriori d'explication de classifieurs et en particulier à la recherche d'une méthode agnostique, c'est-à-dire capable d'expliquer n'importe quel classifieur. En effet, cette caractéristique assure de pouvoir fournir une explication pour le meilleur classifieur sur n'importe quel jeu de données. Nous avons extrait, à partir de deux méthodes a posteriori d'explication agnostiques de classifieur LIME [RSG16] et SHAP [LL17], un processus pour pouvoir développer des méthodes agnostiques et locales d'explication de classifieurs. Ce processus consiste à apprendre un classifieur interprétable (un proxy) capable de retourner les mêmes résultats que le classifieur expliqué dans le voisinage de la série à expliquer. L'explication du résultat du classifieur est ensuite tirée du proxy. À partir de ce processus, nous avons proposé LEFTIST, une méthode pour expliquer le résultat de n'importe quel classifieur de séries temporelles. Nous avons évalué les proxys appris par notre méthode sur deux aspects : la fidélité des proxys avec les classifieurs et l'interprétabilité des explications extraites des proxys. Nous avons montré que notre méthode est fidèle dans le voisinage des séries expliquées pour des classifieurs interprétables et non-interprétables. Enfin, nous avons réalisé une étude utilisateur qui montre la pertinence des explications retournées.

7.1 Perspectives

7.1.1 Seuil de distance guidé par l'utilisateur

Dans le Chapitre 4, nous avons introduit l'utilisation de mesures de distance élastiques dans la recherche d'occurrences de règles temporelles dans des séries temporelles. Notre objectif était de permettre la capture de sous-séries légèrement déformées mais associées au même comportement dans la source des séries. Dans nos données industrielles, les experts d'Energiency estiment que cette notion de similarité est préférable car ils sont conscients que de nombreux facteurs peuvent déformer les séries temporelles.

Dans la pratique, implémenter une définition de la similarité entre deux sous-séries est une tâche difficile. En effet, la notion de similarité est propre à l'utilisateur, à ses besoins, aux caractéristiques des données. Pour certains, la similarité doit être exacte, pour d'autres les sous-séries peuvent être décalées sur un axe ou les deux. Deux éléments sont essentiels pour définir la similarité entre des sous-séries dans un algorithme : une mesure de distance évaluant numériquement la similarité entre des sous-séries, et un seuil de distance statuant si deux sous-séries sont similaires ou pas à partir de la distance calculée entre elles.

Pour résoudre ce second problème, nous avons proposé une méthode de calcul du seuil de similarité qui repose sur la sélection d'un quantile de la distribution des distances entre les sous-séries de la série temporelle et le motif recherché. Cependant, notre méthode nécessite que l'utilisateur sache analyser une distribution de distances entre un motif et des sous-séries, ce qui n'est pas toujours le cas.

Nous pensons qu'il est possible de proposer une méthode de sélection de seuil de similarité qui soit plus intuitive pour l'utilisateur. Dans la pratique, la méthode de recherche d'occurrences capable de satisfaire la notion de similarité de l'utilisateur consiste à lui présenter le motif et chaque sous-série sous forme graphique, et à lui demander si ils sont similaires. Il s'agit là d'une méthode simple mais extrêmement fastidieuse pour l'utilisateur. Notre piste de recherche consiste à intégrer ce concept avec l'apprentissage du seuil de similarité.

Soient un ensemble de sous-séries E, un motif s et une mesure de distance d. L'objectif est d'apprendre le seuil de similarité th à partir de l'ensemble des distances calculées par d entre s et un sous-ensemble de E. Itérativement, nous demandons à l'utilisateur s'il estime que le motif et les sous-séries testées sont similaires à ses

yeux. Le nombre de sous-séries doit être limité pour ne pas rebuter l'utilisateur en lui présentant trop d'exemples. Ce concept peut être utilisé dans plusieurs approches.

Une première approche simple est d'utiliser les réponses de l'utilisateur pour affiner un seuil de similarité sur la distance d.

Une seconde approche est d'avoir plusieurs mesures de distance $\{d_1, \ldots, d_k\}$ de choisir la mesure d_i (et un seuil) correspondant le mieux aux choix de l'utilisateur (voir une combinaison linéaire de ces mesures).

Une dernière approche est d'apprendre directement la mesure de distance d la plus adaptée grâce à des techniques d'apprentissage de métriques. Ces méthodes peuvent toutefois nécessiter plus d'exemples que ce que l'utilisateur est prêt à donner.

7.1.2 Composants interprétables et séries temporelles

Dans le Chapitre 6, nous avons introduit LEFTIST une méthode pour expliquer un résultat de n'importe quel classifieur de séries temporelles. Pour fournir une explication compréhensible à l'utilisateur, notre méthode extrait des composants interprétables à partir de la série dont la classification doit être expliquée. Ces composants sont ensuite utilisés pour construire un espace de représentation de la série expliquée et de son voisinage sur lequel est appris le proxy interprétable du classifieur à expliquer. C'est à partir de ce proxy qu'est extraite l'explication de la classification de la série.

Les composants interprétables que nous avons choisis dans LEFTIST sont des sous-séries de la série à expliquer. Ce choix est motivé par le fait qu'une sous-série peut être associée à un comportement dans la source de la série (ex : une augmentation de la consommation électrique dans la courbe de consommation d'une machine due à sa mise en fonctionnement). Dans la pratique, il est difficile de trouver a priori la ou les sous-séries les plus importantes dans la classification d'une série. Par conséquent, une explication basé sur une seule segmentation n'est pas assurée d'être optimale.

Une piste d'amélioration consiste à ne plus se limiter à l'apprentissage d'une explication mais à itérer sur plusieurs explications jusqu'à trouver la segmentation qui identifie les sous-séries les plus importantes dans la classification. L'algorithme commence par segmenter une première fois la série à expliquer, puis apprend un premier proxy. À partir de ce proxy, nous extrayons la contribution de chaque segment dans la classification, enfin un critère d'arrêt est testé pour voir si l'itération sur les explica-

tions doit continuer. Si c'est négatif, alors une nouvelle segmentation de la série est construite à partir des contributions des précédents segments et un nouveau proxy est appris. Cette opération est réitérée jusqu'à la vérification d'un critère d'arrêt. Le plus grand défi de cette approche consiste à trouver un critère d'arrêt satisfaisant. Différentes propositions sont envisagées : fixer un nombre d'itérations, fixer une limite de temps ou déterminer une mesure d'optimalité d'une explication.

Une autre piste proposée consiste, au lieu d'itérer sur les explications, à fusionner des explications tirées de différentes segmentations en une unique explication. L'idée est d'associer à chaque valeur de la série un score calculé à partir des contributions des différents segments couvrant cette valeur dans chaque explication. Une visualisation présenterait les scores associés aux valeurs de la série et mettrait en évidence les parties de la série où les contributions sont maximales, par exemple une carte de chaleur (*heatmap*).

Dans ces deux approches, différentes segmentations peuvent être testées comme la segmentation aléatoire, TICC [Hal+17], voire proposer à l'utilisateur de définir sa propre segmentation dans le but de tester une de ses hypothèses.

7.1.3 Évaluation de l'interprétabilité des explications

L'évaluation de l'interprétabilité reste une tâche difficile pour le moment. Nous avons analysé les résultats d'une étude utilisateur, que nous avons réalisée pour évaluer la compréhension des explications retournées LEFTIST. Cette étude, nous a conduit à nous poser des questions sur la façon de présenter les résultats, sur combien et quelles explications nous devons présenter. Certaines de ces questions commencent seulement à être explorées. Récemment, Fürnkranz et al. [FK18] ont montré l'impact des biais cognitifs sur la perception de l'explication de résultats. Cette étude a remis en cause, au travers de nombreuses expérimentations avec des utilisateurs, des idées communément admises sur l'interprétabilité de modèles. Par exemple, ils ont montré qu'une explication spécifique pouvait être perçue comme plus générale qu'elle ne l'était. Nous pensons qu'une collaboration avec des experts des sciences cognitives dans nos futures expérimentations permettrait de palier ce problème.

Un autre point à explorer est la nature du public interrogé dans l'étude utilisateur. Dans notre cas, nous avons interrogé des personnes non-expertes des séries temporelles. Cela nous a permis d'évaluer l'impact de l'utilisation d'un tutoriel, ainsi d'in-

terroger une grande base d'utilisateurs. Cependant, l'interrogation d'utilisateurs non-expert limite la portée technique des questions posées. Par conséquent, nous avons comme perspective d'interroger des experts des séries temporelles, par exemple des spécialistes des données travaillant chez Energiency. Cependant, une étude utilisateur auprès d'experts se confronte a des problèmes logistiques et scientifiques. Dans le premier cas, il s'agit de pouvoir réunir assez d'experts des séries temporelles et que ces derniers soient disponibles pour répondre à l'étude. Dans le second, il s'agit de créer un nouveau questionnaire cherchant à évaluer l'interprétabilité des explications attendues par l'utilisateur final.

Bibliographie

[Ach+12] Radhakrishna ACHANTA, Appu SHAJI, Kevin SMITH, Aurélien LUCCHI, Pascal FUA et Sabine SÜSSTRUNK, « SLIC Superpixels Compared to State-of-the-Art Superpixel Methods », in : *IEEE Trans. Pattern Anal. Mach. Intell.* 34.*11* (2012), p. 2274–2282.

[ADT95] Robert ANDREWS, Joachim DIEDERICH et Alan B. TICKLE, « Survey and critique of techniques for extracting rules from trained artificial neural networks », in : *Knowl.-Based Syst.* 8.*6* (1995), p. 373–389.

[AIS93] R. AGRAWAL, T. IMIELIŃSKI et A. SWAMI, « Mining association rules between sets of items in large databases », in : *Acm sigmod record*, t. 22, 2, ACM, 1993, p. 207–216.

[AK] William Vickers ANTHONY BAGNALL Jason Lines et Eamonn KEOGH, *The UEA & UCR Time Series Classification Repository*, `https://timeseriesclassification.com`.

[AK12] M. Gethsiyal AUGASTA et T. KATHIRVALAVAKUMAR, « Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems », in : *Neural Processing Letters* 35.*2* (2012), p. 131–150.

[All83] James F. ALLEN, « Maintaining Knowledge about Temporal Intervals », in : *Commun. ACM* 26.*11* (1983), p. 832–843.

[AS95] Rakesh AGRAWAL et Ramakrishnan SRIKANT, « Mining Sequential Patterns », in : *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, 1995, p. 3–14.

[Åst69] Karl Johan ÅSTRÖM, « On the choice of sampling rates in parametric identification of time series », in : *Inf. Sci.* 1.*3* (1969), p. 273–278.

[Bag+06] Anthony J. BAGNALL, Chotirat (Ann) RATANAMAHATANA, Eamonn J. KEOGH, Stefano LONARDI et Gareth J. JANACEK, « A Bit Level Representation for Time Series Data Mining with Shape Based Similarity », in : *Data Min. Knowl. Discov.* 13.*1* (2006), p. 11–40.

[Bag+15] Anthony J. BAGNALL, Jason LINES, Jon HILLS et Aaron BOSTROM, « Time-Series Classification with COTE : The Collective of Transformation-Based Ensembles », in : *IEEE Trans. Knowl. Data Eng.* 27.*9* (2015), p. 2522–2535.

[Bag+17] Anthony J. BAGNALL, Jason LINES, Aaron BOSTROM, James LARGE et Eamonn J. KEOGH, « The great time series classification bake off : a review and experimental evaluation of recent algorithmic advances », in : *Data Min. Knowl. Discov.* 31.*3* (2017), p. 606–660.

[Bai+15] Adeline BAILLY, Simon MALINOWSKI, Romain TAVENARD, Laetitia CHAPEL et Thomas GUYET, « Dense Bag-of-Temporal-SIFT-Words for Time Series Classification », in : *Advanced Analysis and Learning on Temporal Data - First ECML PKDD Workshop, AALTD 2015, Porto, Portugal, September 11, 2015, Revised Selected Papers*, 2015, p. 17–30.

[BB15] Aaron BOSTROM et Anthony J. BAGNALL, « Binary Shapelet Transform for Multiclass Time Series Classification », in : *Big Data Analytics and Knowledge Discovery - 17th International Conference, DaWaK 2015, Valencia, Spain, September 1-4, 2015, Proceedings*, 2015, p. 257–269.

[BB94] Marko BOHANEC et Ivan BRATKO, « Trading Accuracy for Simplicity in Decision Trees », in : *Machine Learning* 15.*3* (1994), p. 223–250.

[BC94] Donald J. BERNDT et James CLIFFORD, « Using Dynamic Time Warping to Find Patterns in Time Series », in : *Knowledge Discovery in Databases : Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03*, 1994, p. 359–370.

[BGV92] Bernhard E. BOSER, Isabelle GUYON et Vladimir VAPNIK, « A Training Algorithm for Optimal Margin Classifiers », in : *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992.* 1992, p. 144–152.

[BR16] Mustafa Gokce BAYDOGAN et George C. RUNGER, « Time series representation and similarity based on local autopatterns », in : *Data Min. Knowl. Discov.* 30.*2* (2016), p. 476–509.

[Bre+84] Leo BREIMAN, J. H. FRIEDMAN, R. A. OLSHEN et C. J. STONE, *Classification and Regression Trees*, Wadsworth, 1984, ISBN : 0-534-98053-8.

[BRT13] Mustafa Gokce BAYDOGAN, George C. RUNGER et Eugene TUV, « A Bag-of-Features Framework to Classify Time Series », in : *IEEE Trans. Pattern Anal. Mach. Intell.* 35.*11* (2013), p. 2796–2802.

[BRY98] Andrew R. BARRON, Jorma RISSANEN et Bin YU, « The Minimum Description Length Principle in Coding and Modeling », in : *IEEE Trans. Information Theory* 44.*6* (1998), p. 2743–2760.

[CB17] Marco CUTURI et Mathieu BLONDEL, « Soft-DTW : a Differentiable Loss Function for Time-Series », in : *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, p. 894–903.

[CF99] Kin-pong CHAN et Ada Wai-Chee FU, « Efficient Time Series Matching by Wavelets », in : *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia, March 23-26, 1999*, 1999, p. 126–133.

[CFY03] Kin-pong CHAN, Ada Wai-Chee FU et Clement T. YU, « Haar Wavelets for Efficient Similarity Search of Time-Series : With and Without Time Warping », in : *IEEE Trans. Knowl. Data Eng.* 15.*3* (2003), p. 686–705.

[CKL03] Bill Yuan-chi CHIU, Eamonn J. KEOGH et Stefano LONARDI, « Probabilistic discovery of time series motifs », in : *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, 2003, p. 493–498.

[CN04] Lei CHEN et Raymond T. NG, « On The Marriage of Lp-norms and Edit Distance », in : *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004*, 2004, p. 792–803.

[Coh95] William W. COHEN, « Fast Effective Rule Induction », in : *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, 1995, p. 115–123.

[CS02] Paul COTOFREI et Kilian STOFFEL, « Classification Rules + Time = Temporal Rules », in : *Computational Science - ICCS 2002, International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part I*, 2002, p. 572–581.

[Cut11] Marco CUTURI, « Fast Global Alignment Kernels », in : *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, 2011, p. 929–936.

[Das+98] Gautam DAS, King-Ip LIN, Heikki MANNILA, Gopal RENGANATHAN et Padhraic SMYTH, « Rule Discovery from Time Series », in : *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York City, New York, USA, August 27-31, 1998*, 1998, p. 16–22.

[Den+13] Houtao DENG, George C. RUNGER, Eugene TUV et Vladimir MARTYANOV, « A time series forest for classification and feature extraction », in : *Inf. Sci.* 239 (2013), p. 142–153.

[DK17] Finale DOSHI-VELEZ et Been KIM, « Towards a rigorous science of interpretable machine learning », in : *arXiv preprint arXiv :1702.08608* (2017).

[Efr+04] Bradley EFRON, Trevor HASTIE, Iain JOHNSTONE, Robert TIBSHIRANI et al., « Least angle regression », in : *The Annals of statistics* 32.*2* (2004), p. 407–499.

[Faw+19] Hassan Ismail FAWAZ, Germain FORESTIER, Jonathan WEBER, Lhassane IDOUMGHAR et Pierre-Alain MULLER, « Deep learning for time series classification : a review », in : *Data Min. Knowl. Discov.* 33.*4* (2019), p. 917–963.

[FG11] Eugene FINK et Harith Suman GANDHI, « Compression of time series by extracting major extrema », in : *J. Exp. Theor. Artif. Intell.* 23.*2* (2011), p. 255–270.

[FK18] Johannes FÜRNKRANZ et Tomás KLIEGR, « The Need for Interpretability Biases », in : *Advances in Intelligent Data Analysis XVII - 17th International Symposium, IDA, 's-Hertogenbosch, Netherlands, October 24-26, 2018, Proceedings*, 2018, p. 15–27.

[Fre13] Alex Alves FREITAS, « Comprehensible classification models : a position paper », in : *SIGKDD Explorations* 15.*1* (2013), p. 1–10.

[FRM94] Christos FALOUTSOS, M. RANGANATHAN et Yannis MANOLOPOULOS, « Fast Subsequence Matching in Time-Series Databases », in : *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, USA, May 24-27, 1994.* 1994, p. 419–429.

[Fu11] Tak-Chung FU, « A review on time series data mining », in : *Eng. Appl. of AI* 24.*1* (2011), p. 164–181.

[GBB11] Xavier GLOROT, Antoine BORDES et Yoshua BENGIO, « Deep Sparse Rectifier Neural Networks », in : *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, 2011, p. 315–323.

[GH06] Liqiang GENG et Howard J. HAMILTON, « Interestingness measures for data mining : A survey », in : *ACM Comput. Surv.* 38.*3* (2006), p. 9.

[GL13] Tomasz GÓRECKI et Maciej LUCZAK, « Using derivatives in time series classification », in : *Data Min. Knowl. Discov.* 26.*2* (2013), p. 310–331.

[GL14] Tomasz GÓRECKI et Maciej LUCZAK, « Non-isometric transforms in time series classification using DTW », in : *Knowl.-Based Syst.* 61 (2014), p. 98–108.

[Gra+14] Josif GRABOCKA, Nicolas SCHILLING, Martin WISTUBA et Lars SCHMIDT-THIEME, « Learning time-series shapelets », in : *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, 2014, p. 392–401.

[Gui+19a] Riccardo GUIDOTTI, Anna MONREALE, Salvatore RUGGIERI, Franco TURINI, Fosca GIANNOTTI et Dino PEDRESCHI, « A Survey of Methods for Explaining Black Box Models », in : *ACM Comput. Surv.* 51.*5* (2019), 93 :1–93 :42.

[Hal+17] David HALLAC, Sagar VARE, Stephen P. BOYD et Jure LESKOVEC, « Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data », in : *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, 2017, p. 215–223.

[Har89] George W HART, « Residential energy monitoring and computerized surveillance via utility power flows », in : *IEEE Technology and Society Magazine* 8.*2* (1989), p. 12–16.

[Hil+14] Jon HILLS, Jason LINES, Edgaras BARANAUSKAS, James MAPP et Anthony J. BAGNALL, « Classification of time series by shapelet transformation », in : *Data Min. Knowl. Discov.* 28.*4* (2014), p. 851–881.

[Hir77] Daniel S. HIRSCHBERG, « Algorithms for the Longest Common Subsequence Problem », in : *J. ACM* 24.*4* (1977), p. 664–675.

[HL00] David W. HOSMER et Stanley LEMESHOW, *Applied Logistic Regression, Second Edition*, Wiley, 2000, ISBN : 978-0-47135632-5.

[Ho95] Tin Kam HO, « Random decision forests », in : *Third International Conference on Document Analysis and Recognition, ICDAR 1995, August 14 - 15, 1995, Montreal, Canada. Volume I*, 1995, p. 278–282.

[Höp01] Frank HÖPPNER, « Discovery of Temporal Patterns. Learning Rules about the Qualitative Behaviour of Time Series », in : *Principles of Data Mining and Knowledge Discovery, 5th European Conference, PKDD 2001, Freiburg, Germany, September 3-5, 2001, Proceedings*, 2001, p. 192–203.

[HPO16] Mario HERMANN, Tobias PENTEK et Boris OTTO, « Design Principles for Industrie 4.0 Scenarios », in : *49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, HI, USA, January 5-8, 2016*, 2016, p. 3928–3937.

[HPS16] Moritz HARDT, Eric PRICE et Nati SREBRO, « Equality of Opportunity in Supervised Learning », in : *Advances in Neural Information Processing Systems 29 : Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, p. 3315–3323.

[HS05] Magnus Lie HETLAND et Pål SÆTROM, « Evolutionary Rule Mining in Time Series Databases », in : *Machine Learning* 58.*2-3* (2005), p. 107–125.

[IS15] Sergey IOFFE et Christian SZEGEDY, « Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift », in : (2015), p. 448–456.

[Ita75] Fumitada ITAKURA, « Minimum prediction residual principle applied to speech recognition », in : *IEEE Transactions on Acoustics, Speech, and Signal Processing* 23.*1* (1975), p. 67–72.

[JJO11] Youngseon JEONG, Myong Kee JEONG et Olufemi A. OMITAOMU, « Weighted dynamic time warping for time series classification », in : *Pattern Recognition* 44.*9* (2011), p. 2231–2240.

[JKN04] Ulf JOHANSSON, Rikard KÖNIG et Lars NIKLASSON, « The Truth is In There - Rule Extraction from Opaque Models Using Genetic Programming », in : *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, 2004, p. 658–663.

[JLS02] Xiaoming JIN, Yuchang LU et Chunyi SHI, « Distribution Discovery : Local Analysis of Temporal Rules », in : *Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference, PAKDD 2002, Taipei, Taiwan, May 6-8, 2002, Proceedings*, 2002, p. 469–480.

[JNK04] Ulf JOHANSSON, Lars NIKLASSON et Rikard KÖNIG, « Accuracy vs. comprehensibility in data mining models », in : *Proceedings of the seventh international conference on information fusion*, t. 1, 2004, p. 295–300.

[Kat16] Rohit J. KATE, « Using dynamic time warping distances as features for improved time series classification », in : *Data Min. Knowl. Discov.* 30.*2* (2016), p. 283–312.

[Keo+04] Eamonn KEOGH, Selina CHU, David HART et Michael PAZZANI, « Segmenting time series : A survey and novel approach », in : *Data mining in time series databases*, 2004, p. 1–21.

[Keo02] Eamonn J. KEOGH, « Exact Indexing of Dynamic Time Warping », in : *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, 2002, p. 406–417.

[KJF97] Flip KORN, H. V. JAGADISH et Christos FALOUTSOS, « Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences », in : *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA.* 1997, p. 289–300.

[KLC02] Eamonn J. KEOGH, Stefano LONARDI et Bill Yuan-chi CHIU, « Finding surprising patterns in a time series database in linear time and space », in : *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, 2002, p. 550–556.

[KLT03] Eamonn J. KEOGH, Jessica LIN et Wagner TRUPPEL, « Clustering of Time Series Subsequences is Meaningless : Implications for Previous and Future Research », in : *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, 2003, p. 115–122.

[KP00] Eamonn J. KEOGH et Michael J. PAZZANI, « A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases », in : *Knowledge Discovery and Data Mining, Current Issues and New Applications, 4th Pacific-Asia Conference, PAKDD 2000, Kyoto, Japan, April 18-20, 2000, Proceedings*, 2000, p. 122–133.

[KPB16] Isak KARLSSON, Panagiotis PAPAPETROU et Henrik BOSTRÖM, « Generalized random shapelet forests », in : *Data Min. Knowl. Discov.* 30.*5* (2016), p. 1053–1085.

[KR05] Eamonn J. KEOGH et Chotirat (Ann) RATANAMAHATANA, « Exact indexing of dynamic time warping », in : *Knowl. Inf. Syst.* 7.*3* (2005), p. 358–386.

[KR13] Eamonn J. KEOGH et Thanawin RAKTHANMANON, « Fast Shapelets : A Scalable Algorithm for Discovering Time Series Shapelets », in : *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA.* 2013, p. 668–676.

[Kul+15] Todd KULESZA, Margaret M. BURNETT, Weng-Keen WONG et Simone STUMPF, « Principles of Explanatory Debugging to Personalize Interactive Machine Learning », in : *Proceedings of the 20th International Conference on Intelligent User Interfaces, IUI 2015, Atlanta, GA, USA, March 29 - April 01, 2015*, 2015, p. 126–137.

[Lak+17] Himabindu LAKKARAJU, Ece KAMAR, Rich CARUANA et Jure LESKOVEC, « Interpretable & Explorable Approximations of Black Box Models », in : *CoRR* abs/1707.01154 (2017).

[LB15] Jason LINES et Anthony J. BAGNALL, « Time series classification with ensembles of elastic distance measures », in : *Data Min. Knowl. Discov.* 29.*3* (2015), p. 565–592.

[LBL16] Himabindu LAKKARAJU, Stephen H. BACH et Jure LESKOVEC, « Interpretable Decision Sets : A Joint Framework for Description and Prediction », in : *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, p. 1675–1684.

[Lee+18] Ritchie LEE, Mykel J. KOCHENDERFER, Ole J. MENGSHOEL et Joshua SILBERMANN, « Interpretable Categorization of Heterogeneous Time Series Data », in : *Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018, May 3-5, 2018, San Diego Marriott Mission Valley, San Diego, CA, USA.* 2018, p. 216–224.

[Lin+02] J LIN, E KEOGH, P PATEL et S LONARDI, « Finding Motifs in Time Series, In proceedings of the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining », in : *Edmonton, Alberta, Canada* (2002), p. 53–58.

[Lin+07] Jessica LIN, Eamonn J. KEOGH, Li WEI et Stefano LONARDI, « Experiencing SAX : a novel symbolic representation of time series », in : *Data Min. Knowl. Discov.* 15.*2* (2007), p. 107–144.

[Lin+12] Jason LINES, Luke M. DAVIS, Jon HILLS et Anthony J. BAGNALL, « A shapelet transform for time series classification », in : *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, 2012, p. 289–297.

[Lip18] Zachary C. LIPTON, « The mythos of model interpretability », in : *Commun. ACM* 61.*10* (2018), p. 36–43.

[LKK01] Mark LAST, Yaron KLEIN et Abraham KANDEL, « Knowledge discovery in time series databases », in : *IEEE Trans. Systems, Man, and Cybernetics, Part B* 31.*1* (2001), p. 160–169.

[LKL12] Jessica LIN, Rohan KHADE et Yuan LI, « Rotation-invariant similarity in time series using bag-of-patterns representation », in : *J. Intell. Inf. Syst.* 39.*2* (2012), p. 287–315.

[LL17] Scott M. LUNDBERG et Su-In LEE, « A Unified Approach to Interpreting Model Predictions », in : *Advances in Neural Information Processing Systems 30 : Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, p. 4765–4774.

[Low99] David G. LOWE, « Object Recognition from Local Scale-Invariant Features », in : *ICCV*, 1999, p. 1150–1157.

[LTB18] Jason LINES, Sarah TAYLOR et Anthony J. BAGNALL, « Time Series Classification with HIVE-COTE : The Hierarchical Vote Collective of Transformation-Based Ensembles », in : *TKDD* 12.*5* (2018), 52 :1–52 :35.

[Mar07] Pierre-Francois MARTEAU, « Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching », in : *CoRR* abs/cs/0703033 (2007).

[MTV97] Heikki MANNILA, Hannu TOIVONEN et A. Inkeri VERKAMO, « Discovery of Frequent Episodes in Event Sequences », in : *Data Min. Knowl. Discov.* 1.*3* (1997), p. 259–289.

[Mue+09] Abdullah MUEEN, Eamonn J. KEOGH, Qiang ZHU, Sydney CASH et M. Brandon WESTOVER, « Exact Discovery of Time Series Motifs », in : *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, 2009, p. 473–484.

[Mül07] Meinard MÜLLER, *Information retrieval for music and motion*, t. 2, Springer, 2007, "69–84".

[Pap+09] Panagiotis PAPAPETROU, George KOLLIOS, Stan SCLAROFF et Dimitrios GUNOPULOS, « Mining frequent arrangements of temporal intervals », in : *Knowl. Inf. Syst.* 21.*2* (2009), p. 133–171.

[Pat+02] Pranav PATEL, Eamonn J. KEOGH, Jessica LIN et Stefano LONARDI, « Mining Motifs in Massive Time Series Databases », in : *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, 2002, p. 370–377.

[PC01] Sanghyun PARK et Wesley W. CHU, « Discovering and Matching Elastic Rules from Sequence Databases », in : *Fundam. Inform.* 47.*1-2* (2001), p. 75–90.

[Per+00] Chang-Shing PERNG, Haixun WANG, Sylvia R. ZHANG et Douglas Stott Parker JR., « Landmarks : a New Model for Similarity-based Pattern Querying in Time Series Databases », in : *Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 - March 3, 2000*, 2000, p. 33–42.

[Pia91] Gregory PIATETSKY-SHAPIRO, « Discovery, Analysis, and Presentation of Strong Rules », in : *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991, p. 229–248.

[PM02] Ivan POPIVANOV et Renée J. MILLER, « Similarity Search Over Time-Series Data Using Wavelets », in : *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, 2002, p. 212–221.

[Qui93] J. Ross QUINLAN, *C4.5 : Programs for Machine Learning*, Morgan Kaufmann, 1993, ISBN : 1-55860-238-0.

[RAM05] Juan José RODRIGUEZ, Carlos J. ALONSO et José A. MAESTRO, « Support vector machines of interval-based features for time series classification », in : *Knowl.-Based Syst.* 18.*4-5* (2005), p. 171–178.

[Rat+05] Chotirat (Ann) RATANAMAHATANA, Eamonn J. KEOGH, Anthony J. BAGNALL et Stefano LONARDI, « A Novel Bit Level Time Series Representation with Implication of Similarity Search and Clustering », in : *Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005, Proceedings*, 2005, p. 771–777.

[Ren+15] Xavier RENARD, Maria RIFQI, Walid ERRAY et Marcin DETYNIECKI, « Random-shapelet : An algorithm for fast shapelet discovery », in : (2015), p. 1–10.

[RHD17] *Right for the Right Reasons : Training Differentiable Models by Constraining their Explanations*, 2017, p. 2662–2670.

[RSG16] Marco Túlio RIBEIRO, Sameer SINGH et Carlos GUESTRIN, « "Why Should I Trust You ?" : Explaining the Predictions of Any Classifier », in : *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, p. 1135–1144.

[RSG18] Marco Túlio RIBEIRO, Sameer SINGH et Carlos GUESTRIN, « Anchors : High-Precision Model-Agnostic Explanations », in : *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2018, p. 1527–1535.

[RT04] Simona E. ROMBO et Giorgio TERRACINA, « Discovering Representative Models in Large Time Series Databases », in : *Flexible Query Answering Systems, 6th International Conference, (FQAS) 2004, Lyon, France, June 24-26, 2004, Proceedings*, 2004, p. 84–97.

[SAD13] Alexandra STEFAN, Vassilis ATHITSOS et Gautam DAS, « The Move-Split-Merge Metric for Time Series », in : *IEEE Trans. Knowl. Data Eng.* 25.*6* (2013), p. 1425–1438.

[SC78] Hiroaki SAKOE et Seibi CHIBA, « Dynamic programming algorithm optimization for spoken word recognition », in : *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.*1* (1978), p. 43–49.

[Sca+17] Simone SCARDAPANE, Danilo COMMINIELLO, Amir HUSSAIN et Aurelio UNCINI, « Group sparse regularization for deep neural networks », in : *Neurocomputing* 241 (2017), p. 81–89.

[Sch15] Patrick SCHÄFER, « The BOSS is concerned with time series classification in the presence of noise », in : *Data Min. Knowl. Discov.* 29.*6* (2015), p. 1505–1530.

[SG91] Padhraic SMYTH et Rodney M. GOODMAN, « Rule Induction Using Information Theory », in : *Knowledge Discovery in Databases*, 1991, p. 159–176.

[Sha53] Lloyd S SHAPLEY, « A value for n-person games », in : *Contributions to the Theory of Games* 2.*28* (1953), p. 307–317.

[Sho+15] Mohammad SHOKOOHI-YEKTA, Yanping CHEN, Bilson J. L. CAMPANA, Bing HU, Jesin ZAKARIA et Eamonn J. KEOGH, « Discovery of Meaningful Rules in Time Series », in : *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, 2015, p. 1085–1094.

[Sil+18] Diego Furtado SILVA, Rafael GIUSTI, Eamonn J. KEOGH et Gustavo E. A. P. A. BATISTA, « Speeding up similarity search under dynamic time warping by pruning unpromising alignments », in : *Data Min. Knowl. Discov.* 32.*4* (2018), p. 988–1016.

[Sim+13] Noah SIMON, Jerome FRIEDMAN, Trevor HASTIE et Robert TIBSHIRANI, « A sparse-group lasso », in : *Journal of Computational and Graphical Statistics* 22.*2* (2013), p. 231–245.

[SK14] Erik STRUMBELJ et Igor KONONENKO, « Explaining prediction models and individual predictions with feature contributions », in : *Knowl. Inf. Syst.* 41.*3* (2014), p. 647–665.

[SM13] Pavel SENIN et Sergey MALINCHIK, « SAX-VSM : Interpretable Time Series Classification Using SAX and Vector Space Model », in : *2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013*, 2013, p. 1175–1180.

[SPK18] Joan SERRÀ, Santiago PASCUAL et Alexandros KARATZOGLOU, « Towards a Universal Neural Network Encoder for Time Series », in : *Artificial Intelligence Research and Development - Current Challenges, New Trends and Applications, CCIA 2018, 21st International Conference of the Catalan Association for Artificial Intelligence, Alt Empordà, Catalonia, Spain, 8-10th October 2018*, 2018, p. 120–129.

[Tav+17] Romain TAVENARD, Simon MALINOWSKI, Laetitia CHAPEL, Adeline BAILLY, Heider SANCHEZ et Benjamin BUSTOS, « Efficient Temporal Kernels Between Feature Sets for Time Series Classification », in : *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part II*, 2017, p. 528–543.

[TF18] Tomás TEIJEIRO et Paulo FÉLIX, « On the adoption of abductive reasoning for time series interpretation », in : *Artif. Intell.* 262 (2018), p. 163–188.

[TFV17] Romain TAVENARD, Johann FAOUZI et Gilles VANDEWIELE, *tslearn : A machine learning toolkit dedicated to time-series data*, `https://github.com/rtavenar/tslearn`, 2017.

[TH12] Tijmen TIELEMAN et Geoffrey HINTON, « Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude », in : *COURSERA : Neural networks for machine learning* 4.*2* (2012), p. 26–31.

[Tib96] Robert TIBSHIRANI, « Regression shrinkage and selection via the lasso », in : *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* (1996), p. 267–288.

[TIU05] Yoshiki TANAKA, Kazuhisa IWAMOTO et Kuniaki UEHARA, « Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle », in : *Machine Learning* 58.*2-3* (2005), p. 269–300.

[Tou+10] Vincent TOUBIANA, Arvind NARAYANAN, Dan BONEH, Helen NISSENBAUM et Solon BAROCAS, « Adnostic : Privacy Preserving Targeted Advertising », in : *Proceedings of the Network and Distributed System Security Symposium, NDSS 2010, San Diego, California, USA, 28th February - 3rd March 2010*, 2010.

[Wan+13] Xiaoyue WANG, Abdullah MUEEN, Hui DING, Goce TRAJCEVSKI, Peter SCHEUERMANN et Eamonn J. KEOGH, « Experimental comparison of representation methods and distance measures for time series data », in : *Data Min. Knowl. Discov.* 26.*2* (2013), p. 275–309.

[WSZ04] Huanmei WU, Betty SALZBERG et Donghui ZHANG, « Online Event-driven Subsequence Matching over Financial Data Streams », in : *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, 2004, p. 23–34.

[WYO17] Zhiguang WANG, Weizhong YAN et Tim OATES, « Time series classification from scratch with deep neural networks : A strong baseline », in : (2017), p. 1578–1585.

[Xu+15] Kelvin XU, Jimmy BA, Ryan KIROS, Kyunghyun CHO, Aaron C. COURVILLE, Ruslan SALAKHUTDINOV, Richard S. ZEMEL et Yoshua BENGIO, « Show, Attend and Tell : Neural Image Caption Generation with Visual Attention », in : *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, p. 2048–2057.

[YF00] Byoung-Kee YI et Christos FALOUTSOS, « Fast Time Sequence Indexing for Arbitrary Lp Norms », in : *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, 2000, p. 385–394.

[YK09] Lexiang YE et Eamonn J. KEOGH, « Time series shapelets : a new primitive for data mining », in : *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, 2009, p. 947–956.

[YL06] Ming YUAN et Yi LIN, « Model selection and estimation in regression with grouped variables », in : *Journal of the Royal Statistical Society : Series B (Statistical Methodology)* 68.*1* (2006), p. 49–67.

[You85] H Peyton YOUNG, « Monotonic solutions of cooperative games », in : *International Journal of Game Theory* 14.*2* (1985), p. 65–72.

[Zho+16] Bolei ZHOU, Aditya KHOSLA, Àgata LAPEDRIZA, Aude OLIVA et Antonio TORRALBA, « Learning Deep Features for Discriminative Localization », in : *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, p. 2921–2929.

[Zhu+17] Yan ZHU, Makoto IMAMURA, Daniel NIKOVSKI et Eamonn J. KEOGH, « Matrix Profile VII : Time Series Chains : A New Primitive for Time Series Data Mining (Best Student Paper Award) », in : *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, 2017, p. 695–704.

Mes publications

[Gui+17] Maël GUILLEMÉ, Laurence ROZÉ, Véronique MASSON, Cérès CARTON, René QUINIOU et Alexandre TERMIER, « Improving Time-Series Rule Matching Performance for Detecting Energy Consumption Patterns », in : *Data Analytics for Renewable Energy Integration : Informing the Generation and Distribution of Renewable Energy - 5th ECML PKDD Workshop, DARE 2017, Skopje, Macedonia, September 22, 2017, Revised Selected Papers*, 2017, p. 59–71.

[Gui+19b] Maël GUILLEMÉ, Simon MALINOSKY, Romain TAVENARD et Xavier RENARD, « Localized Random Shapelet », in : *AALTD@ ECML/PKDD*, 2019.

[Gui+19c] Maël GUILLEMÉ, Véronique MASSON, Laurence ROZÉ et Alexandre TERMIER, « Agnostic Local Explanation For Time Series Classification », in : *ICTAI*, 2019.

UNIVERSITE BRETAGNE LOIRE MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Extraction de connaissances interprétables dans des séries temporelles

Mot clés : Apprentissage automatique, Séries temporelles, Interprétabilité

Résumé : Energiency est une entreprise qui vend à des industriels une plate-forme pour leur permettre d'analyser leurs données de consommation d'énergie et de production, représentées sous la forme de séries temporelles. Cette plate-forme intègre des modèles d'apprentissage automatique pour répondre aux besoins des clients. L'application de tels modèles sur des séries temporelles rencontre deux problèmes : d'une part certaines approches classiques d'apprentissage automatique ont été conçues pour des données tabulaires et doivent être adaptées aux séries temporelles, d'autre part les résultats de certaines approches sont difficilement compréhensibles par les utilisateurs finaux.

Dans la première partie, nous adaptons une méthode de recherche d'occurrences de règles temporelles sur des séries temporelles issues de machines et d'infrastructures industrielles. Une règle temporelle capture des relations de succession entre des comportements dans les séries temporelles (ex : un pic de valeur suivi d'un creux). Dans des séries industrielles, à cause de la présence de nombreux facteurs extérieurs, ces comportements réguliers peuvent présenter des perturbations. Par conséquent, deux occurrences du même comportement produisent deux suites de valeurs légèrement différentes. Les méthodes de recherche d'occurrences de règles temporelles actuelles utilisent une mesure de distance pour évaluer la similarité entre des sous-séries. Cependant, ces mesures ne sont pas adaptées pour évaluer la similarité de séries déformées tel que dans les séries temporelles industrielles. La première contribution de cette thèse est la proposition d'une méthode de recherche d'occurrences de règles temporelles capable de capturer cette variabilité dans des séries temporelles industrielles. Pour cela la méthode intègre l'utilisation de mesures de distance élastiques capables d'évaluer la similarité entre des séries temporelles légèrement déformées. Une analyse qualitative montre que notre méthode améliore la capture d'occurrences de règles temporelles dans des séries temporelles industrielles.

La seconde partie de la thèse est consacrée à l'interprétabilité de méthodes de classification de séries temporelles, c'est-à-dire la capacité d'un classifieur à retourner des explications à ses résultats. Ces explications devant être compréhensibles par un humain. La classification est la tâche d'associer une série temporelle à une catégorie (ex : une série de consommation électrique associée à l'état de la machine). Pour qu'un utilisateur final soit enclin à prendre des décisions basées sur les résultats d'un classifieur, il aimerait avoir des assurances sur comment ces résultats sont produits. Dans le cas contraire, cela revient à avoir une confiance aveugle dans le classifieur. La seconde contribution de cette thèse est un classifieur, dit interprétable, car nous pouvons en extraire directement l'explication des résultats. Ce classifieur utilise des informations locales sur les séries temporelles pour les discriminer. Nous présentons comment extraire une explication d'un résultat. Une analyse qualitative montre que notre méthode a une meilleure précision que des classifieurs interprétables existants mais en deçà de classifieurs non-interprétables. À partir de ce constat, nous avons donc proposé dans la troisième et dernière contribution de cette thèse, une méthode pour expliquer a posteriori un résultat de n'importe quel classifieur. Cette méthode peut être utilisée pour expliquer les résultats de classi-

fieurs non-interprétables. Cette méthode apprend un classifieur interprétable, dit proxy, sur le voisinage de la séries temporelle dont nous souhaitons expliquer la classification. Ce proxy doit imiter les résultats du classifieur à expliquer dans ce voisinage. Nos expérimentations montrent que notre méthode retourne des proxy fidèles aux classifieurs expliqués. Enfin, nous avons réalisé une étude utilisateur pour évaluer l'interprétabilité de notre méthode. Ces résultats montrent que notre méthode fournit des explications compréhensibles pour un utilisateur.

Title: Extraction of interpretable knowledge in time series

Keywords: Machine learning, Time series, Interpretability

Abstract: Energiency is a company that sells a platform toallow manufacturers to analyze their energy consumption and production data, represented in the form of time series. This platform integrates machine learning models to meet customer needs. The application of such models to time series encounters two problems: on the one hand, some classical machine learning approaches have been designed for tabular data and must be adapted to time series, on the other hand, the results of some approaches are difficult for end users to understand.

In the first part, we adapt a method to search for occurrences of temporal rules on time series from machines and industrial infrastructures. A temporal rule captures successional relationships between behaviors in time series (e.g., a value peak followed by a trough). In industrial series, due to the presence of many external factors, these regular behaviours can be disruptive. Therefore, two occurrences of the same behaviour produce two sequences of slightly different values. Current methods for searching the occurrences of a rule use a distance measure to assess the similarity between sub-series. However, these measurements are not suitable for assessing the similarity of distorted series such as those in industrial settings. The first contribution of this thesis is the proposal of a method for searching for occurrences of temporal rules capable of capturing this variability in industrial time series. For this purpose, the method integrates the use of elastic distance measurements capable of assessing the similarity between slightly deformed time series. Qualitative analysis shows that our method improves the capture of temporal rule occurrences in industrial time series.

The second part of the thesis is devoted to the interpretability of time series classification methods, i.e. the ability of a classifier to return explanations for its results. These explanations must be understandable by a human. Classification is the task of associating a time series with a category (e.g., a series of power consumption associated with the condition of the machine). For an end user inclined to make decisions based on a classifier's results, understanding the rationale behind those results is of great importance. Otherwise, it is like having blind confidence in the classifier. The second contribution of this thesis is an interpretable time series classifier that can directly provide explanations for its results. This classifier uses local information on time series to discriminate against them. We present how to extract an explanation for a result. A qualitative analysis shows that our method has a better accuracy than existing interpretable classifiers but worse than non-interpretable classifiers. Based on this observation, we have therefore proposed in the third and last contribution of this thesis, a method to explain a posteriori any result of any classifier. This method can be used to explain the results of non-interpretable classifiers. This method learns

an interpretable classifier, called a proxy, on the neighbourhood of the time series whose classification we want to explain. This proxy must mimic the behaviour of the classifier to be explained in this neighborhood. Our experiments show that our method returns proxies that are faithful to the explained classifiers. Finally, we carried out a user study to evaluate the interpretability of our method. These results show that our method provides explanations that are understandable to users.