Détection d'anomalies dans les flux de données par structure d'indexation et approximation : Application à l'analyse en continu des flux de messages du système d'information de la SNCF

Lucas Foulon

N° d'ordre NNT : 2020LYSEI082

Thèse de doctorat de l'Université de Lyon
opérée au sein de
L'INSA de Lyon

Ecole Doctorale N° 512
Mathématiques et Informatique (InfoMaths)

Spécialité / discipline de doctorat : Informatique

Soutenue publiquement le 16/10/2020 par
Lucas Foulon

Détection d'anomalies dans les flux de données par structure d'indexation et approximation. Application à l'analyse en continu des flux de messages du système d'information de la SNCF.

Devant le jury composé de:

Pr. Thierry Charnois	Université Paris 13	Rapporteur
Pr. Alain Cournier	Université de Picardie Jules Verne	Rapporteur
Pr. Sylvie Calabretto	INSA de Lyon	Examinatrice
Pr. Élisa Fromont	Université de Rennes 1	Examinatrice
Dr. Peter Sturm	INRIA Grenoble Rhône-Alpes	Examinateur
Dr. Christophe Rigotti	INSA de Lyon	Directeur de thèse
Dr. Serge Fenet	Université Claude Bernard Lyon 1	Encadrant de thèse
Dr. Denis Jouvin	SNCF Voyageurs Direction Industrielle	Invité

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage **secretariat@edchimie-lyon.fr** INSA : R. GOURDON	M. Stéphane DANIELE Institut de recherches sur la catalyse et l'environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert EINSTEIN 69 626 Villeurbanne CEDEX **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN **ecole-doctorale.eea@ec-lyon.fr**	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy DE COLLONGUE 69 134 Écully Tél : 04.72.18.60.97 Fax 04.78.43.37.17 **gerard.scorletti@ec-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : H. CHARLES **secretariat.e2m2@univ-lyon1.fr**	M. Philippe NORMAND UMR 5557 Lab. d'Ecologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://www.ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 INSA : M. LAGARDE **secretariat.ediss@univ-lyon1.fr**	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 Avenue Jean CAPELLE INSA de Lyon 69 621 Villeurbanne Tél : 04.72.68.49.09 Fax : 04.72.68.49.16 **emmanuelle.canet@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 **infomaths@univ-lyon1.fr**	M. Luca ZAMBONI Bât. Braconnier 43 Boulevard du 11 novembre 1918 69 622 Villeurbanne CEDEX Tél : 04.26.23.45.52 **zamboni@maths.univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **ed.materiaux@insa-lyon.fr**	M. Jean-Yves BUFFIÈRE INSA de Lyon MATEIS - Bât. Saint-Exupéry 7 Avenue Jean CAPELLE 69 621 Villeurbanne CEDEX Tél : 04.72.43.71.70 Fax : 04.72.43.85.28 **jean-yves.buffiere@insa-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bât. Direction **mega@insa-lyon.fr**	M. Jocelyn BONJOUR INSA de Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69 621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Véronique GUICHARD INSA : J.Y. TOUSSAINT Tél : 04.78.69.72.76 **veronique.cervantes@univ-lyon2.fr**	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69 365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Remerciements

Tout d'abord je tiens à remercier chaleureusement Thierry Charnois, Professeur à l'Université Paris 13, et Alain Cournier, Professeur à l'Université de Picardie Jules Verne, d'avoir accepté la relecture de ce travail de thèse, pour leurs judicieuses analyses et leurs pertinents retours.

Je suis également très reconnaissant auprès de Sylvie Calabretto, Élisa Fromont et Peter Sturm, respectivement Professeur à l'INSA de Lyon, Professeur à l'Université de Rennes 1 et Directeur de Recherche à l'INRIA Grenoble, d'avoir accepté de faire partie de mon jury de thèse et d'avoir apporté de nouvelles pistes de réflexion et de bienveillantes recommandations.

Je tiens à remercier la société SNCF Voyageurs, et plus particulièrement la Direction Industrielle, qui ont corroboré cette thèse et qui ont tout fait pour qu'elle se déroule dans de bonnes conditions de travail. Je remercie l'équipe Train Communicant et Geoservice pour leur accueil, leur professionnalisme et leur soutien durant ces trois années, et l'équipe du réseau Synapse pour l'organisation des évènements en interne. Et je tiens à remercier plus particulièrement Baptiste, Florian, lord Gaétan, Gilles, Denis, Virginie, Grant, Philippe (et ses tirs croisés) et Carlos, pour leur aide et leur amitié. Un bien plus grand merci à Denis Jouvin, architecte SI et mon encadrant à la SNCF, qui a élaboré et donné vie à ce sujet de thèse, m'a intégré professionnellement, et qui a toujours continué à défendre et façonner toutes les idées liées à ce travail.

Je voudrais aussi remercier le laboratoire LIRIS et tout d'abord les membres de l'équipe SMA qui m'ont accompagné durant la fin de mon Master. Mais aussi les membres de l'équipe DM2L pour leur soutien. Je remercie aussi certains enseignants de l'Université Lyon 1 pour avoir su transmettre leur passion. Un très grand merci à Serge Fenet et Christophe Rigotti, respectivement encadrant et directeur de thèse, pour avoir accepté et m'avoir accompagné tout au long de cette aventure. Votre professionnalisme et vos conseils ont été pour moi indispensables et m'ont permis de me forger une véritable méthodologie de travail de recherche. Merci pour le temps que vous y avez consacré, et pour ces agréables temps de réflexion partagés ensemble. De plus, merci pour vos qualités personnelles : sympathie, écoute et compréhension (et je ne cite que quelques uns des éléments de l'union de vos qualités respectives).

Un super merci à mes super camarades de bureau au laboratoire : Julien pour avoir accepté ses défaites à Micro Machines et ses méditations scien-

tifiques sur les domaines qui nous rapprochaient, Romain, Yann pour ses belles planètes, Omar. Également mes camarades de l'étage, pour n'en citer que quelques uns : Tarek pour ses débats de qualité, Adnane simplement pour son style (parce qu'il a trop la classe), Anes pour ses inimitables imitations, Aimene, n'arrête surtout jamais d'arrêter, Marie pour toutes les pauses thé. Et mes camarades d'avant thèse : Aram pour sa théorie des jeux, Jeremy pour ses beaux dessins, Binôme et ses 15 minutes d'avance, Carole pour nous avoir montré l'exemple, Antoine pour ses répliques cultes, Amine, JC, Anna, Dadoo, Vamara, Babs, Sami, et bien d'autres. Et un merci spécial à Seb et Julie pour leur soutien et les maintes relectures.

Bien évidemment, un grand merci à ma famille qui sont mes modèles d'apprentissage les plus performants : mes cousins cousines, mes grands-parents d'ici et d'ailleurs, mon frère et sa "petite" famille. Et mes parents qui m'ont donné beaucoup plus que simplement leurs ADN, mais aussi leur esprit critique et mon premier ordi. Et pour celle qui était là avant, pendant et après, qui a toujours cru que c'était possible, un merci d'amour à Roxane, c'est toi la plus belle œuvre.

"*Je ne suis qu'une goutte de plus noyée dans l'océan qui sait*
Qu'à nous tous, on pourrait faire plein de vagues et tout éclabousser"
Keny Arkana

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Avec le développement d'Internet et des technologies d'échange et de stockage des données, le volume des informations échangées ne cesse de croître de façon exponentielle, tandis que la vitesse de ces échanges s'accélère. Dans les années 90, l'arrivée du Web a permis que cette croissance devienne mondiale et de plus en plus connectée, notamment grâce à la participation du grand public. Cependant, les internautes ne sont pas les seuls acteurs de cette expansion puisque la majorité des secteurs d'activités, y compris privés (par exemple Wallmart pour la grande distribution ou Amazon pour le commerce en ligne), produisent et consomment également de plus en plus de données.

En parallèle de l'accroissement des données récoltées, la puissance de calcul a également augmenté au point de se rapprocher des suppositions faites par les lois de Moore dans les années 60 et 70, telle que celle faisant l'hypothèse que le nombre de transistors dans les microprocesseurs, indicateur de leur puissance de calcul, doublerait tous les 24 mois (loi de Moore de 1975).

Ces augmentations conjointes ont été un tremplin pour le développement de l'apprentissage automatique et de la fouille de données. Ces dernières décennies, ces deux domaines ont bénéficié de la réduction des temps de traitement, tout en diminuant le coût et la taille du matériel informatique nécessaire à leur application. Les micro-ordinateurs grand public et les smartphones sont aujourd'hui assez puissants pour être des vecteurs de consommation massive de données, mais également de production. Les champs d'applications se sont alors démocratisés, pouvant aller de la reconnaissance des plantes à l'aide de photographies de feuillages, à la création d'œuvres musicales et graphiques, en passant par la recommandation d'achats. Dans les deux dernières décennies, l'utilisation des méthodes d'apprentissage automatique et de fouille de données est l'une des principales motivations des différents acteurs du secteur privé et public pour l'utilisation des outils informatiques et numériques. Fournissant parfois directement un service au client,

comme les assistants vocaux, ces méthodes permettent aussi de prendre des décisions internes tel que le renouvellement de stocks des produits.

Bien qu'ils puissent être perçus comme des assistants, ces services numériques et technologiques ont principalement un attrait commercial pour les entreprises. Il est important pour elles de pouvoir les intégrer, les faire évoluer, mais aussi de pouvoir les maintenir fonctionnels de façon à ne pas perdre de plus-value. Un des grands sous-domaines de l'apprentissage automatique et de la fouille de données est celui de la détection d'anomalies, qui possède de nombreux cas d'usages, tels que par exemple : la détection d'éléments anormaux pour l'aide aux diagnostics médicaux, la détection de transactions financières frauduleuses, la détection d'intrusions informatiques ou physiques (par l'intermédiaire de caméras de surveillance), etc.

La Société Nationale des Chemins de fer Français (SNCF), entreprise ferroviaire publique française, est impliquée dans cette dynamique à plusieurs niveaux. D'une part, elle offre de nouveaux services aux passagers, comme la réservation, la consultation des horaires, l'assistance, ou encore le suivi des trains directement sur support numérique. D'autre part, son matériel change en permanence afin de fiabiliser les services de transport proposés.

Avec l'évolution des technologies embarquées et mobiles, le parc ferroviaire de la SNCF collecte en interne de plus en plus de données et le nombre de communications est en augmentation constante. Les différents corps de métiers ont accompagné ces modifications, au point d'être parfois complètement dépendants des outils mis en place. En effet, le système d'information interne actuel permet de collecter, traiter, stocker et distribuer les informations entre les différents protagonistes comme les trains, les passagers, les employés... Il maintient à jour et transmet en temps-réel les informations concernant le système ferroviaire, et assure, grâce à la transmission des connaissances métier récoltées, un trafic fluide. Ce système d'information s'est imposé comme l'une des pierres angulaires de l'activité de la SNCF, et il est indispensable qu'il reste opérationnel en permanence, sans ralentissement ni interruption. C'est pourquoi la SNCF surveille son fonctionnement en temps-réel. Toutefois, son infrastructure est très complexe : différentes générations d'engins communiquent différents types de messages entre différents interlocuteurs sous différents protocoles. L'une des principales difficultés de l'analyse de l'activité de ce système d'information est évidemment la forte hétérogénéité, qui engendre une grande diversité dans les comportements des messages qui circulent.

L'objectif de notre travail est de proposer une approche concrète, capable d'analyser en temps-réel le système d'information, de façon à alerter au plus tôt en cas d'apparition d'un comportement anormal, et d'ainsi éviter une

éventuelle indisponibilité d'un ou plusieurs services dépendant de ce système.

Les flux étudiés sont créés à partir de traces indiquant le passage des messages sur certains points de collecte du système d'information. Ces flux sont considérés comme les descripteurs les plus pertinents par les superviseurs [1] SNCF, mais ils sont nombreux et tous très différents. Notre outil d'aide à la supervision automatique doit alerter rapidement de l'éventualité d'un dysfonctionnement, afin de non seulement permettre aux superviseurs de détecter très tôt les anomalies, mais aussi pour agir à la fois plus rapidement et sur un périmètre d'intervention plus fin.

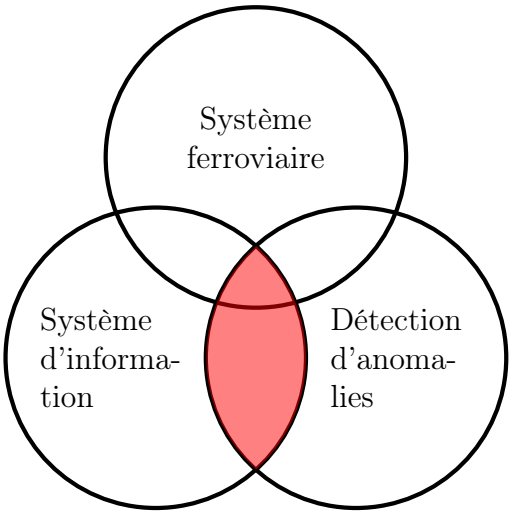


FIGURE 1.1 – Intersections entre les trois domaines de cette thèse : système ferroviaire, système d'information et détection d'anomalies. Le travail de cette thèse s'étend sur la zone colorée en rouge.

Afin de répondre au mieux à cette problématique, nous présentons dans le chapitre 2 une définition du terme "anomalie" et des principales caractéristiques des phénomènes qu'il englobe. Nous décrivons ensuite les différents domaines d'application concernés par la détection d'anomalies, avant de développer les méthodes que nous avons rencontrées dans la littérature.

Cet état de l'art nous a permis d'identifier un score d'anomalie particulièrement adapté aux objets issus de flux et codés sous forme de séquences. Ce score, nommé CFOF (pour "*Concentration Free Outlier Factor*"), a été proposé récemment par ANGIULLI [Angiulli, 2017, Angiulli, 2020], qui a également montré sa résistance à l'augmentation de la dimensionnalité des don-

1. Experts chargés de la surveillance du système d'information.

nées. Toutefois, il n'existe pas de méthode performante pour le calcul de ce score sur des données arrivant en flux, et c'est sur cet aspect que nous proposons deux contributions. La première, présentée au chapitre 3, est une méthode d'approximation du score CFOF couplée à une technique d'élagage mise en place lors des parcours du voisinage des objets. La seconde contribution est basée sur une projection des données sur plusieurs sous-espaces, permettant de calculer en parallèle des scores partiels sur ces sous-espaces. Ces scores sont ensuite agrégés pour obtenir une approximation du score global d'un objet. Cette méthode est détaillée au chapitre 4. Ces deux contributions se complètent et sont évaluées sur des jeux de données synthétiques, ainsi que sur des jeux de données réels provenant du benchmark NAB [2] proposé dans [Lavin et Ahmad, 2015]. Ces expérimentations montrent que les méthodes proposées conduisent à des réductions importantes des temps de calcul sur les flux, tout en fournissant des approximations de bonne qualité permettant de préserver la qualité de la détection.

Le chapitre 5 se concentre sur une application de notre travail au système d'information de la SNCF et illustre des résultats de détections effectuées sur des données réelles. Celle-ci ont été validées par les acteurs de la supervision à la SNCF. Enfin, au chapitre 6, nous concluons et abordons les différentes perspectives de ce travail.

2. Numenta Anomaly Benchmark

Chapitre 2

État de l'art

Sommaire

2.1 Introduction : la détection d'anomalies

La détection d'anomalies est un vaste domaine traité depuis de nombreuses années et connexe à différents domaines de recherche eux-mêmes

en plein essor, comme l'apprentissage automatique. Dans un premier temps nous aborderons la définition du terme "anomalie" et les problématiques qui en découlent. Dans un second temps nous listerons ses principaux domaines d'application, avant de détailler les différents types de données qu'il est possible de traiter.

2.1.1 Qu'est-ce qu'une anomalie

Une anomalie peut être définie comme une observation (portant sur un objet, une donnée, ...) inattendue au regard d'un ensemble d'autres observations préétablies considérées comme normales. Plus formellement, dans un ensemble $\mathcal{D}$ contenant n observations notées x_i, alors $x_p \in A$ sera considérée comme anormale si elle diffère, par ses caractéristiques, des autres observations, c'est-à-dire de celles contenues dans l'ensemble $A \backslash \{x_p\}$.

La définition du terme *anomalie* est spécifique au cas d'usage. La plus courante dans le domaine de la détection est une observation qui se distingue des autres par sa singularité : elle pourrait résulter d'un ensemble de règles différentes par rapport aux autres observations [Hawkins, 1980]. Une anomalie est donc inattendue ; il s'agit d'une irrégularité, une incohérence, une valeur aberrante ou très rare, si l'on tient compte du système qui l'a générée.

Dans le cadre de notre problématique soulevée au sein de la SNCF, la supervision des flux d'informations en temps réel, il est possible de considérer au sens large une anomalie comme une perturbation sur l'infrastructure de communication. Plus précisément, la perturbation sera révélée par un ou plusieurs indicateurs, qui seront eux-mêmes mis en lumière par une ou plusieurs données anormales par rapport à notre historique de données. Autrement dit, une situation globalement anormale sur l'infrastructure est souvent induite par une anomalie d'un ou plusieurs composants locaux. L'analyse se fait sur deux niveaux de granularités différentes.

À présent, nous allons nous intéresser aux limites rencontrées dans le domaine de la détection.

2.1.2 Difficultés

La frontière entre ce qui est normal et ce qui ne l'est pas est souvent floue. Cela peut être dû à une dépendance directe au contexte défini localement et non pris en compte lors de l'étude de l'observation ; ce qui peut rendre le travail de détection d'autant plus difficile. Si l'anomalie est parfois clairement identifiable, sa détection peut être plus délicate dans d'autres situations.

2.1.2.1 Exemple : une rencontre fortuite

Commençons avec une illustration simple permettant de mettre en évidence quelques difficultés couramment rencontrées, même dans nos vies quotidiennes.

Prenons un instant pour imaginer que nous sommes en pleine promenade dans le centre de la ville de Lyon. À cet instant, nous croisons des passants, différents moyens de transport (voitures, bus, ...), et de nombreux commerces rattachés à de grands immeubles accompagnés parfois d'îlots de verdure. Pour une ville ayant connu une forte urbanisation, la situation semble normale. Mais voilà que tout au bout d'une rue, nous apercevons une vache — animal appartenant à la famille des bovidés. Ce ruminant de carrure plutôt imposante trouve généralement sa place dans des milieux plus ruraux, sauf peut-être en Inde où elle est considérée comme sacrée.

Cette observation quelque peu inattendue soulève trois questions. Premièrement, sommes-nous certains de ce que nous venons de voir ? Peut-être que cela n'est que le fruit de notre imagination, et qu'il s'agit d'une mauvaise interprétation de ce que nous venons d'observer. Dans ce cas, ce ne serait qu'une mauvaise "mesure" de notre esprit. Deuxièmement, il se peut qu'un évènement particulier ait lieu non loin de nous, comme par exemple une foire agricole. Alors la situation serait appropriée et l'animal est sûrement bien accompagné. Cette observation ne serait finalement pas si anormale si nous tenions compte de cet évènement. Troisièmement, si nous sommes certains de ce que nous venons de voir, et qu'aucun évènement particulier n'a lieu, quelle est la raison, ou plutôt la cause, anormale de sa visite en ville ? Trouver une explication devient presque plus difficile que de constater l'impromptu.

Ces trois points sont une simplification des questionnements que peut avoir tout individu lorsqu'il fait face à une observation inattendue. Et nous traitons ces types d'interrogations tous les jours lorsque nous faisons face à un évènement inopiné ou indésirable. Il n'est pas toujours évident d'observer une singularité, sans douter dans un premier temps, de notre objectivité, et dans un second temps de l'expliquer.

Nous venons de voir une schématisation des problématiques que nous pouvons rencontrer avec un exemple simplifié, mais dans un contexte plus technique, des caractéristiques intrinsèques aux données considérées vont rendre la détection plus complexe. Nous allons à présent analyser ces problématiques dans un cadre plus technique et scientifique.

2.1.2.2 Données bruitées

L'un des critères qui conditionne notre analyse est le niveau de bruit contenu dans la mesure des données. C'est une des grandes difficultés dans le traitement des données. En effet, en fonction des moyens de mesure disponibles pour la récolte des données à analyser, le signal obtenu peut être plus ou moins bruité. La présence du bruit va impacter notre capacité à différencier les données normales et anormales. Comme illustré sur la figure 2.1 proposée par [Aggarwal, 2017], la frontière entre les deux types de données sera masquée par le bruit. Dans certaines situations, ce sera le bruit lui-même qui sera détecté comme anormal durant l'analyse.

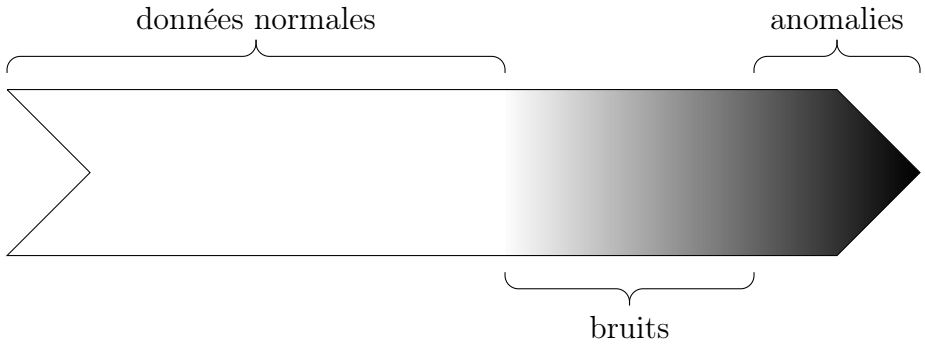


FIGURE 2.1 – Le spectre des données normales aux données anormales, séparées par les données bruitées [Aggarwal, 2017].

La suppression ou l'altération du bruit peut parfois nécessiter de faire appel à des méthodes particulières, utilisées en pré-traitement avant toute détection. C'est le cas par exemple lorsque nous voulons étudier le chant d'une espèce maritime spécifique : il est nécessaire de supprimer tous les bruits environnants pour mieux analyser le signal sonore visé.

2.1.2.3 Dynamique des données

En fonction du processus qui les génère, les données observées peuvent exhiber une forme non-stationnaire. Cela va impacter leur dynamique et générer une variabilité qui peut facilement être interprétée comme une anomalie. Nous pouvons définir trois types de variations : les cycles (saisonnalité), les tendances, et les changements de régime.

Des variations périodiques (saisonnalité) demandent de définir une normalité dépendante de ces cycles (anomalie contextuelle). Par exemple, l'analyse de la consommation électrique à grande échelle est soumise à des cycles.

Il se peut aussi que l'objet en cours de supervision évolue de façon progressive. Sur l'étude d'une ligne de transport en commun, il est possible d'observer une hausse (tendance à la hausse) ou une baisse de la moyenne de la fréquence quotidienne de la ligne si par exemple le réseau de transport ou une zone desservie par cette ligne ont été réaménagés.

Les changements de régime, des variations survenant de manière définitive, obligent une adaptation de l'analyse. Ces variations peuvent être dues à une modification matérielle ou logicielle. La notion de normalité observée à un instant donné se retrouve sans intérêt quelques observations plus tard.

2.1.2.4 Hétérogénéité

La grande hétérogénéité des données observées et des connaissances portant sur les processus les ayant générés, nécessaire à une analyse pertinente, est une difficulté incontournable. En effet, les différents types de données (et les différents domaines d'applications) peuvent rendre futiles certaines techniques de détection d'anomalies d'une problématique à l'autre.

Pour un même type de données, les définitions de ce qui est normal et de ce qui ne l'est pas doivent être revues d'une application à l'autre. Ainsi, une des principales complexités à rendre la tâche de détection automatisée cohérente vient de la nécessité de prendre en compte toutes ces différences. C'est pourquoi, pour certaines applications, l'expertise métier sera très précieuse et fournira un certain nombre d'informations de contexte basées sur des observations historiques. La supervision des flux d'informations en temps réel dans un système d'information est effectuée à travers de nombreux indicateurs, comme les liens entre les différents équipements, permettant de résumer le comportement global des communications.

Dans le cadre des approches de détection non supervisées (voir Section 2.2.3) la majeure partie de la complexité de la détection d'anomalies est inhérente à la nécessité de découvrir ces informations de contexte au sein des données elles-mêmes.

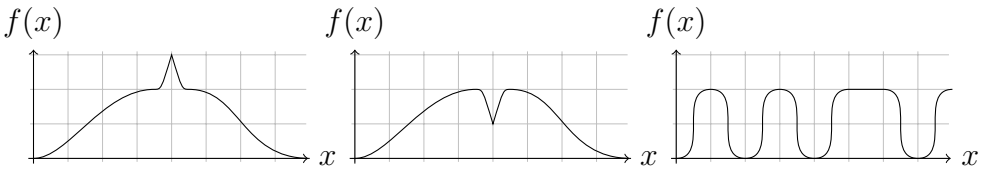
Il existe donc principalement quatre obstacles dans la mise en place même d'un outil de détection : la définition de la frontière normal/anormal qui est rarement statique et évidente à distinguer, la présence de bruit dans les données, les dynamiques propres du système à étudier et l'hétérogénéité des données et des applications. Nous allons voir à présent les différentes caractéristiques des anomalies et des annotations utilisézs pour signaler celles-ci.

2.1.3 Types d'anomalies et d'annotations

D'après [Chandola *et al.*, 2009], il existe trois types d'anomalies. Pour donner une meilleure interprétation à chacune de ces définitions, prenons l'ensemble de départ $\mathcal{D}$ contenant n observations x_i avec $0 < i \leq n$.

2.1.3.1 Anomalie dite "unitaire" ou globale

Un objet est considéré comme une anomalie globale s'il dévie fortement des autres objets. Un exemple d'anomalie globale est illustré sur la figure 2.2a.



(a) Anomalie globale (b) Anomalie locale (c) Anomalie collective

FIGURE 2.2 – Trois types d'anomalies sur des séries temporelles : (a) pic anormal, (b) baisse anormale, (c) continuité anormale des valeurs de la série.

Dans notre ensemble de départ $\mathcal{D}$, les observations x_i sont des entiers compris entre 1 et 5. Si une nouvelle observation x_{n+1} déroge à la règle en ayant une valeur entière de 12, elle sera identifiée dans ce cas de figure comme une anomalie globale par rapport à l'ensemble $\mathcal{A}$. Son comportement diffère des autres données contenues dans cet ensemble.

2.1.3.2 Anomalie contextuelle

Une anomalie est contextuelle si elle dépend de la situation où elle se trouve, tel qu'une chute brutale de la température au cours d'une journée ensoleillée. De manière conditionnelle, on ne peut pas décider (ou ne pas décider correctement) sans avoir d'information sur le contexte. Une anomalie contextuelle se définit à une échelle locale, en tenant compte du contexte de son voisinage et des instances qui l'entourent, si bien qu'à l'échelle globale elle n'est pas caractérisée comme une anomalie ainsi présenté sur la figure 2.2b.

Pour ce nouvel exemple, les observations x_i de $\mathcal{D}$ avec un indice i impair ont une valeur entière strictement négative tel que $x_1 = -3$, $x_3 = -5$, etc, et les observations avec un indice i pair possèdent des valeurs entières strictement positives tel que $x_2 = 2$, $x_4 = 5$, etc. Si une nouvelle observation x_{n+1} avec $n + 1$ impair a une valeur positive tel que $x_{n+1} = 3$, elle est dans ce cas une anomalie contextuelle. Elle aurait pu ne pas être considérée comme

anormale si nous n'avions pas tenu compte de la parité de l'indice i ou si elle avait été observée dans un autre contexte, ici avec un indice pair. Cette observation se distingue des autres par la nature même de son contexte, ici la parité de l'indice.

Cet indice aurait pu représenter les heures d'une journée, ou toute autre information temporelle. Ce type d'anomalie est donc très spécifique aux données temporelles telles que les séries temporelles ou les flux de données que nous décrirons dans la section 2.1.5.

2.1.3.3 Anomalie collective

Une observation seule ne peut être une anomalie collective. Ce type d'anomalie est un ensemble de plusieurs observations normales comme présenté sur la figure 2.2c. En analysant de façon collective un ensemble d'observations normales et en tenant compte des liens entre celles-ci, elles reflètent un comportement anormal par rapport à l'ensemble des données.

Par exemple, durant une analyse des commandes exécutées sur un ordinateur, une tentative d'authentification à distance seule ne semble pas anormale. Si cette tentative est exécutée de façon répétée, il se peut qu'il y ait une tentative d'intrusion à distance. Dans ce cas, la suite répétée d'une même observation donne lieu à une anomalie collective. Alors qu'en regardant les observations indépendamment les unes par rapport aux autres, elles ne présentent rien d'anormal.

Nous venons d'aborder trois grands types d'anomalies. À présent nous allons voir comment les signaler et informer l'utilisateur du résultat de la détection.

2.1.3.4 Annotations : score ou label

À la sortie d'une méthode d'apprentissage quelconque, après avoir analysé une ou plusieurs observations, l'utilisateur de la méthode peut être informé de deux façons : par un score ou par un label.

Notons à présent, pour toute observation x, la fonction $f(x) = y$ représentant une méthode d'apprentissage quelconque. La sortie y décrit l'annotation attribuée à l'observation x, par cette méthode d'apprentissage. Cette sortie décrit alors l'état normal ou anormal de l'observation y. La sortie de la fonction f d'une nouvelle observation x_{n+1} sera alors notée y_{n+1}. Durant le processus de détection, pour indiquer si l'observation x en cours d'annotation est normale ou anormale, notons y le label ou le score attribué à x. Un label peut prendre la valeur $y = 0$ s'il s'agit d'une observation normale, ou

$y = 1$ si elle ne l'est pas. Pour résumer, labelliser la sortie consiste alors à lui attribuer une valeur binaire.

Si nous décidons d'attribuer un score, normalisé entre $0.0 \leq y \leq 1.0$, plus la valeur du score sera proche de 0.0 et plus l'observation sera considérée comme normale. À l'inverse, plus le score est proche de 1.0 et plus l'observation sera considérée comme anormale. Si son score est $y = 0.5$, cela signifie que nous n'avons aucune indication permettant de trancher entre normale et anormale. Il est possible de transformer un score en label en définissant un seuil. Tous les scores strictement supérieurs à 0.6 seront labellisés anormaux, et normaux dans le cas contraire.

Les différents types d'anomalies à détecter et la façon de les étiqueter sont deux caractéristiques qui influencent le choix de la méthode d'apprentissage. Toutefois ces caractéristiques sont elles-mêmes définies par la problématique de détection spécifique au domaine d'application et dépendent de ce que l'on souhaite déceler et du contexte dans lequel la détection est appliquée. C'est pourquoi, nous allons à présent aborder les différents domaines de la détection d'anomalies.

2.1.4 Les différents domaines d'application de la détection d'anomalies

La détection d'anomalies ou détection d'évènements inattendus est un défi que nous retrouvons dans de nombreux domaines d'application. Ces derniers font varier les problématiques et les types de données à traiter, ce qui influence fortement sur le choix de la technique utilisée. Quelques-uns de ces grands domaines d'application sont présentés ici. Le choix s'est porté sur des domaines où nous retrouvons un grand nombre de travaux et de méthodes différentes, et sur les domaines proches de notre cas d'application : la supervision des flux du système d'information.

2.1.4.1 Sécurité et intrusion dans les systèmes informatique

Les intrusions dans les systèmes informatiques sont de nos jours monnaie courante, notamment pour les entités possédant un large réseau ou des données sensibles. La supervision des interactions entre utilisateur et système mais aussi celle au sein même d'un réseau dans les systèmes informatiques est un des premiers domaines à avoir utilisé des méthodes d'apprentissage pour faire de la détection d'anomalies. Cela s'explique par le fait qu'il a pu très tôt fournir des quantités suffisantes de données à manipuler.

Dans ce domaine de détection, l'objectif est de repérer un comportement qui semble malintentionné, que ce soit sur une machine ou sur un réseau. Les obstacles de cette détection résident dans la volumétrie des données à analyser, et dans le fait qu'elle doit se faire au plus tôt c'est-à-dire en temps-réel.

Sur une machine, une méthode peut être d'analyser une suite de traces d'appels au système, et de déterminer s'il y a tentative d'intrusion [Forrest *et al.*, 1996]. Sur un réseau, la démarche peut être similaire, mais elle pourra analyser des paquets de données [Garcia-Teodoro *et al.*, 2009]. Dans le premier cas, l'analyse porte sur des données symboliques (ou discrètes), tandis que dans le second cas l'analyse peut être composée d'un mélange de données symboliques et continues.

2.1.4.2 Anomalie dans le contexte industriel

La détection d'anomalies dans le contexte industriel est applicable à différents domaines tels que la maintenance prédictive, la supervision d'un ou plusieurs composants industriels (e.g. chaîne de production), le contrôle de qualité lors de la fabrication d'un produit, la détection des défauts structurels importants, et enfin, la supervision des flux de messages dans un système d'information qui correspond au cadre d'application de cette thèse.

Pour la détection, l'analyse s'effectue principalement sur des indicateurs permettant de mettre en lumière le bon ou le mauvais fonctionnement du système. En règle générale, l'évaluation des indicateurs au cours du temps permet d'avoir une visualisation intuitive du fonctionnement dynamique du système. Les flux analysés peuvent provenir de multiples sources. Cependant les données traitées peuvent être aussi sous forme d'évènement discret et ne décrire que l'état d'un composant, comme l'état ouvert ou fermé d'une valve.

Dans le cas de la maintenance prédictive et de la supervision de composants industriels, les indicateurs peuvent provenir de capteurs de température, de pression, ou encore de vitesse. Cela permet d'intervenir avant que l'un des composants ne dysfonctionne, à cause de l'usure, de la fatigue ou encore d'un mauvais réglage.

La supervision des systèmes de communication informatique peut être mise en place pour deux raisons : anticiper ou alerter les pannes, ou bien détecter les intrusions et autres actions malintentionnées, voire les deux en même temps. Il existe de nombreuses publications sur la détection d'intrusions [Ahmed *et al.*, 2016] mettant en lumière plusieurs méthodes pour différents types d'anomalies. Dans le contexte de cette thèse, la supervision a pour objectif de s'assurer que le système fonctionne correctement et, comme nous le verrons dans la section 5, de garantir que les messages circulent bien

dans les chaînes applicatives. À l'instar de la supervision des composants industriels, nous surveillons différents indicateurs permettant de fournir l'état de santé du système.

Les systèmes de communications sont souvent très complexes et ont tendance à évoluer dans le temps, ce qui peut donner parfois des comportements inattendus. Ce sous-domaine est assez peu développé, mais nous noterons tout de même qu'il se rapproche de la détection d'intrusions dans les réseaux informatiques par la nature des instances analysées telles que l'étude statistique de traces (appelées aussi logs) ou le nombre de messages reçus [Garcia-Teodoro *et al.*, 2009, Gupta et Sekar, 2003], et de la supervision de composants industriels par la volonté de prévenir une potentielle panne [Fujimaki *et al.*, 2005].

2.1.4.3 Détection de fraude

Avec la dématérialisation des paiements et des transactions, les échanges financiers se sont considérablement démocratisés. La lutte contre les transactions frauduleuses est un enjeu majeur au sein des établissements financiers, les amenant à développer des méthodes de détection d'anomalies.

Les deux types de fraudes les plus étudiées sont la fraude à la carte bleue et la fraude à l'assurance. Pour le premier, une base des transactions bancaires est maintenue pour chaque utilisateur, avec pour information la date de l'évènement, le lieu, le type, mais aussi le montant. Pour le second type, bien que la problématique puisse sembler assez similaire, les informations seront légèrement différentes, et les types d'anomalies recherchées n'ont pas les mêmes indicateurs révélateurs. D'autant plus que les catégories d'assurances peuvent être variées (assurance vie, maladie, ...), les indicateurs révélateurs seront probablement différents au sein de cette même problématique.

Dans la surveillance de marché financier, domaine plus récent, les indicateurs seront essentiellement basés sur les prix des marchés et les échanges [Donoho, 2004]. En effet, le changement soudain du prix d'un produit suivi par la vente massive de ce même produit peut être le signe d'une préparation à une transaction frauduleuse, dans le but de racheter le produit à bas prix ou au contraire de le vendre au prix fort durant un temps anormalement court. Cependant, il est aussi possible d'ajouter un bon nombre d'indicateurs statistiques autour de ces deux valeurs, le prix du produit et le nombre de transaction sur ce produit. De nouveaux acteurs tels que BlackRock, une entreprise financière gestionnaire d'actifs, utilisent des méthodes d'apprentissages (ALADDIN dans le cas de BlackRock) pour analyser le système financier dans sa globalité dans le but de pérenniser les transactions, et dans le même temps, de les augmenter et de les accélérer. La puissance financière

du groupe et les accusations dont il fait l'objet se retrouve au centre des préoccupations comme le rapporte un documentaire diffusé par la chaîne ARTE, sujet repris par de nombreux médias tels que [Leder, 2020]. C'est pourquoi il devient indispensable de pouvoir, au même rythme que ces entreprises, analyser ces transactions pour s'assurer qu'il n'y a rien de frauduleux derrière ces actions financières.

Pour résumer, le domaine de la détection de fraude s'applique principalement à de la fraude financière et transactionnelle, mais également dans de nombreux cas d'usage différents. La nature des données varie entre ces deux grands types de fraude. La fraude à la carte bleue et à l'assurance se focalise plus particulièrement sur les caractéristiques de la transaction, comme son emplacement géographique lors du retrait bancaire ou de l'accident, pour détecter un vol de numéro de carte de retrait ou une fausse déclaration. Dans l'analyse du système financier, l'étude se fera plutôt sur les valeurs des marchandises et des différentes transactions entre les entités financières sur ces marchandises, pour éviter des arrangements financiers portant préjudice au bon fonctionnement du marché économique.

2.1.4.4 Diagnostic médical

Le milieu médical utilise toutes sortes de données pour diagnostiquer les patients. Pour un même diagnostic, les données peuvent être le témoignage écrit des symptômes du patient, jusqu'à la prise de mesures sous forme de série temporelle pouvant durer plusieurs heures, en passant par la tomodensitométrie, qui consiste à scanner le corps humain à l'aide de rayons X, ou autre procédé d'imagerie médicale (IRM, échographie Doppler, ...). Les données sont donc très variées, ce qui rend ce domaine d'autant plus complexe à appréhender.

Les méthodes classiques de diagnostic médical utilisent des arbres de décisions [Soni *et al.*, 2011] suivant les symptômes décrits par le patient. Elles sont assez similaires à la tâche effectuée par le médecin lorsqu'il s'adresse à son patient.

Depuis peu, l'utilisation des méthodes d'apprentissage appliquées au milieu médical est en plein essor. Par exemple dans les mammographies [Spence *et al.*, 2001] pour la détection d'éventuelles tumeurs dans le diagnostic du cancer du sein, ou pour la détection de la rétinopathie diabétique via la détection de taches [Chakrabarty, 2018]. L'avantage ici est de pouvoir analyser en détail une image et de détecter des symptômes qui ne sont pas toujours visibles à l'œil nu. Dans de multiples situations d'analyses, on étudie principalement une ou plusieurs images en 2D ou en 3D. Dans d'autres cas, on étudie des données temporelles comme dans l'analyse des électrocardiogrammes [Lin

et al., 2005] où l'anomalie serait représentée par une période de trouble dans la périodicité du signal (saut trop fréquent ou signal trop lisse). Une des difficultés dans le diagnostic médical est d'éviter les faux négatifs, qui pourraient avoir des conséquences vitales pour le patient.

2.1.4.5 Détection dans les réseaux sociaux

L'extension d'Internet, principal réseau de communication, a fait croître nos échanges. Les outils de communications sont de plus en plus développés et de plus en plus rapides. La quantité de données échangées a donc elle aussi fortement augmenté, et les informations qu'elles contiennent se sont diversifiées. De plus, les échanges se font sur de multiples supports et parfois sous différents protocoles de communication.

Une quantité importante de contenus et de données complexes est partagée sur les réseaux sociaux, par courriel ou sur des sites personnels, sous forme de texte, d'image, voire les deux. La difficulté réside dans le traitement d'un contenu très vaste afin d'identifier ceux qui pourraient être qualifiés d'inappropriés (par exemple, de la publicité non signalée comme telle ou encore des propos haineux). La méthode la plus simple est d'analyser si le nouveau contenu est en adéquation avec les autres. Sur un forum par exemple, un texte venant d'être écrit sera analysé pour s'assurer qu'il correspond à la thématique du forum. Si ce texte contient un discours approprié il sera officiellement posté par les modérateurs. S'il contient un discours de haine [Ribeiro *et al.*, 2017], il ne sera pas posté ou retiré s'il a déjà été posté en ligne. Mais dans un premier temps, il est nécessaire d'avoir analysé le forum et repéré les différents sujets traités [Allan *et al.*, 1998, Kasiviswanathan *et al.*, 2011]. Pour renforcer l'analyse, il est aussi possible d'étudier les liens entre les contenus par le biais de structure de graphe [Savage *et al.*, 2014].

Les spécificités des réseaux sociaux sont leur large quantité de données, de sources, et l'hétérogénéité des contenus. C'est un domaine présentant de nombreuses méthodes d'apprentissage pour assurer cohérence et adéquation du contenu sur les réseaux sociaux.

La liste des différents domaines présentés n'est pas exhaustive, il existe de nombreux champs d'application liés à la détection d'anomalies tels que l'analyse de perturbations dans l'écosystème présentée par [Verbesselt *et al.*, 2012], la vidéo-surveillance [Collins *et al.*, 2000], ou bien encore l'analyse des trajectoires des transports en commun [Kong *et al.*, 2018].

Les domaines d'applications ont leurs spécificités et leurs caractéristiques propres, mais il semble que la nature des données s'inscrit comme une problématique récurrente. Dans la prochaine section nous tenterons donc d'aborder

et d'expliciter cette particularité.

2.1.5 Nature des données

Comme nous avons pu le voir dans la section 2.1.4, il existe un large panel de données dans lesquelles il est possible de détecter des anomalies. Cette variabilité aura une grande influence sur les techniques de détection employées. Les données, ou observations, univariées ne possèdent qu'un attribut, aussi appelé variable dans le domaine des statistiques. Un attribut représente une valeur, une caractéristique de représentation de la donnée. Les données multivariées possèdent plusieurs attributs.

2.1.5.1 Données univariées

De façon plus formelle, un ensemble $\mathcal{D}$ de n données univariées sera décrit de la façon suivante :

$$D = (x_1, x_2, \ldots, x_n), 0 < i \leq n$$

les x_i prenant leur valeur dans un domaine défini.

2.1.5.2 Données multivariées

Pour un ensemble $\mathcal{D}$ de n données multivariées de dimension $\mathcal{D}$ on notera :

$$D = (x_1, x_2, \ldots, x_n) \text{ avec } x_i \text{ un vecteur noté } \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,d} \end{pmatrix} \text{ et } 0 < i \leq n$$

Dans l'ensemble $\mathcal{D}$, les données x_i sont communément appelées "vecteur d'attribut". La représentation de cet ensemble sous forme matricielle s'écrit :

$$D = \begin{pmatrix} x_{1,1} & x_{2,1} & \ldots & x_{n,1} \\ x_{1,2} & x_{2,2} & \ldots & x_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,d} & x_{2,d} & \ldots & x_{n,d} \end{pmatrix}$$

Chaque ligne de la matrice représente un attribut, c'est-à-dire une dimension. Chaque colonne représente une observation contenant $\mathcal{D}$ attributs. Dans le cas multivarié, les attributs peuvent être ou non du même type.

Nous allons voir ici en détail les différentes caractéristiques que peuvent avoir les données à traiter.

2.1.5.3 Types de données

2.1.5.3.a Attributs discrets ou continus

Un attribut discret possède un ensemble fini ou infini dénombrable de valeurs, qui peuvent être ou non des entiers [Han *et al.*, 2012]. Les couleurs, les formes géométriques sont des attributs de description qui peuvent être considérés comme des attributs discrets. Dans la catégorie des ensembles infinis dénombrables, ce sont la plupart du temps des entiers permettant d'identifier un état ou de comptabiliser un effectif.

Dans le cas contraire, si un attribut n'est pas dénombrable, alors il est continu. Ces attributs sont généralement représentés par des nombres réels.

Nous allons voir quelques catégories de données et dans quelle mesure ces dernières influencent le processus de détection d'anomalies avec des difficultés qui leurs sont propres.

2.1.5.3.b Données textuelles

Il existe une large variété de type de données représentées sous forme de texte. Lorsque des données textuelles sont utilisées, elles sont généralement considérées comme des données discrètes [1]. Les données textuelles sont composées de mots, eux-mêmes composés de lettres, et le tout est porteur de sens. La Grammaire Universelle, théorie proposée par Noam Chomsky [Chomsky, 1986], est une explication plutôt claire sur la faculté humaine à utiliser ce type de données. D'après cette théorie, il existe des règles de structure du langage innées à l'humain, indépendamment de toute expérience sensorielle. L'être humain a alors, tout comme pour l'expression et la compréhension orale, des facilités avec la communication écrite, ce qui peut expliquer pourquoi il existe tant de données textuelles.

Dans certaines situations, la chaîne de caractères s'étudie en tant que telle, par exemple pour effectuer des corrections orthographiques. Un mot mal orthographié devra être détecté comme anormal. Dans d'autres cas, nous privilégierons l'étude de la syntaxe pour vérifier le respect des règles grammaticales qui permettent différentes combinaisons de mots ou de logs informatiques bien précises. Une règle non assimilée durant la phase d'apprentissage est détectée comme anormale. Et enfin, l'analyse sémantique est l'étude du sens d'une phrase (ou d'un "sème" : la plus petite unité de sens). Une phrase ou un texte vide de sens est détecté comme anormal. Le test de Turing [Turing, 1950] est probablement le test le plus connu pour ce

1. Il est possible d'encoder chaque chiffre par une lettre, et les utiliser dans un ensemble infinis non dénombrable, mais dans ce cas on s'éloigne du sens du mot "textuel".

cas d'usage, sauf qu'ici la détection se fait par un individu qui doit juger à l'aveugle si son interlocuteur est un humain ou une machine. Dans cet exemple, la pertinence des propos joue un rôle important.

On trouve de nombreux travaux sur l'analyse des commandes système en informatique [Forrest *et al.*, 1996], l'analyse du contenu d'Internet, ou bien encore l'analyse des séquences ADN [Gupta *et al.*, 2014]. Comme nous venons de voir, la variabilité du type d'information est large, ce qui rend le domaine d'autant plus intéressant qu'il propose de nombreuses méthodes spécifiques aux données discrètes.

2.1.5.3.c Données visuelles

Les images occupent une place de plus en plus importante grâce au développement du numérique et des puissances de calculs toujours plus grandes. Des entreprises comme Google stockent de plus en plus de photos pour alimenter leurs bases de données et améliorer leur système de reconnaissance d'images [Szegedy *et al.*, 2015].

Les domaines d'application des méthodes d'apprentissage automatique sont nombreux, et peuvent aller du domaine médical [Spence *et al.*, 2001] à la conduite autonome, en passant par la vidéo-surveillance [Basharat *et al.*, 2008]. Les images peuvent être une représentation de ce que nous connaissons, comme ce que nous voyons et que nous immortalisons à l'aide d'un appareil photo ou d'une caméra. Au format numérique, la photographie est une matrice de pixels (où chaque pixel représente une couleur), mais il existe d'autres façons de capturer des images, par exemple grâce à la peinture, ou bien encore grâce à l'interférométrie, technique capable de capturer des ondes invisibles à l'œil nu, utilisée dans de nombreux domaines (astronomie, océanographie, ...). Il est possible grâce à l'interférométrie de détecter des micro fractures dans des objets métalliques [Friesem et Vest, 1969] ou de détecter des cancers du sein à l'aide d'enregistrements interférométriques à déphasage [Woisetschläger *et al.*, 1994].

Pour ce type de données, la principale difficulté réside dans la taille des images à analyser par l'utilisateur, où une seule image peut contenir plusieurs millions de pixels. La seconde difficulté est l'identification des anomalies et leur interprétation, souvent complexe à identifier, comme la détection de druses dans des images de rétines de l'œil humain [Freund *et al.*, 2009] pour anticiper et éviter la dégradation de la vue liée à la dégénérescence maculaire.

2.1.5.3.d Réseau et graphe

Avec le développement d'Internet, les réseaux de communications sont toujours plus grands. Même au sein d'une entreprise, le réseau interne est parfois très complexe et peut transporter des données sensibles. Les graphes sont des structures de données qui illustrent les relations entre les entités et sont par conséquent universellement applicables.

Par exemple, une machine permettant l'émission et la réception de données est décrite comme un nœud, et les arêtes entre les différents nœuds directement interconnectés [Jiang *et al.*, 2014] symbolisent les connexions entre ces machines. Suivant le type d'anomalie que l'on souhaite détecter, l'analyse peut s'effectuer sous différentes granularités [Ahmed *et al.*, 2016]. Pour des attaques DoS (attaque par déni de service), l'analyse porte principalement sur le nombre de données envoyées car l'inondation de requêtes sur un serveur peut empêcher son fonctionnement normal. Tandis que pour les attaques dites "par sondage", il est nécessaire de se tourner vers une analyse des requêtes effectuées (saint, portsweep, mscann nmap, ...), ce qui permet de trouver les failles d'une machine [Hoque *et al.*, 2012].

Les réseaux sociaux permettent aussi d'obtenir des graphes grâce aux liens obtenus entre les individus, et entre les sujets abordés [Hamilton *et al.*, 2017]. Il est nécessaire d'avoir une connaissance sur ces liens pour soulever l'hypothèse que la proximité entre deux objets sur les réseaux sociaux apporte de l'information sur leur similitude. Cela permet par exemple de mieux identifier, en regroupant les voisins, les objets anormaux.

2.1.5.3.e Données temporelles

Ces données comportent une information complémentaire : le temps. Il permet de contextualiser les données, notamment les unes par rapport aux autres. C'est pourquoi, la détection d'anomalies dans les données temporelles va principalement se focaliser sur les anomalies contextuelles, mais aussi détecter des ruptures dans les données, c'est-à-dire des anomalies globales [Gupta *et al.*, 2014]. Notons l'ensemble $\mathcal{D}$ de n données contenant deux attributs, dont l'attribut temporel t :

$$D = (x_1, x_2, \ldots, x_n) \text{ et } x_i \text{ un tuple noté } (a_i, t_i) \text{ sachant } 0 < i \leq n$$

avec a_i représentant la valeur observée à l'instant t_i. Les données peuvent aussi posséder plusieurs attributs en plus de l'attribut temporel. Dans ce cas, on notera :

$$x_i = (a_{i,1}, a_{i,2}, \ldots, a_{i,d}, t_i) \text{ avec } x_i \text{ possèdant } d \text{ attributs.}$$

Les attributs temporels peuvent aussi représenter une durée (par exemple 30 secondes, 1 heure, 3 jours et 12 heures, ...), un horaire (c'est-à-dire une durée *modulo* un jour), ou un moment précis appelé horodatage. L'horodatage indique la date et l'heure de l'observation.

Dans la représentation matricielle multivariée de l'ensemble $\mathcal{D}$, définissons le vecteur t, tel que pour tout i tel que $0 < i \leq n$ on obtient l'attribut temporel t_i de l'observation x_i indiqué à la $i^{\text{ème}}$ colonne de la matrice de l'ensemble $\mathcal{D}$:

$$D = \begin{pmatrix} x_{1,1} & x_{2,1} & \dots & x_{n,1} \\ x_{1,2} & x_{2,2} & \dots & x_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1,d} & x_{2,d} & \dots & x_{n,d} \end{pmatrix} \text{ et } t = (t_1, t_2, \dots, t_n)$$

Nous pouvons constater qu'il existe une séquentialité dans les données permettant de définir pour chaque observation une notion de voisinage temporel avec les autres observations, grâce à cette notion d'ordre intrinsèque aux données temporelles. La problématique de prévision (ou *forecasting*) est très proche de la détection d'anomalies temporelles, car les anomalies recherchées à l'état courant sont souvent définies comme une déviation de ce qui est attendu [Gupta *et al.*, 2014] en comparaison à ce qui a été observé précédemment.

La temporalité de ces données permet de les lier les unes aux autres en prenant en considération leurs voisinages respectifs. De plus, une pratique assez courante consiste à agréger durant l'analyse deux observations très proches ou égales temporellement, pour ne faire qu'une seule et même observation. Par exemple l'observation de la valeur de $x_{1,1}$ a été générée à $t_{1,1}$ le 2 février à 16h et l'observation de la valeur de $x_{1,2}$ a été générée à $t_{1,2}$ le 2 février à 21h. Si nous posons t_1 représentant l'attribut temporel du 2 février, alors l'observation multivariée x_1 contient les deux valeurs $x_{1,1}$ et $x_{1,2}$. Mais encore, il est possible de tenir compte de l'attribut a_1 de x_1 généré à l'instant t_1 pour l'analyse de la pertinence de l'attribut a_2 de x_2 généré à l'instant t_2. Par exemple, l'ouverture d'une fenêtre quelques minutes après une augmentation du chauffage dans une même pièce sont deux actions indépendantes et assez normales, mais l'ordre et la proximité temporelle de ces deux actions renvoient à un comportement anormal.

C'est aussi le cas pour une suite de commandes système, considérée comme une anomalie collective si une suite de plusieurs commandes s'avère anormale [Forrest *et al.*, 1996]. Comme expliqué dans la section 2.1.3.3, lorsqu'une même commande de tentative de connexion à une machine est répétée trop souvent, dans une fenêtre temporelle courte, cette sous-séquence,

comprenant une répétition de tentative de connexion, sera annotée comme anormale.

Les données temporelles sont définies sous différentes caractéristiques. Elles peuvent être synchrones ou asynchrones, et dans ce dernier cas le décalage temporel entre différentes sources de données peut compliquer l'analyse. Les données temporelles peuvent aussi être périodiques car elles sont parfois la représentation d'observations répétées à intervalles réguliers (quotidien, mensuel, ...) et dans le cas contraire, elles sont apériodiques. Certaines méthodes sont spécialisées dans l'apprentissage de cette caractéristique, telle que ARIMA [Bianco *et al.*, 2001].

Nous pouvons alors analyser les données temporelles point par point, ou bien par séquence de valeurs temporelles appelée "série temporelle". Les données temporelles peuvent être aussi bien univariées (même si la temporalité peut finalement être vue comme une variable particulière) ou multivariées, comme expliqué plus haut dans ce paragraphe. Il existe aussi certains cas où les données sont récoltées en continu, sans interruption, formant ainsi un flux de données.

Nous allons à présent nous intéresser à deux catégories de données temporelles, les séries temporelles et les flux de données (communément appelé *data stream* dans la littérature).

Séries temporelles. Les séries temporelles sont des séquences de valeurs ordonnées dans le temps, comme par exemple les mesures d'un électrocardiogramme [Keogh *et al.*, 2005]. Mais le terme "série temporelle" est parfois utilisé sans qu'il y ait réellement de notion temporelle dans les données brutes, mais seulement une notion de voisinage, comme dans les travaux de [Wei *et al.*, 2008] qui propose une méthode pour accélérer la comparaison des images entre elles grâce à la conversion en données univariées des formes que contiennent ces images. Elles peuvent représenter différents types de données telles que le cours d'une valeur boursière, la consommation électrique, ou les formes contenues dans une image [Wei *et al.*, 2008].

Les séries temporelles sont généralement vues comme des données univariées, dans lesquelles chaque valeur temporelle t_i possède un attribut a_i représentant le même type de données. Pour une série temporelle s de longueur fixe w :

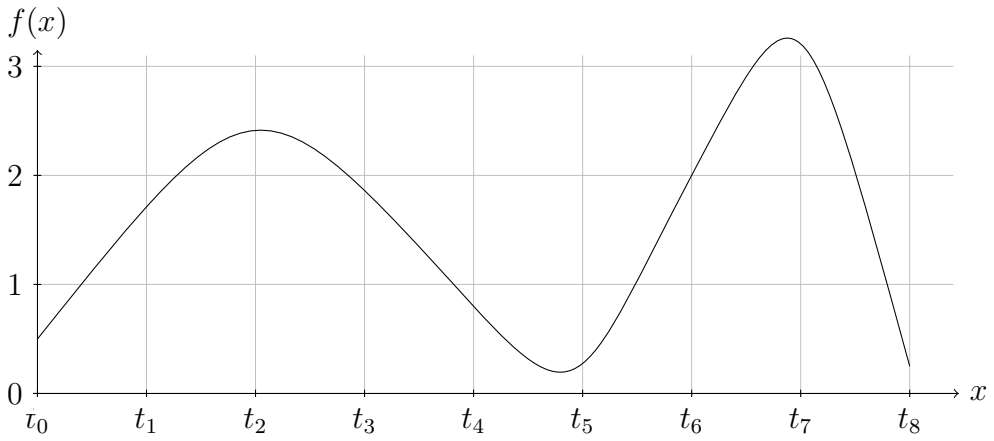
$$s = [(a_1, t_1), (a_2, t_2), \ldots, (a_w, t_w)]$$

Mais les séries temporelles peuvent aussi être multivariées et contenir des types de données différents pour chacun des attributs. Dans ce cas, si plusieurs attributs sont observés durant une même valeur temporelle t_i, l'en-

semble des $\mathcal{D}$ attributs est noté dans un vecteur colonne et la série temporelle s est sous forme d'une matrice $d \times w$:

$$s = \begin{pmatrix} (a_{1,1}, t_1) & (a_{2,1}, t_2) & \dots & (a_{w,1}, t_w) \\ (a_{1,2}, t_1) & (a_{2,2}, t_2) & \dots & (a_{w,2}, t_w) \\ \vdots & \vdots & \ddots & \vdots \\ (a_{1,d}, t_1) & (a_{2,d}, t_2) & \dots & (a_{w,d}, t_w) \end{pmatrix}$$

Un exemple de série temporelle multivariée est présenté sur la figure 2.3.



(a)

temps	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
couleur	bleu	vert	rouge	rouge	bleu	rouge	vert	bleu

(b)

FIGURE 2.3 – Un exemple de série temporelle multivariée de longueur 8. La figure (a) représente la courbe des valeurs du premier attribut, et la figure (b) représente les valeurs du second attribut. Par exemple, la série temporelle multivariée à l'instant t_1 retourne les valeurs 38 et "bleu".

Lors de l'analyse classique de plusieurs séries temporelles, ces dernières possèdent la même temporalité, c'est-à-dire qu'elles possèdent les mêmes attributs temporels, et contiennent le même nombre de valeur. En règle générale, les méthodes d'analyse et de détection utilisent des mesures de distance entre les séries temporelles d'une même base de séries. Ces méthodes permettent par exemple de les trier par similarité, ou retourner celles qui sont

les plus aberrantes. Nous verrons quelques-unes de ces méthodes en détail dans la section 2.2.

Si les séries temporelles à comparer ont des longueurs différentes, et donc un nombre d'observations différent, il est envisageable d'effectuer un pré-traitement sur celles-ci. Par exemple, il est possible d'utiliser la méthode d'approximation PAA (*Piecewise Aggregate Approximation*) pour obtenir des séquences de longueur fixe contenant les moyennes des valeurs sur chaque intervalle dans les séries, comme expliqué par [Lin *et al.*, 2007a].

Le pré-traitement peut aussi être nécessaire lorsque les séries temporelles se trouvent sur des temporalités différentes, c'est-à-dire représentant des créneaux temporels différents. Un créneau temporel représente une fenêtre temporelle qui peut être de l'ordre de plusieurs secondes, plusieurs jours, ou plusieurs années. Par exemple, le sens et l'intensité du trafic routier urbain dépend fortement du créneau temporel qui est observé. Sur un même tronçon routier, le sens du trafic sera différent entre le matin et le soir d'une même journée, et l'intensité sera différente entre un jour de semaine et un jour de week-end. Il est alors possible de mesurer la distance temporelle entre les séries pour ne comparer que celles qui représentent des créneaux temporels identiques ou proches. Ainsi, pour un système avec un comportement périodique basé sur la journée, il semble pertinent de vouloir comparer seulement les séquences placées sur le même créneau horaire provenant de journées différentes. C'est souvent le cas des systèmes qui dépendent du comportement humain à grande échelle, comme la consommation électrique globale. Pourtant, deux séquences décalées dans le temps ne sont pas forcément très différentes, comme nous pouvons le constater sur la figure 2.4.

Durant la comparaison de séries temporelles dont les créneaux temporels se chevauchent mais qui ne sont pas identiques, il est envisageable de confronter entre elles les sous-séquences qui ont les mêmes valeurs temporelles, pour éviter la méthode de force brute, la plus coûteuse, qui consiste à comparer les valeurs deux à deux. Cependant, la distance obtenue est relative au nombre d'observations temporellement identiques entre deux séquences, et est difficilement comparable avec les autres distances obtenues entre les autres séries temporelles dont la taille d'intersection temporelle des observations est différente.

Pour contrer ces difficultés, il existe différentes méthodes. Nous pouvons citer *Dynamic Time Wrapping* [Berndt et Clifford, 1994] qui permet de donner de l'élasticité durant l'analyse. Cette mesure de distance est largement utilisée dans la littérature. Elle permet de comparer des séries de longueurs différentes, et de créer un décalage sur la comparaison entre les valeurs de deux séries temporelles en faisant intervenir plusieurs fois la même valeur

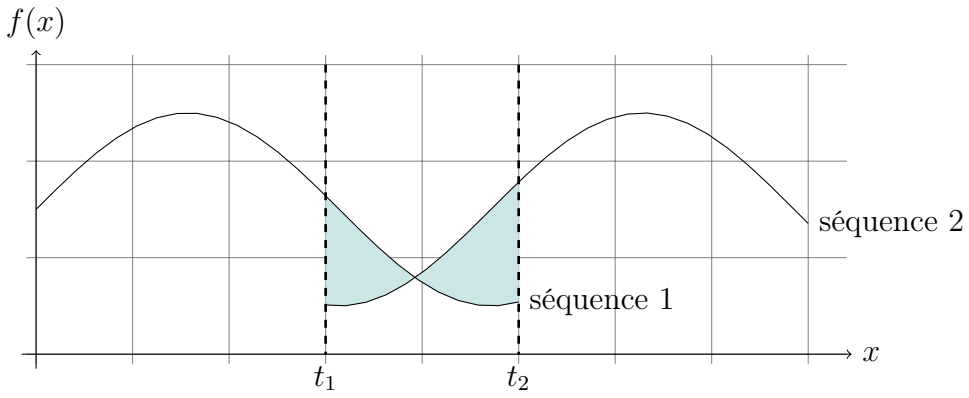


FIGURE 2.4 – Deux séquences de même longueur, décalées dans le temps. Bien qu'elles soient très similaires, la distance sur la fenêtre temporelle $[t_1; t_2]$ ne donne pas d'informations pertinentes quant à leurs ressemblances.

lors de l'utilisation de la mesure de distance.

Flux de données. Les flux de données se différencient des autres types de données par le fait qu'ils n'ont pas de longueur fixe [Gupta *et al.*, 2014]. Avec le développement toujours plus croissant de l'électronique et des réseaux, les capteurs qui délivrent des messages numériques sont de plus en plus nombreux et émettent toujours plus. Le nombre des flux de données disponibles croît, d'autant que le nombre de données générées par unité de temps augmente également. Cela a ouvert depuis quelques décennies la porte à un nouveau domaine de recherche [Aggarwal, 2007, Garofalakis *et al.*, 2016], l'analyse de flux de données.

La spécificité des flux de données est que les données arrivent en continu sans durée déterminée. Il n'y a pas de fin de flux, ni de temps strictement défini entre l'arrivée de deux données. C'est l'une des difficultés intrinsèques de ces types de flux. Chaque donnée contient un ou plusieurs attributs, et le plus souvent possède un horodatage.

Un capteur de température peut par exemple envoyer sa mesure sans interruption, avec un horodatage pour chacune d'entre elles, que ce soit à l'émission ou à la réception. Ce type de flux de données représente une série temporelle. Le choix de la granularité peut revenir au consommateur, qui prendra par exemple la dernière valeur ou la moyenne de ces dernières sur une fenêtre de temps définie. Une autre façon de procéder est de générer des séries temporelles à partir d'un ou plusieurs flux de données, à l'aide d'une fenêtre glissante [Datar et Motwani, 2007].

Comme la détection dans les flux de données est bien différente de celle des

séries temporelles, le jeu de données Numenta Anomaly Benchmark (NAB) de Numenta, proposé par [Lavin et Ahmad, 2015] et disponible sur GitHub [2], a été créé pour pouvoir évaluer les méthodes de détection d'anomalies sur flux de données. Le benchmark NAB sera décrit plus en détail dans le chapitre 3.

2.1.5.4 Données de haute dimension : fléau de la dimensionnalité

Les données sont considérées comme étant en haute dimension lorsqu'elles possèdent au moins quelques dizaines d'attributs. Leur nombre peut s'élever ensuite à des centaines voire des milliers d'attributs.

Le premier obstacle rencontré avec des objets en haute dimension est de les visualiser. En effet, l'esprit humain est capable d'interpréter une visualisation en 2 ou 3 dimensions, avec éventuellement le temps comme dimension supplémentaire. Mais au-delà de 4 dimensions, la visualisation devient inconcevable pour l'esprit humain. Il existe certains travaux sur la visualisation en 2 dimensions pour des objets en haute dimension [Andrews, 1972] avec par exemple l'algorithme SNE (*Stochastic Neighbor Embedding*) [Hinton et Roweis, 2003]. De plus, il est plutôt fréquent de trouver des bases de données contenant de nombreux attributs, nous retrouvons d'ailleurs de nombreux travaux de classification sur des données en haute dimension [Agrawal *et al.*, 1998].

En plus de la complexité de la visualisation, d'autres problématiques ont été incluses dans ce qui est communément appelé "fléau de la dimensionnalité", ou "malédiction de la dimensionnalité", terme utilisé pour la première fois par [Bellmann, 1961]. La recherche par similarité entre objets est particulièrement touchée par ce problème [Beyer *et al.*, 1999]. Dans [Zimek *et al.*, 2012], plusieurs phénomènes de ce fléau sont étudiés, et nous nous intéresserons, dans le cadre de cette thèse, à deux d'entre eux : le phénomène de concentration, et celui que nous nommerons dans ce travail de thèse l'effet *hubness*.

2.1.5.4.a Phénomène de concentration

Avec l'augmentation du nombre de dimension, les distances entre les objets se concentrent et tendent à être toutes similaires. Ce phénomène contre-intuitif est appelé "concentration des distances" [Angiulli, 2020]. Dans cette situation, les objets se répartissent de façon uniforme sur la surface d'une sphère, de sorte que les distances entre les objets deviennent indiscernables.

Nous verrons par la suite à la section 2.2 que ce phénomène met en difficulté les méthodes d'apprentissage utilisant la notion de distance, ou de

2. https://github.com/numenta/NAB

similarité, entre les observations pour fournir un résultat. Dans cette section, nous allons seulement étudier de façon formelle le phénomène de concentration des distances. Mais avant cela, nous allons rappeler de façon semi-formelle quelques concepts statistiques.

Variable aléatoire réelle. Une variable aléatoire réelle X est une fonction mesurable décrite sur un espace de probabilité défini sur $\mathbb{R}$, c'est-à-dire l'ensemble des réels, ou en partie définie sur $\mathbb{R}$. Elle définit l'ensemble des résultats possibles d'une expérience aléatoire. La loi décrivant les valeurs attendues et leurs probabilités d'apparitions se nomme "loi de probabilité".

Une variable aléatoire peut être de nature discrète si les valeurs qu'elle peut prendre appartiennent à une liste finie ou dénombrable de valeurs spécifiques, ou de nature continue dans le cas de valeurs numériques d'un ensemble indénombrable.

Vecteur aléatoire. Un vecteur aléatoire $\boldsymbol{X}$ est une généralisation multivariée d'une variable aléatoire réelle. Pour $\mathcal{D}$ dimensions, on note $\boldsymbol{X}$ un vecteur colonne $(X_1, X_2, \ldots, X_d)^T$ tel que X_i est une variable aléatoire réelle avec $0 < i \leq d$.

Loi de probabilité. Une loi de probabilité permet de décrire le comportement d'une variable aléatoire. Dans le cas d'une variable aléatoire réelle discrète, chaque valeur possible se voit attribuer une probabilité d'apparition. Dans le cas d'un dé à 6 faces non pipé, chacune des faces se voit attribuée une probabilité de $1/6$ à chaque lancé. Dans le cas d'une variable aléatoire réelle continue, le nombre de possibilités étant infini, il n'est pas possible d'attribuer une probabilité raisonnable à chacune des valeurs. Dans ce dernier cas, il est nécessaire d'étudier la probabilité d'apparition d'un intervalle de valeurs.

Fonction de répartition cumulative. La fonction de répartition cumulative (notée *cdf* pour *cumulative distribution function*) d'une variable aléatoire X est la fonction F_X qui pour tout $x \in \mathbb{R}$ retourne la probabilité d'obtenir une valeur inférieure ou égale à x :

$$F_X = \mathbb{P}(X \leq x)$$

La probabilité que X soit dans un intervalle compris entre $\mathcal{A}$ et b semi-ouvert à gauche s'écrit :

$$\mathbb{P}(a < X \leq b) = F_X(b) - F_X(a)$$

Fonction de densité. La fonction de densité ou densité de probabilité (notée *pdf* pour *probability density function*) d'une variable aléatoire X est la fonction f_X qui pour tout $x \in \mathbb{R}$ retourne la probabilité relative d'appartenir à un domaine de valeurs. La probabilité relative de la variable aléatoire réelle X d'obtenir la valeur x, relativement à l'ensemble des autres valeurs, se dérive à partir de la fonction de répartition cumulative F_X de la variable aléatoire réelle X :

$$F_X(x) = \int_{-\infty}^{x} f(x)\,\mathrm{d}x \iff f_X(x) = \frac{\mathrm{d}}{\mathrm{d}x} F_X(x).$$

La notion de probabilité relative s'oppose à la notion de probabilité absolue. Dans le cas de variable aléatoire continue, la probabilité absolue d'obtenir une valeur réelle précise est nulle car il existe une infinité de solutions.

2.1.5.4.b Propriété de la concentration

Nous allons aborder comment se caractérise la concentration des distances de façon plus formelle. Comme développé par [Francois *et al.*, 2007], si l'on note Z et Z' deux variables aléatoires identiquement distribuées, la distribution des distances entre les points générés par Z est équivalente à celle de $||Z - Z'||$. Si l'on note $X = Z - Z'$, alors étudier la distance deux à deux entre les points du jeu de données produit par Z revient à analyser la distribution de $||X||$, la norme de X.

Si l'on considère la variance relative VR, définit par [Beyer *et al.*, 1999] :

$$VR = \frac{\sqrt{Var(||X||)}}{\mathrm{E}(||X||)}$$

comme mesure de concentration de la distribution de $||X||$, on observe que lorsque la variance relative diminue, la distribution de la norme se concentre autour de la moyenne. Ceci pose un problème pour la détection des anomalies lorsque les données proviennent de mesures réelles, c'est-à-dire des mesures réalisées en précision finie. Cependant il a été démontré par [Francois *et al.*, 2007] que la variance relative tend vers 0 que lorsque la dimension de X augmente, et ce pour toute norme L_p avec $p \geq 1$, ce qui rend en pratique l'identification des anomalies plus difficile dans les données de forte dimensionnalité.

C'est également ce qui est affirmé par [Zimek *et al.*, 2012], initialement présenté par [Beyer *et al.*, 1999], avec l'expression du contraste relatif, qui est la différence proportionnelle entre les points les plus éloignés et les points les plus proches. Prenons X_d une variable aléatoire, soit D_{max} la distance entre les points les plus éloignés et D_{min} la distance entre les points les plus

proches. Le contraste relatif est définit par :

$$CR = \frac{D_{max} - D_{min}}{D_{min}}$$

Il a été montré par [Francois *et al.*, 2007] que pour toute norme L_p avec $p \geq 1$ on a :

$$\lim_{d \to \infty} \frac{D_{max} - D_{min}}{D_{min}} \to 0$$

ce qui signifie que pour tout objet du jeu de données produit par la variable aléatoire X_d, les distances de ces objets vers le point le plus proche et vers le point le plus éloigné tendent à être similaires lorsque la dimension augmente. En effet, lorsque la dimension augmente, la moyenne des distances (ou l'espérance de la norme) augmente légèrement avant de se stabiliser, tandis que l'écart-type des distances (ou l'écart-type de la norme) diminue. Sur la figure 2.5 de [Zimek *et al.*, 2012], nous constatons que les distances euclidiennes entre objets se concentrent lorsque la dimension augmente.

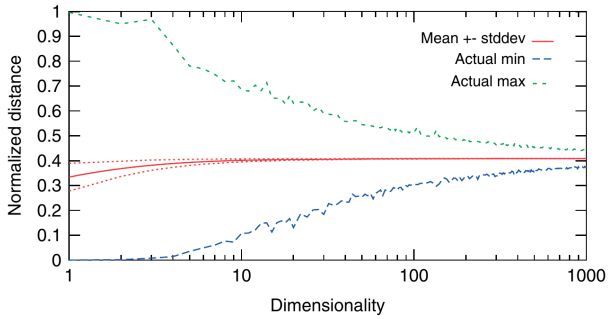


FIGURE 2.5 – Ce graphique est présenté par [Zimek *et al.*, 2012]. Du haut vers le bas : en vert la distance maximale, en rouge la moyenne +- l'écart-type et en bleu la distance minimale des distances euclidiennes normalisées deux à deux. Nous observons que les distances minimum et maximum entre les points se rapprochent, et que l'écart-type diminue lorsque la dimension augmente.

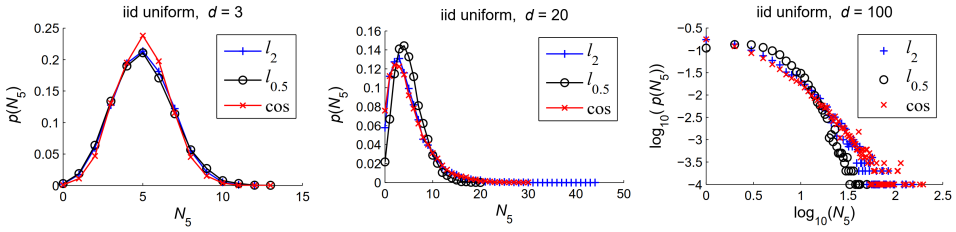
2.1.5.4.c Effet *hubness*

Nous notons $N_k(x)$ le score dit de k-occurrence défini comme le nombre de fois où l'objet x de l'ensemble $\mathcal{D}$ appartient aux k plus proches voisins des autres objets de $\mathcal{D}$. L'effet *hubness* a récemment été examiné par [Radovanović *et al.*, 2010] de façon détaillée. Lorsque la dimension augmente, la distribution de N_k se déséquilibre de sorte qu'un petit nombre d'objets

appelés *hubs* deviennent très fréquemment présents dans la liste des plus proches voisins des autres objets en effectif majoritaire. Ces *hubs* obtiennent une valeur de N_k bien plus grande que celles des autres objets. Pour les autres objets, ils sont très minoritaires dans les listes des plus proches voisins, et obtiennent une faible valeur de N_k (*antihubs*), alors que ces objets sont plus nombreux dans l'ensemble $\mathcal{D}$ par rapport aux objets dit *hubs*. Cet effet se produit même pour une valeur de k très petite par rapport à la taille de l'ensemble $\mathcal{D}$.

Pour résumé, l'effet *hubness* retourne un faible nombre d'objets *hubs* et un grand nombre d'objets *antihubs*. Cela n'est pas sans conséquence pour les méthodes d'apprentissage basées sur le voisinage entre les objets. Lors de la phase de détection d'anomalies, ces méthodes annotent un faible nombre d'objets comme normaux (les *hubs*) car ils sont considérés comme les voisins les plus proches de tous les autres. Ainsi, un grand nombre d'objets sont annotés anormaux (les *antihubs*), puisqu'ils apparaissent très distants par rapport à l'ensemble des objets, y compris par les *antihubs*.

Cette asymétrie du score de k-occurrences est présentée Figure 2.6 par [Radovanović *et al.*, 2009] avec l'observation des distributions de N_5 pour différentes dimensions $d = (3, 20, 100)$ et différentes fonctions de distance (norme L_2, $L_{0.5}$ et cosinus) pour 10000 objets de dimension $\mathcal{D}$ répartis uniformément. Pour $d = 3$, la distribution semble représenter la fonction de densité d'une loi Binomiale. Lorsque la dimension augmente, un déséquilibre est observé à droite de la distribution, un faible nombre d'objets obtient une valeur de N_k élevée, et ce déséquilibre est de plus en plus significatif lorsque la dimension augmente, quelle que soit la fonction de distance.



(a) (b) (c)

FIGURE 2.6 – Distribution des 5-occurrences pour les distances euclidiennes présenté par [Radovanović *et al.*, 2009], $l_{0.5}$ et cosinus pour un ensemble d'objets répartis uniformément de dimension (a) $d = 3$, (b) $d = 20$ et (c) $d = 100$.

Beaucoup de travaux explorent le fléau de la dimensionnalité pour mieux

TABLE 2.1 – Croisement entre les domaines d'applications cités précédemment et les types de données utilisés dans la littérature.

	texte	image	réseau/ graphe	série temporelle	flux de données
sécurité et intrusion	X	X	X		
contexte industrielle				X	X
fraude	X			X	X
diagnostic médical		X		X	X
réseaux sociaux	X	X	X		

comprendre ses origines et ses conséquences. Il touche de nombreux domaines, et deux phénomènes attirent particulièrement notre attention : la concentration des distances et l'effet *hubness*. En effet, pour les méthodes basées sur le voisinage des objets, ces phénomènes peuvent rendre indiscernables les objets anormaux des objets normaux. Nous reviendrons d'ailleurs sur ces aspects dans la section 2.2.3.3.

2.1.6 Résumé des notions abordées Section 2.1

Nous venons d'aborder une liste non exhaustive des différents cas d'utilisation des méthodes de détection d'anomalies que nous avons rencontrés dans la littérature. De plus, une liste des types de données a été présentée permettant au lecteur d'aborder les principaux éléments susceptibles d'être manipulés, en décrivant leurs structures et certaines de leurs applications.

Pour résumer, nous présentons ici le tableau 2.1 présentant les différents types de données en fonction des cas d'usage que l'on a pu aborder dans la section 2.1.4. Il semble évident que pour chaque domaine, il est envisageable d'utiliser tous les types de données selon la spécificité de l'application. Cependant, les paires "type de données/domaine d'application" mises en avant sont principalement celles que nous avons rencontrées dans la littérature et que nous avons choisi de présenter. Notons que tous les domaines d'applications incluent la possibilité de traiter des données en haute dimension.

Les recherches concernant la sécurité et l'intrusion se sont peu portées sur les séries temporelles et les flux de données, types nous avons définis à la section 2.1.5.3.e. Elles se concentrent particulièrement sur des données textuelles et visuelles pour détecter une intrusion ou une tentative. Nous pouvons faire ce même constat pour la détection dans les réseaux sociaux.

Les séries temporelles et les flux de données se retrouvent principalement dans trois domaines d'application : le contexte industriel, la fraude et le

diagnostic médical. Les points communs entre ces trois domaines ne sont, à première vue, pas évidents à identifier. Cependant, l'aspect temporel est essentiel dans ces trois domaines : la répétition et la périodicité des signaux sont les principales caractéristiques de l'analyse, permettant de comparer les nouvelles séries à un historique de séries normales préalablement analysées et validées par des experts du domaine.

Dans le chapitre suivant, nous allons aborder plus en détail différentes méthodes d'apprentissages utilisées dans le domaine de la détection d'anomalies. Nous verrons que ces méthodes présentent certains avantages et inconvénients et que le choix d'utilisation de ces méthodes dépend du type de données à analyser et des domaines d'applications de ces méthodes.

2.2 Différentes familles de méthodes d'apprentissage automatique

L'apprentissage automatique, ou l'apprentissage machine (*machine learning*) consiste à utiliser un ordinateur pour résoudre une tâche en utilisant des modèles statistiques. Ce domaine existe et intéresse les spécialistes du domaine depuis le milieu du XX^{e} siècle. Il est un jeune sous-domaine de l'informatique et de l'intelligence artificielle. Alan Turing, grâce à ses nombreuses contributions durant les années 1940 et son test de Turing [Turing, 1950] qui a posé les fondations des domaines de l'informatique, de l'intelligence artificielle et de l'apprentissage automatique.

Il inventa ce test, qu'il appelait initialement "le jeu d'imitation", pour permettre d'évaluer la faculté d'une machine à imiter un humain durant un échange verbal. L'objectif est simple : une personne ayant le rôle de juge s'adresse à une machine et à un autre individu. Dès le départ, le juge n'a aucune information pour savoir lequel des deux interlocuteurs est la machine. S'il n'est pas capable de juger sans se tromper, au bout de quelques échanges, lequel des deux se trouve être la machine, alors le test est réussi. Ce test se base sur la faculté de la machine à échanger une conversation et rester crédible dans ce scénario. Dans les années 60, des programmes informatiques comme ELIZA, proposé par [Weizenbaum, 1966], étaient déjà capables de jouer le rôle du psychanalyste en retournant des questions formulées à partir des affirmations de l'utilisateur. Et depuis plusieurs années, nous pouvons à présent communiquer avec des assistants vocaux pour différentes tâches telles que envoyer un mail, consulter la météo, ou encore commander des systèmes mécaniques dans une maison connectée (volets roulants, chauffe-

eau, ...). Ce test reste toujours aujourd'hui une référence pour évaluer un système d'apprentissage, et varie parfois en fonction des exigences et de la tâche à effectuer. Nous pourrions imaginer un concours de peinture entre un individu et une machine, et le juge devra évaluer qui, de l'individu et de la machine, a peint quel tableau.

Bien qu'il est possible de trouver dans les années 1930 quelques travaux sur ce domaine, comme ceux de Fisher cherchant à inférer un modèle linéaire à partir d'un jeu de données [Cornuéjols *et al.*, 2018], l'apprentissage automatique connut un réel essor durant la seconde moitié du 20^{e} siècle. Les limites des perceptrons [Minsky et Papert, 1969] furent un coup dur pour le domaine, malgré différents courants qui se sont formés dans le domaine. Le connexionnisme, supporté par de nombreux chercheurs comme Warren McCulloch et Walter Pitts [McCulloch et Pitts, 1943] et directement hérité des sciences cognitives, fut l'un de ces courants touchés par ce revers.

Aujourd'hui, un nombre incalculable de modèles statistiques existe, généralement formés pour exécuter une tâche bien précise à partir d'un jeu d'observation, que nous appelons aussi données d'apprentissage. Le modèle est le résultat de l'analyse du jeu d'observation, et permet de prendre une décision sur les données futures, ou de les prédire. Il fournira une solution en sortie en fonction de la donnée reçue en entrée et de l'apprentissage acquis grâce au jeu d'observations.

Il existe trois catégories de méthodes d'apprentissage automatique que nous aborderons en détail dans la section suivante : supervisée, non supervisée, et semi-supervisée. La différence entre ces trois catégories est liée aux données fournies pour l'apprentissage.

2.2.1 Hiérarchie des méthodes d'apprentissage

Dans un premier temps, nous allons définir les trois grandes catégories de méthodes d'apprentissage, et dans un second temps nous verrons les différentes sous-catégories comme présenté sur la figure 2.7.

2.2.1.1 Méthodes supervisées

Les méthodes d'apprentissage supervisées s'exécutent en deux temps. Tout d'abord la phase d'apprentissage, qui consiste à se former sur un jeu de données. Durant cette phase, la méthode apprend avec les données fournies en entrée, et corrige ses sorties selon les réponses attendues. Les données d'apprentissage sont donc labellisées, permettant à la méthode de développer sa fonction de prédiction. Ensuite vient la phase de test durant laquelle la fonction de prédiction apprise est utilisée sur un nouveau jeu de données.

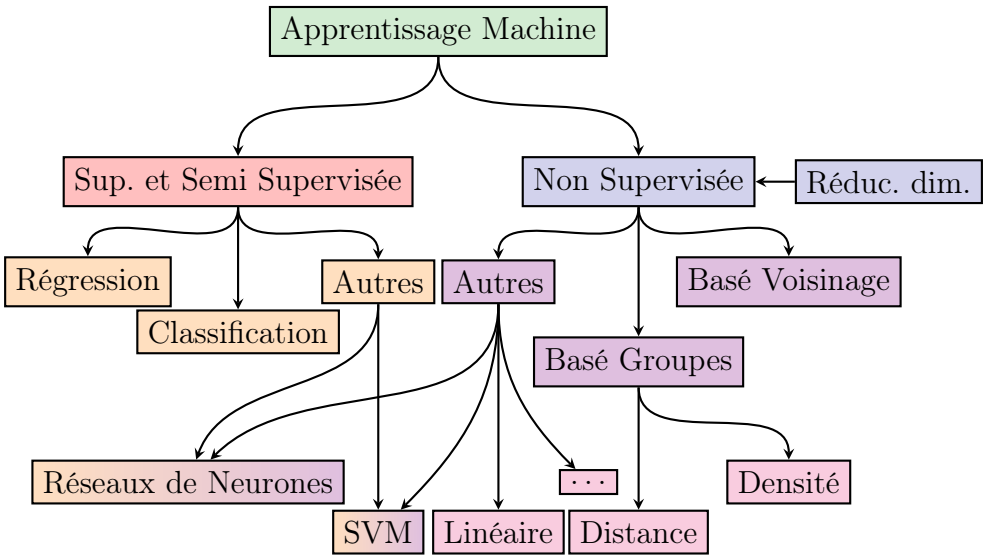


FIGURE 2.7 – Présentation synthétique des approches présentées dans les sections 2.2.2 et 2.2.3.

Dans le contexte de la détection d'anomalies, les données sont souvent divisées en deux classes : les données normales, généralement majoritaires, et les données anormales, correspondant à des anomalies avérées.

2.2.1.2 Méthodes non supervisées

Pour cette famille de méthodes d'apprentissage, les données en entrée ne sont pas étiquetées. Les sorties attendues pour les données fournies ne sont pas connues : le choix de la méthode non supervisée est donc influencé par l'objectif de l'apprentissage. Il existe de nombreuses situations où l'étiquetage de l'historique des données est long et fastidieux [Schneider *et al.*, 2016], ou indisponible car imputable à un manque d'informations sur ces données. Comme précédemment abordé dans la section 2.1.2 : identifier la frontière entre normal et anormal est souvent une tâche délicate ce qui explique notamment la fréquence élevée de l'usage de cette catégorie de méthode dans le domaine de la détection d'anomalies.

Dans le domaine de la détection d'anomalies, il est possible de faire l'hypothèse que les données collectées soient majoritairement normales et donc de s'abstraire du besoin de labelliser les données. Pour déterminer quelles sont les données anormales, ces méthodes utilisent généralement le degré de similitude entre les objets, c'est-à-dire entre les données à évaluer et le reste des données du jeu.

2.2.1.3 Méthodes semi-supervisées

Cette famille propose des méthodes se basant sur l'étiquetage de seulement quelques données qui peuvent être normales, anormales, ou les deux. Ces méthodes s'avèrent efficaces sur des données bien représentées, c'est-à-dire que les quelques données étiquetées permettent de généraliser au mieux les données non-étiquetées qui les entourent et que nous souhaitons apprendre.

Pour cette catégorie de méthodes appliquées à la détection d'anomalies, ce sont généralement des données étiquetées comme étant normales qui sont apprises. C'est le cas pour la détection d'intrusion système où l'historique des données défini comme normal est utilisé pour tester les nouvelles données. Mais il existe certains cas où l'étiquetage des anomalies est plus évident, par exemple sur les réseaux sociaux où il est préférable d'étiqueter les données anormales car la quantité de données est beaucoup trop importante pour pouvoir étiqueter les données normales.

Pour ces trois types de méthodes d'apprentissage (supervisée, non supervisée et semi-supervisée), il est tout à fait envisageable de s'attendre à vouloir détecter des anomalies qui ne ressemblent pas aux données fournies. Une donnée dite "nouvelle" n'appartient à aucune classe précédemment rencontrée. La détection d'anomalie se distingue alors de la détection de "nouveauté", car cette dernière est plutôt amenée à surveiller ce qui est classifié et à signaler ce qui n'a jamais été rencontré. Nous définissons la nouveauté comme un type spécifique d'anomalie. Dans ce travail de thèse, nous souhaitons détecter les anomalies de manière globale, c'est -à-dire à la fois les nouveautés mais aussi celles déjà rencontrées.

Ainsi, le domaine de la détection d'anomalies inclut un nombre important de méthodes non supervisées [Schneider *et al.*, 2016]. En effet, l'anomalie que nous cherchons à identifier possède, dans beaucoup de cas d'application, une structure inattendue et n'appartient à aucune classe clairement et préalablement définie. De plus, il peut être très coûteux de vouloir annoter toutes les données d'apprentissage, car elles peuvent être considérablement volumineuses.

Cependant, il existe certains cas d'usage dans lesquels les types d'anomalies recherchées et les caractéristiques qui permettent de les discriminer sont connus. Dans ce cas, les données anormales peuvent former un ou plusieurs groupes, facilitant l'apprentissage et renforçant la détection. De plus, les anomalies sont souvent très spécifiques aux types d'activités étudiées [Aggarwal, 2017], et demandent alors une expertise métier solide. Superviser les anomalies peut permettre de limiter les faux positifs ou de classifier les dif-

férentes anomalies détectées pour trouver la ou les causes plus rapidement. Le problème de détection d'anomalies supervisée est réduit à un problème de classification, avec une ou plusieurs classes d'anomalies à identifier.

Les méthodes supervisées se catégorisent en deux grandes classes : régression ou classification, choisies en fonction de la tâche à réaliser. Quelques rares exceptions comme les réseaux de neurones peuvent être utilisés pour effectuer l'une ou l'autre de ces deux tâches : nous avons alors décidé de les présenter dans la suite de ce travail comme une classe de méthodes supervisées à part entière.

En ce qui concerne les méthodes non supervisées présentées, nous avons choisi de les séparer dans deux sections distinctes : les méthodes basées sur le voisinage des objets (distance et densité), et celles basées sur la construction de groupe. Le choix entre ces deux approches se fait en fonction du type de données et du type de problème à traiter.

La figure 2.7 illustre cette classification. Nous avons tenu compte de cette classification pour présenter, dans ce travail, les différentes méthodes. À présent nous allons aborder plus en détail ces trois familles et décrire quelques-unes des méthodes les plus considérées dans la littérature. Nos travaux portent sur des données non étiquetées, c'est pourquoi nous verrons les méthodes non supervisées de façon plus précise dans la section 2.2.3. Nous allons alors commencer par présenter les méthodes supervisées et semi-supervisées.

2.2.2 Familles des méthodes supervisées et semi-supervisées

Comme expliqué précédemment, les méthodes supervisées fonctionnent en deux étapes :

— Une étape d'apprentissage : les données sont utilisées pour affiner la fonction de prédiction des données,
— Puis une étape de prédiction : pour une nouvelle observation, utilisation de la fonction de prédiction pour obtenir un résultat (étiquette, valeur continue) pour cette observation.

Les méthodes semi-supervisées travaillent aussi selon ces deux étapes, en prenant en compte que les données d'apprentissages ne sont pas toutes étiquetées. L'inclusion des données non étiquetées pour l'apprentissage est facultative. Les données étiquetées serviront à coup sûr de référence pour l'apprentissage.

Dans la famille des méthodes supervisées et semi-supervisées, il existe différentes sous-familles (classification, régression) avec chacune leurs spécificités. Les méthodes de classification et les méthodes de régression répondent à deux besoins différents. Nous verrons dans la suite de cette section les principales familles de méthodes utilisées et nous détaillerons les différences qui les caractérisent.

2.2.2.1 Méthodes de classification

Pour une donnée en entrée, la classification retourne une valeur discrète en sortie. Cette valeur représente la classe d'appartenance de la donnée. Durant la phase d'apprentissage, le modèle apprend à différencier les différentes classes présentées. Pour une nouvelle donnée, durant la phase de test, la valeur en sortie est une prédiction de la classe d'appartenance de cette donnée. C'est pourquoi certaines méthodes fournissent parfois une prédiction d'appartenance, et non pas une valeur discrète. Il suffit alors, au système ou à l'utilisateur, de choisir la prédiction la plus pertinente, en principe la plus élevée. Si la prédiction d'appartenance retourne le label d'une classe anormale, alors la donnée en entrée est anormale. Il est aussi possible de prédire la classe d'appartenance d'un nouvel objet. Dans ce cas, si la classe prédite est fausse, alors l'objet est considéré comme anormal. Ce second cas d'usage est moins courant.

2.2.2.1.a Méthodes basées proximité et voisinage

Les méthodes basées sur la distance (ou basées proximité) entre les objets (ou sur un groupe d'objets) labellisent une nouvelle donnée selon les étiquettes des données les plus proches. La comparaison entre les objets s'effectue suivant les attributs qui les composent, à l'aide d'une mesure de distance ou de similarité. Les résultats de ce type de méthode sont parfois utilisés dans d'autres méthodes plus complexes, assimilant les informations de proximité et de voisinage comme une connaissance supplémentaire dans le traitement de la méthode de plus haut niveau.

La méthode des k plus proches voisins (kNN ou NN_k pour *k-nearest neighbors* en anglais) s'impose comme une référence dans le domaine, grâce à sa simplicité mais aussi grâce à son efficacité. Dans un ensemble $\mathcal{D}$ contenant n observations X de dimension $\mathcal{D}$, la fonction NN_1 retourne l'observation X_i la plus proche de l'observation passée en paramètre :

$$\mathrm{NN}_1(X_{n+1}) = X_i \text{ tel que } \min_{0<i\leq n} \{||X_i - X_{n+1}||\}$$

Cette fonction est généralisable avec NN_k avec $0 < k \leq n$ pour les k plus proches voisins de X_{n+1} dans l'ensemble $\mathcal{D}$. Il existe de nombreuses distances différentes, dont l'ouvrage [Deza et Deza, 2009] présente un large éventail, permettant de retourner des résultats de distances et donc de voisinages très différents selon la norme ou la métrique utilisées.

Les méthodes des k voisins les plus proches consistent à regarder l'étiquetage des données dans le voisinage proche de la donnée à étiqueter. Comme expliqué par [Aggarwal, 2017], si la majorité des k données les plus proches est étiquetée comme anormale, alors la nouvelle donnée sera labellisée comme telle.

2.2.2.1.b Méthodes basées règles

Ces méthodes sont basées sur les règles apprises durant l'apprentissage. Chacune des règles possède une condition de départ, par exemple une action ou une série d'action, et une conséquence, dans le cas de la détection, un label précisant ou non la nature des données (normale ou non).

En se basant sur le principe du système immunitaire, [Forrest *et al.*, 1994] proposent la méthode de sélection négative qui fonctionne en deux temps. La première étape est de créer des détecteurs. Chaque détecteur est une séquence créée aléatoirement. Il est comparé à l'ensemble de référence constitué de séquences normales appelées *self strings*. Les séquences sont une suite d'octets, pouvant aussi bien être des symboles. Les *self strings* sont les séquences à protéger. Lors de la comparaison, si un détecteur ne correspond à aucun *self string*, il sera préservé pour la seconde étape ou rejeté dans le cas contraire. Durant la seconde étape, pour vérifier l'intégrité d'une séquence à protéger, on la compare à chacun des détecteurs pour s'assurer qu'ils ne correspondent pas. Sinon, la séquence a été modifiée et corrompue. Cette méthode a été reprise pour effectuer de la détection d'anomalies dans des données multi-dimensionnelles [Dasgupta et Majumdar, 2002].

Pour détecter les intrusions, [Lee *et al.*, 1997] proposent l'analyse des traces d'appel système à l'aide d'une fenêtre glissante. Ils parcourent les séries de traces d'appel système pour générer des séquences de longueur fixe qui contiennent elles aussi ces traces. Durant l'apprentissage, ils utilisent RIPPER [Cohen, 1995] pour créer un ensemble de règles. Durant la phase de test, pour une trace donnée, ils analysent toutes les séquences générées à partir de cette trace. Chaque séquence est testée par l'algorithme RIPPER, attribuant l'étiquette correspondante (normale ou anormale) selon le nombre de prédictions anormales retournées par RIPPER. Ensuite, ils analysent la proportion de séquences anormales vis à vis du nombre de séquences totales. Si le nombre (ou le pourcentage) de séquences anormales dépasse un certain

seuil, alors la trace analysée est définie comme anormale.

2.2.2.1.c Arbres de décision

Les arbres de décision partitionnent les données selon une ou plusieurs dimensions, à chaque niveau de l'arbre. Ces méthodes sont très proches, par nature, des méthodes basées règles car il est possible de considérer chaque branche de l'arbre comme une règle à part entière. Pour améliorer la séparation des données à chaque nœud de l'arbre, il est possible d'utiliser des mesures tel que l'indice de GINI. Celui-ci permet de mesurer la pertinence du critère de séparation, en calculant la répartition des étiquettes des données dans les différentes partitions durant la phase d'apprentissage. Plus l'indice de GINI renverra une petite valeur, et plus les groupes de données seront correctement partitionnés, selon la variable et le seuil de séparation choisis pour cette variable. Durant la phase de test, il suffit de parcourir l'arbre de décision ainsi construit pour déterminer le label de la nouvelle donnée.

Il existe d'autres stratégies de séparation des données, comme présentées par [Breslow et Aha, 1997]. Par exemple, l'algorithme ID3 proposée par [Quinlan, 1986], très réputé dans le domaine, utilise l'entropie de SHANNON. Dans le domaine de la détection d'anomalies, nous noterons le travail de [Thaseen et Kumar, 2013] qui compare plusieurs arbres de décision pour la détection d'intrusion réseaux.

2.2.2.1.d Méthodes bayésiennes naïves

Les méthodes de classification bayésiennes naïves font partie des méthodes se basant sur les statistiques des données pour en construire une description abstraite de celles-ci. Elles sont basées sur l'inférence bayésienne qui calcule les probabilités qu'un événement soit le fruit d'une ou plusieurs causes. Si une nouvelle donnée possède une valeur ayant une faible probabilité d'apparition, cette donnée sera étiquetée comme anormale. À partir d'un ensemble d'observation, par exemple le groupe sanguin des parents d'un individu I, d'après le théorème de BAYES, il est possible de calculer la probabilité que l'individu I soit de groupe sanguin B+.

Ces méthodes sont appliquées dans différents types d'analyse comme la détection d'intrusion dans les systèmes [Sebyala *et al.*, 2002, Panda et Patra, 2007]. Il existe des méthodes d'apprentissage plus complexes se basant sur l'inférence bayésienne comme il est possible de voir avec [Domingos, 1999] pour de la classification binaire d'imagerie médicale, pour minimiser l'erreur de classification du jeu d'apprentissage.

2.2.2.2 Méthodes de régression

Contrairement aux méthodes de classification, les méthodes de régression fournissent comme résultat une valeur continue. Dans ce cas de figure, ce n'est plus une classe d'appartenance que nous souhaitons obtenir, mais la prédiction d'une variable continue en sortie, comme par exemple la température extérieure, selon une ou plusieurs variables en entrée. Pour prédire cette valeur, le modèle apprend sur l'historique des températures passées, en tenant compte des différents indicateurs fournis en entrée, comme l'heure de la journée, ou encore la puissance du thermostat. Pour la détection d'anomalies, il est généralement attendu que la prédiction soit proche de la valeur mesurée, indépendamment de la méthode de prédiction. Si l'erreur de prédiction est trop grande, la valeur mesurée sera considérée comme anormale. L'erreur se mesure comme étant la distance entre la valeur obtenue et la valeur mesurée.

2.2.2.2.a Méthodes de régression linéaire

Tout comme pour les méthodes bayésiennes vues précédemment, les méthodes de régression, même les plus simples, font partie des méthodes se basant sur les statistiques des données en s'appuyant sur les données précédentes pour fournir un résultat.

Un modèle de régression estime le ou les cœfficients permettant de construire une fonction de prédiction. Cette dernière, cherchant à établir une relation entre la variable en entrée et la variable en sortie, doit autant que possible minimiser l'erreur entre la sortie estimée (par la fonction) et la sortie attendue. Le modèle ajuste les cœfficients pour atteindre au mieux cet objectif. Le score d'anomalie pour une nouvelle donnée sera déterminé suivant la distance qui le sépare de la valeur prédite par la fonction de régression.

Il existe de nombreux types de régression, la plus connue étant la régression linéaire. Une analyse préalable de la structure des données permet de choisir le type de régression. Pour un modèle linéaire, l'objectif est de trouver les poids w_i de la droite de régression permettant de déduire au mieux, sachant i variables indépendantes X_i, la variable dépendante Y, avec :

$$Y = w_0 + X_1 \times w_1$$

où w_0 est l'ordonnée à l'origine. L'objectif de la phase d'apprentissage étant de trouver les poids w_i qui minimisent l'erreur de prédiction sur un jeu d'apprentissage. Il existe différentes méthodes, comme celle par exemple qui utilise la minimisation de la somme du carré des écarts, permettant ainsi de trouver les valeurs des poids w_i.

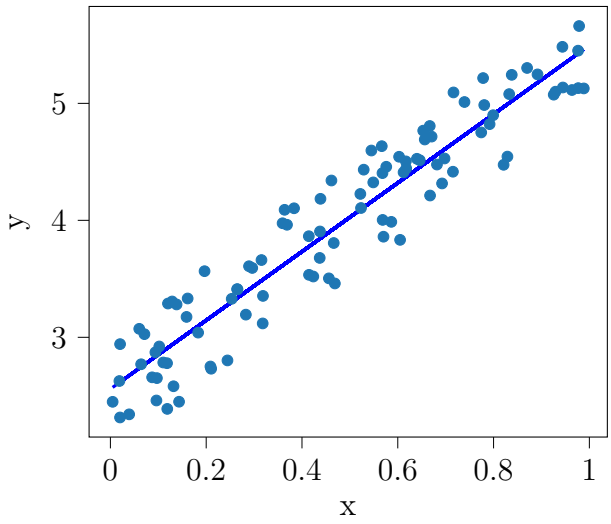


FIGURE 2.8 – Un exemple de régression linéaire.

La méthode lasso, proposée par [Tibshirani, 1996], rajoute une contrainte supplémentaire lors de l'obtention des valeurs des poids. Cette nouvelle contrainte cherche à minimiser le nombre de poids non nuls, permettant ainsi de proposer une régression plus simple pour la prédiction des données.

2.2.2.2.b Méthodes autorégressives

Les modèles de régression ARMA (pour "*AutoRegressive Moving-Average*") sont largement utilisés sur les données temporelles et connaissent de nombreuses variantes notamment pour la détection d'anomalies [Bianco *et al.*, 2001]. L'objectif de ces modèles est de trouver la meilleure régression qui soit, pour prédire au mieux les valeurs futures. Le modèle de référence (ARMA) est la fusion de deux modèles : un modèle autorégressif (AR) et un modèle de moyenne mobile (MA). Le premier modèle utilise les valeurs passées pour prédire, en décalage, la future valeur, tandis que le second modélise l'erreur, ou le bruit, qui se produit à intervalle de temps régulier. Pour le modèle ARMA, notons X_t la variable à l'instant t que l'on souhaite prédire :

$$X_t = \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

avec φ_i les paramètres du modèle AR d'ordre p, θ_i ceux du modèle MA d'ordre q et ε_i les termes d'erreurs. Notons que les valeurs p et q peuvent être différentes.

Le modèle ARMA a inspiré beaucoup d'autres modèles comme notamment ARIMA prenant en compte la non-stationnarité des données, ou encore STARIMA, qui utilise la saisonnalité des données pour améliorer la régression.

2.2.2.3 Méthodes à double usage : classification et régression

Les méthodes présentées dans cette section 2.2.2.3 peuvent à la fois être utilisées pour des problèmes de régression mais aussi pour des problèmes de classification. Nous présentons deux types de méthodes : les réseaux de neurones artificiels, et les Séparateurs à Vastes Marges.

Nous verrons également dans la section 2.2.3.4 que ces méthodes peuvent s'utiliser dans un contexte non supervisé.

2.2.2.3.a Réseaux de neurones artificiels

Ces modèles s'inspirent quelque peu du modèle biologique. Chaque neurone (ou perceptron) est connecté par une ou plusieurs synapses. Chaque synapse fournit une valeur à l'entrée du neurone. Le neurone calcule une valeur qu'il fournit en sortie. Cette sortie peut être ensuite redistribuée à un ou plusieurs neurones (par le biais des synapses), comme représenté sur la figure 2.9, ou simplement fournir le résultat attendu par l'utilisateur. Le neurone renvoie une valeur en sortie, selon les données en entrée et les poids attribués à chaque entrée, à l'instar d'une porte logique. Sa sortie peut être redirigée vers plusieurs neurones, y compris lui-même (apprentissage par récurrence, pour prendre en considération son dernier état pour la future prédiction), ou bien directement à l'utilisateur. Ainsi, il fournit une probabilité d'appartenir à une classe, ou une valeur de prédiction si le réseau est configuré pour effectuer de la régression. Durant la phase d'apprentissage, le réseau va corriger les poids du réseau en fonction de la réponse obtenue et de la réponse attendue.

Ces neurones sont généralement fortement connectés, et les poids entres les neurones sont initialisés aléatoirement. Lors de l'apprentissage, pour chaque donnée en entrée, les neurones fournissent un résultat en sortie (par exemple "normal" ou "anormal" encodé par 0 ou 1), et chacune des sorties prédites est corrigée ou renforcée selon la valeur attendue. Cette dernière action, appelée rétropropagation du gradient (*backpropagation* en anglais), va corriger les poids des synapses. Le sens de cette correction est visible sur la figure 2.9.

Pour la classification, il y a autant de neurones de sortie qu'il y a de classes. Il peut exister une ou plusieurs classes normales et une ou plusieurs classes anormales à apprendre. Chaque neurone de sortie, représentant une

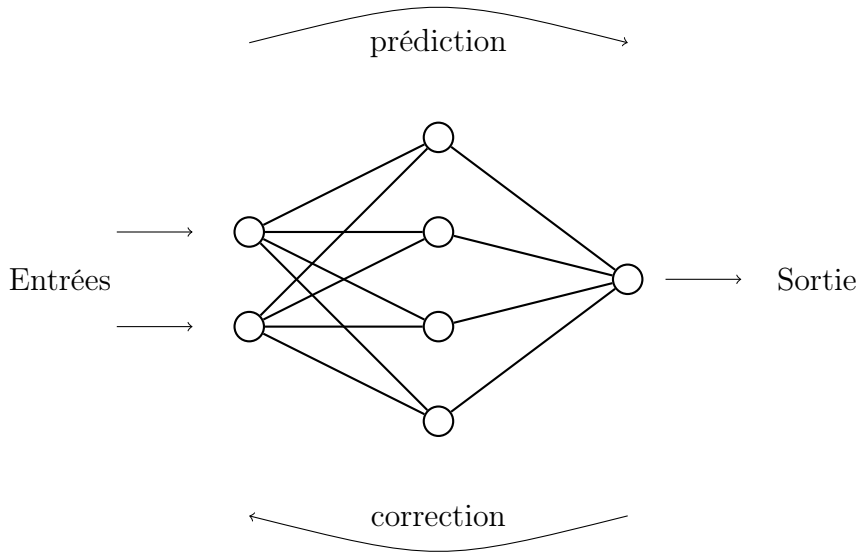


FIGURE 2.9 – Un exemple de réseau de neurones artificiels avec deux neurones sur la couche d'entrée, quatre sur la couche cachée, et un sur la couche de sortie. La flèche du haut indique le sens de la prédiction, et la flèche du bas le sens de la correction durant la phase d'apprentissage (*backpropagation*).

classe différente, donne une probabilité que la donnée en entrée appartienne ou non à la classe en question et corrige les poids. C'est le cas de la méthode de [Kim *et al.*, 2015] testée sur deux jeux médicaux (détection de tumeur et de diabète) et un jeu de simulation de modèle climatique. Pour faire de la régression, le principe est le même, comme expliqué par [Specht, 1991].

Pour la découverte d'inattendu, [Augusteijn et Folkert, 2002] proposent de définir un seuil en sortie. Si aucun neurone de sortie ne dépasse ce seuil, la donnée en entrée est considérée comme inattendue.

2.2.2.3.b SVM et SVR

Les Séparateurs à Vastes Marges (SVM) font partie de la famille des méthodes à noyaux et permettent de travailler sur des problèmes qui ne peuvent être résolus avec des approches linéaires. Une fonction noyau transforme les données non-linéairement séparables dans l'espace d'origine en données linéairement séparables [Cornuéjols *et al.*, 2018]. Les données sont tout d'abord plongées dans un espace de dimension différente, en général de plus haute dimension voire de dimension infinie. L'espace d'arrivée (espace de redescription) permet de trouver une séparation linéaire et d'utiliser alors des méthodes simples pour résoudre le problème en question (classification, ré-

gression). Pour la classification, les SVM sont très couramment employés. Dans le cas d'un problème de classification binaire, un SVM calcule l'hyperplan le plus optimal pour représenter au mieux la séparation entre les deux classes. Si les données ne sont pas linéairement séparables, les données sont projetées dans des espaces de plus haute dimension afin de trouver un ou plusieurs hyperplans séparateurs. Si seulement deux groupes sont séparés (normal et anormal) par l'hyperplan séparateur, la nouvelle observation est étiquetée selon s'il se trouve du côté normal ou du côté anormal de l'hyperplan.

Pour la détection d'attaques système, [Ambwani, 2003] utilise plusieurs SVM : un spécifique à chacun des types d'anomalies connus, et un supplémentaire pour les données normales. Pour le $i^{\text{ième}}$ type d'anomalie, le $i^{\text{ième}}$ SVM renverra 1 pour ce type d'anomalie, 0 pour tous les autres types. Une nouvelle donnée est testée sur chacun de ces classifieurs.

Dans le domaine de la détection, il n'est pas rare de traiter seulement la classe normale durant la phase d'apprentissage, car la classe des données anormales est souvent trop peu représentée. C'est le cas par exemple avec les SVM à une classe [Muñoz-Marí *et al.*, 2010]. Pour ces derniers, initialement proposés par [Schölkopf *et al.*, 2001], l'objectif n'est plus de séparer deux ou plusieurs classes d'objets, mais d'isoler l'unique classe de données normales. Si la nouvelle donnée à étiqueter ne se trouve pas du bon côté de l'hyperplan, elle sera annotée anormale.

Il est par ailleurs possible de faire de la régression avec les SVM ; appelés alors SVR (pour Régression basée sur les Séparateurs à Vastes marges), proposé par [Drucker *et al.*, 1997]. Dans ce cas d'usage, l'hyperplan devient la droite de régression, et doit, durant l'apprentissage, se rapprocher le plus possible des données d'apprentissage en minimisant la distance de l'hyperplan au jeu d'apprentissage. Par exemple, pour détecter l'usure des ponts, [Kromanis et Kripakaran, 2013] mesurent le comportement des variations de température à travers leurs structures. Ils utilisent un SVR pour mesurer l'erreur de prédiction entre la température attendue et la température obtenue.

Nous venons de voir une liste des principales méthodes supervisées et semi-supervisées utilisées dans la détection d'anomalies. À présent, nous allons aborder les méthodes non supervisées.

2.2.3 Famille de méthodes non supervisées

Les méthodes non supervisées sont très répandues dans le domaine de la détection d'anomalies : connaître à l'avance et avec précision ce qui est normal

et ce qui ne l'est pas, de façon à pouvoir les différencier durant l'apprentissage, est souvent bien difficile.

Comme expliqué dans la section 2.2.1, les méthodes d'apprentissage non supervisées traitent avec des données non labellisées. Il peut y avoir deux raisons à cela : soit la tâche de labellisation est trop coûteuse, notamment parce que le volume des données est trop grand, soit l'expertise sur les données est trop légère, ce qui arrive généralement quand les données récoltées sont récentes. Dans tous les cas, il est tout à fait possible d'effectuer un pré-traitement sur les données pour réduire le bruit, ou agréger les données brutes de façon à diminuer la volumétrie et étudier des indicateurs plus compréhensibles pour le superviseur et pour le modèle d'apprentissage. Ce dernier point constitue l'une des étapes de pré-traitement dans ce travail de thèse, comme nous le verrons dans la section 3.6.2.2.

Lors de l'utilisation de méthodes non supervisées pour la détection d'anomalies, il est nécessaire d'effectuer une hypothèse assez forte : une grande partie des données est considérée comme normale et un effectif réduit, voire nul, est anormal. En effet, ces méthodes se basent sur le comportement de groupe au sein des données. Elles extraient des indicateurs résumant leurs comportements, permettant ainsi de vérifier la normalité d'une d'entre elles parmi les autres.

Cette vérification, qui se résume à définir si une donnée est normale ou non, fonctionne pour une donnée fraîchement générée, mais également pour une donnée plus ancienne et incluse dans le jeu de données de départ.

À présent, nous allons voir quelques-unes de ces méthodes permettant d'acquérir des connaissances statistiques sur les données. Dans ce travail de thèse, les données dont nous disposons ne sont pas labellisées, c'est pourquoi nous avons choisi de détailler cette section avec le plus grand soin.

2.2.3.1 Étude des distributions des données

Dans cette section, nous présentons quelques outils utilisés non seulement pour effectuer un pré-traitement et une pré-analyse sur les données, mais aussi permettant de mettre en évidence des données aberrantes. Une des pratiques les plus courantes consiste à observer la distribution des données, visualisable sous différentes formes (une ou plusieurs dimensions).

2.2.3.1.a Boîte à moustache

Cette boîte résume de façon très synthétique la distribution des données sur une dimension en indiquant la médiane (ou la moyenne) et les quartiles.

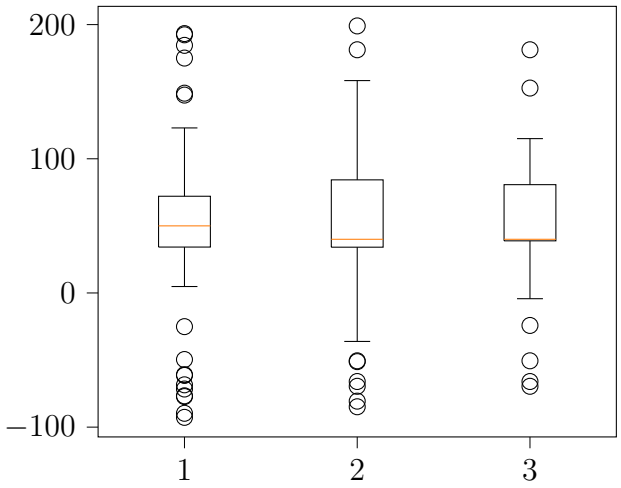


FIGURE 2.10 – Un exemple de boites à moustache. La médiane est représentée par un trait orange, et le premier et le troisième quartiles par les bords respectivement haut et bas du rectangle. Les extrémités des moustaches sont calculées selon les valeurs des quartiles. Les ronds représentent des données potentiellement anormales.

Les extrémités m_1 et m_2 des traits, appelées moustaches, sont utilisées pour représenter les extremums, les déciles ou, comme dans la figure 2.10, sont calculées selon les valeurs du premier et troisième quartile (Q_1 et Q_3), ce qui permet ainsi de mettre en évidence, les données plus grandes, respectivement plus petites, que la valeur maximum, resp. minimum [Le Guen, 2002] : $m_1 = Q_3 + 1.5 \times (Q_3 - Q_1)$ et $m_2 = Q_3 - 1.5 \times (Q_3 - Q_1)$.

Il est possible de comparer rapidement plusieurs boîtes résumant chacune un jeu de données, permettant ainsi d'interpréter les différences entre ces jeux. Pour une même boîte, une donnée possédant une valeur très petite ou très grande par rapport au reste des données est facilement mise en lumière, et vue comme anormale.

2.2.3.1.b Nuage de points

Le nuage de points utilise les valeurs des données comme coordonnées cartésiennes, en 2 dimensions. Il met en évidence les groupes de données les plus distincts, et peut fait apparaître les degrés de corrélation entre les variables. Généralement en 2 dimensions, nous trouvons certaines représentations avec par exemple la couleur des points comme 3^e dimension.

2.2.3.1.c Histogramme

Un histogramme est une représentation en bâton de la distribution des données. Chaque bâton représente en abscisse une valeur ou un intervalle de valeur, et en ordonnée l'effectif de la valeur ou de l'intervalle. Un exemple est illustré sur la figure 2.11. Les bâtons oranges dans l'extrémité droite du graphique sur la figure 2.11 représente probablement des données anormales.

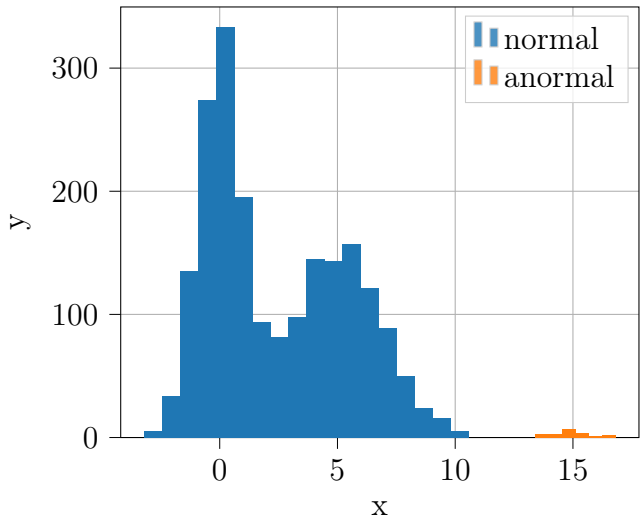


FIGURE 2.11 – Un exemple d'histogramme. Les données anormales sont ainsi mises en évidence par la distance qui les séparent des données normales.

2.2.3.1.d Kurtosis

Le kurtosis donne la probabilité de répartition de l'effectif, selon l'espérance des données normalisées par la valeur de dispersion (c'est-à-dire l'écart-type). Prenons X une variable aléatoire d'espérance μ et d'écart-type σ. Le moment d'ordre n de la variable aléatoire X :

$$\mu_n = \mathrm{E}[(X - \mathrm{E}[X])^n]$$

Sachant que le moment d'ordre 1 de X représente l'espérance de X, le moment d'ordre 2 sa variance, le kurtosis de X est le moment d'ordre 4 standardisé par l'écart-type :

$$\mathrm{Kurt}[X] = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mu_4}{\sigma^4}$$

Il est possible de voir les fonctions de densité de probabilité des données selon trois kurtosis différents, centrés à l'origine, sur la figure 2.12. Une donnée qui se retrouverait sur la courbe avec une probabilité d'apparition faible (en ordonnée sur la figure 2.12) pourrait être annotée comme anormale.

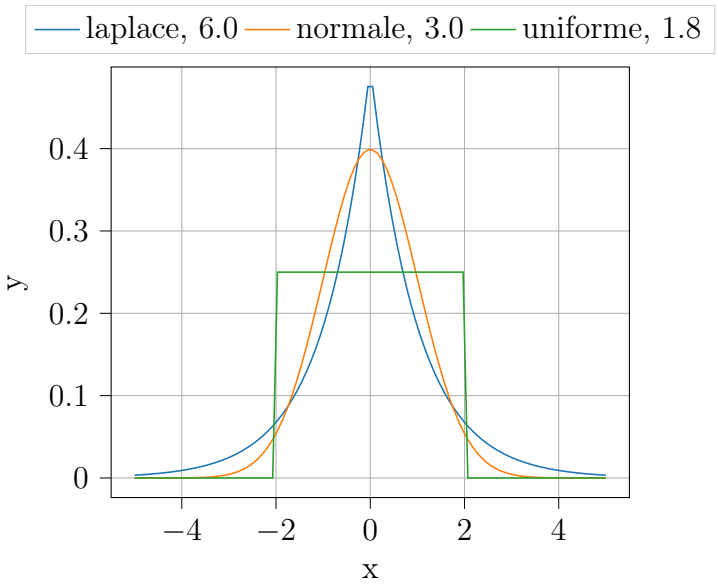


FIGURE 2.12 – Probabilité de distribution pour 3 fonctions différentes de moyenne 0 et d'écart-type 1 : Laplace, Normale et uniforme.

Ces outils (ou méthodes d'analyse) ne sont par définition pas considérés comme étant des méthodes d'apprentissage. Ils permettent cependant d'analyser de manière assez globale les données, et de les observer sous un angle de vision différent. Ils sont parfois suffisants dans certaines tâches de détection, en particulier sur des données ayant une régularité de distribution spécifique et de faible dimension comme expliqué par [Aggarwal, 2017].

Nous constatons ici qu'il est possible avec des méthodes simples et sans annotation de détecter des anomalies. Nous allons voir maintenant les possibilités de détection que nous offrent les méthodes d'apprentissage non supervisées.

2.2.3.2 Famille d'approches basées sur les groupes

Nous nous intéresserons dans un premier temps aux méthodes d'apprentissage non supervisées basées sur le regroupement des données, appelé *clustering* en anglais. L'objectif de ces méthodes est de regrouper les données

de manière à maximiser la compacité de chaque groupe et la séparation des groupes entre eux.

Sur des données non étiquetées, la création de groupe s'effectue selon les similitudes trouvées entre les données. C'est pourquoi le choix de la méthode et celui de la fonction de similarité seront influencés par la problématique de détection. Il existe différentes méthodes permettant de construire ces groupes, et nous aborderons dans cette section les plus plébiscitées dans la littérature.

De façon générale, les méthodes basées sur les groupes construisent des rassemblements parmi les données fournies et permettent à la détection de se faire sur l'ensemble du jeu de données. Ainsi, les données qui n'appartiennent pas ou qui sont trop éloignées des groupes formés sont annotées comme anormales.

Comme abordé dans la section 2.1.3, il existe deux types d'annotations. Nous pouvons annoter avec une valeur binaire : 1 pour anormal lorsque la donnée n'appartient à aucun groupe ou à un groupe d'anomalies, et 0 pour normal si la donnée appartient à un groupe de données normales. Mais il est possible aussi de lui attribuer un score en se référant par exemple à sa distance au groupe le plus proche.

À présent, regardons de plus près comment fonctionnent ces méthodes pour comprendre quels sont leurs critères de regroupement entre les données.

2.2.3.2.a Méthodes basées sur les représentants

Ces méthodes s'appliquent généralement pour des données numériques. Chaque classe possède un représentant, minimisant la distance entre ce représentant et les objets de cette même classe.

Il existe différents types de représentants : le barycentre pour l'algorithme des k-moyennes, le médoïde pour l'algorithme des k-médoïdes, ou encore la médiane pour l'algorithme k-médiane. Pour ces trois algorithmes, l'objectif est de minimiser la distance entre les objets d'un groupe et leur représentant. Pour cela, les groupes se construisent de manière itérative. L'ensemble des représentants vont être amenés à changer au cours du temps durant le processus de construction des groupes, car ils sont généralement choisis au hasard durant la phase d'initialisation. À chaque itération, après avoir distribué les objets dans chaque groupe, l'algorithme cherche, selon ses critères intrinsèques, le meilleur représentant de chaque groupe. Par exemple, si un nouvel objet se retire ou se rajoute dans un groupe, il semble normal que le barycentre soit modifié. L'algorithme k-moyennes itère jusqu'à qu'il y ait convergence, c'est-à-dire dès que la variance dans chaque groupe soit minimisée.

Le score d'anomalie d'un objet se calcule grâce à sa distance au représentant. Dans [Budalakoti *et al.*, 2009], un objet est défini comme anormal si il se trouve trop loin du médoïde le plus proche.

2.2.3.2.b Méthodes hiérarchiques

La hiérarchie dans les données permet de parcourir progressivement, et sur différentes échelles, les groupes mis en évidence par ces méthodes.

Le regroupement peut se faire de manière ascendante, c'est-à-dire en partant d'un seul groupe et en scindant étape par étape les données dans plusieurs sous-groupes. Il peut également se faire de manière descendante, quand toutes les données appartiennent à leur propre groupe d'effectif égale à 1 : ces groupes fusionnent alors les uns après les autres. Dans la première approche, le critère de division cherche à optimiser la séparation entre les objets, par exemple en minimisant la somme des distances deux-à-deux dans les nouveaux sous-groupes. Pour la seconde approche, la fusion se fait sur les objets les plus similaires. Dans un sens comme dans l'autre, l'analyse des discriminants est très intéressante pour acquérir ou valider des connaissances sur les données traitées.

Pour effectuer l'étude de trajectoire d'objets mobiles, [Piciarelli et Foresti, 2006] proposent un regroupement hiérarchique où chaque branche de l'arbre représente une suite de mouvements formant une trajectoire. Chaque nœud de l'arbre possède une probabilité d'apparition selon l'historique des trajectoires déjà vues. Une anomalie sera une trajectoire empruntant un nœud rarement parcouru (probabilité d'apparition faible) ou un nouveau nœud (trajectoire jamais empruntée). Une anomalie ne sera pas forcément une trajectoire dangereuse et pourra, le cas échéant, mettre l'arbre à jour.

2.2.3.2.c Modèle de mélange gaussien

Le modèle de mélange gaussien cherche à exprimer la distribution des distances à l'aide de plusieurs fonctions gaussiennes, où chacune d'entre elles seraient représentatives d'un groupe de données.

Il utilise généralement l'algorithme espérance-maximisation de manière itérative afin d'obtenir les paramètres de ces fonctions avec deux étapes de calcul : une étape d'évaluation et une étape de maximisation. Pour l'évaluation, l'espérance de la vraisemblance est recalculée en incluant les modifications de l'itération précédente, la maximisation est quant à elle calculée sur la fonction de vraisemblance. En résumé, l'objectif est d'inclure, à chaque itération, les objets dans la fonction gaussienne la plus pertinente, et de mettre à jour ces fonctions pour mieux inclure ces objets. L'algorithme peut converger

seul, mais en général un nombre d'itération maximum est défini.

Une fois les fonctions gaussiennes calculées, il suffit d'attribuer à chaque donnée, une appartenance à une fonction gaussienne, c'est-à-dire à un groupe. Si une donnée se trouve trop loin des fonctions générées (c'est-à-dire avec une densité de probabilité trop faible), elle sera définie comme anormale.

La méthode SmartSifter proposée par [Yamanishi *et al.*, 2004] utilise un mélange de gaussiennes pour caractériser la densité de probabilité des variables continues. Cette méthode a été testée sur des traces de commande réseaux pour détecter des intrusions.

Nous venons de voir quelques-unes des méthodes permettant de construire, à partir d'un apprentissage non supervisé, des groupes de données. Il en existent beaucoup d'autres, dont quelques-unes sont très spécifiques et très utilisées dans certains domaines, comme les méthodes de *spectral clustering* en traitement d'image [Ng *et al.*, 2002], ou d'autres approches plus récentes comme le *clustering by synchronisation* [Böhm *et al.*, 2010].

Dans la prochaine section, nous allons présenter quelques méthodes qui se basent sur la notion de voisinage entre les objets pour proposer un score d'anomalie.

2.2.3.3 Famille de méthodes basées sur le voisinage

Cette section présente quelques-unes des grandes méthodes d'apprentissage basées sur le voisinage. Ces méthodes non supervisées sont largement utilisées par la communauté et présentent de nombreux travaux aux applications variées.

Le principe général est de tenir compte du voisinage de l'objet en cours d'analyse pour déterminer si ce dernier est normal ou non. Plus l'objet ressemble aux autres objets proches de lui, et plus ses chances d'être aberrant diminuent. Il existe de nombreuses façons de mesurer la similarité entre objets. Pour des objets possédants des valeurs numériques, la distance euclidienne est couramment utilisée pour les comparer. Mais il est possible d'utiliser d'autres types de distances comme présentés par [Deza et Deza, 2009] et des indicateurs différents, tels que la distance au k plus proche voisin ou la densité autour de ce voisin. La famille des méthodes basées sur le voisinage comprend deux sous-familles : les méthodes basées distance et celles basées densité.

Ces méthodes sont utilisées depuis de nombreuses années, et sont encore aujourd'hui très populaires. Le calcul de la distance et de la densité entre objets peut s'effectuer de différentes façons, et la notion de normalité peut varier d'une méthode à l'autre. Certains comportements (en particulier le

fléau de la dimensionnalité, présenté dans la section 2.1.5.4) restent encore très étudiés et donnent lieu à de nouvelles analyses [Beyer *et al.*, 1999, Francois *et al.*, 2007, Radovanović *et al.*, 2009, Angiulli, 2018, Chen et Shah, 2018]. Les méthodes basées sur le voisinage sont utilisées dans de nombreux domaines d'application (prédiction, compression, détection, ...), c'est pourquoi le type de méthode et la mesure de distance utilisés dépendent fortement des données à analyser et de la problématique à résoudre.

Nous allons présenter quelques-unes des méthodes les plus réputées pour chacune des sous-familles (basées distance puis basées densité).

2.2.3.3.a Méthodes basées sur la distance

Pour ces méthodes, le score d'un objet est défini comme étant la distance à ses plus proches voisins. La définition la plus simple consiste à regarder la distance au k^{e} plus proche voisin, que nous noterons NN_k (pour k *Nearest Neighbor*). Plus la distance sera élevée, et plus l'objet sera considéré comme anormal.

Nous allons dans ce paragraphe rentrer un peu plus en détail dans le formalisme de ces méthodes. En effet, ces définitions nous seront utiles pour la suite, notamment pour la section 3 dans laquelle nous décrirons notre première contribution.

Les k voisins les plus proches. Notons $\text{dist}(x, y)$ la distance entre l'objet x et l'objet y, et $nn_k(x)$ le k^{e} plus proche voisin de x. Le score de x calculé à partir de la distance au k^{e} plus proche voisin de x se note : $\text{dist}(x, nn_k(x))$. À partir des distances entre les objets, [Ramaswamy *et al.*, 2000] définissent un objet x comme anormal s'il existe un trop faible nombre s d'autres objets y avec une distance au k^{e} voisin plus grande que la distance entre x et son k^{e} voisin :

$$\text{Si } |\{y : \text{dist}(y, nn_k(y) \geq \text{dist}(x, nn_k(x)\}| < s \text{ alors } x \text{ est anormal}$$

Ils déclarent donc que les s objets ayant les plus grandes valeurs vers le k^{e} voisins seront annotés comme anormaux.

Si $k = 1$, la granularité est faible, car la comparaison ne se fait qu'avec l'objet le plus proche qui peut lui même être anormal. La prise en compte d'un voisin plus lointain en augmentant k permet d'évaluer notre objet avec une granularité plus fine. Mais une valeur de k trop grande mène à une trop forte généralisation de l'estimation car la considération sera trop large.

Pour tenir compte de plus d'objets durant le calcul et considérer les distances vers les plus proches voisins afin d'obtenir une évaluation plus fine, il est possible de regarder la somme des distances en prenant en compte les voisins au moins aussi près que le k^{e} plus proche voisin. L'ensemble $\text{NN}_k(x)$ des k plus proches voisins de x inclut les k objets au moins aussi proche de x que l'objet $nn_k(x)$ parmi l'ensemble de taille n, c'est-à-dire :

$$\text{NN}_k(x) = \{nn_i(x) : \text{dist}(x, nn_i(x)) \leq \text{dist}(x, nn_k(x)) \text{ pour } 0 < i \leq n\}$$

Cette définition est notamment utilisée par [Angiulli et Pizzuti, 2002]. Dans le cas classique de calcul des kNN, il est nécessaire de connaître la distance entre tous les objets, ce qui nous amène à une complexité polynomiale. C'est pourquoi [Angiulli et Pizzuti, 2002] proposent de linéariser l'espace de recherche en utilisant une courbe de Hilbert (*Hilbert space filling curve* en anglais). Cela permet à moindre coût de sélectionner des objets candidats susceptibles d'être anormaux.

Il est possible de construire des groupements à partir du calcul des plus proches voisins pour accélérer le calcul, comme par exemple [Eskin *et al.*, 2002] qui comptabilisent le nombre de voisins de l'objet x inclus dans le rayon w d'un objet :

$$\text{NC}_w(x) = |\{y : d(x, y) \leq w\}|$$

Pour réduire la complexité de calcul, ils approximent le calcul du nombre de voisins inclus dans le rayon en utilisant un représentant par groupement. Les distances sont calculées seulement entre tous les objets vers les représentants des groupements, qui devraient être moins nombreux si la taille w est pertinente.

HotSAX. L'algorithme HotSAX de [Keogh *et al.*, 2005] retourne la séquence qui possède la plus grande distance vers sa plus proche voisine. Mais contrairement à la méthode classique, qui consiste à calculer les distances entre toutes les séquences avant d'estimer les plus proches voisins de chaque séquence puis de définir la séquence la plus aberrante, l'algorithme HotSAX a pour objectif de diminuer le nombre de calcul des distances entre les séquences tout en garantissant d'obtenir le même résultat.

À l'aide de la discrétisation SAX plus amplement détaillée dans [Lin *et al.*, 2007b], [Keogh *et al.*, 2005] proposent une heuristique permettant d'évaluer toutes les séquences extraites d'un flux de données numériques grâce à une fenêtre glissante et retourner les pairs de séquences les plus distantes. Notons que HotSAX ne compare que les séquences qui ne se chevauchent pas temporellement.

Pour permettre de réduire le nombre de calcul, l'heuristique tire parti de la discrétisation SAX des séquences pour évaluer rapidement celles qui sembleraient être de bonnes candidates pour être anormales, de façon à maximiser les scores d'anomalies obtenus sur les premières séquences évaluées, permettant ainsi de se défaire plus rapidement des autres séquences et limiter le nombre de comparaisons deux-à-deux. SHIEH et KEOGH utilisent une structure hiérarchique pour comptabiliser les séquences ayant la même discrétisation, et calculent le score des séquences isolées dans l'arbre en les comparant aux séquences ayant la même signature SAX. Cette heuristique comprend donc deux parcours sur la totalité des séquences : une boucle externe et une boucle interne.

La première boucle tente de présenter en premier les séquences obtenant les scores les plus aberrants. La seconde essaie alors de trouver rapidement la séquence la plus proche de celle évaluée par la boucle externe. La seconde est interrompue si la distance obtenue entre les deux séquences est considérée comme normale par rapport aux distances des autres séquences déjà évaluées.

Dans le pire des cas, toutes les séquences sont testées deux-à-deux. Le coût de calcul sera alors identique de celui la méthode classique.

Si la boucle externe évalue prioritairement les séquences les plus anormales, et que la boucle interne favorise la comparaison avec des séquences proches, la boucle interne sera, au fil des séquences évaluées par la boucle externe, de moins en moins longue. Dans le meilleur des cas, le coût tend à devenir linéaire par rapport au nombre de séquences présentes dans la boucle externe, car le coût de parcours de la boucle interne devient négligeable.

Les k voisins inverses les plus proches. Une autre notion de voisinage utilisée à partir des distances mesurées entre les objets est celle des plus proches voisins inversés. Notons $\mathrm{N}_k(x)$ le nombre de k plus proches voisins inverses de l'objet x :

$$\mathrm{N}_k(x) = |\{y : x \in \mathrm{NN}_k(y)\}|$$

c'est-à-dire le nombre d'objets y qui incluent x parmi leurs k plus proches voisins. Cette mesure a été proposée par [Korn et Muthukrishnan, 2000], et permet de regarder l'influence de l'objet concerné vis-à-vis des autres objets, plutôt que seulement la distance vers les objets voisins. De nombreux travaux utilisent cette mesure $\mathrm{N}_k(x)$ notamment l'algorithme ODIN (*Outlier Detection Using Indegree Number*) proposé par [Hautamaki *et al.*, 2004] qui présentent leur méthode comme la construction d'un graphe dont les arêtes représentent les voisinages les plus proches, où chaque objet est un nœud. Le

score de chaque objet est le degré de son nœud. Plus la mesure $N_k(x)$ d'un objet x est faible, et plus cet objet est susceptible d'être anormal.

La plus longue sous-séquence commune. La similarité entre objets se mesure de différentes façons : la plus longue sous-séquence commune (LSC) est utilisée comme score de similarité, pour la comparaison entre séquences. Le résultat de similarité entre deux séquences est un entier, représentant la longueur de la sous-séquence commune la plus grande. Une séquence est évaluée comme étant anormale si une ou l'ensemble de ses LSC obtenues sont petites.

Cette algorithme est utilisé par [Budalakoti *et al.*, 2009] pour l'analyse de données aérospatiales, en normalisant le résultat par le produit des longueurs des deux séquences comparées. La complexité de recherche de la plus longue sous-séquence commune est un problème NP-difficile, la complexité de calcul d'un algorithme standard pour ce type de problème sera alors exponentielle. Pour remédier à ce problème, [Bergroth *et al.*, 2000] proposent des méthodes d'estimation de plus longue sous-séquence mais avec des complexités de calcul plus faibles.

Nous venons de voir quelques-unes des principales méthodes basées sur la distance entre les objets. Pour l'ensemble de ces méthodes, il existe de nombreuses autres distances suivant le type de données manipulées et le résultat souhaité.

Par exemple, la distance de Manhattan représente la distance entre deux objets numériques comme étant la somme des différences absolues sur chacune des dimensions. Lorsque la distance euclidienne aura tendance à accentuer les écarts entre les valeurs, la distance de Manhattan (aussi appelé norme-l_1) sera plus modérée sur ces mêmes écarts. C'est notamment pour cette raison que cette distance est parfois utilisée sur des données de haute dimension, dans le but de minimiser les effets décrits Section 2.1.5.4. Comme présenté par [Aggarwal *et al.*, 2001], le contraste relatif converge moins vite avec la norme l_1 qu'avec la norme l_2 (c'est-à-dire la distance euclidienne). Il est possible de retrouver un grand nombre d'autres distances décrites dans [Deza et Deza, 2009].

2.2.3.3.b Méthodes basées sur la densité

La densité autour d'un objet représente le nombre d'autres objets dans une région spécifique et locale par rapport à l'objet en cours d'évaluation. Il existe différentes façons de la mesurer et de l'utiliser pour évaluer celui-ci.

La méthode la plus simple est très analogue à celle des NN_k. Elle fonctionne en deux temps : elle mesure la distance $\mathrm{dist}(x, nn_k(x))$ au k^{e} plus proche voisin de l'objet x à évaluer, puis elle calcule la densité de l'hypersphère, de centre x et de rayon $\mathrm{dist}(x, nn_k(x))$, contenant k objets. Plus la densité est faible, et plus l'objet est considéré comme anormal.

Ces méthodes permettent de mettre en lumière la sensibilité locale des objets lors de la détection. Nous pouvons voir sur la figure 2.13 que l'anomalie A ne sera pas détectée avec une méthode basée distance, tandis qu'elle peut aisément l'être avec une méthode basée densité. Détaillons à présent les principales approches de ce type de méthode.

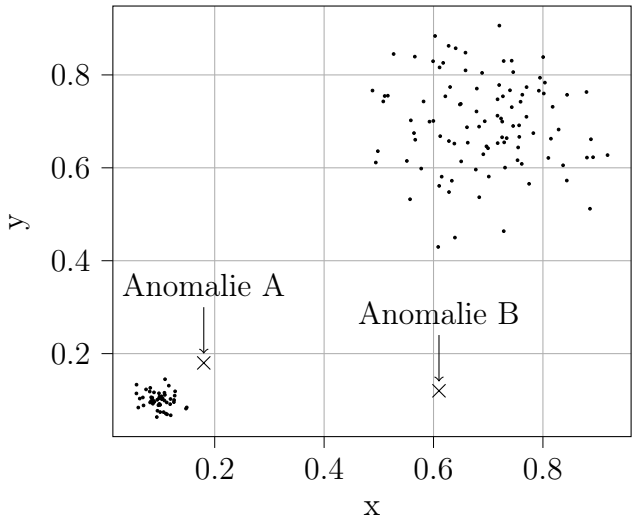


FIGURE 2.13 – Impact de la densité locale présenté par [Aggarwal, 2017].

Local Outlier Factor. La méthode basée sur la densité la plus populaire est très certainement la méthode LOF (pour *Local Outlier Factor*). Proposé par [Breunig *et al.*, 2000], elle consiste à regarder la densité locale de l'objet à évaluer et de la comparer à celle de ses plus proches voisins. Le ratio retourné dépend alors des propriétés de voisinage de ses plus proches voisins.

La distance d'accessibilité de x vers y est définie comme la valeur maximum entre la distance entre x et y et la distance au k^{e} plus proche voisin de y :

$$\mathrm{dist\_acc}_k(x, y) = \max\{\mathrm{dist}(x, y), \mathrm{dist}(y, nn_k(y))\}$$

Si l'objet x est plus loin de y que ce dernier l'est de son k^{e} plus proche voisin de y, ce sera la distance entre les deux objets qui sera employée. Dans le cas contraire ce sera la distance au k^{e} plus proche voisin de y. La distance d'accessibilité n'est alors pas symétrique, ce qui signifie que $\text{dist\_acc}_k(x, y)$ n'est pas systématiquement égale à $\text{dist\_acc}_k(y, x)$.

Cette mesure de distance est ensuite utilisée dans le calcul de la densité d'accessibilité locale lrd, représentant l'inverse de la moyenne des distances d'accessibilité de x vers ses k plus proches voisins :

$$\text{lrd}_k(x) = 1/\left[\frac{\sum_{o \in \text{NN}_k(x)} \text{dist\_acc}_k(x, o)}{k}\right]$$

Le score LOF final pour un objet x est la moyenne des ratios, dont ces derniers sont calculés comme étant les ratios entre la densité d'accessibilité local de x divisé par ceux de ces k voisins :

$$\text{LOF}_k(x) = \frac{\sum_{o \in \text{NN}_k(x)} \frac{\text{lrd}_k(o)}{\text{lrd}_k(x)}}{k}$$

Si l'objet x obtient un score proche de 1, alors sa densité est similaire à celle de ses voisins. Dans le cas où il obtient un score plus petit que 1, l'objet x a une densité plus forte, et il convient de le considérer comme normal. Mais si à l'inverse, son score est supérieur à 1, sa densité est plus faible que celle de ses voisins : plus son score est grand, et plus il sera considéré comme anormal.

Le point fort de cette méthode est de pouvoir s'adapter aux différentes densités au sein d'un même jeu de données. Comme présenté sur la figure 2.13, les deux groupes d'objets possèdent des densités différentes. Contrairement aux méthodes se basant seulement sur les distances entre les objets, la méthode LOF ne sera pas impactée par cette différence de densité et pourra aisément détecter l'objet A comme anormal car sa densité locale sera bien différente de celles des objets de son voisinage.

Grâce à son efficacité d'adaptation aux différentes concentrations de groupe, cette méthode a été largement reprise. Pour une application sur des flux de données avec une mise à jour incrémentale des densités locales, [Salehi *et al.*, 2016] proposent MiLOF et se comparent aussi à une version incrémentale plus ancienne proposée par [Pokrajac *et al.*, 2007], appelée iLOF.

La version iLOF permet de calculer le score LOF à l'arrivée de chaque nouvel objet, sans avoir à recalculer toutes les distances vers tous les objets. Pour un nouvel objet x, iLOF récupère les k plus proches voisins de x, et calcule le score LOF de x en mettant préalablement à jour les valeurs nécessaires

au calcul de LOF pour tous les voisins inverses de x. Cette approche est très efficace, particulièrement lorsque le nombre d'objets à mettre à jour reste faible, mais reste assez coûteux en mémoire dans tous les cas. L'algorithme MiLOF (*Memory efficient incremental Local Outlier Factor*) propose de limiter la taille mémoire exploitée en sélectionnant un certain nombre d'objets fixé à l'avance. Ces derniers sont désignés comme représentant de groupe. Les valeurs nécessaires au calcul de LOF des représentants sont stockées, permettant d'approximer le score LOF d'un nouvel objet.

Isolation Forest. L'approche nommée *Isolation Forest* [Liu *et al.*, 2008] est récente. Elle se base sur un ensemble d'arbres et sur des critères de divisions aléatoires (*random forest*) pour définir si un objet est anormal. Si l'objet est difficile à séparer des autres objets dans de nombreux arbres, alors l'objet est considéré comme normal. Au contraire, si l'objet est facilement séparable, c'est-à-dire avec peu de critères de division, dans de nombreux arbres, alors il est considéré comme anormal.

Il existe plusieurs variantes qui permettent notamment d'améliorer la distribution des scores, en ne se basant plus sur une segmentation des données unidimensionnelle (type arbre binaire), mais en utilisant des hyperplans comme présenté par [Hariri *et al.*, 2019].

DBSCAN et OPTICS. L'algorithme DBSCAN (pour "*Density-Based Spatial Clustering of Applications with Noise*") a été proposé en 1996 par [Ester *et al.*, 1996]. Il repose sur la densité locale des objets pour définir si un objet appartient à un groupe ou non.

Cet algorithme utilise deux paramètres : la distance maximum ϵ pour laquelle les objets sont considérés comme voisins proches entre eux, et minPts, le nombre d'objets minimum pour former un groupe. Si un objet contient au moins minPts dans un rayon ϵ, appelé aussi le ϵ-voisinage de l'objet, alors il devient un groupe, avec les objets qui l'entourent. Sinon, les objets n'ayant pas assez de points dans leur voisinage seront considérés comme étant du bruit. Pour cet algorithme, il n'est pas nécessaire de connaître à l'avance le nombre de groupes recherchés. Cependant, DBSCAN, étant très dépendant de ces deux paramètres, n'est donc pas capable de détecter des groupes de densités différentes, pour des valeurs de paramètre égales.

L'algorithme OPTICS améliore ce dernier point en tenant compte, pour chaque objet, de la distance au minPts$^{\text{ème}}$ objet le plus proche, appelée la *core-distance*. À partir de cette distance, il est possible de calculer la distance d'accessibilité d'un objet p par rapport à un autre objet (appelé *reachability-*

distance de p) défini comme la valeur maximum entre la *core-distance* de l'objet p et la distance d'accessibilité à un autre objet. Pour tout objet, la distance d'accessibilité est calculée par rapport aux objets contenus dans son ϵ-voisinage, et l'algorithme OPTICS retourne la plus petite distance d'accessibilité pour chacun des objets. Tout comme DBSCAN, cela permet de construire des groupes d'objets. Mais contrairement à lui, l'algorithme OPTICS peut adapter la densité des groupes jusqu'à la limite définie indirectement par les paramètres ϵ et minPts. Ainsi, un objet ayant un nombre d'ϵ-voisins plus petit que minPts verra sa distance d'accessibilité indéfinie et se catégorise comme anormal.

Il est également possible d'utiliser la même normalisation que LOF en comparant les densités d'accessibilité de chaque objet par rapport à ceux de son voisinage pour obtenir un score d'anomalie interprétable, comme proposé par [Breunig *et al.*, 1999].

Concentration Free Outlier Factor Une récente approche présentée par [Angiulli, 2017] permet d'évaluer la normalité d'un objet x en mesurant le nombre de voisins inverses nécessaires pour que x soit inclus dans au moins ϱ pour cent de voisinage, ϱ étant un paramètre défini par l'utilisateur. Le score CFOF est normalisé par n, la taille du jeu de données :

$$\mathrm{CFOF}(x) = \min_{1 \leq k \leq n} \left(\frac{k}{n} : \mathrm{N}_k(x) \geq n\varrho \right)$$

La démarche la plus simple pour cette méthode est de connaître le voisinage de tous les objets vers tous les autres objets. Ensuite, il suffit de faire évoluer k de 1 jusqu'à la taille du jeu de données pour pouvoir évaluer tous les objets d'un même jeu de données. L'algorithme d'évaluation CFOF peut s'arrêter lorsque tous les objets à évaluer sont inclus dans au moins ϱ pour cent des voisinages du jeu de données. Tout comme LOF, CFOF est capable de s'adapter à la concentration des différents groupes de données. En effet, [Angiulli, 2020] montre que, pour deux groupes de distributions similaires mais paramétrées différemment, le score de détection obtenu est le même dans chacun des groupes.

La méthode nécessite de connaître la distance entre tous les objets : il s'agit d'une complexité polynomiale. Pour réduire cela, Anguilli propose un algorithme appelé *fast*-CFOF avec lequel il est possible de calculer une estimation du score CFOF en segmentant le jeu de départ en plusieurs échantillons aléatoires. Cela permet de surpasser la complexité de calcul du score CFOF, tout en assurant une précision minimum au calcul des scores. Cette méthode est utile lorsqu'il est nécessaire de calculer les scores de tous les

objets dans un temps réduit.

Comparés aux méthodes basées distance (NN_k, ...) et celles basées densité (LOF, ...), les méthodes CFOF et *fast*-CFOF — qui hérite des points forts de CFOF — se démarquent par leur capacité à évaluer des données en très haute dimension, et cela a été prouvé formellement par Anguilli dans ses différents travaux [Angiulli, 2017, Angiulli, 2018, Angiulli, 2020].

En effet, il prouve notamment que les scores obtenus ne se concentrent pas dans l'espace des distances euclidiennes lorsque la dimension des objets tend vers l'infini. Ils résistent donc au phénomène de concentration et à l'effet *hubness* expliqués dans la section 2.1.5.4.

Anguilli propose également d'analyser, pour quelques-unes des méthodes basées proximité, le ratio entre la moyenne et l'écart-type des scores d'anomalies. Il montre alors que ce ratio tend à devenir de plus en plus petit lorsque la dimension augmente, pour un jeu de données avec une distribution uniforme ou une distribution suivant une loi normale [Angiulli, 2020]. La concentration des distances a un effet néfaste sur les méthodes basées sur les distances entre les objets, ce qui n'est pas le cas pour sa méthode.

Dans le travail de thèse, c'est cette méthode qui sera employée pour la détection d'anomalies. C'est pourquoi elle sera plus amplement détaillée dans la section 3.2.

2.2.3.4 Autres méthodes

2.2.3.4.a One-Class Support Vector Machine

Le classifieur OC-SVM (pour "*One Class Support Vector Machine*") est un type particulier de SVM, abordés Section 2.2.2.3.b. Comme expliqué pour SVM et SVR, les méthodes à noyaux sont généralement utilisées pour des problèmes qui ne sont pas linéairement séparables.

Les OC-SVM sont utilisés de manière non supervisée pour la détection d'anomalies dans de nombreux domaines d'applications. Ils peuvent être vus comme un SVM à deux classes dont l'objectif est de séparer grâce à l'hyperplan les données normales des données anormales.

De la même manière que les SVM multi-classes, l'utilisation d'une fonction noyau permet de projeter les données dans un espace de représentation en plus haute dimension permettant ainsi, grâce à une borne discriminante, de séparer les données.

Dans les travaux de [Schölkopf *et al.*, 2000], ils incluent dix classes nor-

males dans une seule grande classe normale grâce au classifieur OC-SVM. Il convient de préciser que chacune de ces classes représente l'écriture manuscrite d'un des dix chiffres : ces représentations matricielles en 2 dimensions sont très différentes d'une classe à l'autre. Toutes les écritures manuscrites valides, où il est possible de lire un des dix chiffres, se retrouvent du même côté de l'hyperplan, qui ne peut être tracé que dans un espace de dimension supérieure à 2 pour pouvoir inclure tous les chiffres. Toutes les écritures manuscrites qui ne représentent pas un chiffre (ou qui le représentent mal), seront de l'autre côté de l'hyperplan et donc anormales.

Nous avons choisi de classer cette méthode parmi les non supervisées, ni basée groupe, ni basée voisinage. Néanmoins certains l'identifient malgré tout comme une méthode basée groupe [Gupta *et al.*, 2014] ou de régression [Aggarwal, 2017].

2.2.3.4.b Réseaux de neurone artificiel

Déjà présent dans la section 2.2.2.3.a des méthodes supervisées, les réseaux de neurones artificiels s'utilisent aussi de façon non supervisée.

Cartes auto-organisatrices. Les cartes auto-organisatrices (ou SOM pour *self organizing maps* en anglais) font partie des méthodes basées sur des réseaux de neurones artificiels. Elles ont été proposées pour la première fois par Kohonen en 1982 [Kohonen, 1982].

Ce type de réseau de neurones se différencie des autres types de réseau de neurones par sa structure et son type d'apprentissage (pas de rétropropagation). Les SOM sont structurées comme un maillage. Les neurones possèdent chacun un vecteur référent, qui de manière itérative est associé à une zone de l'espace de données et qui propose une représentation discrète de ce dernier. À chaque itération, un neurone dit "gagnant" modifie son vecteur référent dans l'espace pour minimiser la distance entre le vecteur et les données que le neurone représente.

Dans leurs travaux, [Hoglund *et al.*, 2000] définissent le *Best Matching Unit* (BMU) d'un objet comme étant le neurone le plus proche. Dans l'ensemble du jeu de données, ils comptent le nombre d'objets ayant un BMU plus grand que l'objet à évaluer. Si ce nombre est plus faible qu'un certain seuil, alors l'objet est anormal.

La structure interne des neurones qui compose HTM varie quelque peu des modèles de réseau de neurones artificiels classiques. Un neurone renvoie 1 s'il est activé, sinon il retourne 0. Un neurone est stimulé par la propaga-

tion avant, c'est-à-dire par les neurones de la couche inférieure [3]. Cependant, les interactions entre les neurones d'une même couche sont plus spécifiques pour HTM : avec une influence moindre, un neurone peut aussi être stimulé par d'autres neurones d'une même couche, prioritairement proches. En effet, l'ajout de ce dernier stimulus permet de trouver des concordances d'activation entre les neurones d'une même couche pour renforcer la prédiction. Si un neurone n'a pas été activé par la propagation avant, mais qu'un grand nombre de neurones proches de lui le sont, alors ce neurone sera activé.

La structure globale d'HTM est hiérarchique, et le nombre de neurones décroît d'une couche à l'autre dans le sens de la propagation avant. Il est aussi envisageable de fusionner plusieurs HTM pour associer en sortie le traitement de différentes sources.

Ce modèle a été utilisé par [Ahmad *et al.*, 2017] pour de la détection d'anomalies. Dans un contexte d'analyse de signal non supervisée, le modèle prédit pour chaque nouveau point, un score d'anomalie. Le modèle se forme sur les données antérieures et il n'a pas besoin de les stocker. Pour évaluer la robustesse de leur modèle, [Ahmad *et al.*, 2017] utilisent le Numenta Anomaly Benchmark (NAB) [Lavin et Ahmad, 2015] contenant 58 séries temporelles réelles ou artificielles disponibles sur GitHub [4]. Le NAB est décrit plus en détail dans la section 3.6.2. En effet, comme ce benchmark propose des anomalies sur diverses séries temporelles et les résultats de détection de différentes méthodes, il nous a semblé pertinent d'évaluer notre méthode sur ces jeux et de comparer les résultats aux autres méthodes de détection.

Auto-encodeur. Ce réseau de neurones possède autant de neurones en entrée qu'en sortie. Le nombre de neurones décroît dans les couches intermédiaires.

Proposé pour la première fois par [Kramer, 1991], l'objectif de ce réseau est de fournir en sortie les mêmes valeurs qu'en entrée, et d'analyser les valeurs sur les couches intermédiaires. Durant l'apprentissage, le réseau corrige ses poids pour que la sortie obtenue se rapproche de l'entrée.

Les couches intermédiaires, contenant moins de neurones que les couches entrée-sortie, donnent une représentation de plus faible dimension des données en entrée. Ce modèle est donc beaucoup utilisé pour réduire le bruit dans les données, donner une représentation de plus faible dimension de ces dernières, et pouvoir ensuite projeter les données dans un espace de dimen-

3. Ce que HTM considère comme couche inférieure représente la couche précédente pour les réseaux de neurones artificiels classiques.

4. https://github.com/numenta/NAB

sion réduite.

Mais les auto-encodeurs sont également utilisés pour détecter des anomalies. Avec le même apprentissage que pour la tâche précédente, il est ensuite possible de tester la sortie et d'attribuer un score d'anomalie selon la sortie prévue qui n'est autre que l'entrée. [Sakurada et Yairi, 2014] comparent l'efficacité d'un auto-encodeur sur des données non linéairement séparables, et montrent que les neurones activés sur les couches intermédiaires sont différents selon que les données en entrée soient normales ou anormales.

2.2.3.4.c Modèles probabilistes génératifs

Les modèles probabilistes génératifs ont pour objectif de trouver la probabilité conjointe, $\mathrm{p}(x, y)$, entre les entrées x et la sortie y. Ils utilisent les règles bayésiennes pour l'obtention de prédictions sur la valeur de sortie y sachant x, $\mathrm{p}(x|y)$.

Nous présentons alors deux familles de modèles très réputées basées sur ces règles : les modèles de MARKOV, et les modèles de MARKOV à état caché.

Modèles de MARKOV Les modèles de MARKOV sont un type particulier d'automate à état fini. Un automate à état fini est un modèle constitué d'un nombre fini d'états et de transitions entre chaque état représentant les conditions pour passer d'un état à un autre. L'état dans lequel se trouve la machine se nomme l'état courant.

Le modèle de MARKOV se différencie de l'automate classique par le fait d'utiliser la propriété de MARKOV. Cette propriété consiste à conserver, dans l'état courant, une représentation résumant les états précédemment parcourus, dans le but de prédire de manière probabiliste les états futurs. La probabilité du prochain état, appelé probabilité de transition, ne dépend pas directement de l'ensemble des états précédents, mais seulement de l'état courant contenant en son sein l'ensemble des $k-1$ états vus précédemment. Le modèle est dit d'ordre k.

L'ensemble des probabilités de transition peut être représenté dans une matrice de transition. Prenons un exemple avec trois états. La matrice P représente pour chaque ligne i la probabilité $p_{i,j}$ de passer de l'état e_i à l'état e_j. Prenons la matrice P pour un modèle de MARKOV d'ordre 1 :

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} \\ p_{2,1} & p_{2,2} & p_{2,3} \\ p_{3,1} & p_{3,2} & p_{3,3} \end{bmatrix} = \begin{bmatrix} 0.9 & 0.05 & 0.05 \\ 0.7 & 0.0 & 0.3 \\ 0.8 & 0.0 & 0.2 \end{bmatrix}$$

La probabilité $p_{1,2}$ de passer de l'état e_1 à l'état e_2 est égale à 0.05, et de rester dans le même état e_1 de 0.9. Durant la phase de détection d'anomalies, si un état dont la probabilité d'apparition est nulle ou presque est observé, cet instant sera déclaré anormal.

L'utilisation d'une chaîne de MARKOV d'ordre 1 pour la détection d'intrusion système et réseau est possible, comme présenté par [Ye, 2000]. La probabilité que le système soit dans un certain état à l'instant t est calculée comme étant le produit des états vus précédemment contenus dans une séquence de longueur définie.

Modèles de MARKOV cachés Contrairement aux modèles de MARKOV classiques, les modèles de MARKOV cachés masquent les états du système à l'utilisateur. Autrement dit, durant l'exécution du modèle, il n'est pas possible de connaître directement les états parcourus, ni même l'état courant, mais l'utilisateur obtient seulement une séquence de valeurs discrètes, donnant des indications sur le système sans donner formellement les états parcourus. Par exemple, un véhicule peut se trouver dans trois états différents : circulation en ville, en campagne et sur autoroute. Ces états ne nous sont pas directement connus, mais l'information vitesse (lent, moyenne, rapide) nous fournit des indications sur le véhicule. À présent, observons un exemple plus formel où nous pourrions avoir deux états e_1 et e_2 tels que les probabilités de transition d'un état à un autre soient fournies par la matrice :

$$P = \begin{bmatrix} 0.05 & 0.95 \\ 0.8 & 0.2 \end{bmatrix}$$

L'utilisateur ne connaît pas les états parcourus, mais obtient une des deux valeurs discrètes s_1 et s_2, dont les probabilités d'apparition, décrites dans la matrice S, dépendront de l'état courant :

$$S = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

Pour S, les lignes représentent l'état courant e_i, non connu par l'utilisateur, et les colonnes les valeurs s_1 et s_2, valeurs de sortie obtenues par l'utilisateur. Nous constatons que la probabilité d'obtenir la sortie s_1 est plus grande sur l'état e_1 que sur l'état e_2, donc si nous obtenons une longue liste de valeurs de s_1, nous pouvons seulement supposer, et non affirmer, que nous nous trouvons depuis assez longtemps sur l'état e_1.

Pour détecter les fraudes à la carte de crédit, [Srivastava *et al.*, 2008] définissent un état comme un type de transaction. Si une transaction survient

alors que sa probabilité d'apparition était faible, elle sera annotée anormale.

Nous venons de voir une liste des principales méthodes non supervisées utilisées dans la détection d'anomalies. Il existe d'autres méthodes tel que EXPoSE qui est également une méthode à noyaux et récemment proposée par [Schneider *et al.*, 2016] applicable de manière incrémentale, permettant d'évaluer la normalité des flux de données, ou d'autres méthodes basées proximité, par exemple [Kriegel *et al.*, 2008] qui étudie les angles entre les objets pour déduire la proximité d'un objet à partir de l'ouverture du secteur angulaire formée par deux demi-droites ayant pour origine commune l'objet à évaluer, et passant par deux autres objets du jeu de données.

Nous allons voir à présent comment ces méthodes peuvent être évaluées entre elles suivant le type de données et d'anomalies à détecter.

2.3 Évaluation et comparaison des méthodes de détection d'anomalies

Pour les méthodes d'apprentissage, différents types d'évaluation existent, qu'elles soient utilisées dans un contexte général, ou dans la détection d'anomalies.

Les indicateurs de temps CPU et de mémoire sont souvent mis en avant dans le choix de la méthode. En effet, il existe de nombreuses situations où l'un ou l'autre, voire les deux, sont restreints. Par exemple, lorsqu'une entreprise possède un système d'alarme pour empêcher les intrusions au sein de son siège, l'analyse du visage d'un individu devra se faire en temps raisonnable. Et pour réduire les coûts d'utilisation de cette solution, la mémoire utilisée ne devra pas non plus être excessive.

Bien entendu, ces deux indicateurs sont intéressants à examiner si les méthodes concernées fournissent des résultats concluants. Par exemple, pour les méthodes de classification (y compris d'anomalies), les méthodes seront évaluées selon la pertinence des classements obtenus sur le jeu de test. Pour cela, il est nécessaire de connaître les résultats attendus. Après qu'une méthode ait étiqueté les données de test, ces résultats fournis par la méthode sont comparés avec les vraies étiquettes.

Il existe de nombreux indicateurs. Mais de manière plus spécifique, la famille des méthodes de détection d'anomalies a aussi des méthodes d'évaluations qui lui sont propres. L'objectif étant de comparer les méthodes selon

le nombre d'anomalies détectées. Quatre indicateurs sont très fréquemment — et à juste titre — utilisés dans ce domaine. L'indicateur TP, pour *True Positive*, énumère le nombre d'anomalies qui doivent et qui ont été détectées par la méthode. L'indicateur FN (*False Negative*) comptabilise le nombre anomalies non détectées mais qui auraient dû l'être. Le nombre de fausses anomalies détectées par la méthode sera noté FP (*False Positive*). Et l'indicateur TN (*True Negative*) retourne le nombre de fois où la méthode signale correctement qu'aucune anomalie n'est présente.

Nous pouvons observer ces quatre indicateurs sur une même figure 2.14.

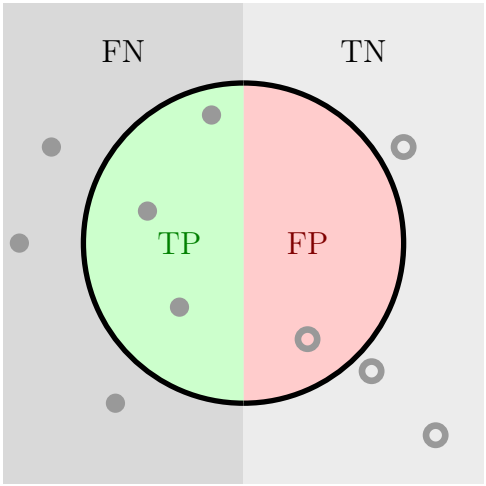


FIGURE 2.14 – Les objets détectés par la méthode comme étant anormaux se trouve dans le cercle, en vert pour les vrais anomalies (TP) et en rouge pour les fausses anomalie (FP).

À présent, nous allons nous intéresser à l'utilisation de ces quatre indicateurs pour comparer les différentes méthodes, puis nous verrons d'autres indicateurs permettant également d'évaluer les méthodes de détection entre elles.

2.3.1 Précision et Rappel

Ces deux mesures sont utilisées pour estimer la pertinence de classification des données. Pour cela, il est alors nécessaire de connaître les labels attendus pour chaque donnée.

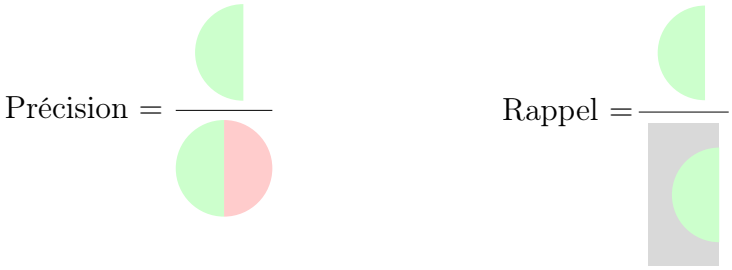
Pour calculer la précision et le rappel pour une classe c, il est nécessaire de connaître les TP, TN et FP de cette classe, comme expliqué précédemment dans cette section. La précision décrite pour c le ratio entre le nombre

d'objets correctement classés dans c et le nombre total d'objets classés dans c (correctement et incorrectement classés) :

$$\text{précision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

En ce qui concerne le rappel de c, il décrit le ratio entre le nombre d'objets correctement classés dans c et le nombre d'objets attendus dans c :

$$\text{rappel} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$



(a) Calcul de la Précision. (b) Calcul du Rappel.

FIGURE 2.15 – Les formules de précision et de rappel sont exprimées ici à partir des sous-ensembles représentés dans la figure 2.14.

La précision mesure alors la pertinence (ou cohérence) de classification dans c, c'est-à-dire parmi les objets classifiés comme appartenant à la classe c, combien en font réellement partie. Et le rappel retourne l'exhaustivité de classification des objets appartenant à la classe c. Par exemple, un rappel de 10% pour la classe c signifie que sur la totalité des objets appartenant à c, un objet sur dix a été correctement classé dans c.

2.3.2 Courbe ROC et aire sous la courbe

La courbe ROC *Receiver Operating Characteristic* met en évidence, pour un modèle de classification binaire, l'aptitude qu'a un modèle à classifier correctement les données selon la variation du seuil de discrimination.

La courbe est tracée en deux dimensions avec en abscisse le taux de faux positifs et en ordonnée le taux de vrais positifs, tous deux exprimés entre 0 et 1. Au point de coordonnée $(0, 0)$, c'est-à-dire lorsque le seuil est au plus bas, le nombre de faux positif est nul tout comme le nombre de vrai positif. Le modèle ne détecte donc aucun objet dans la classe considérée. Par contre, au point de coordonnée $(1, 1)$, le modèle déclare tous les objets inclus

dans la classe considérée, ce qui signifie que le modèle trouve tous les vrais positifs mais aussi classifie tous les autres objets en faux positifs. Prenons un exemple simple avec 10 objets que nous cherchons à classifier dans la classe 1. Le modèle ne connaît pas les classes de ces 10 objets. Nous utilisons un modèle d'apprentissage quelconque qui, pour chaque objet, retourne un score compris entre 0 et 1. Dans chaque tableau, nous inscrivons pour chaque objet le score obtenu par le modèle :

TABLE 2.2 – Tableaux des scores de deux classes d'objets pour le calcul de la courbe ROC. La classe 1 se trouve dans le tableau (a) et la classe 2 dans le tableau (b).

	score d'appartenance à la classe 1
o_0	0.7
o_1	0.8
o_2	0.9
o_3	0.6
o_4	0.9

(a) Objets de la classe 1

	score d'appartenance à la classe 1
o_5	0.4
o_6	0.75
o_7	0.68
o_8	0.3
o_9	0.3

(b) Objets de la classe 2

À présent, nous plaçons le seuil S à 1, ce qui signifie que tous les objets ayant un score s strictement plus petit que 1 ne sont pas prédits, selon les réponses du modèle d'apprentissage dans la classe 1, et tous les objets ayant un score égal ou supérieur à 1 sont dans la classe 1. Aucun des objets n'est alors prédit par le modèle dans la classe 1. Pour construire la courbe ROC, nous pouvons placer un premier point au coordonnée $(0, 0)$. Ensuite, nous diminuons suffisamment le seuil pour que au moins un objet soit prédit dans la classe 1, et atteindre un seuil $S = 0.9$. Nous obtenons un vrai positif sur cinq et toujours zéro faux positif. Ce qui nous permet de placer un deuxième point $(0, 0.2)$ pour construire la courbe ROC. Nous augmentons le seuil alors progressivement pour chaque nouveau point, c'est-à-dire lorsqu'au moins un nouveau positif de plus est prédit. Jusqu'à que tous les objets soit positifs. La courbe obtenue est visible Figure 2.16.

Plus les points de la courbe tendent vers les coordonnées $(0, 1)$, plus les positifs prédits sont vrais, et donc la prédiction efficace. À l'inverse, lorsque les points de la courbe ROC tendent vers $(1, 0)$, le nombre de faux positifs augmente et le nombre de vrais positifs diminue. Si la courbe ROC d'une méthode est toujours au-dessus de la courbe ROC d'une autre méthode, cela signifie que les prédictions de la première méthode sont meilleures, quel que

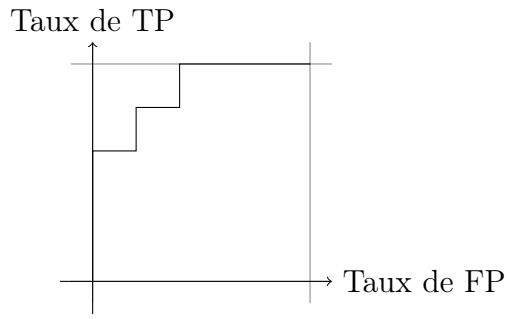


FIGURE 2.16 – La courbe ROC.

soit le seuil appliqué.

Une autre interprétation de la courbe consiste à calculer l'aire sous la courbe ROC. Le résultat obtenu, appelé AUC pour *Area Under the Curve*, permet de comparer différentes courbes ROC de façon à mettre en avant la proportion de vrais positifs selon la proportion de faux positifs prédits. Si l'aire sous la courbe est grande, cela met en avant la capacité de la méthode à prédire un grand nombre de vrais positifs en limitant le nombre de faux. C'est l'une des méthodes d'évaluation employées dans [Campos *et al.*, 2016] pour comparer différentes méthodes d'apprentissage non supervisées pour la détection d'anomalies. Le résultat ROC AUC renvoie à la probabilité de classifier correctement deux objets, un normal et un anormal, comme expliqué par [Hanley et McNeil, 1982]. Un résultat ROC AUC de 1 garantit que le score d'anomalie de l'objet anormal sera toujours plus élevé que celui de l'objet normal, et à l'inverse un ROC AUC de 0 certifie que le score de l'objet normal sera toujours plus élevé.

Le score ROC AUC obtenu sur les données de la table 2.2 est de $22/25 = 0.88$.

2.3.3 Courbe Précision-Rappel et aire sous la courbe

La courbe Précision-Rappel, appelée aussi PRC, retourne la valeur de la précision, comme décrit dans la section 2.3.1, en abscisse, et la valeur du rappel en ordonnée. Le principe est le même que pour la courbe ROC : le seuil de détection augmente progressivement jusqu'à ce que tous les objets se retrouvent avec un score plus petit que ce dernier.

En reprenant l'exemple de la table 2.2, nous traçons la courbe PRC sur la figure 2.17. L'aire sous la courbe est de 0.89.

Nous analysons donc qu'en voulant augmenter la part de TP détecté,

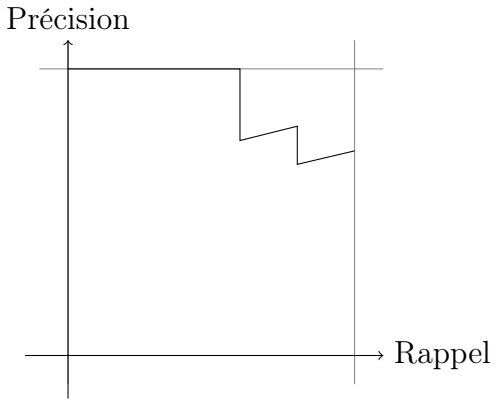


FIGURE 2.17 – La courbe PRC.

c'est-à-dire la valeur de rappel, la précision se détériore, dû à une augmentation des FP. Nos résultats entre la courbe ROC de la figure 2.16 et ceux de la courbe PRC de la figure 2.17 sont assez symétriques : lorsque le nombre de TP augmente, le nombre de FP augmente également.

Cependant, notre exemple est équilibré, le nombre d'objets dans chacune des deux classes est le même. C'est pourquoi les valeurs de rappel et de précision sont proportionnelles aux nombres de TP et de FP. Si ces deux classes étaient déséquilibrées, comme décrit dans [Saito et Rehmsmeier, 2015], les deux courbes ne seraient pas tant similaires, car la courbe PRC tend à donner un ratio du nombre de TP par rapport aux nombres de détections faites et à effectuer. C'est pourquoi, comme expliqué par [Saito et Rehmsmeier, 2015], il semblerait que l'évaluation de la pertinence de la détection soit plus intéressante avec la courbe PRC qui ne sous-estimera pas la classe des anomalies en effectif minoritaire, contrairement à la courbe ROC.

2.4 Conclusion

Nous avons vu dans ce chapitre une définition de la détection d'anomalies et ses domaines d'application les plus récurrents dans la littérature. Ensuite, nous avons présenté différentes méthodes de détection, que nous avons réuni dans deux grandes familles : les méthodes supervisées et semi-supervisées, et les méthodes non supervisées.

Les méthodes supervisées et semi-supervisées sont très efficaces pour détecter des données anormales déjà rencontrées et étiquetées comme telles. Elles le sont aussi pour la détection de nouveautés — données jamais rencontrées — apparaissant comme anormales vis-à-vis des données normales déjà

apprises. Ces méthodes requièrent des données d'entraînement sur lesquelles les experts du domaine apportent, après analyse, la plus-value suffisante.

Le coût de traitement durant la phase d'apprentissage est la principale limite de ces méthodes. Si certains problèmes demanderont peu de données pour résumer correctement les classes et apprendre leurs contours, cela ne se produit que peu souvent. En effet, l'obtention d'une généralisation satisfaisante est souvent faite grâce à l'analyse d'un grand nombre de données, ou grâce à la mise en lumière de discriminants non perçus par les experts lors de l'analyse manuelle. Nos travaux sont directement appliqués sur des données métiers non étiquetées, c'est pourquoi nous avons présenté de manière plus détaillée les méthodes non supervisées dans le domaine de la détection d'anomalies.

Il existe principalement deux types de méthodes de détection non supervisées : les méthodes basées sur les groupes, et les méthodes basées sur le voisinage. Le comparatif présenté dans [Campos *et al.*, 2016] explique qu'aucune méthode non supervisée n'est toujours meilleure que toutes les autres, mais qu'il existe un petit groupe de méthodes se retrouvant souvent en tête des classements, telle que celle utilisant le score LOF présenté Section 2.2.3.3.b. Dans la lignée de ces méthodes basées sur des scores du type LOF, une contribution récente est le score CFOF (décrit Section 2.2.3.3.b) qui est particulièrement intéressant pour sa robustesse en haute dimension, ce qui lui permet notamment d'être plus pertinent que le score LOF sur des objets de dimensions élevées [Angiulli, 2017]. Cependant le score CFOF n'est pas réellement adapté pour être utilisé dans un contexte de flux de données, et c'est sur cet aspect que nous nous concentrons dans la suite de ce mémoire. Nous présentons dans les chapitres 3 et 4 nos contributions permettant de calculer à la volée des approximations de bonne qualité du score CFOF de chaque nouvel objet arrivant dans un flux de données. Ensuite, dans le chapitre 5, nous détaillons la mise en œuvre d'une chaîne de traitements basée sur le calcul de ces approximations pour la détection d'anomalies au sein du système d'information de la SNCF.

Chapitre 3

Calcul du score CFOF par approximation et élagage

Sommaire

3.1 Introduction

Nous allons présenter dans ce chapitre la première proposition de méthode de détection d'anomalies de cette thèse, adaptée au type de flux étudiés dans le contexte industriel de la SNCF. Ces travaux on été présentés en partie dans [Foulon *et al.*, 2019c] et [Foulon *et al.*, 2020].

Dans notre problématique de thèse, les signaux analysés proviennent du système d'information de la SNCF, sous forme de flux de données primaires. Nous construisons, à partir de ceux-ci, un flux contenant les valeurs d'un indicateur (comme par exemple le nombre moyen de messages reçus par minute). Ensuite, nous transformons ce flux en séquences en le découpant en portions contiguës par le biais d'une fenêtre glissante. Nous obtenons alors un ensemble de séquences de même longueur, reflétant l'état du système à différents instants. Notre objectif principal est de connaître l'état de santé du système d'information à l'instant présent. Pour cela nous évaluons la dernière séquence construite en la comparant avec l'historique des séquences observées par le passé. Puisque le mécanisme de construction décrit ci-dessus fournit des séquences en haute dimension (et pouvant potentiellement être basées sur plusieurs flux), nous disposons — pour identifier les anomalies potentielles — d'informations sur le présent, les données les plus récentes, tout en tenant compte du passé proche. Le contexte de ce passé est décrit par les données les plus anciennes de la séquence en cours d'analyse. Les séquences obtenues pour l'historique de référence ne sont pas étiquetées et la méthode proposée fonctionne en non-supervisée. Une caractéristique des jeux étudiés est que les séquences sont majoritairement normales, celles anormales étant minoritaires.

Dans un contexte d'apprentissage non supervisé et dans le cas de données en haute dimension, nous avons pu voir dans l'état de l'art que le score CFOF est un bon candidat. En effet, les résultats présentés dans [Angiulli, 2017, Angiulli, 2020] confirment la pertinence du choix de cette méthode dans ce cas d'usage. Cependant, bien que les méthodes existantes CFOF et *fast*-CFOF proposées par Anguilli soient adaptées pour le calcul des scores de l'ensemble du jeu de données ou sur un échantillon de ce dernier, elles le sont moins pour le calcul incrémental du score d'un nouvel objet qui nécessiterait de mettre à jour les voisinages de la totalité de l'ensemble ou de l'échantillon considéré du jeu de données. En effet il est nécessaire lors de l'arrivée d'un nouvel objet de retrouver sa position dans les voisinages de tous les autres objets déjà stockés. C'est pourquoi nous proposons dans ce travail un algorithme permettant d'obtenir une approximation rapide du score du nouvel objet, avec un coût de calcul moindre, et permettant d'envisager un calcul incrémental sur une base de haute dimension. Dans cette section, les objets que nous considérons pour le calcul du score CFOF sont précisément les séquences de mesures présentées ci-dessus [1].

Tout d'abord, nous allons reprendre la définition du score CFOF donnée

1. Nous utilisons donc indifféremment les termes *séquence* ou *objet* pour les nommer.

par Anguilli, et argumenter sa pertinence dans le cadre qui nous intéresse.

Nous proposerons ensuite un algorithme permettant d'obtenir une approximation du score CFOF pour un nouvel objet. Pour cela, il sera nécessaire de présenter une méthode d'indexation des données. Enfin, nous décrirons les résultats de notre méthode, en comparant les scores obtenus avec les scores CFOF exacts sur un ensemble de jeux de données synthétiques proposé par Anguilli, et sur un benchmark connu de détection d'anomalies, le *Numenta Anomaly Benchmark*.

3.2 Concentration Free Outlier Factor

Pour obtenir le score CFOF [Angiulli, 2017] d'un objet $\boldsymbol{q}$ au regard d'un ensemble d'objets de référence $\mathcal{R}$, il est nécessaire de chercher la taille du voisinage minimal k_m tel que $\boldsymbol{q}$ soit inclus dans les k_m plus proches voisins d'au moins une fraction ϱ des objets de $\mathcal{R}$. Le score CFOF($\boldsymbol{q}$) est le résultat de la normalisation de la valeur de k_m par la taille de $\mathcal{R}$. Le paramétrage de ce score s'effectue par le seuil $\varrho \in [0, 1]$ déterminant la proportion d'objets de $\mathcal{R}$ incluant $\boldsymbol{q}$ dans leur voisinage (de taille k_m). Ce paramètre permet de donner plus ou moins de sensibilité à la valeur du score selon le nombre d'objets de référence auxquels $\boldsymbol{q}$ doit s'apparenter, et comme le montrent les expériences de la section 3.6, son réglage est simple en pratique.

Nous allons définir ce score de façon plus formelle. Soit $\mathrm{nn}_k(\boldsymbol{q})$ le $k^{\text{ième}}$ plus proche voisin d'un objet $\boldsymbol{q}$, c'est-à-dire tel qu'il existe seulement $k-1$ objets plus proches de $\boldsymbol{q}$ que ne l'est l'objet $\mathrm{nn}_k(\boldsymbol{q})$. L'ensemble des k plus proches voisins de $\boldsymbol{q}$ est noté $\mathrm{NN}_k(\boldsymbol{q})$ et inclut tous les objets tel que $\{\mathrm{nn}_i(\boldsymbol{q}) \mid 1 \leq i \leq k\}$. Soit $\mathrm{N}_k(\boldsymbol{q})$ le nombre d'objets de référence ayant $\boldsymbol{q}$ parmi leur k plus proches voisins et défini par $\mathrm{N}_k(\boldsymbol{q}) = |\{\boldsymbol{p} : \boldsymbol{q} \in \mathrm{NN}_k(\boldsymbol{p})\}|$.

Le score CFOF d'un objet $\boldsymbol{q}$ est alors défini par :

$$\mathrm{CFOF}(\boldsymbol{q}) = \min\{k/|\mathcal{R}| : \mathrm{N}_k(\boldsymbol{q}) \geq \varrho \times |\mathcal{R}|\}$$

Actuellement, le score CFOF est le seul pour lequel il a été démontré de façon formelle (et constaté de façon expérimentale) par [Angiulli, 2017] qu'il restait insensible au phénomène de concentration lorsque la dimensionnalité des données croît.

3.3 Proposition d'adaptation du calcul du score CFOF

Les séquences que nous manipulons sont construites à la volée par agrégation des mesures récoltées au fil de l'eau et sont donc par définition de dimension élevée. Cependant, l'algorithme initialement utilisé par Anguilli pour construire son score CFOF n'est pas adapté pour calculer à la volée — et donc dans un contexte de flux de données — le score d'une nouvelle séquence.

En effet, pour calculer le score d'une séquence parmi un ensemble de séquences de référence, nous devons déterminer les voisinages de toutes les séquences de cet ensemble. Cette étape nous amène à une complexité de calcul de $\mathcal{O}(n^2 \times d)$ [Angiulli, 2017], où n est la taille de l'ensemble de référence et d la dimension des séquences. Il est vrai qu'en effectuant la totalité de ces calculs, nous avons ensuite accès rapidement aux scores de toutes les séquences de l'ensemble. Cependant, dans ce travail, nous souhaitons obtenir le score CFOF de la dernière séquence produite au fil de l'eau. À l'arrivée d'une nouvelle séquence, deux possibilités s'offrent donc à nous pour obtenir la position de celle-ci parmi les voisinages des séquences de référence : soit nous recalculons toutes les distances entre toutes les séquences, ce qui nous ramène à la complexité de calcul initial, soit nous maintenons en mémoire la distance entre toutes les séquences et nous pouvons alors atteindre un coût de $\mathcal{O}(n^2)$ en mémoire.

Cependant, lorsqu'une nouvelle séquence est générée, nous n'avons pas besoin de connaître ses voisins les plus proches, mais seulement sa position dans le voisinage des séquences utilisées comme historique de référence. La méthode de force brute, qui consisterait à calculer la distance entre toutes les séquences, est ainsi trop coûteuse et inutile pour le calcul du score sur un flux de données. C'est pourquoi, nous proposons deux étapes permettant par la suite d'approximer rapidement la position d'une nouvelle séquence :

1. Indexer les séquences de l'historique de référence dans un arbre d'indexation,
2. Approximer la distribution des séquences contenues dans les nœuds de l'arbre.

Ces deux étapes permettent ainsi de construire itérativement et maintenir localement dans les nœuds de l'arbre la distribution des séquences qui y sont stockées. Nous les utilisons ensuite pour calculer la position d'une nouvelle séquence dans les voisinages de celles déjà stockées. Nous allons décrire ce processus plus en détail dans les sections suivantes.

3.3.1 Indexation

Soit deux séquences $\boldsymbol{p}$ et $\boldsymbol{q}$ de dimension égale. Les séquences contenues dans l'historique de référence sont indexées dans le but de retrouver plus rapidement la distribution des distances entre ces séquences. Nous définissons le terme "rang" comme la position d'une séquence dans le voisinage d'une autre séquence tel que k est le rang de $\boldsymbol{q}$ dans le voisinage de $\boldsymbol{p}$ sachant la définition de $\mathrm{NN}_k(\boldsymbol{p}) = \boldsymbol{q}$ décrite dans la section 2.2.3.3.a.

Une méthode efficace pour effectuer cette indexation est de considérer la séquence comme un objet dans un espace à plusieurs dimensions, et d'utiliser ces informations dimensionnelles pour structurer l'espace qui les contient. Nous proposons donc d'utiliser un arbre pour organiser hiérarchiquement les séquences qui y sont stockées : ainsi, les séquences sont séparées selon leur répartition dans l'espace, chaque nœud représentant un parallélotope droit de l'espace de départ, c'est-à-dire pavé droit ou parallélépipède rectangle de dimension quelconque. Cette structure permet de retrouver rapidement des séquences proches, car elles sont stockées dans des nœuds voisins.

3.3.1.1 Trois catégories de parallélotopes droits

Comme décrit au début de cette section 3.3, nous cherchons à connaître, à moindre coût, le rang d'une séquence $\boldsymbol{q}$ dans les voisinages de chacune des autres séquences considérées dans l'historique de référence. Pour une séquence $\boldsymbol{p}$ faisant partie de cet ensemble de référence, l'objectif est donc d'obtenir de façon efficace les distances vers toutes les autres séquences, de façon à approximer le rang de la séquence $\boldsymbol{q}$ à évaluer dans le voisinage de $\boldsymbol{p}$.

Pour l'obtention du score CFOF de $\boldsymbol{q}$, seul le rang de la séquence à évaluer nous intéresse. En effet, il n'est pas nécessaire de connaître les distances et le rang exact entre toutes les séquences, mais seulement de savoir si elles ont un rang plus petit ou plus grand que celui de $\boldsymbol{q}$ dans le voisinage de tout $\boldsymbol{p}$. Nous avons donc fait le choix de répartir les séquences dans des parallélotopes droits dont les intersections sont nulles, c'est-à-dire qu'ils ne se chevauchent pas. Une séquence ne peut donc pas se trouver dans deux parallélotopes droits en même temps.

Dès lors que la distance $\mathrm{dist}(\boldsymbol{p}, \boldsymbol{q})$ entre $\boldsymbol{p}$ et $\boldsymbol{q}$ est calculée, il est ainsi possible de ne conserver que les parallélotopes droits traversés par l'hypersphère de centre $\boldsymbol{p}$ et de rayon $\mathrm{dist}(\boldsymbol{p}, \boldsymbol{q})$ et d'élaguer les autres. Cette démarche permet ainsi d'identifier les parallélotopes droits contenant les séquences les plus proches de $\boldsymbol{q}$ dans le voisinage de $\boldsymbol{p}$. Nous pouvons voir dans la figure 3.1 un exemple en 2 dimensions où le rayon r passe par les parallélotopes droits candidats.

Dans l'exemple de la figure 3.1, nous pouvons observer trois catégories de parallélotopes droits :

— les parallélotopes droits traversés par l'hypersphère,
— les parallélotopes droits entièrement inclus dans l'hypersphère,
— les parallélotopes droits entièrement exclus de l'hypersphère.

De façon plus formelle, notons $\text{dist}(\boldsymbol{p}, \boldsymbol{s}_{inclue})$ la distance entre $\boldsymbol{p}$ et toute séquence $\boldsymbol{s}_{inclue}$ contenue dans les parallélotopes droits entièrement inclus dans l'hypersphère, et $\text{dist}(\boldsymbol{p}, \boldsymbol{s}_{exclue})$ la distance entre $\boldsymbol{p}$ et toute séquence $\boldsymbol{s}_{exclue}$ contenue dans les parallélotopes droits entièrement exclus dans l'hypersphère. Il est facile de noter que :

$$\text{dist}(\boldsymbol{p}, \boldsymbol{s}_{inclue}) < \text{dist}(\boldsymbol{p}, \boldsymbol{q}) < \text{dist}(\boldsymbol{p}, \boldsymbol{s}_{exclue}) \tag{3.1}$$

ce qui signifie que les séquences se trouvant dans les parallélotopes droits entièrement inclus — respectivement exclus — dans l'hypersphère se trouvent à une distance plus petite — resp. plus grande — de $\boldsymbol{p}$ que ne l'est $\boldsymbol{q}$. Cet élagage permet d'ordonner rapidement les distances depuis $\boldsymbol{p}$ vers toutes les autres séquences pour ne conserver que celles contenues dans les parallélotopes droits traversés par l'hypersphère. Les autres distances étant bornées par l'équation 3.1, les connaître avec exactitude ne nous apporte pas plus d'informations utiles pour correctement approximer le rang $\boldsymbol{q}$ dans le voisinage de $\boldsymbol{p}$.

Bien sûr, plus la taille des parallélotopes droits est petite, plus leur nombre croît : le nombre de parallélotopes droits à tester augmente également. C'est pourquoi nous avons choisi d'utiliser une structure hiérarchique, un arbre d'indexation, permettant ainsi d'affiner petit à petit la taille des parallélotopes droits à considérer. Cette mise en pratique est présentée dans la section 3.3.1.2 suivante.

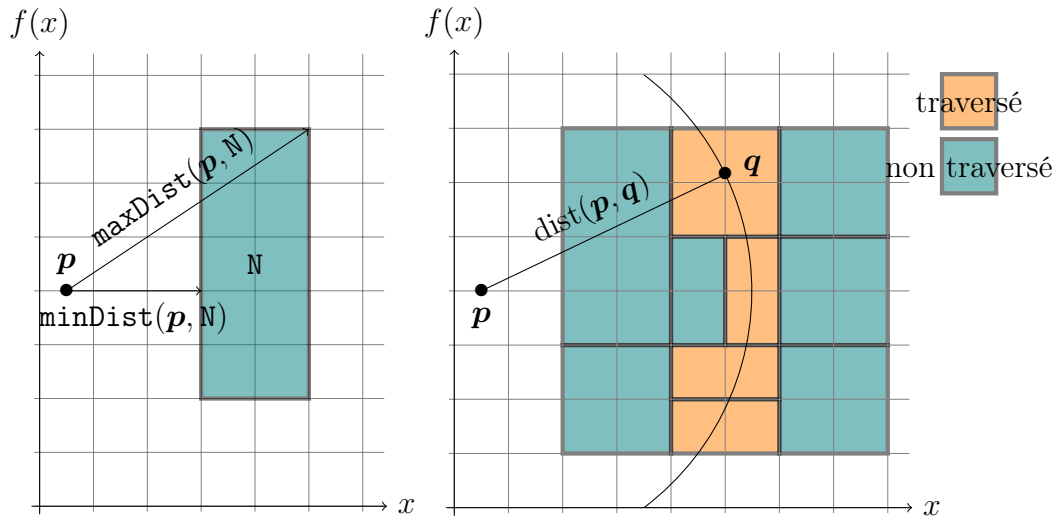
3.3.1.2 Accélération du parcours des séquences : utilisation des parallélotopes droits et élagage des branches de l'arbre d'indexation

Pour accélérer le tri des parallélotopes droits en trois catégories, nous proposons d'utiliser un arbre d'indexation dans lequel chaque nœud de l'arbre représente un parallélotope droit, et dont les nœuds de même niveau ne se chevauchent pas. La taille des parallélotopes droits représentés diminue au fur et à mesure que l'on descend dans l'arbre. Nous associons à chaque séquence les distances minimum et maximum vers chaque nœud de l'arbre : la distance minimum — respectivement maximum — depuis une séquence $\boldsymbol{p}$ vers un

noeud N est définie comme la plus petite — resp. plus grande — distance possible entre $\boldsymbol{p}$ et toutes les séquences incluses dans N.

Ces deux distances extrema sont illustrées sur la figure 3.1a à partir d'un exemple incluant une séquence notée $\boldsymbol{p}$ et un nœud dont l'aire occupée est représentée par un rectangle.

Pour l'élimination des nœuds, nous nous intéressons à deux cas possibles : celui où la distance maximum entre $\boldsymbol{p}$ et le nœud est plus petite que dist$(\boldsymbol{p}, \boldsymbol{q})$ et celui où la distance minimum entre $\boldsymbol{p}$ et le nœud est plus grande que dist$(\boldsymbol{p}, \boldsymbol{q})$ (illustré Figure 3.1b). Comme expliqué dans la section 3.3.1.1, tous les nœuds du premier type — respectivement du second type — contiennent des séquences plus proches — resp. plus éloignées — de $\boldsymbol{p}$ que ne l'est $\boldsymbol{q}$. Autrement dit, le rang de $\boldsymbol{q}$ dans le voisinage de $\boldsymbol{p}$ est au moins plus grand — resp. plus petit — que celui des séquences contenues dans le nœud du premier type — resp. second type. Ces deux types de nœuds ne sont pas visités plus en profondeur, mais l'effectif du premier type est comptabilisé de façon à incrémenter le rang en cours de détermination de $\boldsymbol{q}$ dans le voisinage de $\boldsymbol{p}$, car nous pouvons conclure de façon certaine que toutes les séquences se trouvant dans les nœuds du premier type possèdent un rang plus petit que $\boldsymbol{q}$.



(a) Distances extrema entre $\boldsymbol{p}$ et le nœud N.

(b) Représentation des nœuds traversés et non traversés par le cercle.

FIGURE 3.1 – La figure (a) illustre un exemple de distances minimum et maximum entre $\boldsymbol{p}$ et un nœud N. La figure (b) présente les nœuds se trouvant trop loin, trop près ou traversés par le cercle de centre $\boldsymbol{p}$ et de rayon dist$(\boldsymbol{p}, \boldsymbol{q})$.

Une fois ce premier élagage effectué, la position de la nouvelle séquence dans le voisinage de $\boldsymbol{p}$ reste à approximer dans les nœuds restants, c'est-à-dire les nœuds traversés par l'hypersphère de rayon dist$(\boldsymbol{p}, \boldsymbol{q})$. Au lieu de calculer la distance de $\boldsymbol{p}$ vers toutes les séquences contenues dans ces nœuds, nous allons voir comment approximer leurs positions.

3.3.2 Approximation locale de la distribution des séquences

Pour permettre d'accélérer le calcul du score CFOF, nous proposons d'approximer la position de la nouvelle séquence dans le voisinage de $\boldsymbol{p}$. Cette approximation porte sur les séquences incluses dans les nœuds candidats, telles que définies dans la section 3.3.1. Pour rappel, les nœuds qui n'ont pas été élagués dans la première phase sont ceux pour lesquels les distances minimales et maximales de chacun des nœuds depuis $\boldsymbol{p}$ encadrent la distance dist$(\boldsymbol{p}, \boldsymbol{q})$.

Prenons un nœud `N` parmi la liste des nœuds candidats, et considérons qu'il contient n_s séquences. Nous souhaitons alors connaître la position de la nouvelle séquence dans le voisinage de $\boldsymbol{p}$. Parmi les $n_s + 1$ séquences, composées des n_s séquences et de la séquence $\boldsymbol{q}$, cette dernière sera le $i^{\text{ième}}$voisin de $\boldsymbol{p}$ avec $0 < i \leq n_s + 1$. Pour approximer une valeur plus précise de i, il est possible d'étudier les séquences incluses dans `N` pour en approximer une distribution de leurs distances vers $\boldsymbol{p}$. Nous expliquons plus en détail nos choix concernant la méthode d'approximation dans la section 3.5.2. Mais il est important de noter qu'il est possible d'utiliser une méthode différente suivant l'hypothèse de distribution faite sur les données stockées dans l'arbre.

De manière plus générique, nous utilisons ces approximations de façon à ne pas avoir à calculer la distance vers toutes les séquences contenues dans les nœuds candidats. Pour cela, nous maintenons dynamiquement dans chaque nœud de l'arbre des informations statistiques portant sur les séquences qu'il contient, et qui suffiront pour ensuite calculer la distribution des distances vers n'importe quelle séquence. La position obtenue étant une approximation, cette étape retournera non plus le score CFOF exact, que nous écrivons aussi $\text{CFOF}_{\text{exact}}$, mais une approximation de ce score, appelé CFOF approximé ou $\text{CFOF}_{\text{approx}}$. Nous allons à présent aborder le choix qui a été fait concernant la structure de l'index.

3.4 Arbre d'indexation : *i*SAX

Le but de la structure d'indexation est de nous permettre de retrouver plus rapidement les parallélotopes droits contenant les séquences dans le proche voisinage de $\boldsymbol{p}$, la nouvelle séquence à évaluer.

Les arbres sont des structures d'indexation très utilisées, car ils sont capables de stocker différents types de données, et dans notre cas une grande quantité de séquences, pouvant aller jusqu'à plusieurs milliards. Ils possèdent un nœud principal appelé "racine" qui est l'ascendant de tous les autres. Tous les nœuds sauf le nœud "racine" ont exactement un ascendant direct (nœud parent) qui ne peut pas être eux-mêmes. Par définition, un nœud parent possède au moins un nœud enfant (descendant direct). Si un nœud n'a pas de descendant, il est appelé nœud "feuille".

Dans notre contexte, cette structure permet de séparer rapidement les séquences de référence et d'identifier les nœuds de l'arbre candidats pour retrouver la position d'une nouvelle séquence parmi le voisinage d'une séquence déjà indexée. Nous avons besoin que la structure de l'arbre respecte deux conditions :

1. Les nœuds n'ayant pas de lien d'ascendance (ni de descendance) entre eux dans l'arbre ne doivent pas se chevaucher ;

2. les bornes minimales et maximales, introduites précédemment, doivent être calculables pour n'importe quel point dans l'espace de recherche.

Comme expliqué dans [Lin *et al.*, 2007b], la discrétisation SAX fournit une borne inférieure décrivant la distance minimale entre deux valeurs. De plus, l'arbre d'indexation *i*SAX est capable de contenir des millions de séries temporelles [Shieh et Keogh, 2008] et plusieurs milliards dans la version 2 de l'arbre *i*SAX proposée par [Camerra *et al.*, 2010]. Cette structure adaptée pour l'indexation de série temporelle a été reprise dans de nombreux travaux, comme par exemple [Castro et Azevedo, 2010, McGovern *et al.*, 2011, Hammerla *et al.*, 2013, Baydogan et Runger, 2015].

Dans [Shieh et Keogh, 2008], pour construire l'arbre *i*SAX, nous pouvons voir qu'il est nécessaire de déterminer la moyenne et l'écart-type des données à indexer pour chaque dimension. Ces deux mesures permettent de fixer les séparateurs entre les parallélotopes droits, en divisant en parts égales l'aire sous la loi normale pour chaque dimension. Ensuite, lors de l'insertion de chaque nouvelle donnée, lorsqu'un nœud feuille se retrouve avec un effectif plus grand qu'un seuil fixé à l'avance par l'utilisateur, ce nœud se divise en deux parallélotopes droits selon les séparateurs préalablement définis.

3.5 Utilisation de l'arbre d'indexation pour l'approximation de CFOF

À présent, nous allons voir l'algorithme permettant de calculer une approximation du score CFOF pour une nouvelle séquence.

3.5.1 Algorithme d'approximation

Dans un premier temps, les séquences utilisées comme historique de référence sont indexées dans l'arbre iSAX suivant la démarche proposée par [Camerra *et al.*, 2010]. Une fois les séquences insérées dans l'arbre, nous calculons pour chacune d'entre elles les deux distances, minimale et maximale, vers tous les noeuds qui ont été construits, comme expliqué dans la section 3.3.1.2. Aussi, nous calculons les estimateurs concernant la distribution des séquences. Nous avons rapidement abordé ces calculs et leur utilité dans la section 3.3.2, mais nous reviendrons plus en détail sur ces estimateurs dans la section 3.5.2. Pour le moment, nous les noterons $\widetilde{\text{dist}}(\mathtt{N}_f, p)$ et $\tilde{\sigma}_{\mathtt{N}_f}(p)$, le premier étant l'estimation de la moyenne des distances de p aux séquences contenues dans un noeud $\mathtt{N}_f$, et le second l'écart-type correspondant.

L'étape clef de l'algorithme 1 porte sur le calcul pour chaque séquence de référence $\boldsymbol{p}$ stockée dans l'arbre iSAX et appartenant à l'ensemble $\mathcal{R}$, de la position de $\boldsymbol{q}$, notée $v\text{-}rang_{\boldsymbol{p}}(\boldsymbol{q})$, dans le voisinage de $\boldsymbol{p}$. La valeur de ce rang est de 1 si $\boldsymbol{q}$ est le plus proche voisin de $\boldsymbol{p}$, 2 si $\boldsymbol{q}$ est le second plus proche voisin de $\boldsymbol{p}$, etc. La valeur de ce rang, notée $v\text{-}rang_{\boldsymbol{p}}(\boldsymbol{q})$, est définie plus précisément comme étant la valeur de k pour laquelle $\text{nn}_k(\boldsymbol{p}) = \boldsymbol{q}$.

Une fois que ces valeurs $v\text{-}rang_{\boldsymbol{p}}(\boldsymbol{q})$ sont déterminées pour toutes les séquences $\boldsymbol{p}$ de l'historique de référence, le calcul du score CFOF($\boldsymbol{q}$) s'obtient en trois étapes simples :

1. trier ces valeurs par ordre croissant dans une liste notée $l_{v\text{-}rang}$,
2. considérer l'élément placé à l'indice arrondi à l'entier supérieur $\lceil \varrho \times |\mathcal{R}| \rceil$, qui correspond à la taille de voisinage minimal k_m telle que $\boldsymbol{q}$ soit dans les k_m plus proches voisins d'au moins une fraction ϱ du jeu de données,
3. diviser cette valeur par la taille $|\mathcal{R}|$ du jeu de données pour la normaliser. La valeur CFOF($\boldsymbol{q}$) obtenue est alors $l_{v\text{-}rang}[\lceil \varrho \times |\mathcal{R}| \rceil]/|\mathcal{R}|$.

L'algorithme 1 détermine les $v\text{-}rang_{\boldsymbol{p}}(\boldsymbol{q})$ pour chaque séquence $\boldsymbol{p}$ de l'historique de référence $\mathcal{R}$. Cet historique est parcouru par la boucle commençant à la ligne 2. Pour chaque $\boldsymbol{p}$, la variable *v-rang* est obtenue par le cumul des effectifs des nœuds considérés. Chaque valeur de *v-rang* est insérée dans la

Algorithme 1 : Calcul de l'approximation du score CFOF de $\boldsymbol{q}$ dans un arbre iSAX

Data : La séquence $\boldsymbol{q}$, l'ensemble $\mathcal{R}$ des séquences de référence, la racine $\mathtt{N}_{racine}$ de l'arbre, les valeurs $\widetilde{\text{dist}}(\mathtt{N}_f, p)$ et $\tilde{\sigma}_{\mathtt{N}_f}(p)$, le paramètre CFOF ϱ

1 $l_{v\text{-}rang} \leftarrow \emptyset$
2 **forall** $\boldsymbol{p}$ *dans* $\mathcal{R}$ **do**
3 $\quad v\text{-}rang \leftarrow 0$
4 $\quad dist \leftarrow \mathtt{distance}(\boldsymbol{p}, \boldsymbol{q})$
5 $\quad liste_{\mathtt{N}} \leftarrow [\mathtt{N}_{racine}]$; `// nœuds restants à parcourir`
6 $\quad$ **while** $liste_{\mathtt{N}} \neq \emptyset$ **do**
7 $\quad\quad \mathtt{N}_{courant} \leftarrow liste_{\mathtt{N}}.pop()$
8 $\quad\quad$ **if** $\mathtt{minDist}(\boldsymbol{p}, \mathtt{N}_{courant}) > dist$ **then**
9 $\quad\quad\quad$ Ne rien faire, ignorer simplement $\mathtt{N}_{courant}$
10 $\quad\quad$ **else if** $\mathtt{maxDist}(\boldsymbol{p}, \mathtt{N}_{courant}) \leq dist$ **then**
11 $\quad\quad\quad v\text{-}rang \leftarrow v\text{-}rang + nbrObj(\mathtt{N}_{courant})$
12 $\quad\quad$ **else if** $\mathtt{N}_{courant}$ *est un nœud feuille* **then**
13 $\quad\quad\quad \tilde{\mu} \leftarrow \widetilde{\text{dist}}(\mathtt{N}_{courant}, \boldsymbol{p})$
14 $\quad\quad\quad \tilde{\sigma} \leftarrow \tilde{\sigma}_{\mathtt{N}_{courant}}(\boldsymbol{p})$
15 $\quad\quad\quad v\text{-}rang \leftarrow v\text{-}rang + F_{\tilde{\mu},\tilde{\sigma}}(dist) \times nbrObj(\mathtt{N}_{courant})$
16 $\quad\quad$ **else**
17 $\quad\quad\quad$ **forall** $\mathtt{N}$ *dans* $\mathtt{N}_{courant}.enfants$ **do**
18 $\quad\quad\quad\quad$ Insérer $\mathtt{N}$ dans $liste_{\mathtt{N}}$
19 $\quad\quad\quad$ **end**
20 $\quad\quad$ **end**
21 $\quad$ **end**
22 $\quad$ Insérer $v\text{-}rang$ dans $l_{v\text{-}rang}$ par ordre croissant
23 **end**
24 **return** $l_{v\text{-}rang}[\lceil \varrho \times |\mathcal{R}| \rceil]/|\mathcal{R}|$

liste triée $l_{v\text{-}rang}$, permettant de retourner la valeur de l'approximation de CFOF($\boldsymbol{q}$) (ligne 24).

Comme abordé dans la section 3.3.1.2, l'algorithme 1 parcourt l'arbre en commençant par le nœud racine (boucle interne commençant ligne 6). Le nœud parcouru est ôté de la liste de nœuds restants $liste_{\mathtt{N}}$ à la ligne 7. Mais si le nœud en cours d'analyse a une descendance, alors ils seront potentiellement candidats pour être insérés dans cette liste et être visités par la suite.

Pour tout nœud en cours d'analyse, l'algorithme exploite les deux bornes `minDist` et `maxDist` sur la distance entre $\boldsymbol{p}$ et les séquences contenues dans le

parallélotope droit représenté par le nœud courant. Ces bornes, représentant les distances extrêmes entre $\boldsymbol{p}$ et les séquences du nœud, permettent d'élaguer le sous-arbre, si l'un de ces deux critères est rempli :

1. (ligne 8) la distance minimale entre $\boldsymbol{p}$ et le nœud courant $\mathtt{N}_{courant}$ est strictement supérieure à la distance *dist* entre $\boldsymbol{p}$ et $\boldsymbol{q}$: dans ce cas le sous-arbre ne contient pas de séquences à compter dans $v\text{-}rang_{\boldsymbol{p}}(\boldsymbol{q})$ et l'exploration du sous-arbre peut alors être évitée de façon certaine,
2. (ligne 10) la distance maximale est inférieure à la distance *dist* : toutes les séquences du sous-arbre sont plus proches de $\boldsymbol{p}$ que ne l'est $\boldsymbol{q}$, il est donc possible de réaliser un second type d'élagage sûr, en comptant toutes ces séquences directement dans $v\text{-}rang_{\boldsymbol{p}}(\boldsymbol{q})$ sans parcourir le sous-arbre (cf. ligne 11, avec $nbrObj(\mathtt{N}_{courant})$ représentant le nombre de séquences du sous-arbre du nœud $\mathtt{N}_{courant}$).

Si aucune de ces deux conditions n'est remplie et que le nœud courant est un nœud feuille, nous approximons (ligne 15) la position de $\boldsymbol{q}$ dans le voisinage de $\boldsymbol{p}$ par rapport à toutes les autres séquences comprises dans le nœud et ajoutons la valeur obtenue à *v-rang*. Pour cela, la fonction de répartition $F_{\tilde{\mu},\tilde{\sigma}}$ (prise par défaut comme la fonction de répartition de la loi normale) retourne une approximation de la proportion des séquences incluses dans le nœud qui sont plus proches de $\boldsymbol{p}$ que ne l'est $\boldsymbol{q}$.

Enfin, si le nœud courant est un nœud interne de l'arbre pour lequel les possibilités d'élagage ci-dessus ne s'appliquent pas (ligne 16), alors ses nœuds enfants sont ajoutés à la liste des nœuds restant à visiter.

Nous venons de voir le fonctionnement de notre algorithme, quelle que soit la fonction de répartition utilisée. Nous allons maintenant présenter les approximations que nous avons choisies d'utiliser.

3.5.2 Description des approximations choisies

La fonction de répartition $F_{\tilde{\mu},\tilde{\sigma}}$ utilisée pour l'approximation peut être adaptée par l'utilisateur, selon la distribution des séquences, la précision souhaitée, ou même les potentielles connaissances d'experts sur le domaine d'application. Nous présentons dans cette section le choix que nous avons fait quant à cette fonction, en indiquant d'abord le calcul de la moyenne et de l'écart-type nécessaires.

Pour le paramètre $\tilde{\mu}$, c'est-à-dire la distance moyenne estimée $\widetilde{\mathrm{dist}}(\mathtt{N}, \boldsymbol{p})$ entre une séquence $\boldsymbol{p}$ et une séquence du nœud $\mathtt{N}$, nous utilisons la moyenne

quadratique (*Root Mean Square* ou RMS) suivante :

$$\widetilde{\text{dist}}(\mathtt{N}, \boldsymbol{p}) = \sqrt{\frac{1}{|\mathtt{N}_s|} \times (\sum_{r \in \mathtt{N}_s} |r - \boldsymbol{p}|^2)}$$

avec $\mathtt{N}_s$ dénotant ici l'ensemble des séquences contenues dans le nœud $\mathtt{N}$ lui-même. Dans notre contexte, l'avantage de cette expression, vis-à-vis de la moyenne classique, est qu'elle peut se dériver rapidement pour tout $\boldsymbol{p}$ à partir d'une simple valeur de dispersion pré-calculée et stockée au niveau du nœud $\mathtt{N}$. En effet, soit C le barycentre des séquences de $\mathtt{N}$ (en prenant une masse unitaire pour chaque séquence). Par le théorème de *Huygens* nous avons :

$$\sum_{r \in \mathtt{N}_s} |r - p|^2 = \sum_{r \in \mathtt{N}_s} |r - C|^2 + |\mathtt{N}_s| \times |C - p|^2$$

c'est-à-dire :

$$\widetilde{\text{dist}}(\mathtt{N}, p) = \sqrt{\frac{1}{|\mathtt{N}_s|} \times (\sum_{r \in \mathtt{N}_s} |r - C|^2 + |\mathtt{N}_s| \times |C - p|^2)}$$

Nous pouvons donc pré-calculer $\sum_{r \in \mathtt{N}_s} |r - C|^2$ pour chaque feuille de l'arbre iSAX et ensuite déterminer simplement $|C - \boldsymbol{p}|^2$ lorsqu'il est nécessaire d'obtenir la valeur de $\widetilde{\text{dist}}(\mathtt{N}, \boldsymbol{p})$.

Pour le paramètre $\tilde{\sigma}$, c'est-à-dire $\tilde{\sigma}_{\mathtt{N}}(\boldsymbol{p})$ l'écart-type estimé des distances entre la séquence $\boldsymbol{p}$ et les séquences du nœud $\mathtt{N}$, en considérant qu'en général $\boldsymbol{p}$ est loin de $\mathtt{N}$, nous avons choisi de prendre la moyenne des écarts-types monodimensionnels :

$$\tilde{\sigma}_{\mathtt{N}}(\boldsymbol{p}) = \frac{1}{|D|} \times \sum_{i \in D} \sqrt{\frac{1}{|\mathtt{N}_s|} \times \sum_{r \in \mathtt{N}_s} (r_i - C_i)^2}$$

avec D l'ensemble des dimensions, et r_i et C_i les composantes selon la dimension i de r et du barycentre C de N. On notera qu'avec ce choix, pour un nœud $\mathtt{N}$, la valeur de $\tilde{\sigma}_{\mathtt{N}}(\boldsymbol{p})$ est la même pour toute séquence $\boldsymbol{p}$. Ceci permet, comme pour $\widetilde{\text{dist}}(\mathtt{N}, \boldsymbol{p})$, d'adopter une stratégie de pré-calcul et de stockage au niveau de chaque nœud $\mathtt{N}$.

Nous verrons, dans la section suivante, que ces deux estimations retenues pour $\widetilde{\text{dist}}(\mathtt{N}, \boldsymbol{p})$ et $\tilde{\sigma}_{\mathtt{N}}(\boldsymbol{p})$ dans l'algorithme 1 fournissent des approximations de bonne qualité du score CFOF.

En ce qui concerne la fonction de répartition $F_{\tilde{\mu},\tilde{\sigma}}$ elle-même, nous faisons l'hypothèse d'une distribution des distances selon une loi normale. En

fonction du contexte d'application et des connaissances portant sur les processus qui ont généré les séquences, il est bien sûr possible d'utiliser d'autres paramètres et de faire des hypothèses de distributions différentes.

3.6 Expérimentations

Après avoir formulé les détails de notre méthode d'approximation du score CFOF, nous présentons dans cette section les résultats expérimentaux d'évaluation.

Tout d'abord, nous analysons et mesurons la précision de l'approximation sur l'ensemble de jeux artificiels *Clust2* proposé par [Angiulli, 2017]. Dans un second temps, nous évaluons la méthode sur les jeux de données du *Numenta Anomaly Benchmark* (NAB) proposé par [Lavin et Ahmad, 2015], et disponible sur GitHub [2]. Fournissant des informations sur les anomalies présentes dans les jeux de données, ce benchmark nous permet d'évaluer la pertinence des anomalies détectées par notre méthode dans un contexte de flux de données, c'est-à-dire en évaluant une nouvelle donnée en ne tenant compte que des données vues par le passé.

Pour ces deux familles de jeux de données (*Clust2* et NAB), nous évaluons aussi le gain obtenu grâce à l'utilisation de l'arbre d'indexation en terme de coût de calcul.

Les méthodes $\text{CFOF}_{\text{exact}}$ et $\text{CFOF}_{\text{approx}}$ sont disponibles en Python sur GitHub [3].

3.6.1 Évaluation sur les jeux de la famille *Clust2*

Dans cette section, les résultats obtenus à l'aide de notre méthode sont évalués au regard de ceux obtenus avec la méthode de calcul du CFOF exact proposée par [Angiulli, 2017, Angiulli, 2020].

Dans ces travaux, Anguilli utilise un jeu artificiel qu'il nomme *Clust2*, formé par deux groupes de données. Chaque groupe est distribué selon une loi normale et contient 5000 séquences chacun. Le premier est centré à l'origine et d'écart-type 1 sur toutes les dimensions, et le second centré sur $(4, \ldots, 4)$ et d'écart-type $(0.5, \ldots, 0.5)$. Nous générons un jeu *Clust2* pour des espaces de dimension 2, 5, 10, 20 et 50 afin de suivre l'évolution des résultats de notre méthode selon différentes tailles de séquences. Pour illustrer ces résultats, nous présentons dans la figure 3.2 un exemple de jeu *Clust2* en dimension 2. Ces séquences sont indexées dans un arbre *i*SAX avec un seuil de séparation

2. https://github.com/numenta/NAB
3. https://github.com/luk-f/pyCFOF & https://github.com/luk-f/pyCFOFiSAX

d'une nœud feuille égale à 30 (seuil d'effectif maximal d'une feuille avant sa séparation en deux nœuds).

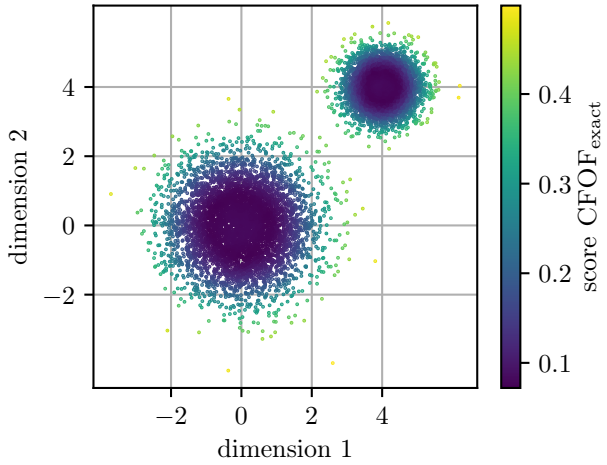


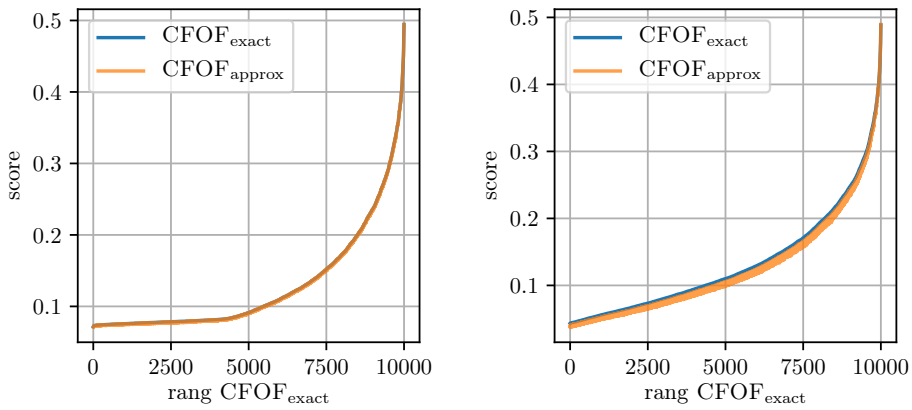
FIGURE 3.2 – Un exemple de jeu *Clust2* en 2 dimensions et les scores CFOF associés pour $\varrho = 0.1$.

Les figures 3.3 et 3.4 comparent les scores CFOF exacts et approximés. Sur chacune d'entre elles, tous les scores sont triés selon le rang des scores CFOF exacts obtenus. Cela permet de s'assurer que les scores de notre méthode discriminent les mêmes séquences que les scores CFOF exacts.

Nous observons que les deux scores sont très proches et que les classements sont très similaires. Ces résultats montrent également que les séquences anormales sont facilement séparables des autres séquences en utilisant le score CFOF exact ou CFOF approximé, quelle que soit la dimension des séquences.

Nous traçons en vert sur les deux figures 3.4b et 3.4d les scores CFOF obtenus en faisant l'hypothèse que les séquences sont uniformément réparties dans chacun des parallélotopes droits (et plus précisément dans chacun des nœuds feuilles de l'arbre). Dans ce cas de figure, les scores ne permettent pas de différencier correctement les séquences décrites comme normales et celle décrites comme anormales par le score CFOF. De plus, nous pouvons voir dans la figure 3.4d que les 500 dernières séquences ont presque les mêmes scores CFOF$_{\text{approx}}$, quel que soit leur rang.

Tout comme le propose [Angiulli, 2017] pour valider les résultats obtenus sur *fast*-CFOF, nous allons examiner la conservation du rang des séquences



(a) Dimension 2 (b) Dimension 5

FIGURE 3.3 – Les deux graphiques comparent les scores CFOF approximés et exacts pour des séquences de dimension 2 et 5.

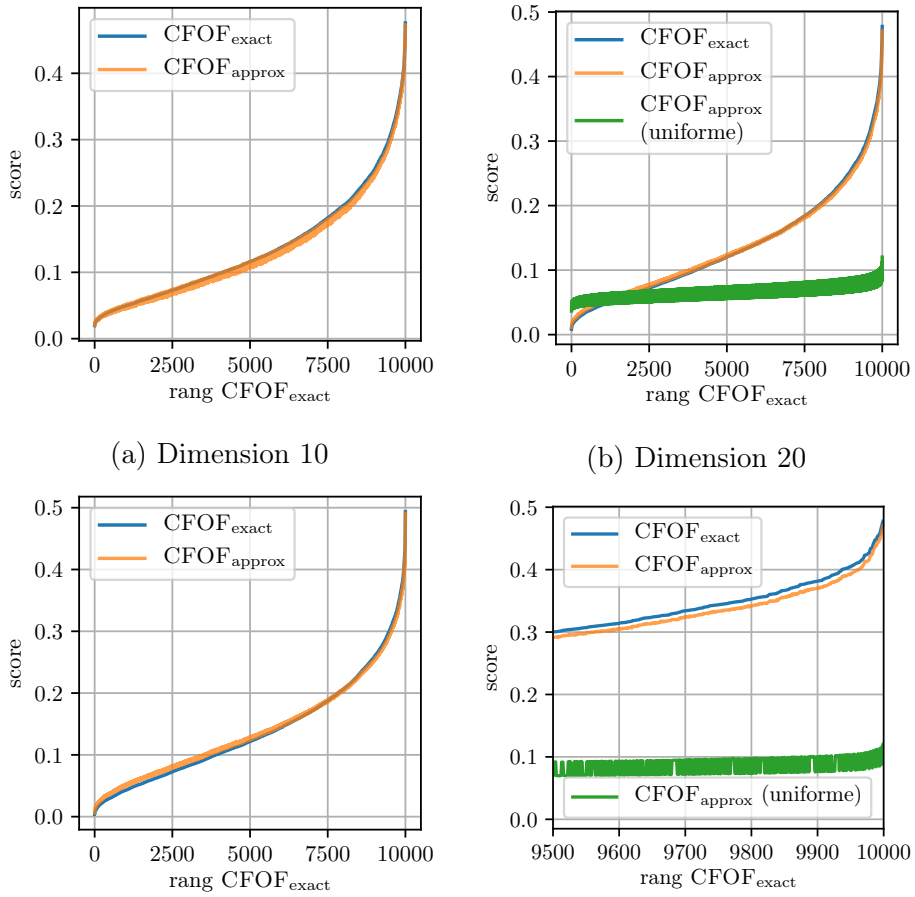
en utilisant comme indicateur la corrélation de SPEARMAN, qui est une mesure de dépendance statistique entre deux variables portant, non pas sur leurs valeurs, mais sur leurs rangs : le coefficient de SPEARMAN atteint un extremum lorsqu'une variable est une fonction monotone de l'autre.

3.6.1.1 Résultats corrélation de SPEARMAN

Cette comparaison des rangs des scores fournis par CFOF exact et approximé est réalisée pour les dimensions 2, 5, 10, 20 et 50 avec différentes valeurs du paramètre ϱ. Les valeurs choisies pour ce paramètre sont 0.005, 0.01, 0.05 et 0.1 et à chaque fois les séquences sont triées selon le score CFOF exact. Les résultats de corrélation de SPEARMAN obtenus sont présentés Table 3.1.

TABLE 3.1 – Corrélation de SPEARMAN entre les résultats des scores CFOF exacts et approximés sur le jeu de données *Clust2*.

ϱ / dimension	2D	5D	10D	20D	50D
0.005	0.876	0.957	0.990	0.997	0.999
0.010	0.939	0.985	0.995	0.999	0.999
0.050	0.996	0.999	0.999	0.999	0.999
0.100	0.999	0.999	0.999	0.999	0.999



(a) Dimension 10

(b) Dimension 20

(c) Dimension 50

(d) Dimension 20

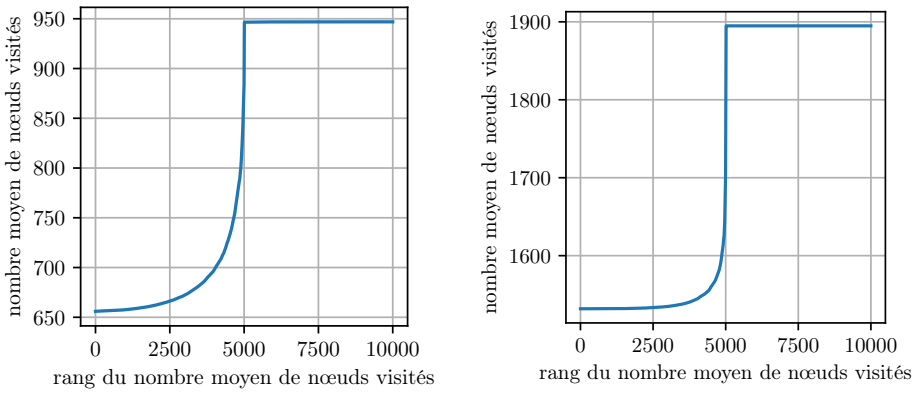
FIGURE 3.4 – Les trois premiers graphiques comparent les scores CFOF approximés et exacts pour des séquences de dimension 10, 20 et 50. Le dernier graphique montre les 500 scores les plus élevés en dimension 20.

Nous constatons que les valeurs de corrélation sont proches de 1, et se dégradent surtout lorsque la dimension des séquences et la valeur de ϱ diminuent. Ces résultats positifs montrent que les valeurs de corrélations tendent vers 1.0 lorsque la dimension augmente, ce qui est très encourageant car le principal atout de la mesure d'anomalie CFOF est de rester robuste en haute dimension.

3.6.1.2 Efficacité de l'élagage

Après avoir montré sur les jeux *Clust2* que le classement des séquences est respecté, et notamment pour les séquences en dimension élevée, nous

allons voir quel est le gain apporté par l'arbre d'indexation. Pour chaque séquence évaluée $\boldsymbol{q}$, il faut parcourir pour chaque séquence de référence $\boldsymbol{p}$ un certain nombre de nœuds dans l'arbre d'indexation. Nous définissons le nombre moyen de nœuds visités comme étant le nombre total de nœuds visités pour l'ensemble des séquences de référence divisé par le nombre de séquences de référence. La figure 3.5 présente le nombre moyen de nœuds visités pour chaque séquence évaluée pour les dimensions 20 et 50.



(a) Dimension 20 (b) Dimension 50

FIGURE 3.5 – Les deux graphiques indiquent, pour chaque séquence évaluée, le nombre moyen de nœuds visités, sur *Clust2* en dimension 20 et 50.

Pour les séquences de dimension 20, le nombre moyen de nœuds visités est compris entre 656 et 947 sur un nombre total de nœuds de 1414. Il est entre 1532 et 1895 pour les séquences en 50 dimensions pour un total de 2193 nœuds. Le nombre moyen de nœuds visités croît lorsque la dimension augmente.

De plus, nous observons que la moitié des séquences évaluées majore le nombre de nœuds visités. Cela s'explique par le fait qu'il y a dans *Clust2* deux groupes de données ayant des écarts-types différents. L'évaluation des séquences contenues dans le groupe ayant le plus petit écart-type impose aux séquences de référence de parcourir plus de nœuds contenant les séquences de ce même groupe pour approximer la position de la séquence à évaluer. Autrement dit, les parallélotopes droits de ce groupe de plus petit écart-type occupent un volume plus réduit, ce qui rend leur élagage plus difficile lorsque la séquence à évaluer et la séquence de référence proviennent toutes les deux de ce groupe.

En outre, le nombre moyen de nœuds visités reste au moins 10 fois inférieur au nombre de séquences totales (i.e., 10000) avec *Clust2* en dimension

20, et 5 fois en dimension 50, ce qui permet de limiter le coût de calcul.

3.6.2 Évaluations sur les jeux du benchmark NAB

Nous venons de montrer sur le jeu de données artificielles *Clust2* que notre méthode reste robuste en haute dimension, tout en limitant le coût de calcul. Nous allons maintenant valider son utilisation sur des données réelles, et l'évaluer sur des jeux de données plus hétérogènes et plus volumineux. En effet, bien que [Angiulli, 2020] ait déjà validé l'efficacité du score CFOF sur plusieurs jeux, nous souhaitons nous assurer que l'approximation que nous proposons reste pertinente pour la détection d'anomalies sur des types de jeux de données variés. Le *Numenta Anomaly Benchmark* (NAB), récemment proposé par [Lavin et Ahmad, 2015], est utilisé dans [Ahmad *et al.*, 2017] afin d'évaluer les méthodes de détection d'anomalies sur des flux de données. C'est sur ce jeu de données que nous proposons donc d'évaluer notre méthode.

3.6.2.1 Présentation du *Numenta Anomaly Benchmark*

Le benchmark NAB est composé de 58 séries temporelles regroupées en 7 familles distinctes. La majorité de ces séries proviennent de situations réelles. Seules 2 familles, préfixées par le terme “artificial”, ne le sont pas. Elles sont composées d'un total de 11 séries temporelles. Parmi les 58 séries temporelles, seules 6 séries ne contiennent pas d'anomalie et 5 d'entre elles sont regroupées dans la famille “artificialNoAnomaly”. Les 7 familles sont décrites brièvement dans le tableau 3.2

Pour chaque série temporelle contenant au moins une anomalie, la période anormale occupe 10% de la série, quelle que soit la taille de la série et le nombre d'anomalies. Il peut y avoir une ou plusieurs anomalies dans le jeu de données, et dans ce dernier cas ces 10% sont partagés en parts égales entre toutes les anomalies.

Une anomalie, définie à un instant t précis, occupe une fenêtre temporelle de longueur lg_w calculée à partir du nombre de données et d'anomalies comprises dans la série, comme expliqué précédemment. Sa fenêtre d'occupation débute à l'instant $t - lg_w/2$ et termine à l'instant $t + lg_w/2$. Les valeurs incluses dans tout cet intervalle sont donc considérées comme anormales, et si deux fenêtres d'anomalies différentes se chevauchent, elles fusionnent pour n'en former qu'une seule.

Comme les modèles d'apprentissage nécessitent d'avoir quelques valeurs non étiquetées de référence avant de pouvoir évaluer une nouvelle donnée, chaque série possède une période dite probatoire définie en fonction de la taille de la série. Cette période, composée des premières valeurs de la série, fournit

TABLE 3.2 – Description des différentes familles de séries proposées dans le NAB de Numenta. La dernière colonne précise le nombre de valeurs contenues dans chaque famille de séries, et la dernière ligne indique le total de ces valeurs.

NAB	Description	Quantité
artificialWithAnomaly	Données générées artificiellement	24192
artificialNoAnomaly	Données générées artificiellement	20160
realAdExchange	Taux de clic des annonces en ligne	9610
realAWSCloudwatch	Métriques du serveur AWS	67740
realKnownCause	Différentes types d'anomalies étiquetées manuellement	69561
realTraffic	Données de trafic	15664
realTweets	Nombre de mentions Twitter de diverses compagnies	158631
		365558

aux différents modèles une base de valeurs de référence pour l'évaluation des valeurs futures. Cette période probatoire ne compte pas dans l'évaluation des méthodes. Notons lg_{ts} la longueur de la série temporelle, celle de sa période probatoire lg_{pp} est définie dans le benchmark par :

$$lg_{pp} = \begin{cases} 0.15 \times lg_{ts} & \text{si } lg_{ts} < 5000 \\ 750 & \text{sinon} \end{cases} \tag{3.2}$$

La première valeur comptabilisée dans l'évaluation commence à l'indice $lg_{pp} + 1$. Comme l'évaluation s'effectue dans un contexte de flux de données, le modèle pourra l'évaluer en prenant en considération les lg_{pp} valeurs précédentes. De la même façon, pour toutes nouvelles valeurs à l'indice i tel que $i > lg_{pp}$, les modèles auront à leur disposition les valeurs d'indices j de la série tel que $j < i$. Ainsi, la série temporelle se voit attribuer un score à chacune des valeurs qui la compose, au fil de l'eau, en ne considérant que celles qui ont été vues antérieurement.

3.6.2.2 Création des séquences : prétraitement

Nous venons de voir la composition du *Numenta Anomaly Benchmark*. Avant de présenter les résultats d'évaluation, nous décrivons ici les paramètres choisis pour la méthode d'approximation du score CFOF à l'aide d'un arbre iSAX.

Nous analysons les séries temporelles en construisant une nouvelle séquence à chaque pas de temps, à l'aide d'une fenêtre glissante qui parcourt chronologiquement la série. La taille des séquences est définie individuellement pour chaque série temporelle, en tenant compte de la longueur de leurs périodes probatoires. Les séquences construites sont composées de plusieurs mesures, représentant chacune une statistique — appelée aussi indicateur — portant sur plusieurs valeurs — appelée aussi données — consécutives de la série temporelle. Nous utilisons principalement deux indicateurs dans ce travail, la moyenne et l'écart-type des valeurs sur la fenêtre temporelle. Nous avons également fait le choix d'utiliser des fenêtres glissantes de taille variable, cette taille augmentant lorsqu'on s'éloigne du présent : cela nous permet d'obtenir une précision plus fine sur les données les plus récentes (petite fenêtre), et de généraliser davantage les données plus anciennes avec une fenêtre plus grande, pour conserver des informations statistiques sur les valeurs passées.

Prenons lg_{pp} la taille de la période probatoire. La taille lg_{m_1} de la première mesure m_1 se calcule par $lg_{m_1} = \lfloor \lfloor lg_{pp}/2 \rfloor /2 \rfloor$, c'est-à-dire le résultat de la division par 2 de lg_{pp}, exécutée deux fois, et arrondi à l'entier inférieur. Ensuite, la taille lg_{m_2} pour le seconde mesure m_2 est l'arrondi à l'entier inférieur du résultat de la division de lg_{m_1} par 2 ($lg_{m_2} = \lfloor lg_{m_1}/2 \rfloor$). La longueur lg_{m_i} des prochaines mesures m_i se calcule aussi en tenant compte de la longueur de la mesure $i-1$ précédente ($lg_{m_i} = \lfloor lg_{m_{i-1}}/2 \rfloor$). On itère cette division jusqu'à ce que $lg_{m_i} \leq 5$.

Cette dernière mesure obtenue représente statistiquement les valeurs les plus récentes de la série temporelle. Pour un même indicateur, les mesures ne se chevauchent pas, et la séquence construite se voit attribuer l'horodatage de la valeur la plus récente présente dans la fenêtre glissante. Un exemple est présenté sur la figure 3.6. Dans le cas du benchmark NAB, et en fonction des séries considérées, nous obtenons un nombre de mesures par séquence valant 8, 10, ou 12 (ce qui représente aussi le nombre de dimensions des séquences).

La longueur de la séquence (la somme des longueurs des mesures) pour un indicateur j est notée $lg_{indicateur_j} = \sum^{i} lg_{m_i}$, la longueur totale de la séquence $lg_{sequence_f} = \sum^{j} lg_{indicateur_j}$.

Dans l'application au jeu de données NAB, une séquence à l'instant t est la concaténation des mesures générées par la fenêtre glissante pour les deux

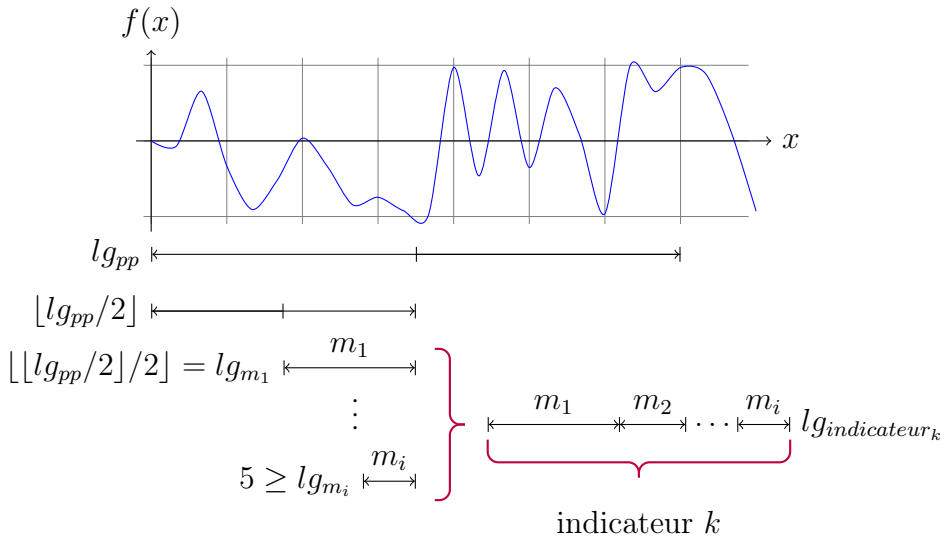


FIGURE 3.6 – Présentation de la création des séquences, pour un indicateur k, selon la taille de la période probatoire lg_{pp}.

indicateurs de moyenne et d'écart-type. Le résultat de cette concaténation est visible sur la figure 3.7.

Les étapes d'initialisation et de construction de l'arbre sont effectuées avec les $lg_{pp} - lg_{sequence_f}$ premières séquences.

Durant le processus d'évaluation, les séquences sont générées, évaluées et indexées au fur et à mesure du parcours de la fenêtre glissante sur la série temporelle. Chaque séquence générée se voit attribuer une approximation du score CFOF comme expliqué dans les sections 3.3 et 3.5. Les statistiques des nœuds de l'arbre portant sur les séquences insérées sont mises à jour à l'arrivée de chaque nouvelle séquence. L'arbre est détruit et entièrement reconstruit toutes les lg_{pp} nouvelles séquences insérées. La $i^{\text{ième}}$ initialisation/construction est effectuée avec les $lg_{pp} \times i - lg_{sequence_f}$ séquences générées par le passé.

Nous choisissons un seuil de séparation des nœuds feuilles égal à 10 lorsque le nombre de séquences insérées est inférieur à 3000 à l'initialisation de l'arbre, et de 20 sinon.

Tout d'abord, nous analysons la qualité de détection de notre méthode en comparant ses résultats avec les autres méthodes proposées par Numenta dans le benchmark NAB. Ensuite, nous évaluons la qualité de l'approximation et le gain obtenu par l'utilisation de l'arbre d'indexation iSAX, en terme de

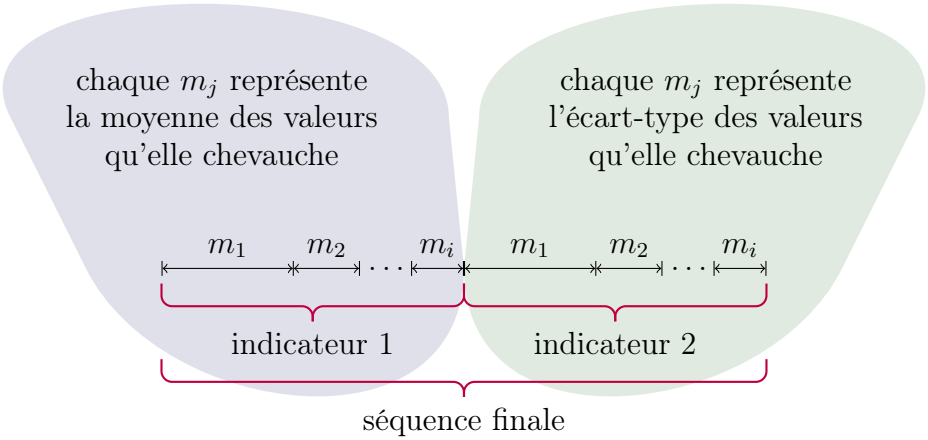


FIGURE 3.7 – Création des séquences sur les jeux du benchmark NAB avec les mesures de deux indicateurs différents : les moyennes, à gauche sur la séquence finale et les écarts-types, à droite. La dimension de la séquence est $2 \times i$.

coût de calcul par rapport au coût initial qu'aurait engendré la méthode CFOF classique de [Angiulli, 2020]. Enfin, nous évaluons le gain selon le seuil de séparation choisi, et selon la dimension des séquences.

3.6.2.3 Qualité de la détection

Pour connaître l'efficacité de la détection avec le score CFOF, nous utilisons l'indicateur ROC décrit dans la section 2.3.2. Sur les 58 séries temporelles disponibles dans le benchmark NAB, nous avons retiré les 6 séries ne contenant pas d'anomalie. En effet, l'évaluation ROC ne peut se faire s'il n'existe qu'une seule classe de données normales dans une même série.

Nous présentons sous forme de boîte à moustache dans la figure 3.8 les résultats du score ROC AUC des méthodes de détection proposées dans le benchmark NAB (indicateur présenté dans la section 2.2.3.1.a). Les résultats ROC AUC pour les scores CFOF exacts et approximés sont représentés pour chaque série par un triangle de couleur respectivement bleu ou rouge.

À première vue, la différence entre les résultats ROC AUC des scores CFOF exacts et approximés est faible. Mais nous discuterons plus en détail de leur similitude dans la section 3.6.2.4.

Nous constatons surtout que le score CFOF, qu'il soit exact ou approximé, se retrouve dans la partie haute du classement des scores ROC AUC pour une majorité de séries temporelles. Ils ne se trouvent qu'une

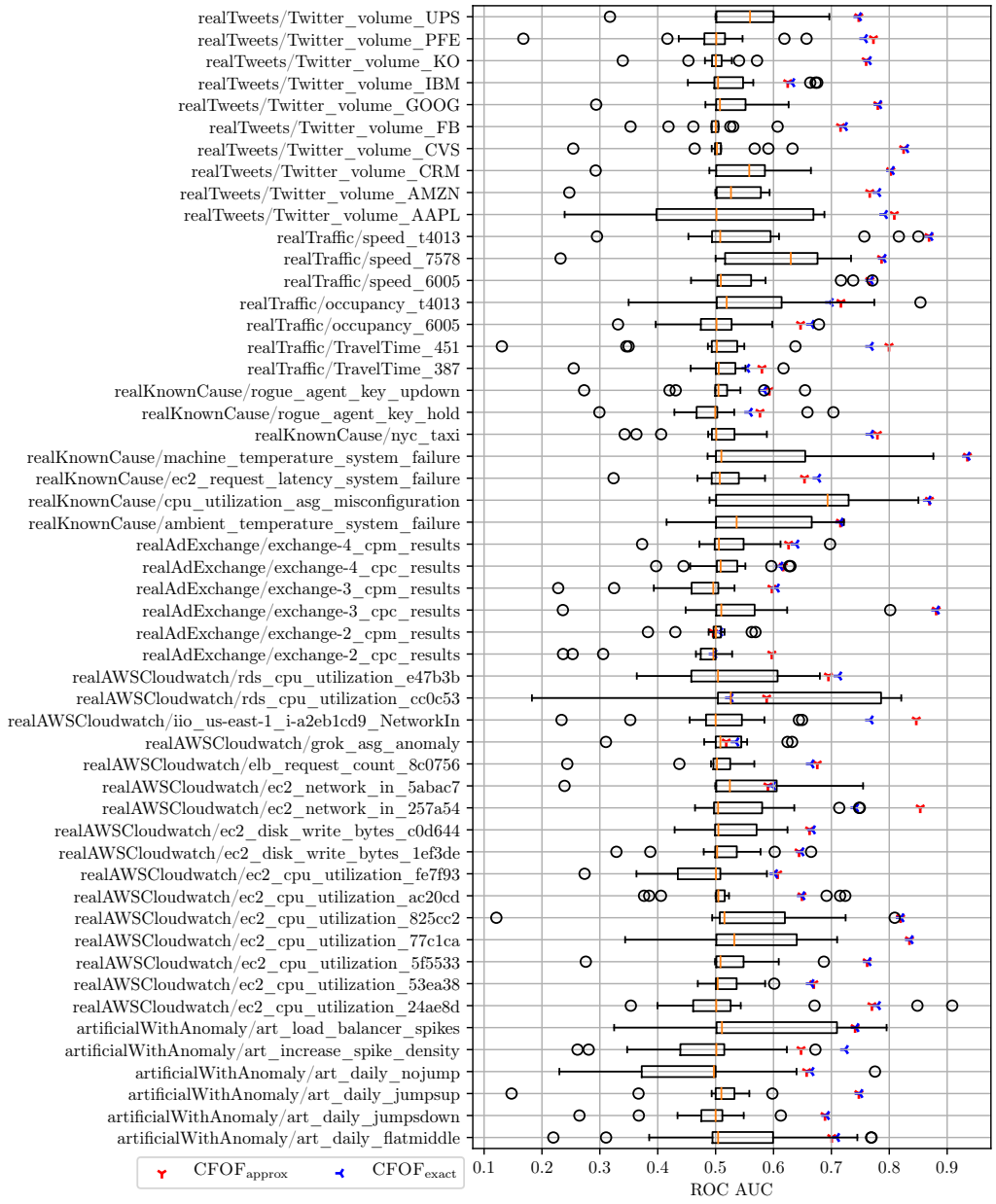


FIGURE 3.8 – Comparaison des résultats ROC AUC pour les scores CFOF exacts et approximés comparés aux scores des autres méthodes proposées dans le benchmark NAB.

seule fois sous la moyenne pour les deux scores, et qu'une seule fois sous la médiane pour le score CFOF$_{exact}$ ("realAWSCloudwatch/rds_cpu_utilization_cc0c53"). En observant les résultats pour la famille de séries tempo-

relles "realTweets", le score CFOF arrive en tête sur toutes les séries sauf une (troisième sur "Twitter_volume_IBM"). Les anomalies de cette famille étant généralement des fortes montées en charge des valeurs, le score CFOF les détecte sans trop de difficulté. De même, la forte périodicité "realKnownCause/nyc_taxi" explique la réussite du score CFOF.

TABLE 3.3 – Moyenne et écart-type des scores ROC AUC pour l'ensemble des méthodes de détection ainsi que leur classement.

méthode/ROC AUC	moyenne	écart-type	class.
bayesChangePt	0.503225	0.008663	14
$\text{CFOF}_{\text{approx}}$	**0.712945**	**0.100189**	1
$\text{CFOF}_{\text{exact}}$	0.708183	0.100501	2
contextOSE	0.347730	0.136428	17
earthgeckoSkyline	0.501772	0.002408	15
expose	0.562708	0.113633	6
htmjava	0.578539	0.138069	3
knncad	0.536839	0.092588	10
null	0.500000	0.000000	16
numenta	0.558001	0.127363	8
numentaTM	0.560009	0.146204	7
random	0.504226	0.016633	12
randomCutForest	0.573264	0.106666	4
relativeEntropy	0.504263	0.004539	11
skyline	0.564601	0.076908	5
twitterADVec	0.503917	0.003585	13
windowedGaussian	0.540573	0.126220	9

Le tableau 3.3 résume les scores ROC AUC pour chaque méthode de détection sur l'ensemble des 52 séries temporelles.

En comparant la moyenne des scores de chaque méthode, nous observons que la méthode CFOF approximé arrive largement en tête avec un peu plus de 0.13 d'avance sur le troisième, *htmjava* obtenant le meilleur score parmi les méthodes proposées dans NAB. De façon étonnante, le score CFOF approximé possède un ROC AUC plus élevé que celui du CFOF exact, mais seulement de 0.005.

Nous présentons ci-dessous les courbes ROC de trois séries temporelles pour lesquelles le score CFOF obtient des résultats divergents d'une série à

l'autre : "machine_temperature_system_failure", "nyc_taxi" et "exchange-2_cpm_results".

Pour la première série, le score CFOF approximé obtient son meilleur score ROC AUC, mais les écarts avec le second et le troisième sont serrés, comme nous pouvons le voir sur la figure 3.9.

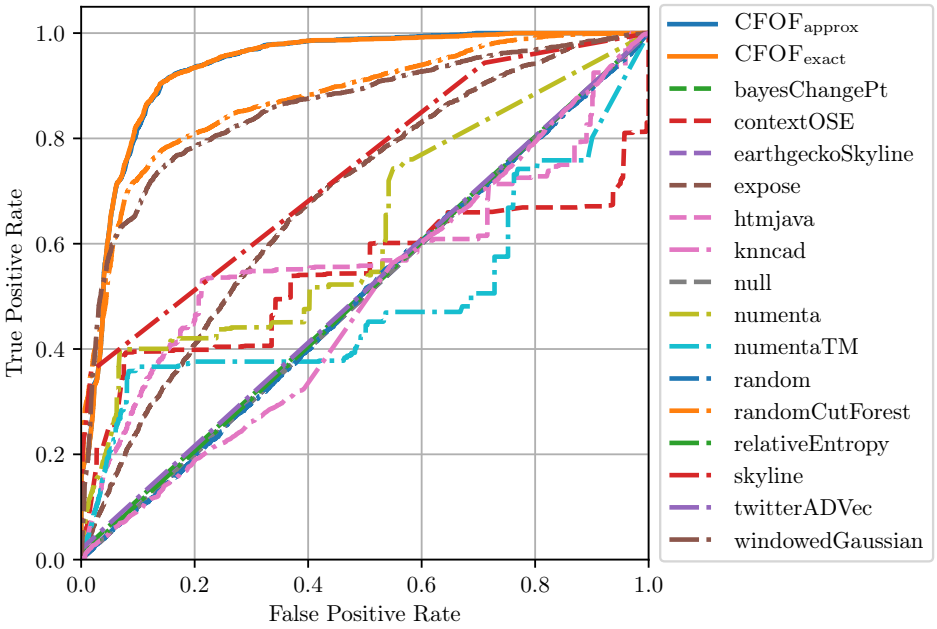


FIGURE 3.9 – Les courbes ROC pour la série "machine_temperature_system_failure". Les courbes CFOF exact et CFOF approximé se superposent.

Sur la seconde série, notre score obtient un score honorable, mais distance les scores ROC AUC des autres méthodes. Ce résultat est d'autant plus satisfaisant que cette série temporelle, "nyc_taxi", a de fortes ressemblances avec les données métiers SNCF que nous présenterons dans le chapitre 5. En effet, les données de cette série sont fortement périodiques et se répètent jour après jour. Les courbes ROC pour cette série sont visibles sur la figure 3.10.

Pour la dernière série temporelle, dont les courbes ROC sont présentées sur la figure 3.11, les scores CFOF exacts et CFOF approximés ne fournissent pas de très bons résultats ROC AUC, mais c'est également le cas pour les autres méthodes de détection où seule la méthode *knncad* se démarque légèrement.

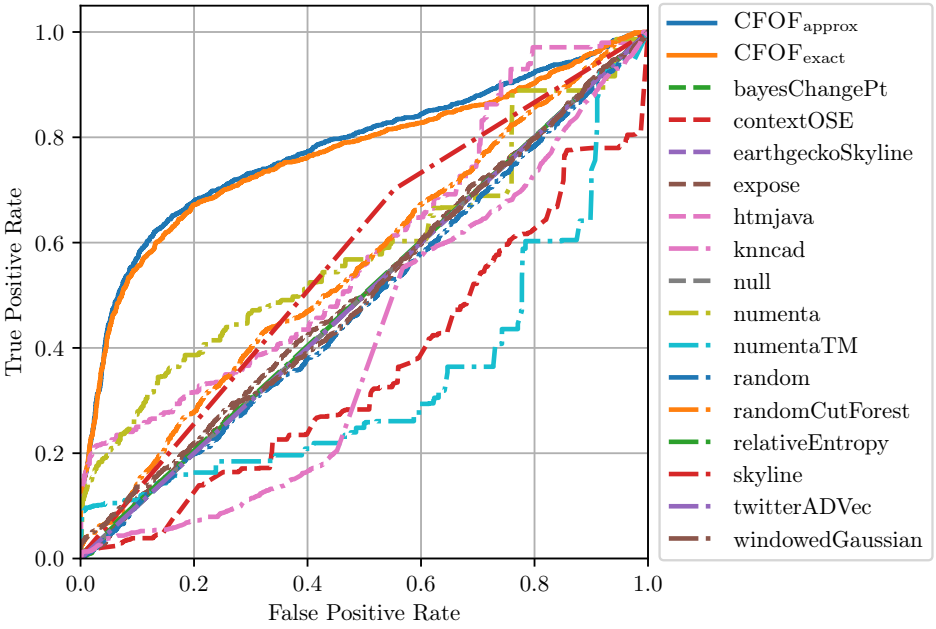


FIGURE 3.10 – Les courbes ROC pour la série “nyc_taxi”.

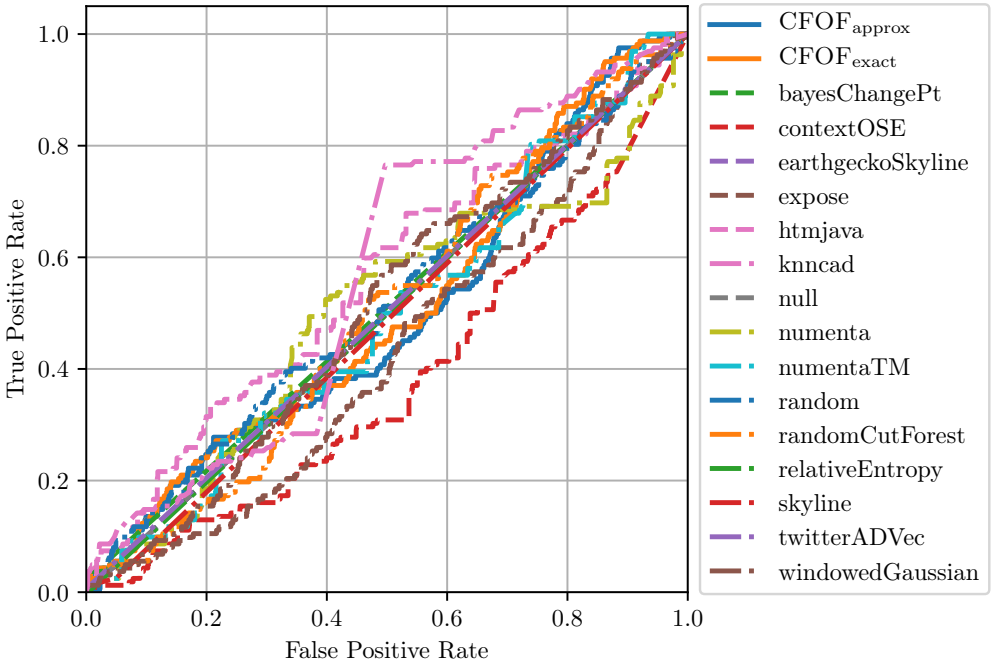


FIGURE 3.11 – Les courbes ROC pour la série “exchange-2_cpm_results”.

Le calcul de la courbe ROC et du score ROC AUC associé sont largement utilisés dans le domaine de la classification, tout comme dans celui de la détection d'anomalies. À leur lumière, les résultats obtenus ici confortent les capacités des scores CFOF exacts et approximés à effectuer de la détection efficace sur les différents types de séries temporelles. Cependant, comme décrit dans la section 2.3, plusieurs travaux dont ceux de [Saito et Rehmsmeier, 2015] préconisent l'utilisation d'indicateurs autres que ROC et ROC AUC lorsque le jeu de données est déséquilibré, ce qui est très souvent le cas pour la détection d'anomalies : les cas anormaux sont par définition bien moins nombreux que les cas normaux.

Les deux indicateurs recommandés par [Saito et Rehmsmeier, 2015] sont la courbe précision-rappel (PRC) et le calcul de l'AUC associé, abordés dans la section 2.3.3. C'est pourquoi nous présentons également les résultats pour ces deux indicateurs.

Nous présentons donc les résultats de l'AUC de PRC pour chacune des séries temporelles sur la figure 3.12. Les scores PRC AUC des mesures CFOF exactes et approximées sont indiqués par un triangle respectivement bleu ou rouge. Les distributions des scores PRC AUC des autres méthodes sont aussi présentées, pour chaque série temporelle, sous la forme de boîte à moustache.

Nous avons retiré des résultats la méthode *null* proposée parmi les méthodes implémentées dans le benchmark NAB, car le nombre de points générés pour le tracé de la courbe PRC n'était que de 2. Comme cette méthode fournit le même score 0.5 à tout instant sur chaque série temporelle, il est seulement possible de considérer tous les résultats comme positifs ou négatifs selon la position du seuil de détection. Dans [Saito et Rehmsmeier, 2015], il est précisé qu'il est fragile de faire de trop grandes interpolations pour la construction de la courbe PRC entre les différentes valeurs PRC, car cela fausse l'interprétabilité de cette dernière. Par exemple, nous obtenons seulement deux points de coordonnées $(1.0, 0.0)$ et $(0.0, 1.0)$ pour la méthode *null*, ce qui pourrait laisser penser que pour beaucoup de séries temporelles, la méthode *null* serait la meilleure méthode de détection. C'est pourquoi, pour chaque boîte à moustache, nous ne comptabilisons pas les méthodes dont les résultats PRC AUC sont définis sur moins de 10 points, ce qui en exclut complètement les méthodes *skyline*, *earthgeckoSkyline*, *relativeEntropy* et *twitterADVec*. Toutes les autres méthodes sont présentes pour toutes les séries, sauf *bayesChangePt* exclu de deux séries, et *contextOSE* exclu une seule fois.

Les résultats des scores PRC AUC pour les scores CFOF exacts et approximés, représentés par des triangles bleus et rouges, restent globalement meilleurs par rapport aux autres méthodes. De plus, en mettant en parallèle

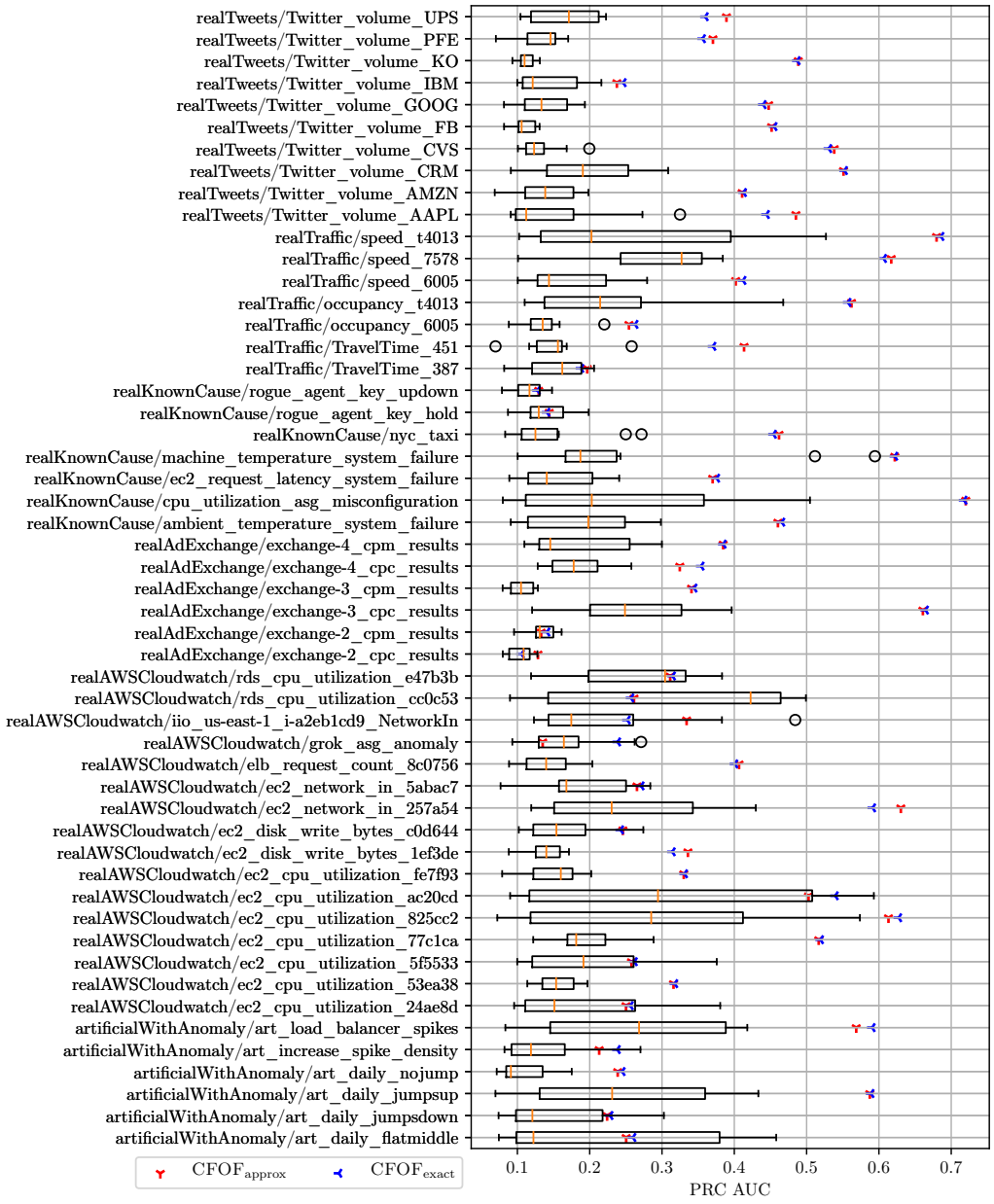


FIGURE 3.12 – Comparaison des résultats PRC AUC pour les scores CFOF exacts et approximés avec les autres méthodes proposées dans le benchmark NAB, présentés sous forme de boîte à moustache.

ces résultats PRC AUC avec ceux de la figure 3.8 présentant de la même manière les scores ROC AUC de la page 96, nous constatons que les scores CFOF sont bons sur les mêmes séries. Ils se retrouvent parfois près de la

moyenne des scores des autres méthodes pour quelques séries (ex. "exchange-2_cpm_results", "grok_asg_anomaly", ...), que ce soit avec PRC AUC ou ROC AUC. Ces mesures permettent de confirmer les résultats obtenus avec ROC AUC : les qualités des scores AUC pour les scores CFOF par rapport aux autres méthodes restent pratiquement inchangées de ROC à PRC.

Bien que les scores PRC AUC des approches CFOF exact et approximé aient les plus forts écarts-types, les résultats du tableau 3.4 montrent que la moyenne des scores pour les mesures CFOF est meilleure que les autres moyennes, devançant celle de *numentaTM* de 0.14 point. Les méthodes *htmjava*, *numenta*, *numentaTM* et *randomCutForest* confirment leur bonnes positions obtenues dans le tableau 3.3. Cependant, la méthode *expose* voit son score se dégrader par rapport à ses résultats avec le score ROC AUC.

TABLE 3.4 – Moyenne et écart-type des scores PRC AUC pour l'ensemble des méthodes de détection ainsi que leur classement. La dernière colonne indique, pour chaque méthode de détection, le nombre de séries temporelles ayant au moins 10 points sur la courbe PRC.

méthode / PRC AUC	moyenne	écart-type	class.	nb. points ≥ 10
baseline	0.113063	0.007496	13	0
bayesChangePt	0.129687	0.043641	10	50
$\text{CFOF}_{\text{approx}}$	**0.387886**	0.159344	1	52
$\text{CFOF}_{\text{exact}}$	0.387604	0.157213	2	52
contextOSE	0.121671	0.071690	11	51
expose	0.169448	0.076705	8	52
htmjava	0.246111	0.110690	4	52
knncad	0.136964	0.038967	9	52
numenta	0.235370	0.103608	5	52
numentaTM	0.246172	0.117930	3	52
random	0.116469	0.011753	12	52
randomCutForest	0.221088	0.115606	6	52
windowedGaussian	0.203955	0.118911	7	52

Nous présentons ci-dessous le détail de trois des courbes PRC. Comme préconisé par [Saito et Rehmsmeier, 2015], nous avons rajouté la droite représentant la *baseline* calculée selon la répartition des données dans chaque classe (proche de 10%, mais sans l'atteindre car nous ne comptabilisons pas les périodes probatoires). Nous observons d'ailleurs que la courbe PRC de la méthode *random*, qui consiste à attribuer un score aléatoire tout au long de

la série, est très similaire à cette droite, et ce pour toutes les séries. Toutes les méthodes se trouvent en moyenne au-dessus de cette *baseline*.

Comme les scores PRC AUC sont très similaires avec ceux de ROC AUC, nous avons choisi de reprendre les mêmes trois séries que précédemment : "machine_temperature_system_failure", "nyc_taxi" et "exchange-2_cpm_results", représentatives de trois comportements dans les performances obtenus.

Les courbes de la première série "machine_temperature_system_failure" sont présentées sur la figure 3.13. Bien que le score PRC AUC pour cette série ne soit plus le meilleur score de la mesure CFOF approximée, il reste tout de même un des rares scores au-dessus de 0.6.

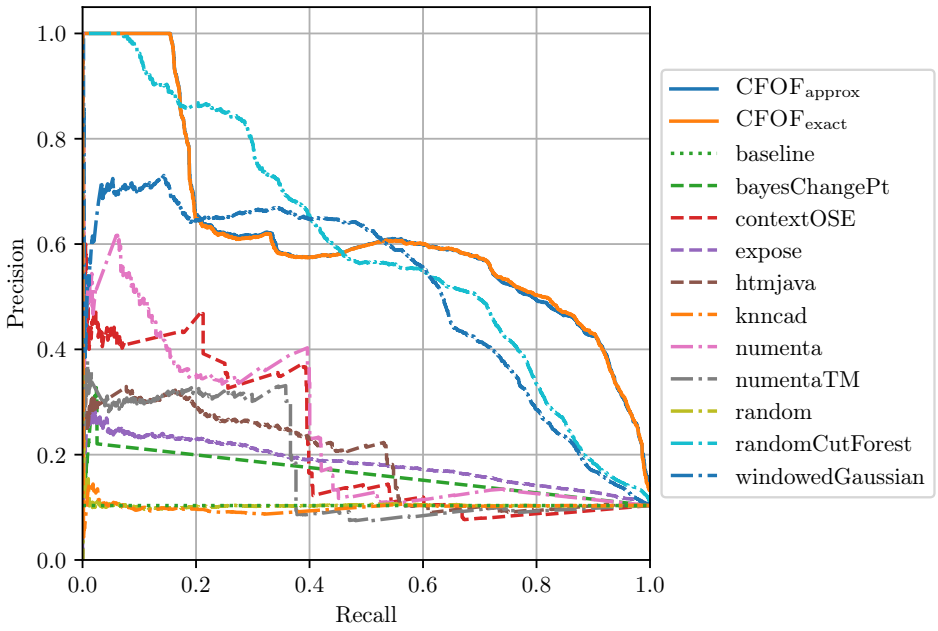


FIGURE 3.13 – Les courbes PRC pour la série "machine_temperature_system_failure". Sur ce jeu de données, les courbes des scores CFOF exacts et approximés se superposent.

Pour la série "nyc_taxi", les courbes PRC pour les scores CFOF exacts et approximés restent toujours au-dessus des autres méthodes. Seule la méthode *htmjava* arrive à légèrement concurrencer la mesure CFOF aux alentours d'une valeur de rappel à 0.18. Les résultats des courbes sont visibles sur la figure 3.14. De nombreuses méthodes tendent rapidement vers la *baseline*. Les scores CFOF exacts et approximés continuent de dominer pour cette série,

dont la forte périodicité fait écho à celle des données industrielles SNCF que nous verrons au chapitre 5.

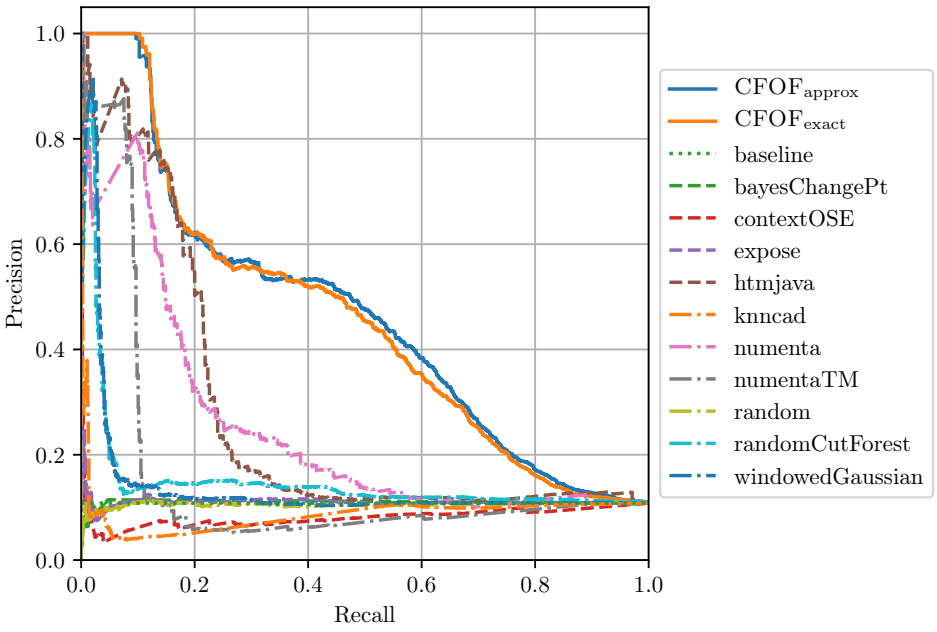


FIGURE 3.14 – Les courbes PRC pour la série "nyc_taxi".

Pour la dernière série analysée "exchange-2_cpm_results", dont les résultats des courbes PRC sont présentés sur la figure 3.15, toutes les méthodes de détection semblent faire défaut. Elles tendent toutes rapidement vers la courbe *baseline*, ce qui signifie qu'elles ne font pas mieux, pour cette série, qu'une méthode de détection aléatoire.

Comme nous pouvons le constater avec les résultats obtenus sur les deux indicateurs, ROC et PRC, notre méthode de calcul de score CFOF approché donne de très bons résultats pour la détection d'anomalies sur un flux de données lorsqu'elle est comparée aux autres. C'est aussi le cas du score CFOF exact, si nous faisons abstraction du temps nécessaire à son calcul, qui le rend difficilement utilisable en pratique sur un flux de données. Vu l'hétérogénéité des séries temporelles fournies par le benchmark NAB, notre méthode est évidemment plus efficace sur certaines séries que sur d'autres. Elle reste cependant très compétitive sur une majorité de séries par rapport aux méthodes de détection proposées dans le benchmark, et garde la tête du classement que ce soit à la lumière du score ROC AUC ou du score ROC PRC.

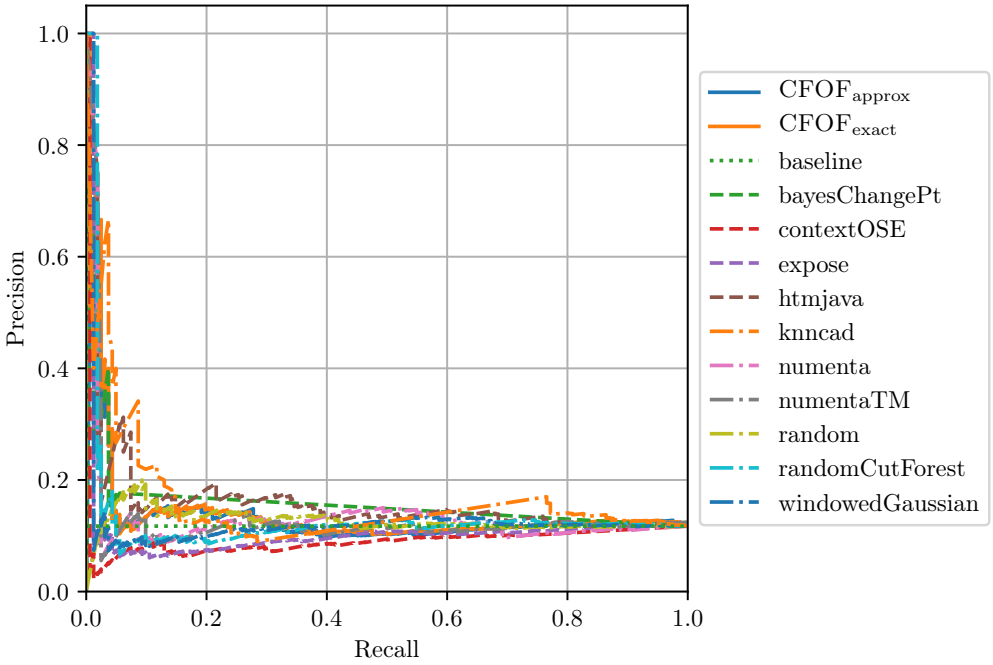


FIGURE 3.15 – Les courbes PRC pour la série "exchange-2_cpm_results".

3.6.2.4 Qualité de l'approximation

Comme nous avons pu le voir, les scores CFOF exacts et approximés sont applicables pour la détection d'anomalies sur les séries temporelles proposées dans le benchmark NAB, et sont très compétitifs par rapport aux autres méthodes proposées.

Maintenant, nous allons comparer la qualité de l'approximation par rapport aux scores CFOF exacts. En effet, comme abordé précédemment dans la section 3.5, le score obtenu à l'aide de l'arbre d'indexation n'est qu'une approximation du score CFOF exact. Lors du calcul de la position d'une séquence dans les voisinages des autres séquences — lors du calcul du *v-rang* — nous utilisons des approximations afin de ne pas avoir à calculer les distances entre toutes les séquences.

Dans la figure 3.16, nous présentons les résultats de corrélation de SPEARMAN entre les scores CFOF exacts et approximés, pour chacune des séries temporelles du benchmark NAB. A chaque fois, les séquences sont triées selon le score CFOF exact. Nous obtenons une moyenne de corrélation de 0.9600 et un écart-type de 0.0962.

Sur l'ensemble des 57 séries temporelles, seule la série "artificialNoAnomaly/art_flatline" a été retirée car toutes les séquences créées pour l'analyse

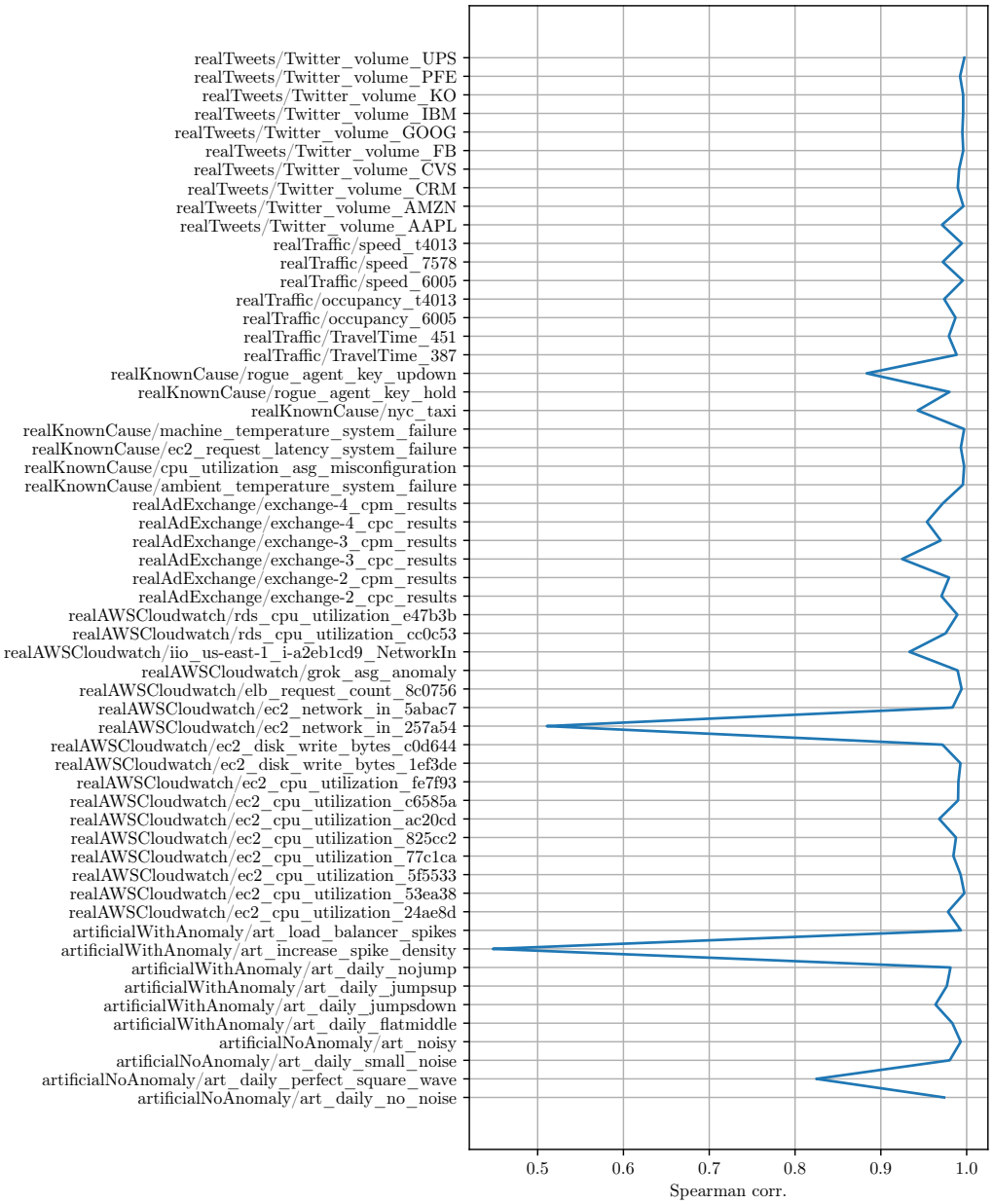


FIGURE 3.16 – Résultats de la corrélation de SPEARMAN entre les scores CFOF exacts et approximés, pour toutes les séries temporelles incluses dans le benchmark NAB.

sont égales, et il n'y a pas de réel classement possible. Seules 4 séries se retrouvent avec une corrélation de SPEARMAN strictement inférieure à 0.9.

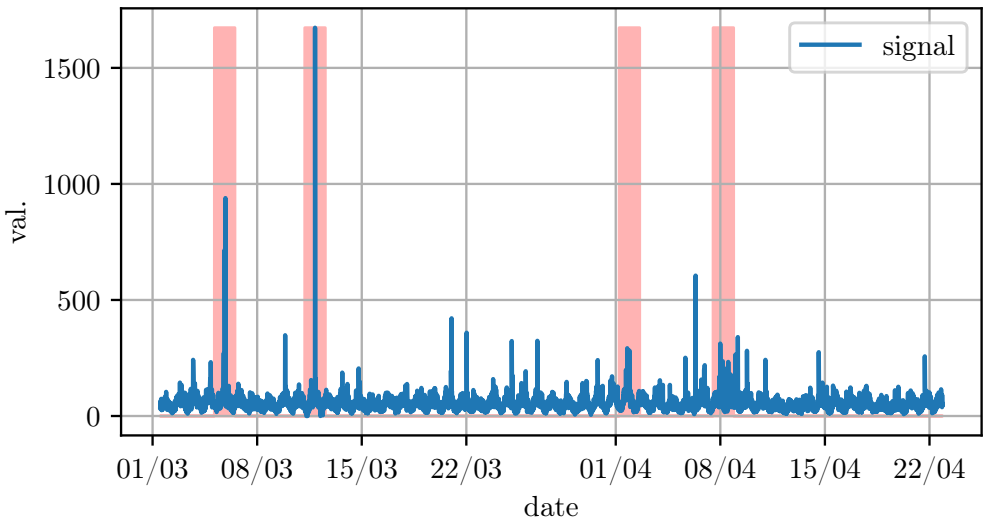
Nous avons choisi trois séries pour lesquelles nous analysons en détail la

différence entre le score CFOF exact et le score CFOF approximé : "Twitter_volume_AMZN", "signal_art_increase_spike_density" et "rogue_agent_key_updown".

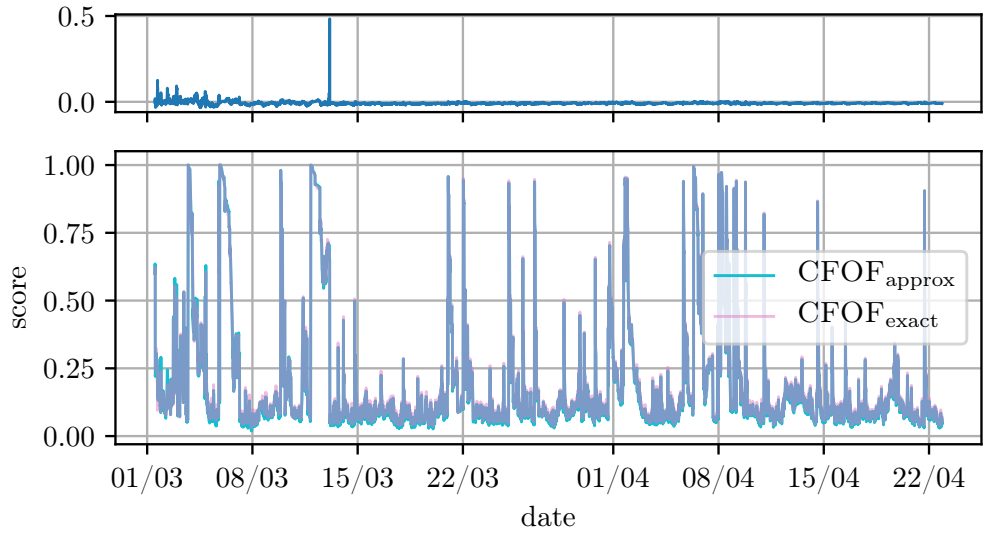
La série "Twitter_volume_AMZN" est celle pour laquelle la corrélation de SPEARMAN est la meilleure, avec un score de 0.9962. Le signal de la série est visible sur la figure 3.17a (page 108). Les quatre zones rouges représentent les zones anormales à détecter. Les scores CFOF exacts et approximés sont représentés sur la figure 3.17b. La courbe en haut sur cette figure montre la différence entre les deux scores en fonction de l'horodatage. La différence observée entre les deux scores est très faible. Le pic survenant le 13 mars est dû à une chute du score CFOF exact juste 5 minutes avant celui du score CFOF approximé, c'est-à-dire à une valeur près.

La deuxième série analysée, présentée sur la figure 3.18a (page 109), présente une corrélation de seulement 0.4479, ce qui est le plus mauvais résultat obtenu par le score approximé. Cependant, en observant sur la figure 3.18b la différence entre le score CFOF exact et approximé, nous constatons que durant la période anormale les deux scores sont très proches. Les divergences de scores se faisant surtout pour des scores CFOF faibles. Nous pouvons noter une différence de score d'environ 0.41 au tout début de la série, dû au fait que le score $\text{CFOF}_{\text{exact}}$ chute une mesure plus tôt que le score $\text{CFOF}_{\text{approx}}$.

Enfin, la dernière série analysée ("rogue_agent_key_updown") visible sur la figure 3.19a (page 110), la corrélation de SPEARMAN est de 0.8835 soit la 4$^{\text{e}}$ plus mauvaise corrélation obtenue. En effet, pour le signal "rogue_agent_key_updown", les scores CFOF exacts et approximés, présentés sur la figure 3.19b, subissent quelques écarts d'approximation (notamment les 11 et 14 juillet). Toutefois, les informations de détection apportées par des deux scores concordent, les ordres de grandeur des deux scores restent similaires, et les périodes détectées comme anormales se chevauchent.

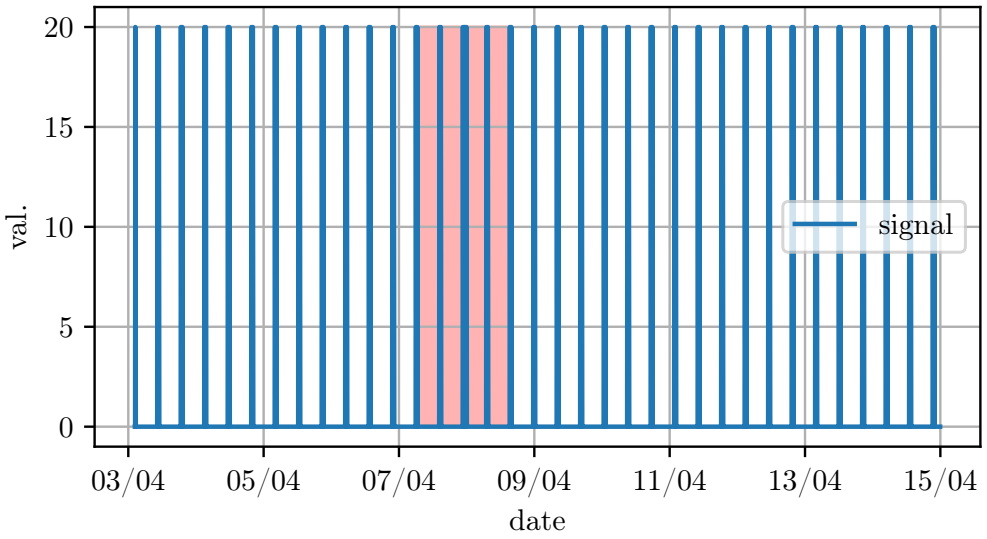


(a) Signal de la série "Twitter_volume_AMZN". Les zones en rouge indiquent les périodes anormales.

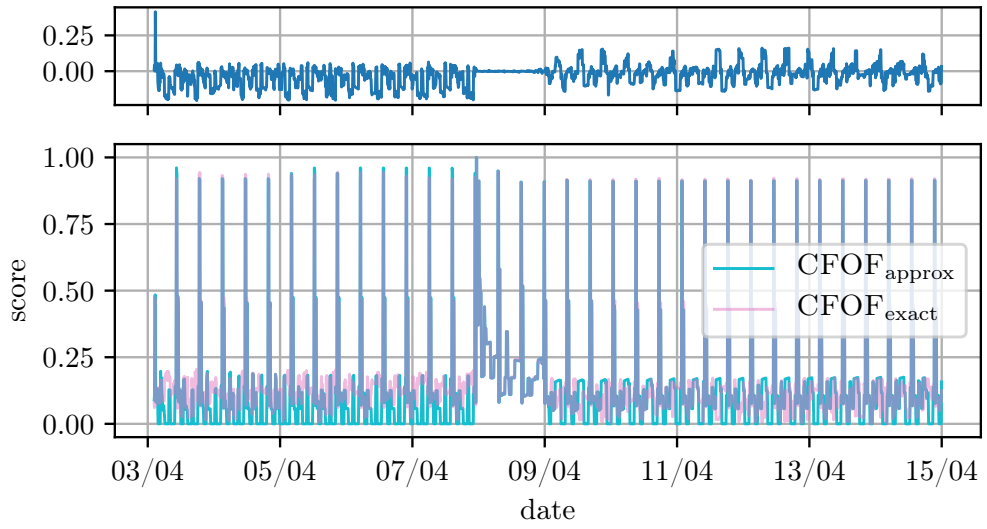


(b) Comparaison des scores CFOF exacts et approximés. Différence entre les scores sur la courbe du haut.

FIGURE 3.17 – Présentation des détections CFOF sur la série "Twitter_volume_AMZN". Le signal est présenté sur la figure (a) et les scores sont comparés sur la figure (b).

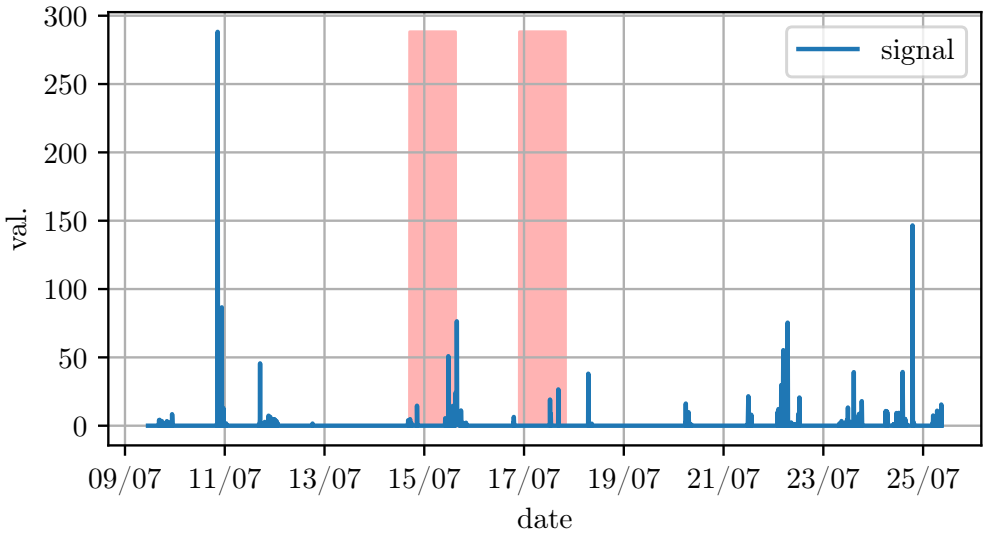


(a) Signal de la série "art_increase_spike_density". La zone en rouge indique la période anormale.

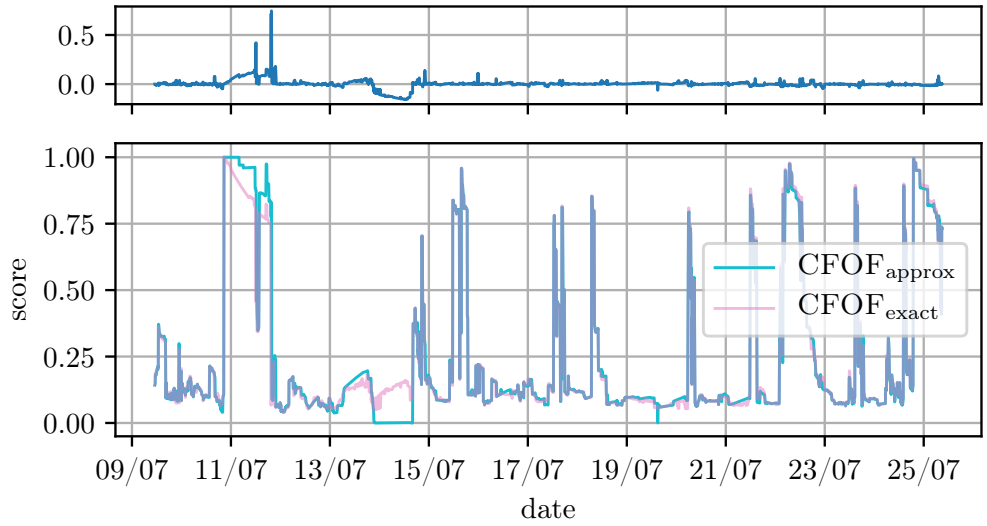


(b) Comparaison des scores CFOF exacts et approximés. Différence entre les scores sur la courbe du haut.

FIGURE 3.18 – Présentation des détections CFOF sur la série "art_increase_spike_density". Le signal est présenté sur la figure (a) et les scores sont comparés sur la figure (b).



(a) Signal de la série “rogue_agent_key_updown”. Les zones en rouge indiquent les périodes anormales.



(b) Comparaison des scores CFOF exacts et approximés. Différence entre les scores sur la courbe du haut.

FIGURE 3.19 – Présentation des détections CFOF sur la série “rogue_agent_key_updown”. Le signal est présenté sur la figure (a) et les scores sont comparés sur la figure (b).

3.6.2.5 Gain sur le coût de calcul

Les résultats de la corrélation de Spearman nous confortent sur le fait que l'approximation du score CFOF reste très pertinente pour remplacer le score exact, et permet donc de réduire le coût de calcul du score pour pouvoir l'utiliser dans un contexte de flux de données.

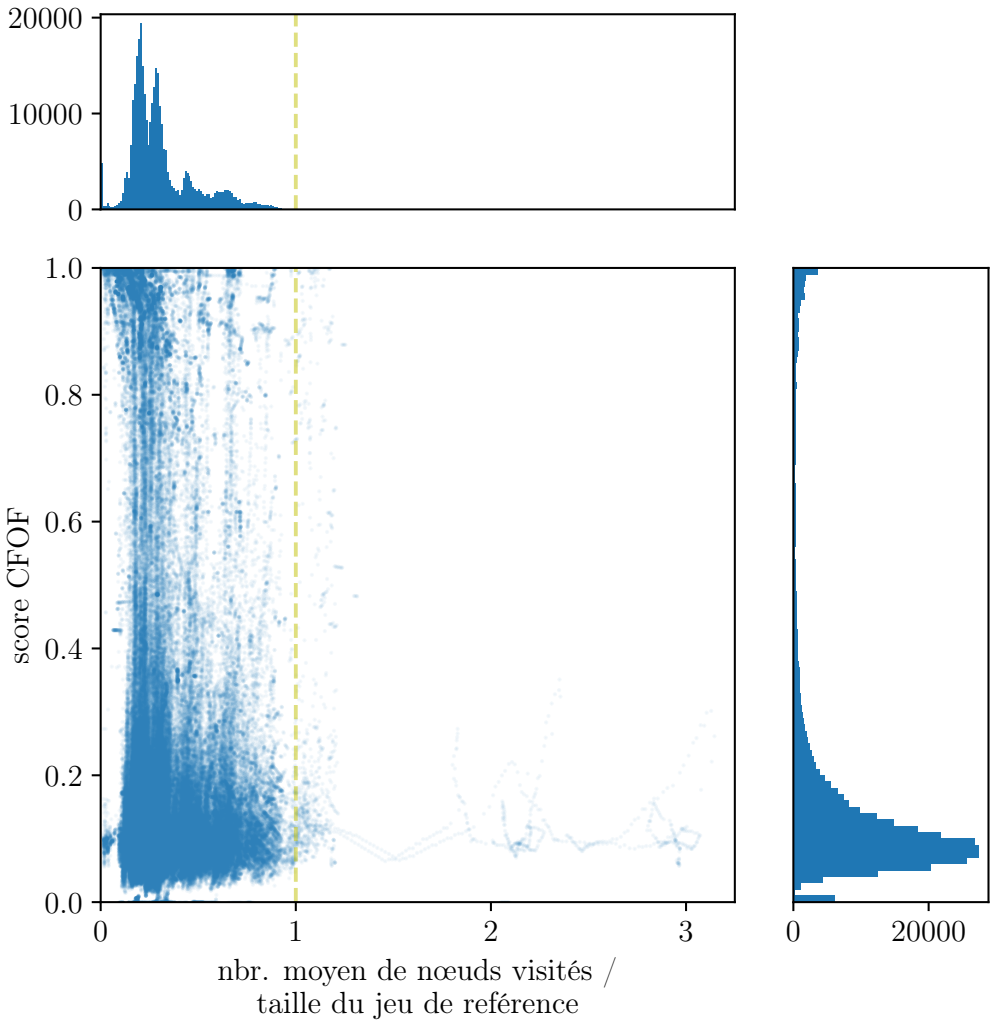


Figure 3.20 – Représentation, pour chaque séquence évaluée, du score $\text{CFOF}_{\text{approx}}$ selon le ratio entre le nombre moyen de nœuds visités et la taille du jeu de référence.

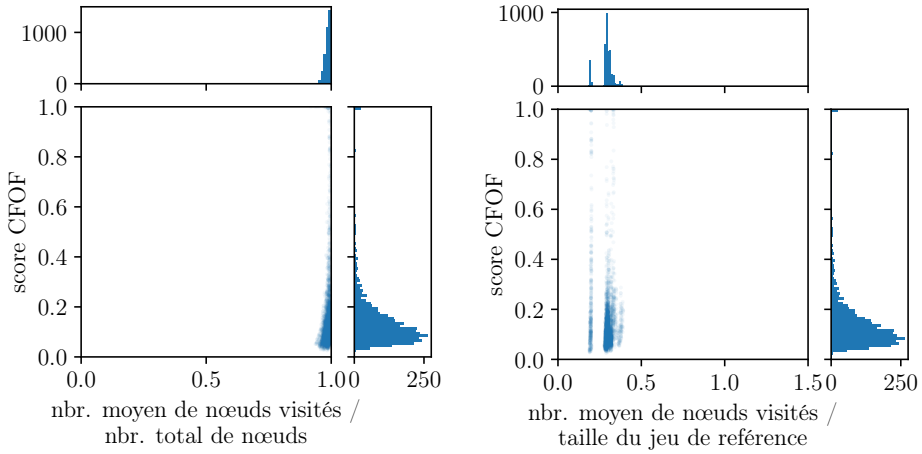
Nous analysons dans la figure 3.20 le ratio entre le nombre moyen de nœuds visités et la taille du jeu de référence. Chaque point sur la figure représente une séquence évaluée selon son score $\text{CFOF}_{\text{approx}}$ en ordonnée. La

valeur en abscisse représente la moyenne du nombre de nœuds visités pour l'approximation du score CFOF pour chacune des séquences de référence utilisées.

Lorsque l'approximation de ce score nécessite en moyenne un ratio inférieur à 1.0, cela signifie que la structure d'indexation permet un gain à la hauteur de ce ratio : ce gain vaut 1 moins le ratio, en pourcentage.

Notons d'ores et déjà que très peu de séquences ont un ratio supérieur à 1.0, mais il en existe (seulement 0.443%). Nous verrons plus en détail les cinq séries concernées par ce mauvais ratio dans la section 3.6.2.6. Il est aussi intéressant de constater que lorsque le score $\text{CFOF}_{\text{approx}}$ obtenu est proche de 1, le ratio se concentre vers 0. Ce constat est particulièrement intéressant dans notre contexte de flux de données, pour lequel notre contrainte est de détecter une anomalie le plus rapidement possible. Majoritairement, le ratio est plus faible que 0.5, ce qui donne un gain supérieur à 50%.

Nous analysons plus en détail ce ratio sur trois séries, en présentant notamment un second ratio : le nombre moyen de nœuds parcourus par rapport au nombre total de nœuds de la structure *i*SAX. Ce nouveau ratio indique donc l'efficacité de l'élagage effectué lors du parcours de l'arbre.



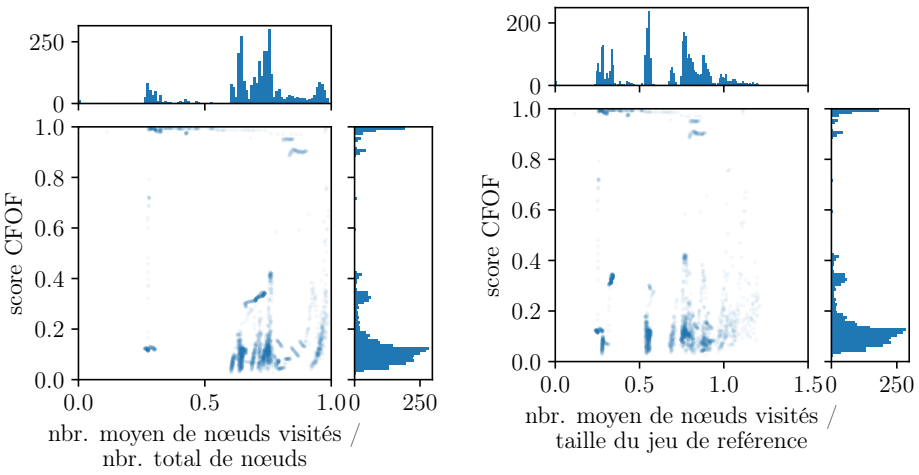
(a) Ratio entre le nombre moyen de nœuds visités et le nombre total de nœuds

(b) Ratio entre le nombre moyen de nœuds visités et la taille du jeu de référence

FIGURE 3.21 – Analyse des gains selon le score $\text{CFOF}_{\text{approx}}$ pour la série "elb_request_count_8c0756" : (a) efficacité de l'élagage et (b) gain de calcul.

La première série analysée est "elb_request_count_8c0756". Nous observons dans la figure 3.21a que les séquences de référence visitent quasiment

l'intégralité des nœuds qui constituent l'arbre. Cependant, comme le montre la figure 3.21b, le nombre de nœuds est de moitié plus petit que le nombre de séquences de références, ce qui permet de diviser au moins par 2 le coût de l'approximation du score CFOF des séquences.



(a) Ratio nombre moyen de nœuds visités et nombre total de nœuds

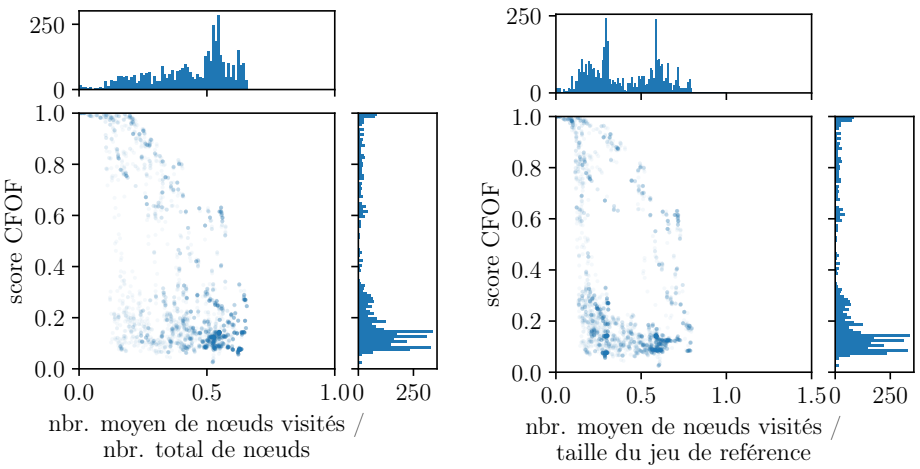
(b) Ratio entre le nombre moyen de nœuds visités et la taille du jeu de référence

FIGURE 3.22 – Analyse des gains selon le score $\text{CFOF}_{\text{approx}}$ pour la série "rds_cpu_utilization_e47b3b" : (a) efficacité de l'élagage et (b) gain de calcul.

Pour la seconde série, "rds_cpu_utilization_e47b3b", l'algorithme 1 permet d'élaguer environ 30% des nœuds lors du parcours de l'arbre d'indexation. Comme présenté sur la figure 3.22a, le ratio du nombre moyen de nœuds visités se situe principalement entre 0.5 et 1.0.

Toutefois, les gains obtenus (égaux à 1 moins le ratio du nombre moyen de nœuds parcourus par rapport à la taille du jeu de référence) comme présenté sur la figure 3.22b, restent globalement moins bons que ceux de la série précédente. On observe même un surcoût (ratio plus grand que 1), mais qui survient pour des scores CFOF plutôt faibles.

Enfin, pour la dernière série analysée ("ec2_disk_write_bytes_1ef3de"), l'algorithme 1 élague en moyenne 57% des nœuds. Les valeurs présentées sur la figure 3.23b montrent que le gain est toujours positif, et la figure 3.23a indique que la structure d'indexation réduit fortement le nombre moyen de nœuds parcourus pour les séquences ayant un score CFOF élevé. Cette série



(a) Ratio entre le nombre moyen de nœuds visités et le nombre total de nœuds

(b) Ratio entre le nombre moyen de nœuds visités et la taille du jeu de référence

FIGURE 3.23 – Analyse des gains selon le score $\text{CFOF}_{\text{approx}}$ pour la série "ec2_disk_write_bytes_1ef3de" : (a) efficacité de l'élagage et (b) gain de calcul.

est donc un exemple de jeu de données pour lequel on bénéficie pleinement de l'élagage implémenté par notre méthode.

En observant ces résultats, nous constatons des gains certains liés au processus d'approximation et d'élagage. Ces gains varient d'une série à une autre, ainsi que pour les séquences d'une même série. Un faible nombre d'approximations semble plus coûteux que les calculs exacts. Ces cas restent néanmoins peu nombreux dans nos expérimentations, et une analyse des différentes séries semble montrer que seules les séquences ayant un score faible sont touchées par ce phénomène. Ils nécessitent toutefois une analyse plus détaillée pour les caractériser plus précisément.

Pour cela, dans un premier temps, nous allons analyser les effets du changement de seuil d'occupation maximale des nœuds feuilles sur les gains, puis nous considérerons dans un second temps l'impact de la dimension des séquences.

3.6.2.6 Gain selon structure des arbres d'indexation : manipulation du seuil

Pour rappel, l'arbre d'indexation iSAX nécessite de définir un seuil sur le nombre de séquences à partir duquel un nœud feuille se scinde en deux. Ce seuil est défini pour la construction de l'arbre et ne change pas par la suite. Nous définissons ce seuil lors de l'initialisation de l'arbre : si le nombre de séquences à indexer est inférieur à 3000 le seuil est de 10, sinon il est défini à 20.

Nous présentons donc dans cette section de nouveaux résultats, pour lesquels le seuil a été fixé à 30, quel que soit le nombre de séquences à indexer, et nous comparons ces nouveaux résultats avec les précédents.

Nous présentons dans la figure 3.24 la distribution des ratios du nombre moyen de nœuds visités par rapport à la taille du jeu de référence, selon le seuil choisi pour la construction de l'arbre.

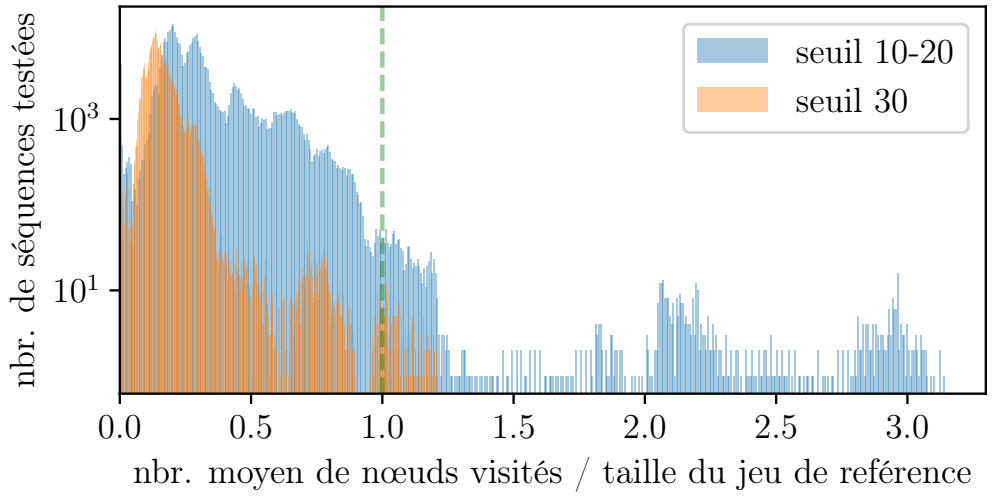


FIGURE 3.24 – Comparaison pour deux seuils différents de la distribution du ratio du nombre moyen de nœuds visités par rapport à la taille du jeu de référence.

Nous pouvons observer que des seuils de 10 à 20 obtiennent parfois un coût de calcul supérieur au coût de CFOF exact (équivalent à un ratio de 1), alors que c'est beaucoup plus rare pour un seuil de 30 (seulement 0.073% contre 0.443%). Ainsi, en augmentant la capacité des nœuds feuilles, nous réduisons leur nombre total, ce qui a pour conséquence de réduire le coût de calcul.

Ce résultat est confirmé par la figure 3.25 pour laquelle les points sont triés en abscisse selon le nombre de séquences de référence. Pour le seuil à 10-20, nous observons qu'après la barre verticale des 3000 — en jaune — nous n'obtenons plus de coût de calcul supérieur au coût de CFOF exact.

Les coûts de calcul supérieurs aux coûts des CFOF exacts sont dus à deux phénomènes. D'abord, nous pouvons observer que, sur la figure 3.25, plusieurs séquences, pour lesquelles la taille du jeu de référence reste inférieure à 1050, possèdent un coût de calcul supérieur au coût de CFOF exact. Cela s'explique par le fait que l'historique de référence est trop petit, et que l'algorithme de construction de l'arbre génère un nombre trop élevé de nœuds pour indexer ce faible nombre de séquences. Cependant, plus le nombre de séquences indexées augmente, et plus le phénomène s'estompe. Ces séquences proviennent des séries "ec2_cpu_utilization_c6585a", "grok_asg_anomaly" et "rds_cpu_utilization_e47b3b", dont les ratios de la dernières sont visibles sur la figure 3.22b de la section 3.6.2.5. Et ensuite, nous remarquons qu'un pic est visible aux alentours de 3000 séquences de référence. Cela est dû aux calculs des séquences de deux séries : "art_daily_no_noise" et "art_daily_perfect_square_wave", qui montrent une grande périodicité et ne contiennent pas de perturbation. Les valeurs d'une période à l'autre sont identiques, et les séquences générées le sont donc également. Lors des expérimentations portant sur ces séries, le seuil de séparation des nœuds feuilles est fixé à 10 lorsque le nombre de séquences de référence tend vers 3000.

Cependant, l'arbre d'indexation a de plus en plus de mal à construire des séparateurs satisfaisants, puisqu'il doit indexer uniquement des séquences similaires. L'effectif de l'arbre croît, dépassant 3000 séquences de référence, mais en conservant une valeur de seuil de 10. Lors de la nouvelle initialisation de l'arbre pour ces deux séries, avec un nombre de séquences de référence supérieur à 3000, le seuil passe à 20 : l'arbre d'indexation permet à nouveau d'obtenir un gain de calcul. Lors de l'utilisation d'un seuil de 30, le gain se maintient et il est globalement meilleur que pour les seuils 10-20.

Les résultats de la corrélation de Spearman pour les différents seuils sont montrés sur la figure 3.26. Les différences sont faibles : la plus grande est constatée pour la série "exchange-3_cpm_results", avec une différence de 0.0768. Seules 2 séries sur 57 ("art_increase_spike_density" et "ec2_network_in_257a54") montrent une corrélation faiblement plus élevée avec le seuil de 30. La moyenne des corrélations de Spearman se dégrade peu de 0.955 à 0.942 lorsque le seuil passe de 10-20 à 30. L'écart-type des corrélations reste identique quelle que soit la valeur du seuil.

Comme les corrélations de Spearman avec des seuils différents sont très

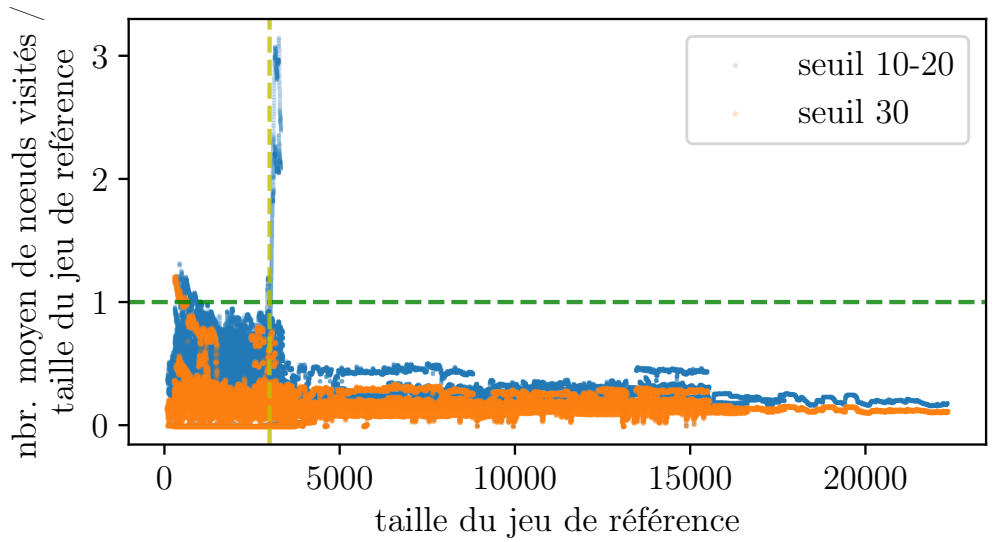


FIGURE 3.25 – Comparaison pour deux définitions différentes du seuil du ratio du nombre moyen de nœuds visités par rapport à la taille du jeu de référence (en abscisse).

proches, les résultats de détection, visible sur la figure 3.27, sont de ce fait très similaires. Ainsi, pour la série "exchange-3_cpm_results", qui montre le plus grand écart entre les corrélations de SPEARMAN de chaque seuil, la différence de ROC AUC n'est que de 0.034, et celle de PRC AUC de 0.005.

Ainsi, l'augmentation du seuil ne dégrade que très peu les résultats obtenus, que ce soit en termes de détection et d'approximation. Notons aussi que le nombre de nœuds construit dans l'arbre sera plus petit, et le gain en coût de calcul sera meilleur. Une augmentation non démesurée du seuil est donc une bonne solution pour améliorer le coût de l'approximation.

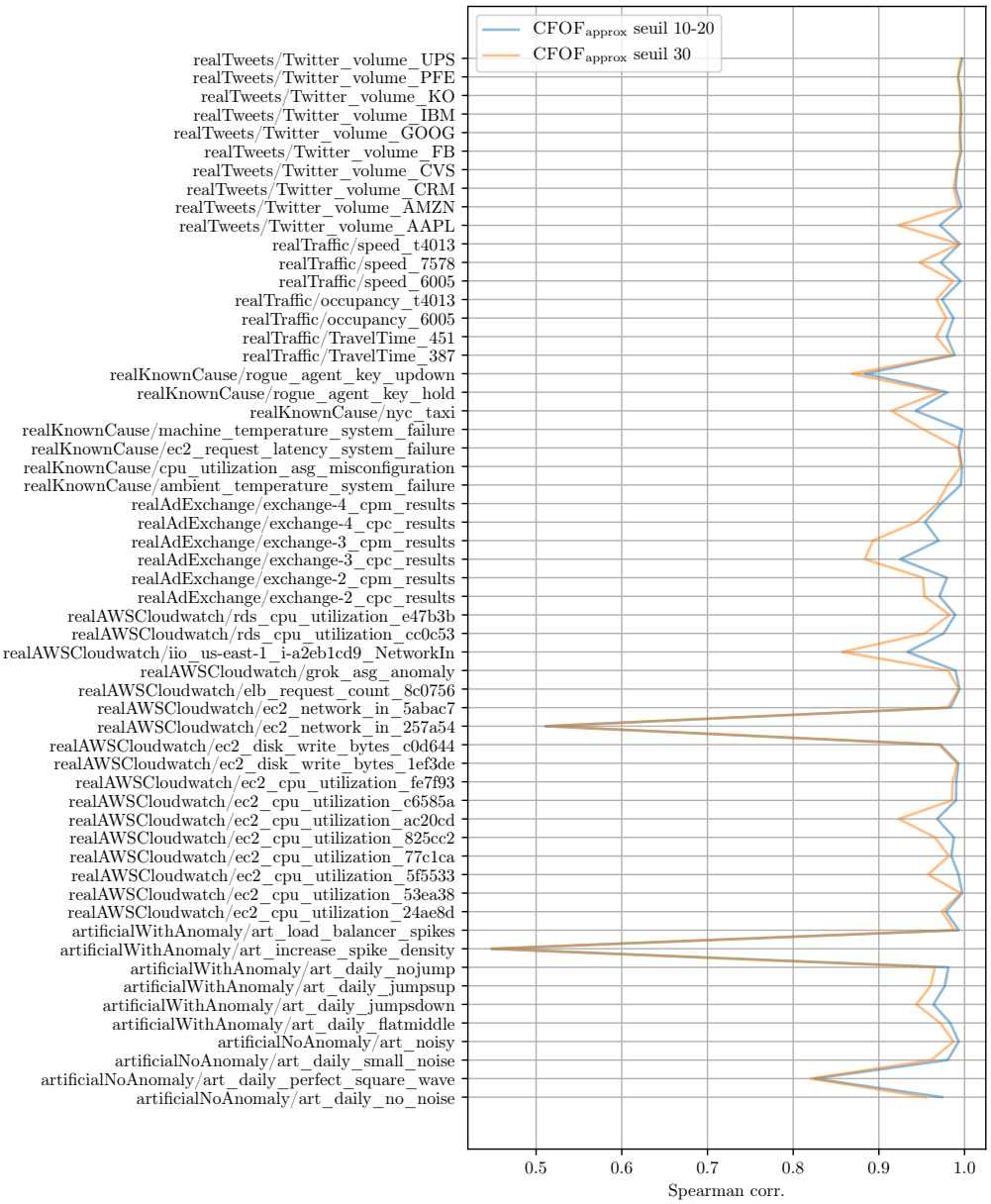


FIGURE 3.26 – Comparaison pour deux définitions différentes du seuil de la corrélation de SPEARMAN sur 57 séries du benchmark NAB.

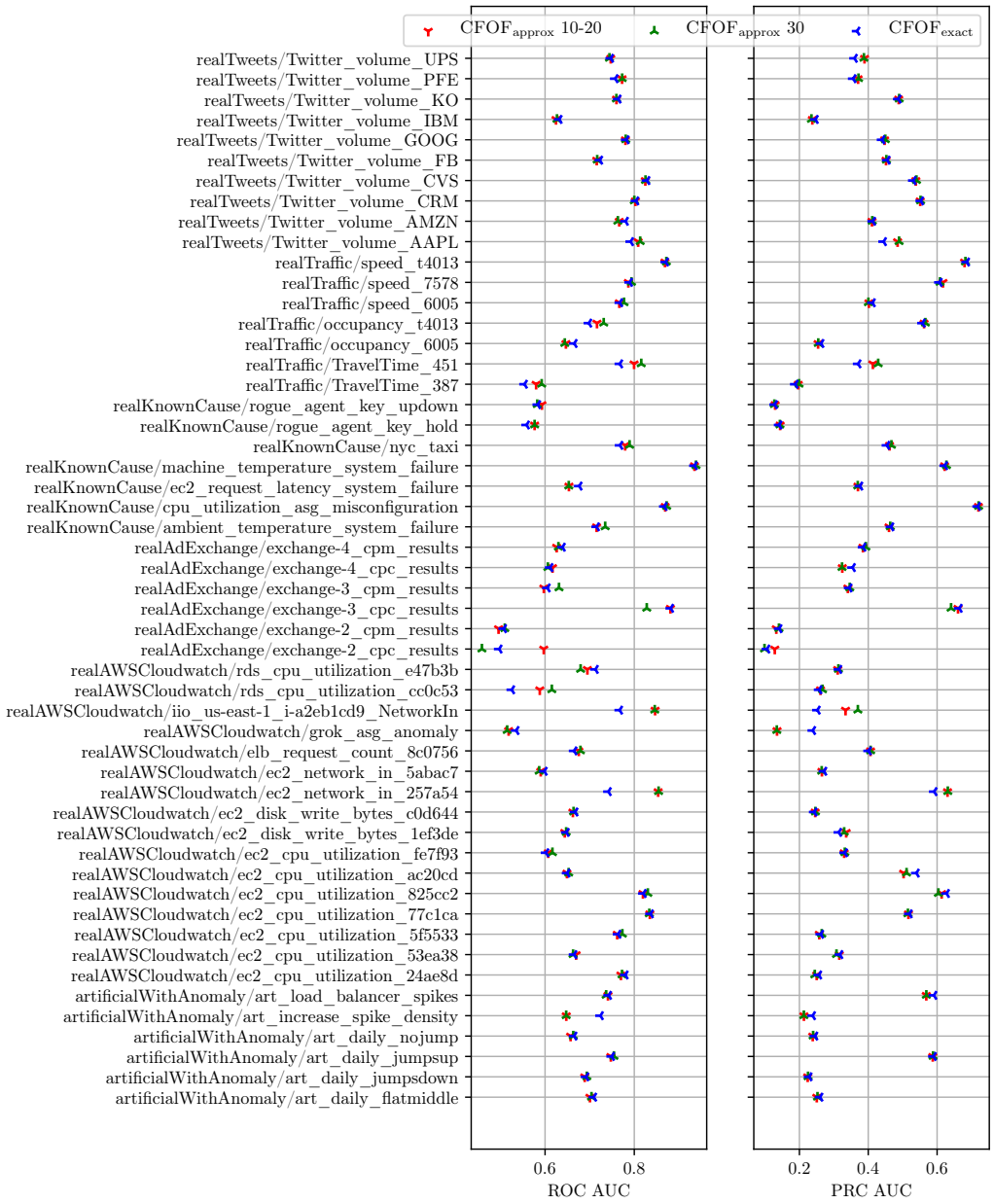


FIGURE 3.27 – Comparaison des résultats ROC AUC et PRC AUC pour les scores CFOF exacts et approximés avec seuil de 10-20 et 30.

3.6.2.7 Gain selon dimensionnalité des séquences

Les séquences sont construites selon la démarche expliquée dans la section 3.6.2.2, et c'est pourquoi les dimensions des séquences générées sont différentes d'une série à l'autre, variant entre 8, 10 et 12.

Dans cette section, nous analysons plus en détail la distribution du nombre moyen de nœuds visités normalisé par la taille du jeu de référence, par dimension, et pour un seuil fixé à 10-20.

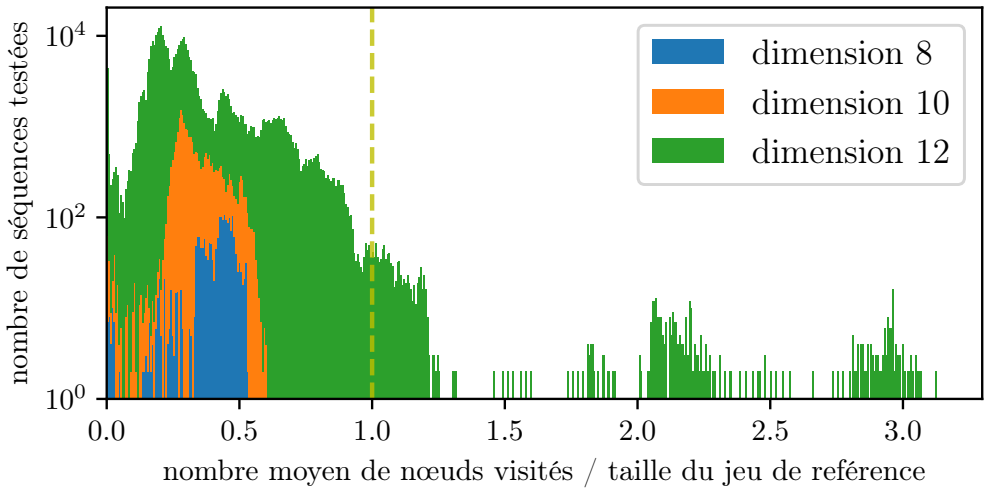


FIGURE 3.28 – Comparaison de la distribution du nombre moyen de nœuds visités normalisé par la taille du jeu de référence, selon la dimension des séquences, avec un seuil paramétré à 10-20.

Nous constatons sur la figure 3.28 que la distribution varie selon la dimension. Pour les dimensions 8 et 10, le ratio du nombre de nœuds visités par rapport à la taille du jeu de référence ne dépasse pas respectivement 0.53 et 0.60. Le gain en coût de calcul est donc toujours positif. Pour les séquences de dimension 12, des coûts de calcul supérieurs aux coûts des CFOF exacts apparaissent, mais ils ne représentent que 0.48% des séquences de même dimension.

Le risque d'obtenir un coût de calcul supérieur au coût de CFOF exact est minimisé lorsque le seuil est fixé à 30, comme montré sur la figure 3.29.

Nous pouvons confirmer les résultats de la section 3.6.2.6 : le gain est de meilleure qualité lorsque le seuil est à 30. Seule la série "rds_cpu_utilization_e47b3b" montre un coût de calcul supérieur au coût de CFOF exact, pour les même raisons que celles identifiées dans la section 3.6.2.6. Ce résultat se confirme dans la figure 3.29.

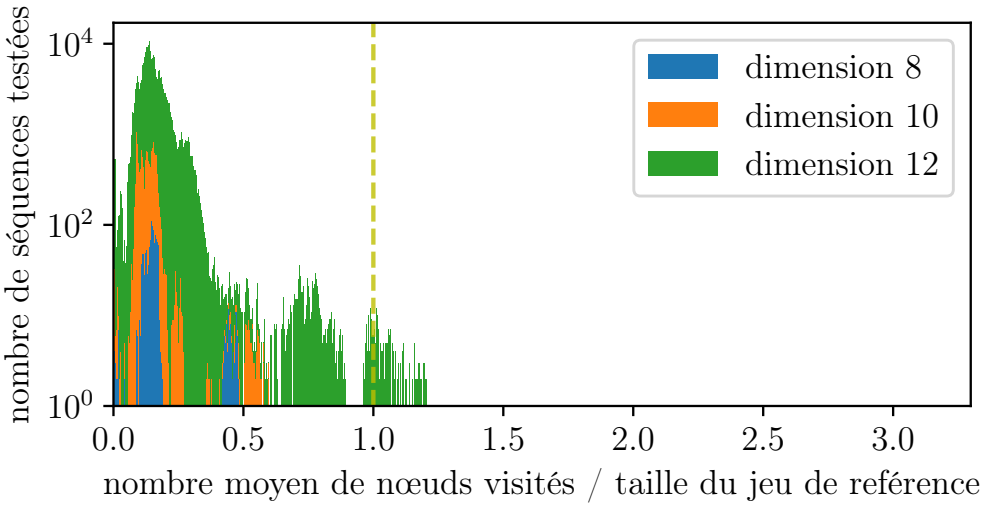


FIGURE 3.29 – Comparaison de la distribution du nombre moyen de nœuds visités normalisé par la taille du jeu de référence, selon la dimension des séquences, avec un seuil paramétré à 30.

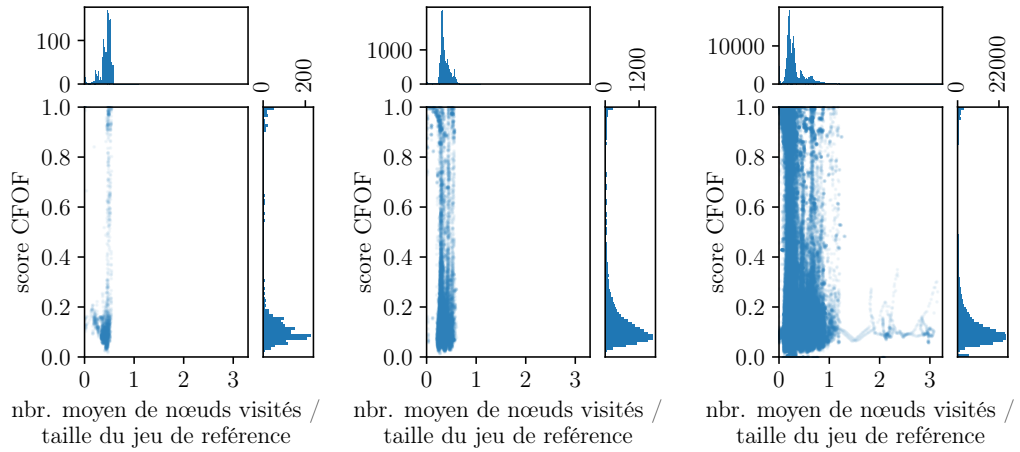
Mais c'est aussi le cas lorsque le seuil est modifié : les gains en dimension 8 restent meilleurs que ceux en dimension 12 quel que soit le seuil.

Ces résultats se confirment sur la figure 3.30. Nous constatons qu'il est alors possible de réduire le coût de calcul en modifiant le seuil, mais cela s'avère n'être une bonne option que lorsque la dimension des séquences n'est pas trop élevée. Dans le cas contraire, c'est la diminution de la dimension des séquences qui permettra de limiter le coût de l'approximation et de conserver un gain par rapport au calcul du score exact.

3.7 Conclusion

Nous avons présenté dans ce chapitre une méthode permettant d'approximer le score CFOF à l'aide d'un arbre d'indexation. Ce dernier fournit une approximation sur la distribution des séquences indexées dans les différents nœuds de l'arbre. Ainsi, pour évaluer une nouvelle séquence, l'algorithme que nous avons proposé utilise ces distributions afin de positionner rapidement cette séquence dans le voisinage de chaque séquence de référence.

Nous avons testé notre méthode sur l'ensemble de jeux purement synthétiques *Clust2* initialement proposé par [Angiulli, 2017], ainsi que sur le benchmark NAB, proposé par [Lavin et Ahmad, 2015], disponible sur Gi-



(a) Dimension 8 (b) Dimension 10 (c) Dimension 12

FIGURE 3.30 – Représentation par dimension, pour chaque séquence évaluée, du nombre moyen de nœuds visités normalisé par la taille du jeu de référence.

tHub [4] et composé de 58 séries temporelles différentes.

Sur les jeux artificiels, les résultats de corrélation de SPEARMAN montrent que les scores CFOF approximés maintiennent correctement les rangs des scores CFOF exacts. Nous faisons ce même constat pour une grande majorité des séries du benchmark NAB.

En ce qui concerne la détection d'anomalies, nous avons comparé notre méthode et celles évaluées dans le benchmark NAB, grâce aux scores ROC AUC et PRC AUC : les détections basées sur les scores CFOF exacts et approximés se retrouvent généralement bien classés sur l'ensemble des séries. Ils sont en tête des deux classements globaux (ROC AUC et PRC AUC) obtenus en faisant la moyenne sur toutes les séries.

Le but de notre méthode est de réduire les temps de calcul en obtenant une approximation du score CFOF sur des flux de données. Le gain en terme de coût de calcul s'avère indéniable sur les jeux *Clust2*, de 2 à 50 dimensions. Pour les séries du benchmark NAB, quelques séquences dans ces séries montrent des coûts de calcul de score approché supérieurs à ceux des valeurs de CFOF exact, mais elles sont peu nombreuses. Les résultats d'expérimentations ont montré qu'il existe deux paramètres pouvant être la cause de ces coûts de calcul plus élevés : 1) la diminution du seuil définissant le nombre maximum de séquences dans une feuille de l'arbre d'indexation et 2) l'augmentation de la dimension de ces séquences.

4. https://github.com/numenta/NAB

L'augmentation de la capacité de stockage des nœuds feuilles de l'arbre d'indexation, en la fixant à 30, a ainsi permis de limiter le nombre de séquences nécessitant des coûts de calcul supérieurs à ceux des scores CFOF exacts, au prix d'une légère diminution des résultats de corrélation de SPEARMAN. Cependant, une augmentation plus forte de la capacité des nœuds feuilles impliquerait une réduction du nombre de nœuds total sur lesquels les distributions des voisinages sont approximées. Cela pourrait mener à une trop forte généralisation de la distribution des séquences indexées et à une réelle dégradation des résultats de corrélation de SPEARMAN. L'augmentation de cette capacité de stockage a donc ses limites et il serait utile de pouvoir réduire les coûts de calcul tout en conservant des résultats de corrélation satisfaisants.

Dans ce but, il semble possible d'exploiter le phénomène 2) mentionné ci-dessus, c'est-à-dire le lien entre temps de calcul et dimension des séquences. Pour ce faire, nous proposons dans le prochain chapitre d'approximer le score CFOF d'une séquence $\boldsymbol{q}$ à partir de l'agrégation des résultats d'approximation des scores CFOF obtenus pour des sous-séquences qui composent $\boldsymbol{q}$. Chacune de ces sous-séquences, de plus petites dimensions que celle de la séquence d'origine, sera évaluée dans un arbre d'indexation distinct, contenant des sous-séquences des séquences de référence. En utilisant des sous-séquences de dimensions réduites, l'objectif sera donc obtenir plus rapidement les résultats de leur approximation, et de réduire ainsi les coûts de calcul de l'approximation du score CFOF de la séquence d'origine.

Chapitre 4

Calcul des scores CFOF par agrégation à partir de sous-espaces

4.1 Introduction

L'utilisation de CFOF est particulièrement adaptée pour des séquences en haute dimension vis-à-vis des autres méthodes non supervisées de détection d'anomalies [Angiulli, 2020]. Nous avons proposé, au chapitre 3, une méthode d'approximation du score CFOF permettant de réduire le coût de calcul dans le contexte de la recherche d'anomalies dans des flux de données, tout en préservant la qualité de détection. Cependant, les résultats présentés dans le chapitre 3 montrent que l'augmentation de la dimension peut mener parfois à un surcoût de calcul de l'approximation par rapport au calcul classique du score CFOF exact par parcours de l'ensemble des séquences. En effet, il semble que la structure d'indexation utilisée puisse dans certains cas produire un nombre important de nœuds, supérieur au nombre de séquences indexées, et ne permette pas à la méthode de parcourir l'arbre tout en limitant le coût de calcul.

Nous allons voir dans ce chapitre l'origine de ce phénomène lorsque la dimension augmente, avant de proposer un complément à notre méthode, afin d'éviter les cas particuliers causant la dégénération de la structure d'indexation.

4.2 Problématique

Nous avons choisi d'utiliser un arbre d'indexation *i*SAX pour adapter le calcul du score CFOF dans un contexte où les données sont fournies au fil de l'eau. Cependant, nous avons observé dans les résultats précédents (Chapitre 3) que le parcours de l'arbre est, dans certains cas, plus coûteux qu'un parcours simple de l'ensemble des séquences. Nous avons également vu que si le réglage du seuil du nombre maximum d'éléments par feuille de l'arbre permet de résoudre partiellement ce problème, une autre difficulté survient lorsque la dimension augmente fortement. En effet, l'arbre d'indexation contient alors un très grand nombre de nœuds au niveau 1 qui est le premier niveau nécessitant d'être parcouru. Pour rappel, l'unique nœud de niveau 0 est le nœud racine, et les nœuds de niveau 1, qui sont ses descendants directs, sont tous visités lors du parcours de l'arbre, c'est-à-dire qu'ils ne peuvent être élagués.

Les nœuds de niveau 1 représentent toutes les possibilités de coupure en deux portions de toutes les dimensions autour de leurs moyennes, le nombre maximum de nœuds de niveau 1 est donc de 2^d pour un jeu comportant d dimensions [Camerra *et al.*, 2014]. Si les dimensions sont indépendantes, le processus d'indexation pourra être amené à construire tous les nœuds du niveau 1 pour répartir les séquences [1]. Cette situation ne pose pas de problème si le nombre de séquences à indexer est largement supérieur au nombre de nœuds de niveau 1 que peut produire l'arbre. En effet, ces nœuds, ou les sous-arbres qui leur sont rattachés, sont alors suffisamment peuplés pour maintenir l'efficacité de la structure. En revanche, si le nombre de séquences est inférieur au nombre maximum de nœuds de niveau 1 (ou du même ordre), et que les dimensions de l'espace sont plutôt indépendantes entre elles, chaque nœud ne contiendra que très peu de séquences (voire une seule).

Nous allons illustrer cela sur un ensemble de jeux artificiels nommé *Clust1* et généré pour l'occasion en faisant varier la dimensionnalité. Un jeu est ici formé d'un seul groupe (cluster) de 10000 séquences, distribuées selon une loi normale sur chaque dimension, centrée à l'origine et d'écart-type 1. Pour chaque jeu, nous comptons le nombre de nœuds créés au niveau 1, et calculons la moyenne et l'écart-type du nombre de séquences placées dans ces nœuds (et cumulé dans les sous-arbres qui en sont issus). Le tableau 4.1 présente les résultats obtenus pour des jeux de dimensions 2, 5, 10, et 20.

Le nombre de nœuds créés au niveau 1 croît très vite lorsque la dimension des séquences augmente, jusqu'à atteindre quasiment le nombre de séquences

1. D'une façon plus générale, la croissance du nombre de noeuds dans tout l'arbre a été soulignée dans [Shieh et Keogh, 2009] et aussi dans [Wang *et al.*, 2013] où il est montré que ce nombre croît exponentiellement en fonction de la dimension de l'espace.

TABLE 4.1 – Distribution du nombre de séquences dans les nœuds de niveau 1 (et cumulé sur leurs descendants), selon la dimension des séquences des jeux *Clust1*.

Dimension	Moyenne nbr. séquences	Écart-type nbr. séquences	Nbr. nœuds niveau 1
2	2500.000	38.814	4
5	312.500	17.780	32
10	9.766	3.121	1024
20	1.005	0.069	9954

insérées (i.e., 10000) lorsqu'elles sont de dimension 20. La très grande majorité de ces 9954 nœuds contient alors une seule séquence. Dans ces expériences, le seuil du nombre maximum de séquences par feuille a été fixé à 30, mais nous noterons que ce seuil ne change pas le nombre de nœuds de niveau 1. Le modifier ne changerait donc pas les résultats obtenus ici.

Si nous considérons des jeux de la famille *Clust2* — jeux artificiels contenant chacun deux clusters de 5000 séquences et décrits au chapitre 3 — nous obtenons les résultats indiqués dans le tableau 4.2.

TABLE 4.2 – Distribution du nombre de séquences dans les nœuds de niveau 1 (et cumulé sur leurs descendants), selon la dimension des séquences des jeux *Clust2*.

Dimension	Moyenne nbr. séquences	Écart-type nbr. séquences	Nbr. nœuds niveau 1
2	2500.000	2399.070	4
5	714.286	1647.570	14
10	175.439	833.275	57
20	45.045	395.363	222
50	7.862	147.283	1272
100	3.029	87.214	3301
200	2.057	71.216	4862
500	1.972	69.105	5072

Nous remarquons que la tendance observée est plus lente que précédemment. Ceci peut s'expliquer essentiellement par la distribution des séquences en deux groupes rendant les dimensions moins indépendantes que dans les

jeux *Clust1*. En effet, de telles dépendances conduisent à avoir besoin de moins de nœuds de niveau 1 pour couvrir tout l'espace. Mais le constat reste similaire, plus la dimension augmente et plus alors le nombre de nœuds de niveau 1 s'accroit, mais en même temps plus le nombre moyen de séquences associées à un nœud de niveau 1 (et son sous-arbre) décroit. Le parcours de l'arbre nécessitant au minimum l'exploration de tous les nœuds de niveau 1, l'élagage lors des recherches devient de moins en moins efficace. Ces résultats sont également cohérents avec les observations faites à la section 3.6.1.2 qui montrent que le nombre moyen de nœuds visités pour les jeux *Clust2* double lorsque nous passons de 20 à 50 dimensions.

4.3 Proposition

Nous souhaitons maintenir un gain, en terme de coût de calcul de l'approximation de CFOF, en haute dimension dans certains cas particuliers vus dans la section précédente pour lesquels nous avons constaté un surcoût. Pour cela, nous proposons d'utiliser conjointement plusieurs structures d'index de plus faibles dimensions. De façon plus précise, nous découpons l'espace de départ en s sous-espaces $E_{d_1}, E_{d_2}, \ldots, E_{d_s}$ deux à deux disjoints et tels que l'union des dimensions de $E_{d_1}, E_{d_2}, \ldots, E_{d_s}$ redonne l'espace de départ. Chaque sous-espace est alors associé à un arbre iSAX qui lui est propre. Puis, chaque séquence de référence est projetée sur chacun des sous-espaces et indexée dans chacun des arbres correspondants.

A partir des arbres d'indexation de ces sous-espaces, nous proposons deux méthodes d'approximation du score CFOF d'une séquence :

1. La première consiste à projeter la séquence dans les sous-espaces et à agréger les scores CFOF obtenus à partir de chacun des arbres associés. Pour construire cette approximation par agrégation nous tirerons partie d'une propriété établie par Angiulli dans [Angiulli, 2020].
2. La seconde méthode est basée sur une modification de l'algorithme 1 proposé au chapitre 3. Elle consiste, pour le calcul du *v-rang* de chaque séquence $\boldsymbol{p}$, à calculer les valeurs de *v-rang* obtenues par $\boldsymbol{p}$ dans chacun des sous-espaces et à retenir alors la moyenne de ces *v-rang* comme valeur de *v-rang* de $\boldsymbol{p}$ sur l'espace complet.

Nous allons exposer ces deux méthodes avant de les évaluer. La section 4.3.1.2 détaillera l'approximation par agrégation de scores CFOF sur des sous-espaces, et la section 4.3.1.3 présentera l'approche basée sur l'agrégation des *v-rang* obtenus sur ces sous-espaces.

Dans les deux cas, l'objectif est de réduire le nombre de nœuds parcourus tout en préservant la qualité de l'approximation du score CFOF et donc celle de la détection des anomalies. Un autre aspect intéressant de cette stratégie d'agrégation est de permettre aisément de mettre en œuvre un niveau de parallélisation du traitement. En effet, les calculs peuvent être effectués en parallèle sur chacun des arbres avant de réaliser l'étape d'agrégation qui se trouve être très simple (avec un coût constant). La durée du traitement avec un arbre unique indexant tout l'espace, peut alors être remplacée par le maximum de la durée nécessaire pour le calcul sur chacun des arbres associés aux sous-espaces.

4.3.1 Approximation du score CFOF à partir des résultats de plusieurs arbres

4.3.1.1 Rappel d'une propriété du score CFOF

Angiulli présente dans [Angiulli, 2020] une approximation du score CFOF, lorsque la dimensionnalité est grande, dans le cas d'un jeu de données contenant les réalisations d'un vecteur aléatoire $\boldsymbol{X}_d$ (à valeurs dans $\mathbb{R}^d$) dont les composantes sont des variables aléatoires indépendantes et identiquement distribuées. Par souci d'homogénéité avec le reste du mémoire nous appellerons *séquence* une réalisation d'un tel vecteur aléatoire, alors qu'Angiulli lui utilise le terme plus général d'*objet*. Son étude se base sur ses travaux antérieurs [Angiulli, 2018] dans lesquels il démontre des propriétés des k voisins inverses les plus proches (notion abordée dans la section 2.2.3.3.a) dans les espaces de grande dimensionnalité.

Soit $\mu_{\boldsymbol{X}_d}$ le vecteur moyen de $\boldsymbol{X}_d$ (i.e., les composantes de $\mu_{\boldsymbol{X}_d}$ sont les moyennes des composantes de $\boldsymbol{X}_d$) et soit $\boldsymbol{x}_d$ une réalisation de $\boldsymbol{X}_d$. Angiulli utilise le carré standardisé de la différence entre $\boldsymbol{x}_d$ et $\mu_{\boldsymbol{X}_d}$ [2], noté $z\text{-score}_{\boldsymbol{x}_d,\boldsymbol{X}_d}$ et défini par :

$$z\text{-score}_{\boldsymbol{x}_d,\boldsymbol{X}_d} = \frac{\|\boldsymbol{x}_d - \mu_{\boldsymbol{X}_d}\|^2 - \mu_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2}}{\sigma_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2}} \tag{4.1}$$

avec $\mu_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2}$ (resp. $\sigma_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2}$) la moyenne (resp. l'écart-type) de la différence entre $\boldsymbol{x}_d$ et $\mu_{\boldsymbol{X}_d}$ au carré.

2. Ce carré représente le carré de la distance entre les points dont les vecteurs positions seraient $\boldsymbol{x}_d$ et $\mu_{\boldsymbol{X}_d}$.

Angiulli démontre alors que lorsque la dimension est grande le score CFOF($\boldsymbol{x}_d$) tend vers une expression dépendant de z-score$_{\boldsymbol{x}_d,\boldsymbol{X}_d}$:

$$\text{CFOF}(\boldsymbol{x}_d) \approx \Phi\left(\frac{z\text{-score}_{\boldsymbol{x}_d,\boldsymbol{X}_d}\sqrt{\kappa-1}+2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}}\right) \tag{4.2}$$

où ϱ est le paramètre de seuil du score CFOF, Φ la fonction de répartition de la loi normale $\mathcal{N}(0,1)$ et κ le kurtosis des composantes de $\boldsymbol{X}_d$ (égal pour toutes les composantes car elles sont identiquement distribuées [3]). On notera que la version du kurtosis utilisée ici est le moment centré réduit d'ordre 4, c'est-à-dire : $\kappa = \mathbb{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right]$ où X est une variable aléatoire correspondant à une des composantes de $\boldsymbol{X}_d$ avec μ et σ sa moyenne et son écart-type.

4.3.1.2 Agrégation des scores CFOF exacts provenant de plusieurs sous-espaces

Nous allons démontrer, en utilisant la propriété rappelée dans la section précédente, qu'il est possible d'approximer le score CFOF d'une séquence à partir de ses scores CFOF calculés sur plusieurs sous-espaces deux à deux disjoints et dont l'union forme l'espace d'origine.

Pour cela, nous allons tout d'abord étudier la décomposition de z-score$_{\boldsymbol{x}_d,\boldsymbol{X}_d}$ (lorsque $d > 1$), et nous verrons ensuite qu'il est possible, à partir des scores CFOF obtenus dans différents sous-espaces, de retrouver le score CFOF de $\boldsymbol{x}_d$ dans son espace d'origine, grâce à cette décomposition.

4.3.1.2.a Décomposition préliminaire

Soit deux sous-espaces E_{d_1} et E_{d_2} supplémentaires de $\mathbb{R}^d$ avec $d > 1$. Ils sont supplémentaires, leur intersection se réduit donc au vecteur nul et $d = dimension(E_{d_1}) + dimension(E_{d_2})$. Les projections de $\boldsymbol{X}_d$ dans E_{d_1} et E_{d_2} seront notées respectivement $\boldsymbol{Y}_{d_1}$ et $\boldsymbol{Z}_{d_2}$.

$\boldsymbol{Y}_{d_1}$ et $\boldsymbol{Z}_{d_2}$ sont eux-mêmes des vecteurs aléatoires à valeur dans $\mathbb{R}^{d_1}$ et $\mathbb{R}^{d_2}$. Soit $\boldsymbol{x}_d$ une réalisation de $\boldsymbol{X}_d$, ses projections sur E_{d_1} et E_{d_2} sont des réalisations de $\boldsymbol{Y}_{d_1}$ et $\boldsymbol{Z}_{d_2}$ que nous noterons $\boldsymbol{y}_{d_1}$ et $\boldsymbol{z}_{d_2}$.

Nous travaillons dans un premier temps sur une décomposition en deux sous-espaces, mais nous donnerons ensuite la généralisation pour un nombre quelconque de sous-espaces.

Considérons la décomposition des différents termes de z-score$_{\boldsymbol{x}_d,\boldsymbol{X}_d}$. Notons avec des indices i et j les composantes de nos vecteurs, le terme $\|\boldsymbol{x}_d - \mu_{\boldsymbol{X}_d}\|^2$ s'écrit alors :

3. [Angiulli, 2020] présente une généralisation de ce résultat au cas où elles ne sont pas identiquement distribuées.

$$\begin{aligned}\|\boldsymbol{x}_d - \mu_{\boldsymbol{X}_d}\|^2 &= \sum_{i=1}^{d}(x_i - \mu_{X_i})^2 \\ &= \sum_{i=1}^{d_1}(y_i - \mu_{Y_i})^2 + \sum_{j=1}^{d_2}(z_j - \mu_{Z_j})^2 \\ &= \|\boldsymbol{y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2 + \|\boldsymbol{z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2\end{aligned}$$

De la même façon $\mu_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2}$ s'écrit :

$$\mu_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2} = \mu_{\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2} + \mu_{\|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2}$$

En ce qui concerne l'écart-type, comme $\|\boldsymbol{x}_d - \mu_{\boldsymbol{X}_d}\|^2$ est la somme de $\|\boldsymbol{y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2$ et $\|\boldsymbol{z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2$, il se décompose en :

$$\sigma_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2} = \sqrt{\sigma^2_{\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2} + \sigma^2_{\|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2} + 2\mathrm{Cov}[\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|, \|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|]}$$

Les variables aléatoires correspondant aux composantes étant indépendantes, la covariance entre $\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|$ et $\|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|$ est nulle. Nous obtenons alors :

$$\sigma_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2} = \sqrt{\sigma^2_{\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2} + \sigma^2_{\|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2}}$$

A l'aide de ces deux décompositions, nous pouvons à présent réécrire z-score$_{\boldsymbol{x}_d, \boldsymbol{X}_d}$ défini par l'équation 4.1 :

$$\begin{aligned}z\text{-score}_{\boldsymbol{x}_d, \boldsymbol{X}_d} &= \frac{\|\boldsymbol{x}_d - \mu_{\boldsymbol{X}_d}\|^2 - \mu_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2}}{\sigma_{\|\boldsymbol{X}_d - \mu_{\boldsymbol{X}_d}\|^2}} \\ &= \frac{\|\boldsymbol{y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2 + \|\boldsymbol{z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2 - \mu_{\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2} - \mu_{\|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2}}{\sqrt{\sigma^2_{\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2} + \sigma^2_{\|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2}}} \\ &= \frac{z\text{-score}_{\boldsymbol{y}_{d_1}, \boldsymbol{Y}_{d_1}} \times \sigma_{\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2} + z\text{-score}_{\boldsymbol{z}_{d_2}, \boldsymbol{Z}_{d_2}} \times \sigma_{\|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2}}{\sqrt{\sigma^2_{\|\boldsymbol{Y}_{d_1} - \mu_{\boldsymbol{Y}_{d_1}}\|^2} + \sigma^2_{\|\boldsymbol{Z}_{d_2} - \mu_{\boldsymbol{Z}_{d_2}}\|^2}}}\end{aligned} \tag{4.3}$$

Ce résultat va nous permettre de décomposer l'approximation du score CFOF rappelée dans la section 4.3.1.1.

4.3.1.2.b Décomposition de l'approximation dans deux sous-espaces

Nous allons montrer que le score CFOF de $\boldsymbol{x}_d$ peut être approximé à partir des scores CFOF de ses projections $\boldsymbol{y}_{d_1}$ et $\boldsymbol{z}_{d_2}$ dans les sous-espaces E_{d_1} et E_{d_2}.

Dans la suite, pour faciliter la lecture, nous noterons $\boldsymbol{x}_d$, $\boldsymbol{y}_{d_1}$ et $\boldsymbol{z}_{d_2}$ respectivement par $\boldsymbol{x}$, $\boldsymbol{y}$ et $\boldsymbol{z}$. Pour la même raison, nous utiliserons $z\text{-score}_{\boldsymbol{x}}$, $z\text{-score}_{\boldsymbol{y}}$ et $z\text{-score}_{\boldsymbol{z}}$ pour $z\text{-score}_{\boldsymbol{x}_d,\boldsymbol{X}_d}$, $z\text{-score}_{\boldsymbol{y}_{d_1},\boldsymbol{Y}_{d_1}}$ et $z\text{-score}_{\boldsymbol{z}_{d_2},\boldsymbol{Z}_{d_2}}$, ainsi que $\sigma_{\boldsymbol{Y}}$ et $\sigma_{\boldsymbol{Z}}$ pour $\sigma_{\|\boldsymbol{Y}_{d_1}-\mu_{\boldsymbol{Y}_{d_1}}\|^2}$ et $\sigma_{\|\boldsymbol{Z}_{d_2}-\mu_{\boldsymbol{Z}_{d_2}}\|^2}$.

Considérons l'approximation donnée par l'équation 4.2 de la section 4.3.1.1. En remplaçant $z\text{-score}_{\boldsymbol{x}}$ (i.e., $z\text{-score}_{\boldsymbol{x}_d,\boldsymbol{X}_d}$) par son expression fournie par l'équation 4.3 nous obtenons :

$$\begin{aligned}\mathrm{CFOF}(\boldsymbol{x}) &\approx \Phi\left(\frac{(z\text{-score}_{\boldsymbol{y}} \times \sigma_{\boldsymbol{Y}} + z\text{-score}_{\boldsymbol{z}} \times \sigma_{\boldsymbol{Z}}) \times \sqrt{\kappa-1}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}} + \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}}\right)\\ &\approx \Phi\left(\frac{z\text{-score}_{\boldsymbol{y}} \times \sigma_{\boldsymbol{Y}} \times \sqrt{\kappa-1}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}} + \frac{z\text{-score}_{\boldsymbol{z}} \times \sigma_{\boldsymbol{Z}} \times \sqrt{\kappa-1}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}} + \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}}\right)\end{aligned} \tag{4.4}$$

Considérons le terme $\frac{z\text{-score}_{\boldsymbol{y}} \times \sigma_{\boldsymbol{Y}} \times \sqrt{\kappa-1}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}}$, il s'écrit :

$$\begin{aligned}\frac{z\text{-score}_{\boldsymbol{y}} \times \sigma_{\boldsymbol{Y}} \times \sqrt{\kappa-1}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}} &= \frac{z\text{-score}_{\boldsymbol{y}} \times \sqrt{\kappa-1}}{\sqrt{\kappa+3}} \times \frac{\sigma_{\boldsymbol{Y}}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2}}\\ &= \left(\frac{z\text{-score}_{\boldsymbol{y}} \times \sqrt{\kappa-1} + 2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}} - \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}}\right)\\ &\quad \times \frac{\sigma_{\boldsymbol{Y}}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2}}\end{aligned} \tag{4.5}$$

En appliquant l'approximation donnée par l'équation 4.2 à $\mathrm{CFOF}(\boldsymbol{y})$, nous obtenons :

$$\frac{z\text{-score}_{\boldsymbol{y}} \times \sigma_{\boldsymbol{Y}} \times \sqrt{\kappa-1}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}} \approx \left(\Phi^{-1}(\mathrm{CFOF}(\boldsymbol{y})) - \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}}\right) \times \frac{\sigma_{\boldsymbol{Y}}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2}} \tag{4.6}$$

Le même calcul sur le terme $\frac{z\text{-score}_{\boldsymbol{z}} \times \sigma_{\boldsymbol{Z}} \times \sqrt{\kappa-1}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}}$ conduit à :

$$\frac{z\text{-score}_{\boldsymbol{z}} \times \sigma_{\boldsymbol{Z}} \times \sqrt{\kappa-1}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}} \approx \left(\Phi^{-1}(\text{CFOF}(\boldsymbol{z})) - \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}} \right) \times \frac{\sigma_{\boldsymbol{Z}}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2}} \tag{4.7}$$

En substituant dans l'équation 4.4 les approximations fournies par les équations 4.6 et 4.7, nous avons :

$$\begin{aligned} \text{CFOF}(\boldsymbol{x}) \approx \Phi \Bigg(& \left(\Phi^{-1}(\text{CFOF}(\boldsymbol{y})) - \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}} \right) \times \frac{\sigma_{\boldsymbol{Y}}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2}} \\ & + \left(\Phi^{-1}(\text{CFOF}(\boldsymbol{z})) - \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}} \right) \times \frac{\sigma_{\boldsymbol{Z}}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2}} \\ & + \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}} \Bigg) \end{aligned}$$

En isolant les parties dépendantes des scores CFOF, par distribution et refactorisation, l'approximation s'écrit :

$$\text{CFOF}(\boldsymbol{x}) \approx \Phi \left(\frac{\Phi^{-1}(\text{CFOF}(\boldsymbol{y})) \times \sigma_{\boldsymbol{Y}} + \Phi^{-1}(\text{CFOF}(\boldsymbol{z})) \times \sigma_{\boldsymbol{Z}}}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2}} + \lambda \right) \tag{4.8}$$

$$\text{avec } \lambda = \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa+3}} - \frac{2 \times (\sigma_{\boldsymbol{Y}} + \sigma_{\boldsymbol{Z}}) \times \Phi^{-1}(\varrho)}{\sqrt{\sigma_{\boldsymbol{Y}}^2+\sigma_{\boldsymbol{Z}}^2} \times \sqrt{\kappa+3}}$$

Cette équation nous fournit le moyen d'approximer le score CFOF de $\boldsymbol{x}$ à partir des scores CFOF de ses projections $\boldsymbol{y}$ et $\boldsymbol{z}$ dans deux sous-espaces supplémentaires.

4.3.1.2.c Généralisation de la décomposition pour s sous-espaces

Soit une collection de s sous-espaces $E_{d_1}, \ldots, E_{d_k}, \ldots, E_{d_s}$ de $\mathbb{R}^d$, tels que leurs intersections deux à deux se réduisent au vecteur nul et tels que $\sum_{k=1}^{s} dimension(E_{d_k}) = d$. La projection dans E_{d_k} du vecteur aléatoire $\boldsymbol{X}_d$ et d'une de ses réalisations $\boldsymbol{x}_d$ seront notées $\boldsymbol{X}_{d_k}$ et $\boldsymbol{x}_{d_k}$.

Considérons tout d'abord l'expression de $z\text{-score}_{\boldsymbol{x}_d, \boldsymbol{X}_d}$. Comme dans la section 4.3.1.2.a, puisque les variables aléatoires correspondant aux composantes sont indépendantes, la covariance entre $\|\boldsymbol{X}_{d_i} - \mu_{\boldsymbol{X}_{d_i}}\|$ et $\|\boldsymbol{X}_{d_j} - \mu_{\boldsymbol{X}_{d_j}}\|$

est nulle pour toute paire $i \in \{1, \ldots, s\}$, $j \in \{1, \ldots, s\}$ avec $i \neq j$. De la même façon qu'en 4.3.1.2.a nous avons alors :

$$z\text{-score}_{\boldsymbol{x}_d, \boldsymbol{X}_d} = \frac{\sum_{k=1}^{s} z\text{-score}_{\boldsymbol{x}_{d_k}, \boldsymbol{X}_{d_k}} \times \sigma_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}}{\sqrt{\sum_{k=1}^{s} \sigma^2_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}}} \tag{4.9}$$

Ensuite, par le même calcul que dans la section 4.3.1.2.b, nous pouvons également généraliser à s sous-espaces l'équation 4.8, et nous obtenons :

$$\text{CFOF}(\boldsymbol{x}) \approx \Phi\left(\frac{\sum_{k=1}^{s} \Phi^{-1}(\text{CFOF}(\boldsymbol{x}_{d_k})) \times \sigma_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}}{\sqrt{\sum_{k=1}^{s} \sigma^2_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}}} + \lambda\right) \tag{4.10}$$

$$\text{avec } \lambda = \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa + 3}} - \frac{2 \times (\sum_{k=1}^{s} \sigma_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}) \times \Phi^{-1}(\varrho)}{\sqrt{\sum_{k=1}^{s} \sigma^2_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}} \times \sqrt{\kappa + 3}}$$

L'équation 4.10 donne la formulation générale que nous avons obtenue pour l'approximation du score CFOF à partir des scores CFOF sur des sous-espaces. Nous allons ci-dessous en donner une version plus spécifique, dans le cas où les valeurs $\sigma_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}$ sont les mêmes dans tous les sous-espaces E_{d_k}. C'est un cas important, car il se présente notamment lorsque les objets sont des séquences générées à partir d'une fenêtre glissante sur une série temporelle et que les sous-espaces sont de même dimensionnalité. En effet les lois, les écarts-types et les moyennes de chacune des composantes des vecteurs aléatoires sont alors égaux, et si les E_{d_k} ont le même nombre de dimensions, les valeurs de $\sigma_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}$ sont elles aussi égales.

Dans ces conditions, pour s sous-espaces, l'équation 4.9, après factorisation par $\sigma_{\|\boldsymbol{X}_{d_k} - \mu_{\boldsymbol{X}_{d_k}}\|^2}$ et simplification, s'écrit :

$$z\text{-score}_{\boldsymbol{x}_d, \boldsymbol{X}_d} = \frac{\sum_{k=1}^{s} z\text{-score}_{\boldsymbol{x}_{d_k}, \boldsymbol{X}_{d_k}}}{\sqrt{s}} \tag{4.11}$$

Et le même calcul conduit alors pour l'équation 4.10 à la version simplifiée suivante :

$$\text{CFOF}(z\text{-score}_{\boldsymbol{x}}) \approx \Phi\left(\frac{\sum_{k=1}^{s} \Phi^{-1}(\text{CFOF}(\boldsymbol{x}_{d_k}))}{\sqrt{s}} + \lambda\right) \tag{4.12}$$

$$\text{avec} \quad \lambda = \frac{2\Phi^{-1}(\varrho)}{\sqrt{\kappa + 3}} - \frac{s \times 2\Phi^{-1}(\varrho)}{\sqrt{s \times (\kappa + 3)}} = \frac{(1 - \sqrt{s}) \times 2\Phi^{-1}(\varrho)}{\sqrt{(\kappa + 3)}}$$

Par la suite, lorsque l'approximation du score CFOF sera calculée à partir des équations 4.12 ou 4.10, par agrégation des scores obtenus sur des sous-espaces, nous emploierons la notation $\text{CFOF}_{\text{agg}}(.)$ en précisant entre les parenthèses la façon dont sont déterminés les scores pour les sous-espaces. Nous utiliserons notamment la notation $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{exact}})$ lorsque ce sont les scores CFOF exacts sur les sous-espaces qui ont été calculés, et $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ quand il s'agit des scores CFOF approximés par sous-espace (CFOF approximé tel que nous l'avons proposé au chapitre 3).

4.3.1.3 Variante par agrégation des *v-rang* dans les sous-espaces

Notre réflexion sur les possibilités de répartition des calculs du score CFOF sur plusieurs arbres, nous a également conduit à élaborer une stratégie alternative basée sur une modification de l'algorithme 1 présenté dans la section 3. Bien que cette extension de l'algorithme ne présente pas les mêmes garanties théoriques que l'approche présentée dans les sections précédentes, nous verrons dans les expérimentations qu'elle permet, elle aussi, d'obtenir des résultats pratiques intéressants.

Les modifications apportées sont les suivantes. Au niveau de l'entrée de l'algorithme 1, au lieu de recourir à un seul arbre pour indexer les séquences de références, nous utilisons plusieurs arbres, un arbre par sous-espace. Ensuite, lors du calcul du score CFOF d'une séquence $\boldsymbol{q}$ par l'algorithme 1, pour chaque séquence de référence $\boldsymbol{p}$ nous calculons ses valeurs de *v-rang* pour chacun des arbres (i.e., les *v-rang* de ses projections dans les sous-espaces). Enfin, après la ligne 21 de l'algorithme, la valeur de *v-rang* retenue pour $\boldsymbol{p}$ est la moyenne de ses *v-rang* obtenus dans les sous-espaces.

Dans la suite, lorsque l'approximation sera calculée à partir de cette méthode, nous utiliserons la notation $\text{CFOF}_{\text{agg}}(\textit{v-rang})$.

4.4 Expérimentations

Nous testons les deux méthodes d'approximation proposées, agrégation à partir des scores CFOF et à partir des *v-rang*, sur les jeux *Clust1* (cf. Section 4.2), *Clust2* (cf. Section 3.6.1) et sur les jeux du benchmark *Numenta Anomaly Benchmark* (NAB) [Lavin et Ahmad, 2015] (cf. Section 3.6.2).

Nous maintenons le seuil du nombre maximum d'éléments par feuille de l'arbre d'indexation à 30, car les résultats de la section 3.6.2.6 du chapitre 3 montrent que le gain sur le coût de calcul était significatif tandis que les valeurs de corrélation de Spearman ne diminuaient que très faiblement.

Les méthodes $\text{CFOF}_{\text{exact}}$ et $\text{CFOF}_{\text{approx}}$ sont disponibles en Python sur GitHub [4].

4.4.1 Évaluations sur les jeux de la famille *Clust1*

4.4.1.1 Approximation du score $\text{CFOF}_{\text{exact}}$ à partir des scores $\text{CFOF}_{\text{exact}}$ obtenus dans les sous-espaces

Dans cette section nous évaluons en pratique la pertinence de l'équation 4.12 sur des cas simples, en comparant sur les jeux *Clust1* les scores $\text{CFOF}_{\text{exact}}$ avec les scores $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{exact}})$.

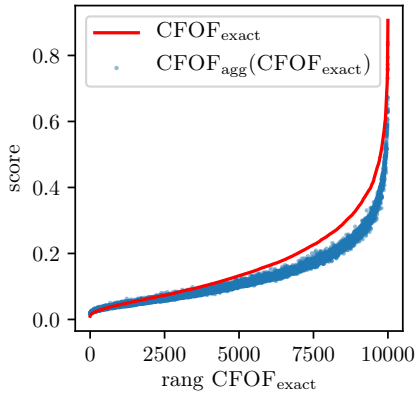
Nous réalisons ces tests sur quatre jeux *Clust1* de dimensions différentes : 20, 50, 100, 200. Pour chaque jeu, tout le jeu est indexé comme base de référence et ensuite le score de chacun des 10000 objets du jeu est calculé vis-à-vis de cette base de référence. Le tableau 4.1 de la section 4.2 a montré que pour 5 dimensions le nombre total de nœuds de niveau 1 restait très limité (32 nœuds), nous choisissons donc ici des sous-espaces ayant 5 dimensions. Ainsi, par exemple, pour le jeu *Clust1* en dimension 20 nous approximons le score CFOF en agrégeant 4 scores $\text{CFOF}_{\text{exact}}$ obtenus dans 4 sous-espaces différents, alors que pour le jeu *Clust1* en dimension 200 nous obtenons l'approximation à partir de 40 scores $\text{CFOF}_{\text{exact}}$ de 40 sous-espaces différents.

TABLE 4.3 – Corrélations de SPEARMAN entre les scores $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{exact}})$ et $\text{CFOF}_{\text{exact}}$ selon la dimension d'origine des séquences pour les jeux *Clust1*. Tous les sous-espaces sont de dimension 5.

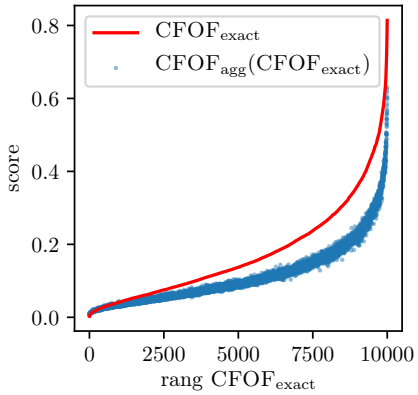
Dimension	Nbr. sous-espaces	Corrélation de SPEARMAN
20	4	0.9972
50	10	0.9971
100	20	0.9970
200	40	0.9971

La figure 4.1 montre que les approximations de la valeur du score sont moins bonnes que celles obtenues au chapitre 3, et se dégradent quand le nombre de sous-espaces augmente. Mais la valeur elle-même du score n'est pas cruciale, le point essentiel est de maintenir l'ordre des scores afin de pouvoir identifier comme anomalies les séquences ayant les scores les plus forts

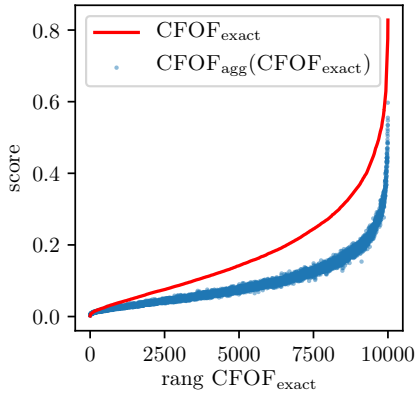
4. https://github.com/luk-f/pyCFOF & https://github.com/luk-f/pyCFOFiSAX



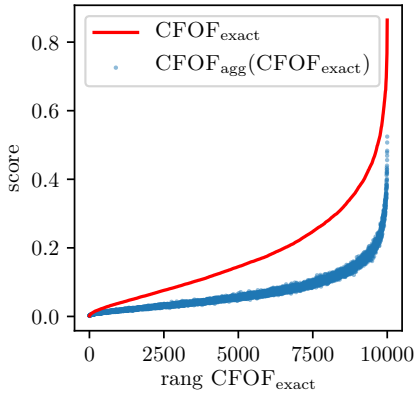
(a) Dimension 20 et 4 arbres.



(b) Dimension 50 et 10 arbres.



(c) Dimension 100 et 20 arbres.



(d) Dimension 200 et 40 arbres.

FIGURE 4.1 – Comparaison, pour quatre jeux *Clust1* de dimension 20, 50, 100 et 200, des scores CFOF$_{exact}$ et des scores CFOF$_{agg}$(CFOF$_{exact}$), calculés à partir de plusieurs scores CFOF$_{exact}$ provenant de différents sous-espaces de dimension 5. Pour chaque jeu, les scores CFOF$_{agg}$(CFOF$_{exact}$) sont triés selon le rang des scores CFOF$_{exact}$ associés.

relativement aux autres. Sur ce point, le tableau 4.3 indique que cet ordre est globalement bien préservé, avec des valeurs de corrélation de SPEARMAN élevées et qui varient très peu avec le nombre de sous-espaces (pour des sous-espaces de même dimension).

4.4.1.2 Approximation du score $\mathbf{CFOF_{exact}}$ à partir des scores $\mathbf{CFOF_{agg}(CFOF_{approx})}$ et $\mathbf{CFOF_{agg}(v\text{-}rang)}$

Avant de rapporter dans la section suivante les résultats détaillés obtenus sur les jeux *Clust2*, nous donnons sur un des jeux de la série *Clust1* un aperçu du comportement des deux méthodes $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et $\text{CFOF}_{\text{agg}}(v\text{-}rang)$. Nous avons choisi le jeu en 20 dimensions réparties en 4 arbres de dimension 5. Nous observons les résultats correspondants sur la figure 4.2. Nous remarquons que comme pour $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{exact}})$ dans la section précédente, $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ semble sous-estimer le score $\text{CFOF}_{\text{exact}}$, alors que $\text{CFOF}_{\text{agg}}(v\text{-}rang)$ le surestime (sauf pour les séquences ayant un score $\text{CFOF}_{\text{exact}}$ élévé). Toutefois, ici encore, le rang est lui bien préservé, avec des corrélations de SPEARMAN vis-à-vis de $\text{CFOF}_{\text{exact}}$ de 0.9961 pour $\text{CFOF}_{\text{agg}}(v\text{-}rang)$ et de 0.9983 pour $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$.

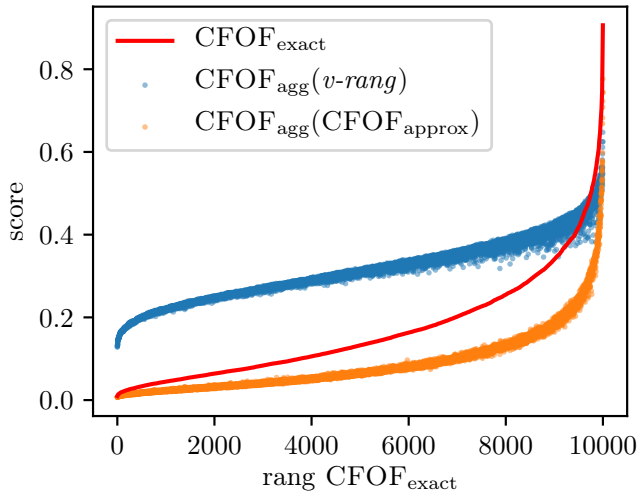


FIGURE 4.2 – Comparaison des scores $\text{CFOF}_{\text{exact}}$, $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et $\text{CFOF}_{\text{agg}}(v\text{-}rang)$ pour *Clust1* en 20 dimensions avec 4 arbres de dimensions 5. Tous les scores sont triés selon les valeurs de $\text{CFOF}_{\text{exact}}$.

En ce qui concerne le parcours des arbres, nous nous intéressons au nombre moyen de nœuds visités pour calculer le score d'un objet tel que défini Section 3.6.1.2. Pour rappel, pour calculer le score d'un objet $\boldsymbol{q}$, il faut parcourir pour chaque objet de référence $\boldsymbol{p}$ un certain nombre de nœuds dans l'arbre d'indexation. Le nombre moyen de nœuds visités pour calculer le score de $\boldsymbol{q}$ est alors défini comme étant le nombre total de nœuds visités pour l'ensemble des objets de référence divisé par le nombre d'objets de référence. Comme à présent la base de référence a été projetée sur plusieurs

arbres (4 dans le cas présent) nous récupérons le nombre moyen de nœuds visités pour chacun de ces arbres, et nous retenons le minimum, le maximum, la moyenne et la médiane de ces quatre valeurs. Enfin, nous noterons que les nombres de nœuds visités sont les mêmes pour les deux méthodes $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et $\text{CFOF}_{\text{agg}}(v\text{-}rang)$, car en fait elles ne diffèrent que dans la manière de réaliser les agrégations et non dans les élagages et les parcours d'arbres.

Les résultats sont présentés Figure 4.3 et indiquent des moyennes de nombres de nœuds visités situées plutôt entre 830 et 900. Dans cette expérience le plus grand des arbres possède 973 nœuds, c'est donc la limite haute du nombre de nœuds pouvant être parcourus. Si nous totalisons sur les 4 arbres ceci représente un nombre moyen de nœuds visités strictement inférieur à 4000. Sur le tableau 4.1 de la section 4.2, nous avons constaté que le nombre de nœuds de niveau 1, pour un arbre d'indexation contenant des séquences *Clust1* de dimension 20, était d'environ 10000. Remplacer la visite nécessaire de ces 10000 nœuds de niveau 1 par un nombre moyen de nœuds visités de moins de 4000 représente une réduction de 60%. De plus, si le traitement est parallélisé sur les 4 arbres, dans ce cas le coût en temps sera celui du plus long parcours, c'est-à-dire moins de 1000 nœuds, pouvant permettre selon les choix d'implantation un gain allant jusqu'à 90%.

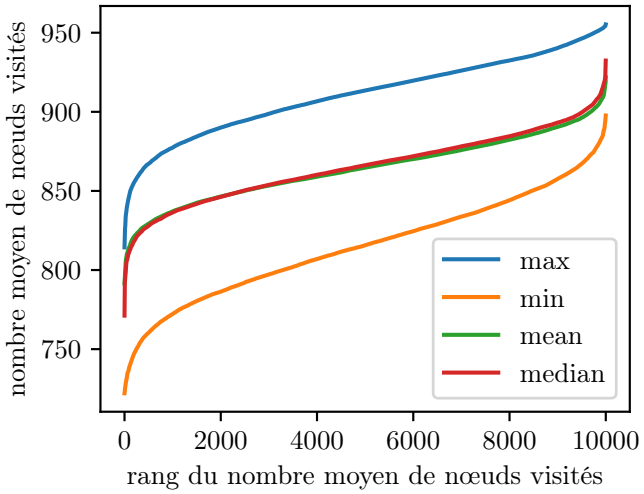


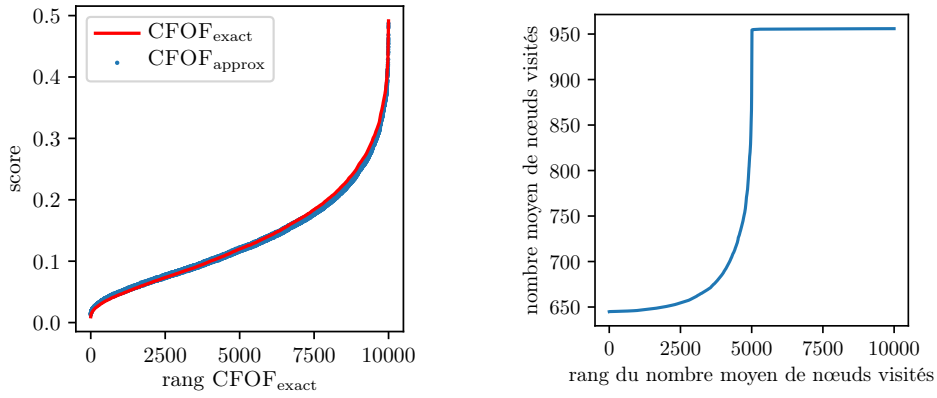
FIGURE 4.3 – Statistiques des nombres moyens de nœuds visités sur chacun des 4 arbres de dimension 5 pour le calcul d'un score sur le jeu *Clust1* en dimension 20. Les points sont triés par valeur croissante du nombre moyen de nœuds visités.

4.4.2 Évaluations sur les jeux de la famille *Clust2*

À présent nous comparons les approches sur les jeux *Clust2* en dimension 20 et 200 en faisant varier le nombre de sous-espaces pour évaluer la pertinence des approximations obtenues selon le nombre d'arbres et la dimension des sous-espaces.

Nous rappellerons les comportements obtenus avec un seul arbre (un seul sous-espace égal à l'espace d'origine) puis nous indiquerons les résultats obtenus avec 2, 4, 5, 10 et enfin 20 sous-espaces. Pour le jeu en dimension 200, nous présenterons également les expériences réalisées pour 40, 100 et 200 sous-espaces.

4.4.2.1 Avec *Clust2* en 20 dimensions

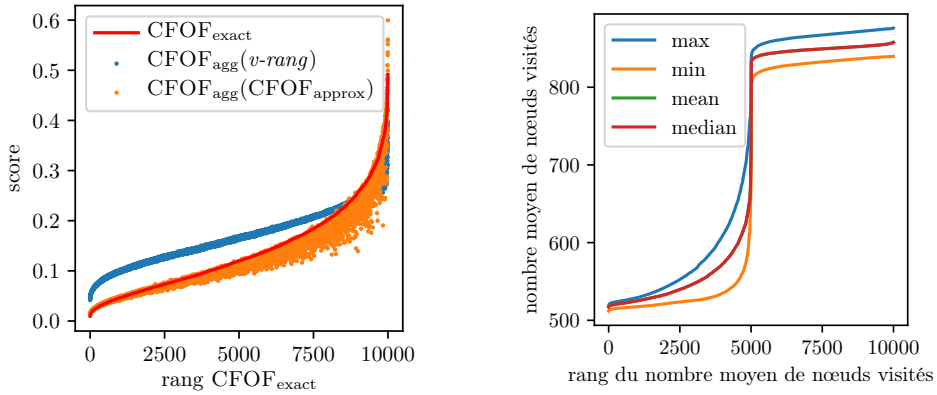


(a) Comparaison des scores.

(b) Nœuds visités.

FIGURE 4.4 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{approx}$, triés selon le rang des scores CFOF$_{exact}$, et nombres moyens de nœuds visités pour le calcul de CFOF$_{approx}$, triés par ordre croissant, pour *Clust2* en dimension 20 avec un seul arbre.

Comme repère de base, nous présentons de nouveau sur la figure 4.4 les résultats obtenus pour *Clust2* en dimension 20 avec un seul arbre dans la section 3.6.1. Les mêmes paramètres que dans la section 3.6.1 ont été utilisés mais ce jeu aléatoire a été régénéré et les résultats peuvent donc être légèrement différents ici. La figure confirme de nouveau que les scores CFOF$_{approx}$ sont très proches des valeurs de CFOF$_{exact}$ et le calcul donne une corrélation de SPEARMAN de 0.9996. Le nombre moyen de nœuds visités, visible sur la figure 4.4b, varie de 645 à 956 pour un arbre qui en contient 1162.



(a) Comparaison des scores.

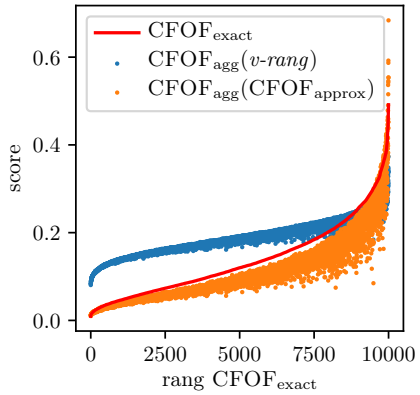
(b) Nœuds visités.

FIGURE 4.5 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 2 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 20 avec 2 arbres de dimension 10. Pour les nombres de nœuds visités la moyenne et la médiane sont superposées.

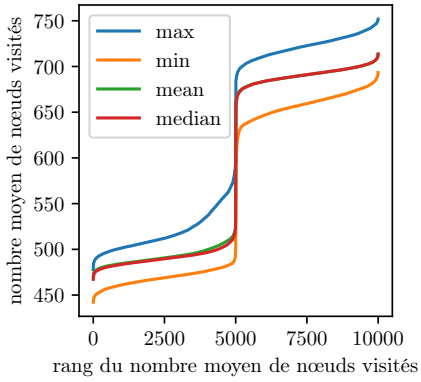
Nous observons les scores CFOF$_{agg}$(CFOF$_{approx}$) et CFOF$_{agg}$(*v-rang*), et les quantités moyennes de nœuds parcourus avec deux arbres de dimension 10 sur la figure 4.5. Les deux scores obtiennent de fortes corrélations de SPEARMAN vis-à-vis du score CFOF$_{exact}$, avec une valeur de 0.9994 pour CFOF$_{agg}$(*v-rang*) et de 0.9955 pour CFOF$_{agg}$(CFOF$_{approx}$). Ces valeurs sont très proches, mais nous notons que les scores CFOF$_{agg}$(CFOF$_{approx}$) semblent plus dispersés que les scores CFOF$_{agg}$(*v-rang*). Le nombre moyen de nœuds parcourus, avec un minimum de 512 et un maximum de 876, diminue par rapport aux résultats obtenus avec un seul arbre.

En ce qui concerne les résultats pour *Clust2* en dimension 20 avec respectivement 4 et 5 arbres de dimensions 5 et 4, les résultats sont visibles sur les figures 4.6 et 4.7. La réduction du nombre moyen de nœuds visités se poursuit, avec pour 5 arbres un minimum de 394 et un maximum de 757. Les dispersions des scores CFOF$_{agg}$(*v-rang*) et CFOF$_{agg}$(CFOF$_{approx}$) s'accroissent, mais les corrélations de SPEARMAN restent élevées avec des valeurs supérieures à 0.98 dans tous les cas.

Sur les expérimentations avec 10 et 20 arbres de dimensions 2 et 1, présentées sur les figures 4.8b et 4.9b, la tendance s'amplifie : les nombres moyens

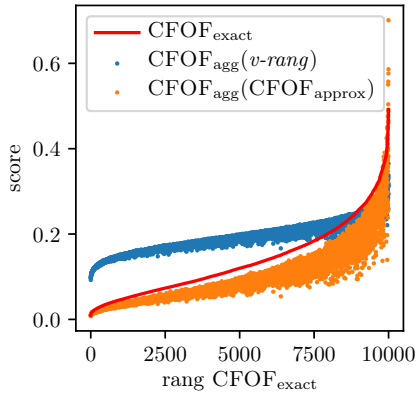


(a) Comparaison des scores.

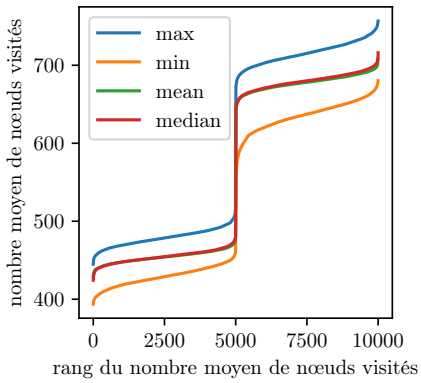


(b) Nœuds visités.

FIGURE 4.6 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 4 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 20 avec 4 arbres de dimension 5.



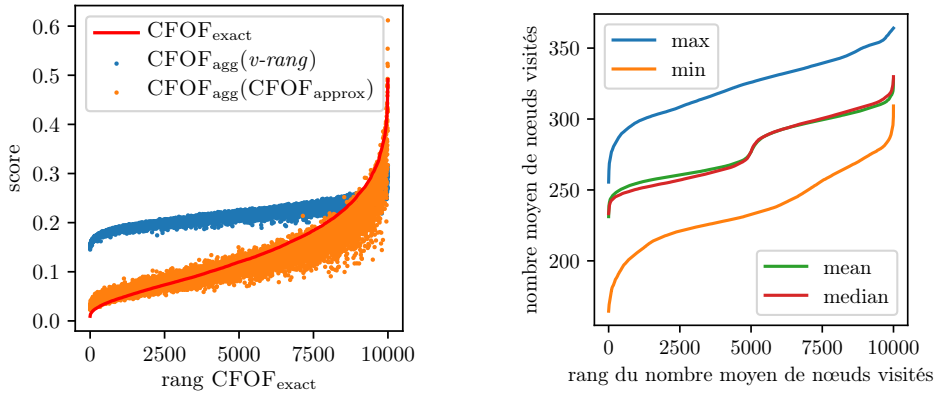
(a) Comparaison des scores.



(b) Nœuds visités.

FIGURE 4.7 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 5 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 20 avec 5 arbres de dimension 4.

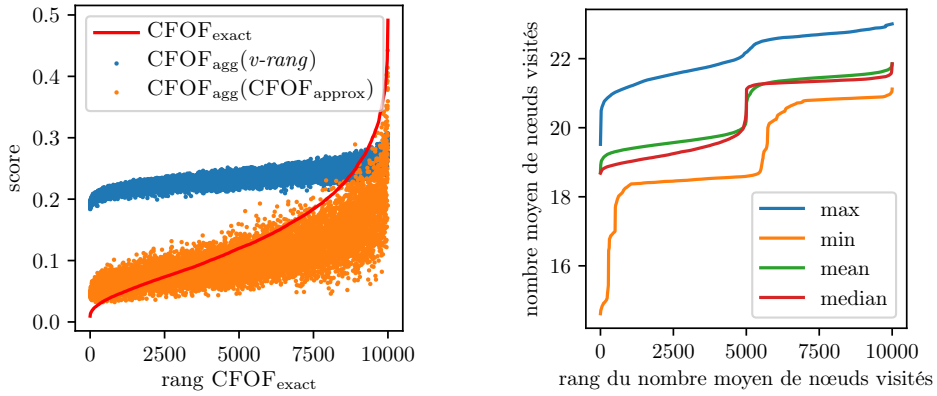
de nœuds visités sont beaucoup plus bas que dans les expérimentations précédentes. Pour 10 arbres de dimension 2, le minimum est de 165 tandis que le



(a) Comparaison des scores.

(b) Nœuds visités.

FIGURE 4.8 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 10 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 20 avec 10 arbres de dimension 2.



(a) Comparaison des scores.

(b) Nœuds visités.

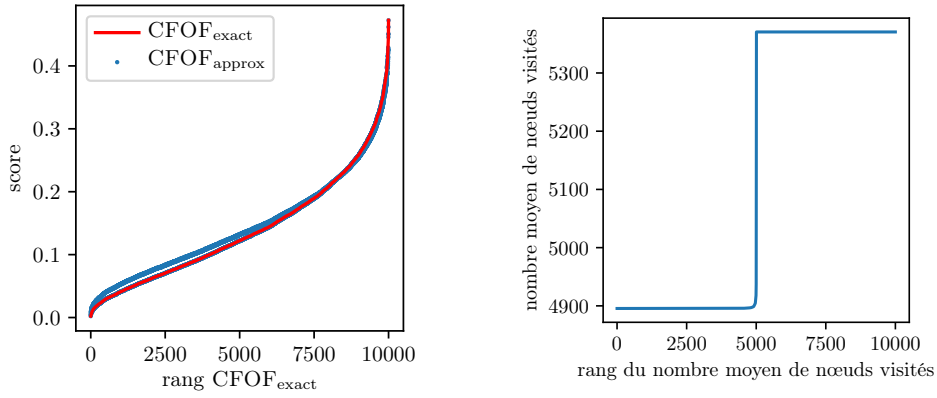
FIGURE 4.9 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 20 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 20 avec 20 arbres de dimension 1.

maximum est de 364. Avec 20 arbres de dimension 1, ces valeurs passent à 23 et 15. Cependant, nous constatons que la dispersion des scores s'accroît en-

core, et que les résultats de corrélation de SPEARMAN diminuent. Ils restent bons pour 10 arbres, avec des valeurs supérieures à 0.96 pour les deux scores CFOF_{agg}, mais diminuent pour 20 arbres à 0.8757 pour $\text{CFOF}_{\text{agg}}(v\text{-}rang)$ et 0.8519 pour $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$. Dans cette série d'expériences sur le jeu *Clust2* en dimension 20, il semble que 10 arbres en dimension 2 soit un bon compromis entre la qualité de l'approximation et le coût de calcul.

4.4.2.2 Avec *Clust2* en 200 dimensions

Comme dans la section précédente, nous évaluons les scores CFOF_{agg} et le nombre moyen de nœuds parcourus, mais cette fois pour le jeu *Clust2* en dimension 200.



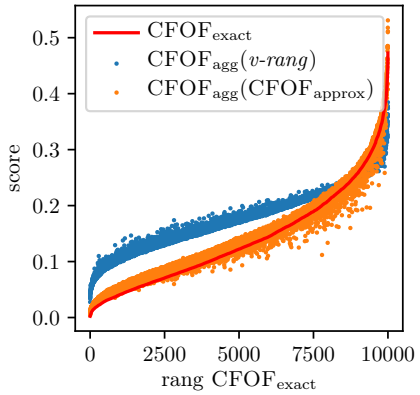
(a) Comparaison des scores.

(b) Nœuds visités.

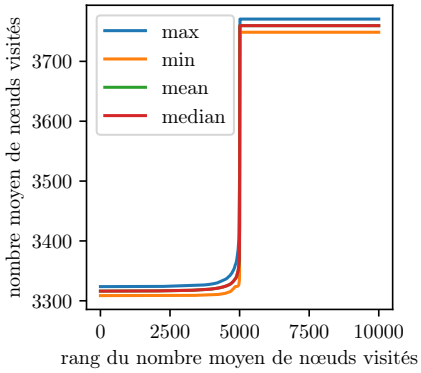
FIGURE 4.10 – Comparaison des scores $\text{CFOF}_{\text{exact}}$ avec les scores $\text{CFOF}_{\text{approx}}$, triés selon le rang des scores $\text{CFOF}_{\text{exact}}$, et nombres moyens de nœuds visités pour le calcul de $\text{CFOF}_{\text{approx}}$, triés par ordre croissant, pour *Clust2* en dimension 200 avec un seul arbre.

Ici aussi, nous donnons sur la figure 4.10 les résultats pour le calcul de $\text{CFOF}_{\text{approx}}$ avec un seul arbre comme point de départ. La corrélation de SPEARMAN pour ce score est très bonne avec une valeur de 0.9972.

Nous observons une amélioration du nombre moyen de nœuds visités avec 2 arbres sur la figure 4.11 tout en conservant de bons résultats de corrélation de SPEARMAN, que ce soit pour le score $\text{CFOF}_{\text{agg}}(v\text{-}rang)$ ou $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$, avec des valeurs de corrélation respectives de 0.9972 et 0.9945, même si nous constatons un début de dispersion des scores.

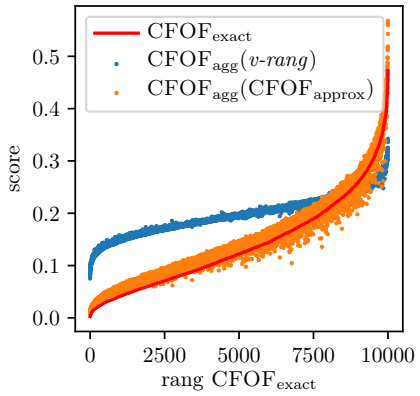


(a) Comparaison des scores.

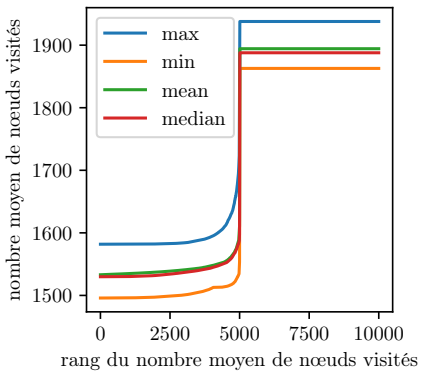


(b) Nœuds visités.

FIGURE 4.11 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 2 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 200 avec 2 arbres de dimension 100. Pour les nombres de nœuds visités la moyenne et la médiane sont superposées.



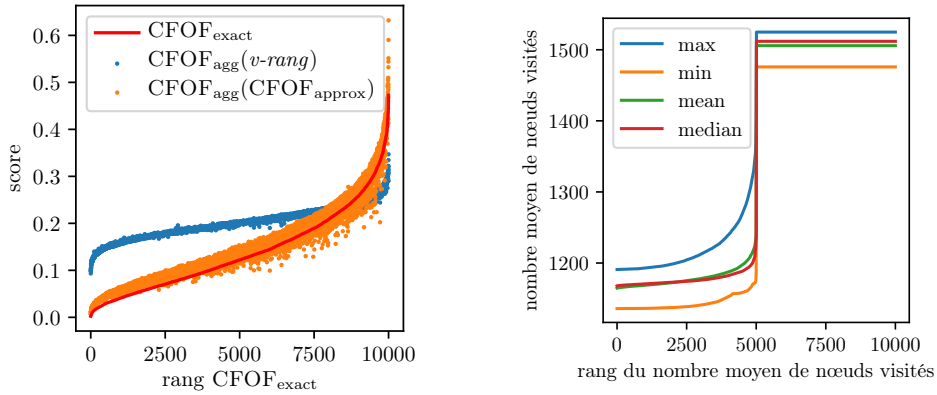
(a) Comparaison des scores.



(b) Nœuds visités.

FIGURE 4.12 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 4 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 200 avec 4 arbres de dimension 50.

En augmentant le nombre d'arbres à 4, chacun de dimension 50, puis



(a) Comparaison des scores.

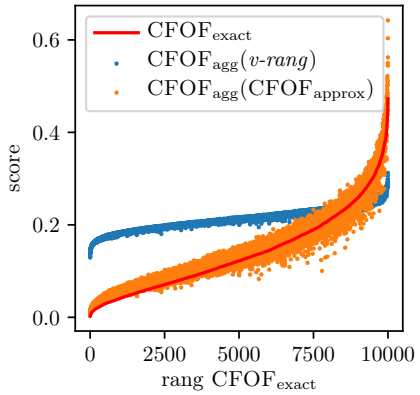
(b) Nœuds visités.

FIGURE 4.13 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 5 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 200 avec 5 arbres de dimension 40.

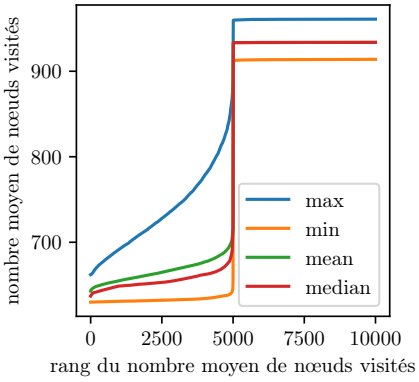
à 5 de dimension 40 le nombre de nœuds visités continue de diminuer. Ces résultats sont visibles sur les figures 4.12 et 4.13. Les corrélations de SPEARMAN restent bonnes (la plus petite étant de 0.9928, obtenue pour CFOF$_{agg}$(CFOF$_{approx}$) avec 5 arbres).

Les résultats pour 10 et 20 arbres sont donnés Figures 4.14 et 4.15. Le nombre moyen de nœuds visités varie entre un minimum de 495 avec 20 arbres et un maximum de 961 avec 10 arbres. Rappelons que le nombre moyen de nœuds visités maximum avec un seul arbre dépassait les 5000 (visible sur la figure 4.10b). L'augmentation du nombre d'arbres et la diminution des dimensions des sous-espaces a permis de diviser le nombre moyen de nœuds visités par plus de 5, tout en conservant ici des coefficients de corrélation de SPEARMAN au-dessus 0.983 dans tous les cas. Dans cette configuration, les dimensions des arbres sont identiques à ceux de *Clust2* en dimension 20 avec 1 et 2 arbres. Les résultats du nombre de nœuds visités sont également très proches et seront comparés dans la section 4.4.2.3.

Avec 40 arbres de dimension 5 (résultats présentés sur la figure 4.16), les nombres moyens de nœuds visités poursuivent leur diminution, accompagnée d'une légère baisse de corrélation de SPEARMAN, qui passe à 0.9712 pour CFOF$_{agg}$(*v-rang*) et à 0.9826 pour CFOF$_{agg}$(CFOF$_{approx}$). Nous remarquons cependant que les scores CFOF$_{agg}$(CFOF$_{approx}$) sous-estiment davantage les

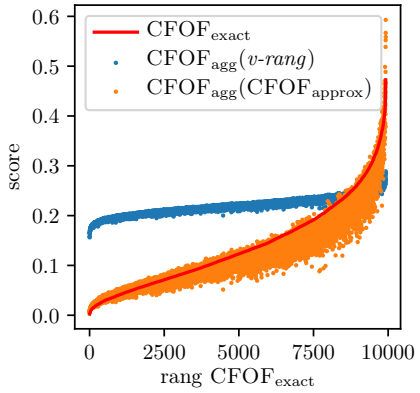


(a) Comparaison des scores.

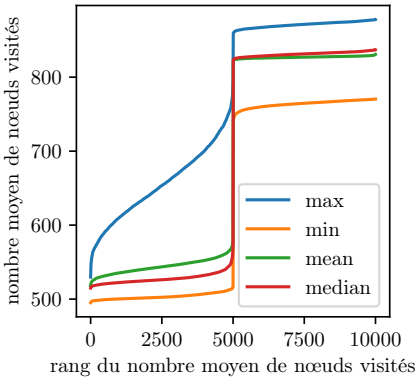


(b) Nœuds visités.

FIGURE 4.14 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 10 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 200 avec 10 arbres de dimension 20.



(a) Comparaison des scores.

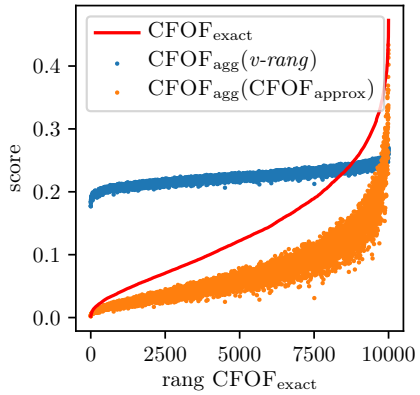


(b) Nœuds visités.

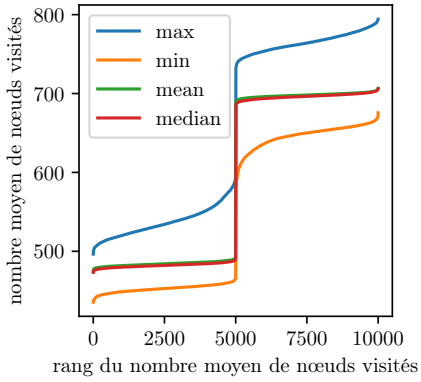
FIGURE 4.15 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 20 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 200 avec 20 arbres de dimension 10.

valeurs des scores CFOF$_{exact}$. Ils se retrouvent tous en dessous de la courbe des CFOF$_{exact}$.

Tout comme pour *Clust2* en dimension 20, des différences plus notables

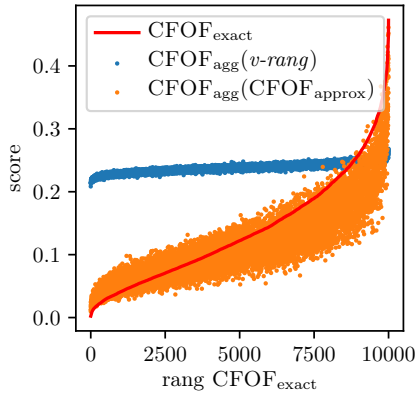


(a) Comparaison des scores.

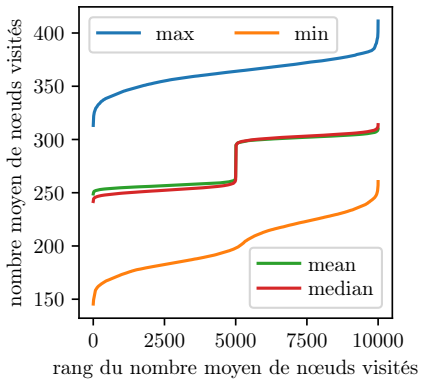


(b) Nœuds visités.

FIGURE 4.16 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 40 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 200 avec 40 arbres de dimension 5.



(a) Comparaison des scores.

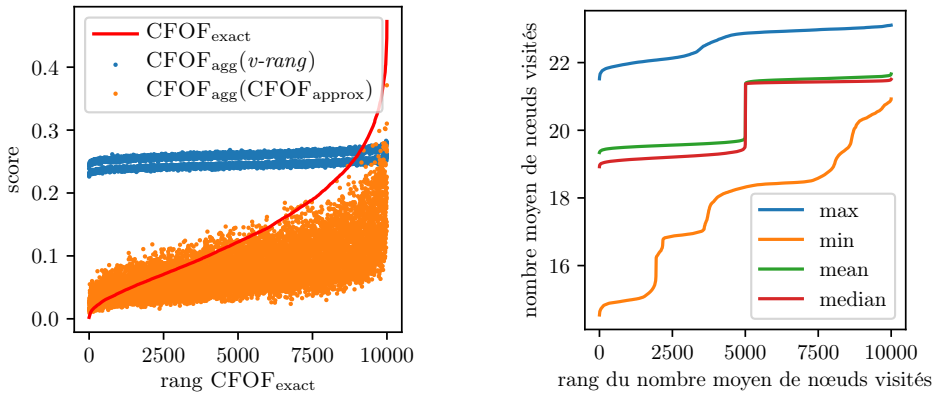


(b) Nœuds visités.

FIGURE 4.17 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 100 arbres pour le calcul de CFOF$_{agg}$, triés par ordre croissant, pour *Clust2* en dimension 200 avec 100 arbres de dimension 2.

apparaissent lorsque les dimensions des sous-espaces diminuent jusqu'à une valeur de 2 (Figure 4.17). Le nombre moyen de nœuds visités varie alors

entre 145 et 411 (l'arbre le plus grand en contenant 1127). Mais les résultats des corrélations de SPEARMAN diminuent plus nettement, et atteignent les valeurs de 0.9316 pour $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et de 0.9427 pour $\text{CFOF}_{\text{agg}}(v\text{-}rang)$.



(a) Comparaison des scores.

(b) Nœuds visités.

FIGURE 4.18 – Comparaison des scores $\text{CFOF}_{\text{exact}}$ avec les scores CFOF_{agg}, triés selon le rang des scores $\text{CFOF}_{\text{exact}}$, et quatre mesures calculées à partir des nombres moyens de nœuds visités sur chacun des 200 arbres pour le calcul de CFOF_{agg}, triés par ordre croissant, pour *Clust2* en dimension 200 avec 200 arbres de dimension 1.

Enfin, avec 200 arbres de dimension 1 (Figure 4.18) le nombre moyen de nœuds visités est compris entre 14 et 23, et il est largement inférieur au nombre de nœuds total de l'arbre le plus grand qui en contient 221. Mais les scores CFOF_{agg} obtenus sont beaucoup moins satisfaisants qu'auparavant, avec des corrélations de SPEARMAN de 0.4895 pour $\text{CFOF}_{\text{agg}}(v\text{-}rang)$ et de 0.7225 pour $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$. Si l'on regarde les scores CFOF_{agg} dans chacun des clusters du jeu (Figure 4.19) nous distinguons clairement les scores des deux groupes $\mathcal{N}(1,0)$ et $\mathcal{N}(4,0.5)$, et la dispersion plus faible du score $\text{CFOF}_{\text{agg}}(v\text{-}rang)$ au sein des clusters. Ceci lui permet d'obtenir de meilleures corrélations de SPEARMAN lorsque nous les calculons séparément sur chacun des clusters, comme présenté dans le tableau 4.4.

4.4.2.3 Synthèse des résultats en dimension 20 et 200

La figure 4.20 fournit une vue d'ensemble des résultats précédents pour *Clust2* en dimension 20 et en dimension 200.

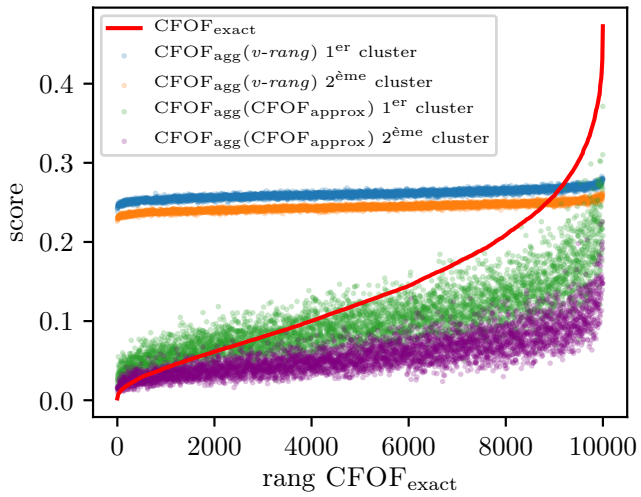


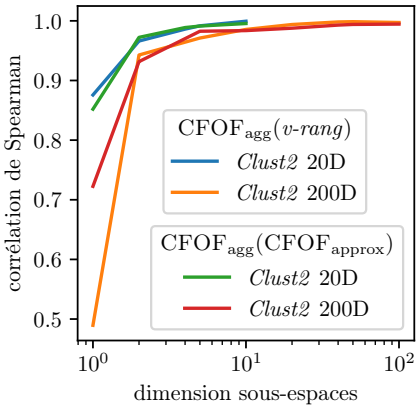
FIGURE 4.19 – Comparaison des scores CFOF$_{exact}$ avec les scores CFOF$_{agg}$, triés selon le rang des scores CFOF$_{exact}$, pour les deux groupes de séquences de *Clust2* en dimension 200 avec 200 arbres de dimension 1.

TABLE 4.4 – Comparaison des corrélations de SPEARMAN selon le type de score CFOF$_{agg}$ utilisé et le groupe de séquences.

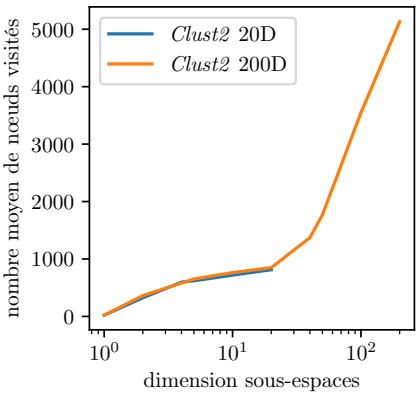
	CFOF$_{agg}$(*v-rang*)	CFOF$_{agg}$(CFOF$_{approx}$)
tout groupe confondu	0.4895	0.7225
1er groupe	0.9090	0.8592
2ème groupe	0.9200	0.8682

Pour la corrélation de SPEARMAN, nous observons sur la figure 4.20a qu'à partir d'une dimensionnalité de sous-espace au moins égale à 2, elle est au-dessus de 0.93 dans tous les cas, et toujours au-dessus de 0.96 à partir des sous-espaces de dimension 10. Nous remarquons également que sauf dans le cas particulier de sous-espaces de dimension 1, les deux méthodes d'agrégation CFOF$_{agg}$(CFOF$_{approx}$) et CFOF$_{agg}$(*v-rang*) obtiennent des performances très proches.

Sur la figure 4.20b, nous constatons que le nombre moyen de nœuds visités varie peu entre la dimension 4 et la dimension 20. Elle devient fortement croissante après la dimension 20, et reste très faible lorsque la dimension des sous-espaces est de 1 mais avec une corrélation de SPEARMAN toujours très inférieure à 0.9.



(a) Corrélation de Spearman.



(b) Nœuds visités.

Figure 4.20 – Corrélations de Spearman et nombres moyens de nœuds visités, pour *Clust2* en dimension 20 et 200 selon la dimension des sous-espaces utilisés pour l'obtention des scores CFOF$_{agg}$.

4.4.3 Évaluation sur les jeux du benchmark NAB

Pour la suite des expérimentations, nous évaluons notre méthode sur les jeux du benchmark *Numenta Anomaly Benchmark* (NAB) [Lavin et Ahmad, 2015], décrit dans la section 3.6.2, dont les jeux sont mis à disposition par les auteurs sur GitHub [5].

Dans ces expérimentations sur les séries NAB, les objets d'un jeu sont des séquences formées à partir de fenêtres glissantes sur une série temporelle. Les composantes des objets sont donc bien identiquement distribuées mais en revanche ne sont pas forcément indépendantes. Nous verrons cependant que ceci ne semble pas dégrader nos résultats.

Dans les sections précédentes, les deux approximations à partir de plusieurs arbres CFOF$_{agg}$(*v-rang*) et CFOF$_{agg}$(CFOF$_{approx}$) ont atteint des corrélations de Spearman assez similaires, ne permettant pas de choisir l'une plutôt que l'autre. Nous utilisons donc les deux lors de l'évaluation sur le benchmark NAB, mais pour ne pas alourdir la présentation nous nous concentrons plutôt sur CFOF$_{agg}$(CFOF$_{approx}$) dans cette section. Les résultats numériques détaillés pour les deux approches sont donnés dans les annexes (les performances pour CFOF$_{agg}$(*v-rang*) restant globalement similaires à celles de CFOF$_{agg}$(CFOF$_{approx}$)).

Les dimensions des séquences provenant de l'encodage des jeux du benchmark NAB (cf. Section 3.6.2) restent modestes (8, 10 et 12 selon les jeux),

5. https://github.com/numenta/NAB

elles ne se prêtent donc pas forcément à l'obtention de gains sur les parcours avec $\text{CFOF}_{\text{agg}}(\textit{v-rang})$ et $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$, mais l'évaluation a ici comme but premier de voir si les performances de détection des méthodes sont maintenues.

Concernant la répartition des dimensions dans les sous-espaces, les jeux encodés contiennent 8, 10 ou 12 dimensions selon les séries et, pour rappel, la première moitié de ces dimensions contient des valeurs de moyennes glissantes et la seconde moitié contient des valeurs d'écarts-types décrivant localement la série. Nous allons dans ce qui suit comparer les résultats de $\text{CFOF}_{\text{approx}}$ (c'est-à-dire l'approximation avec un seul arbre proposée au chapitre 3) avec des approximations utilisant 2 et 4 arbres. Dans les expériences avec 2 arbres, la première moitié des dimensions sera utilisée pour le premier arbre et les autres dimensions pour le second, formant alors des sous-espaces de tailles 4, 5 ou 6 selon les jeux. Pour 4 arbres, les dimensions seront toujours prises dans l'ordre et réparties de la façon suivante. Pour les jeux ayant 8 dimensions nous formerons 4 sous-espaces de tailles 2, 2, 2 et 2. Pour les jeux à 10 dimensions les 4 sous-espaces auront les tailles 3, 3, 2 et 2. Et enfin pour les jeux contenant 12 dimensions nous aurons 4 sous-espaces de tailles 3, 3, 3 et 3.

4.4.3.1 Qualité de la détection

De la même façon que pour les résultats de détection de la section 3.6.2.3, nous évaluons les nouvelles méthodes à l'aide des deux indicateurs ROC AUC et PRC AUC présentés dans la section 2.3.

Nous observons les différences de scores ROC AUC et PRC AUC entre les scores CFOF approximés (avec 1, 2 et 4 arbres) respectivement sur les figures 4.21 et 4.22. Pour ROC AUC et PRC AUC globalement les différences de qualité sont faibles entre 1, 2 et 4 arbres, comme par exemple pour la famille de séries "realTweets". Certains jeux conduisent toutefois à des différences plus marquées comme "ambient_temperature_system_failure" de la famille "realKnownCause" pour le score PRC AUC.

Les résultats ROC AUC moyens, donnés dans la table 4.5 à la page 155, sont également du même ordre quel que soit le type d'approximations utilisé. Le score $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ avec 2 arbres retourne la meilleure moyenne avec une dispersion très proche de $\text{CFOF}_{\text{approx}}$ et $\text{CFOF}_{\text{exact}}$. La seconde meilleure moyenne est obtenue par $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ avec 4 arbres. Les scores $\text{CFOF}_{\text{agg}}(\textit{v-rang})$ avec 2 et 4 arbres sont respectivement $5^{\text{ième}}$ et $6^{\text{ième}}$ au classement.

Comme pour ROC AUC, les résultats PRC AUC moyens, visibles dans la table 4.6 à la page 156, sont similaires entre les différents types de méthodes

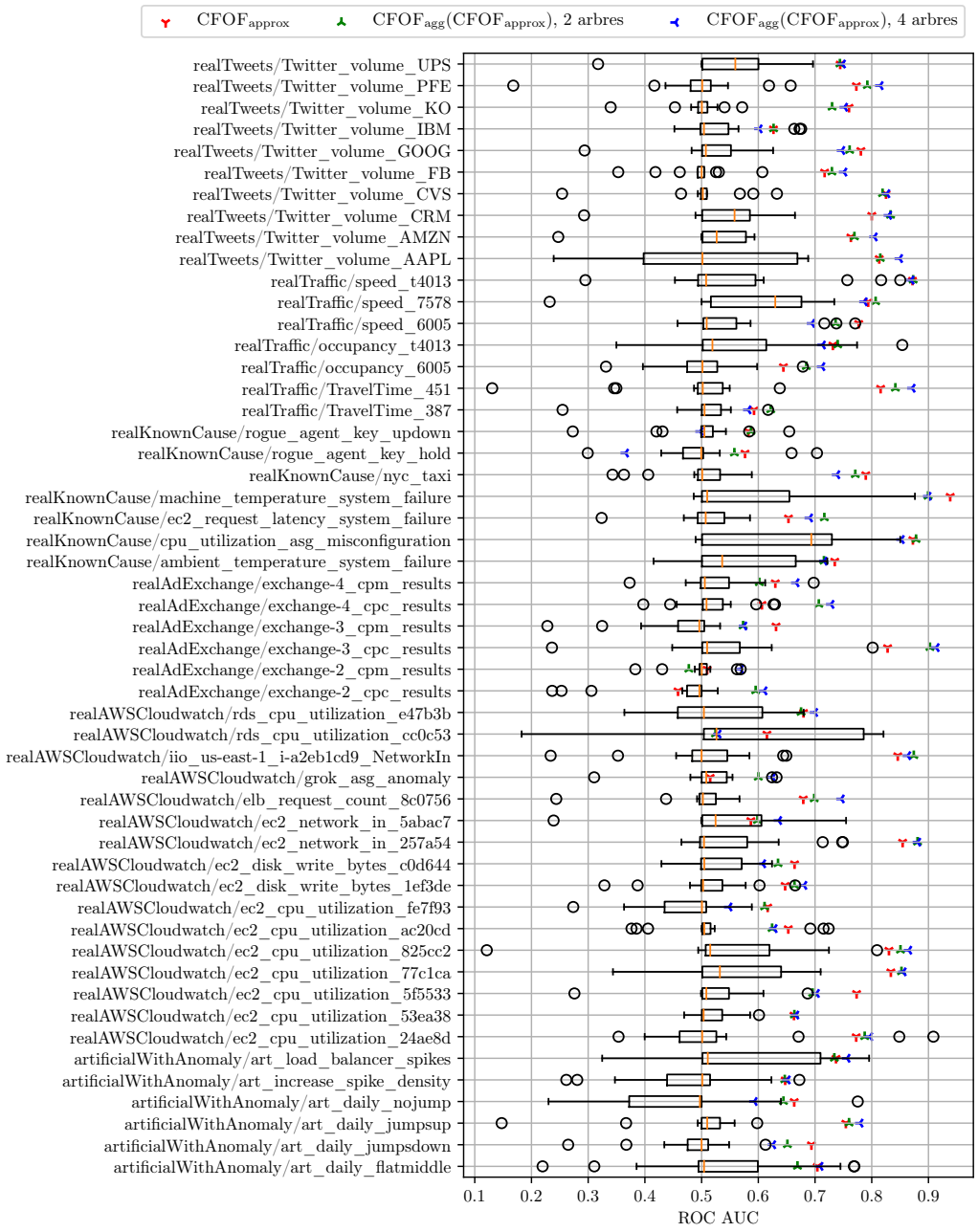


FIGURE 4.21 – Comparaison des résultats ROC AUC sur les séries du benchmark NAB pour CFOF$_{approx}$ et CFOF$_{agg}$(CFOF$_{approx}$) avec 2 et 4 arbres. Les autres méthodes évaluées dans le benchmark NAB sont présentées sous forme de boîtes à moustache.

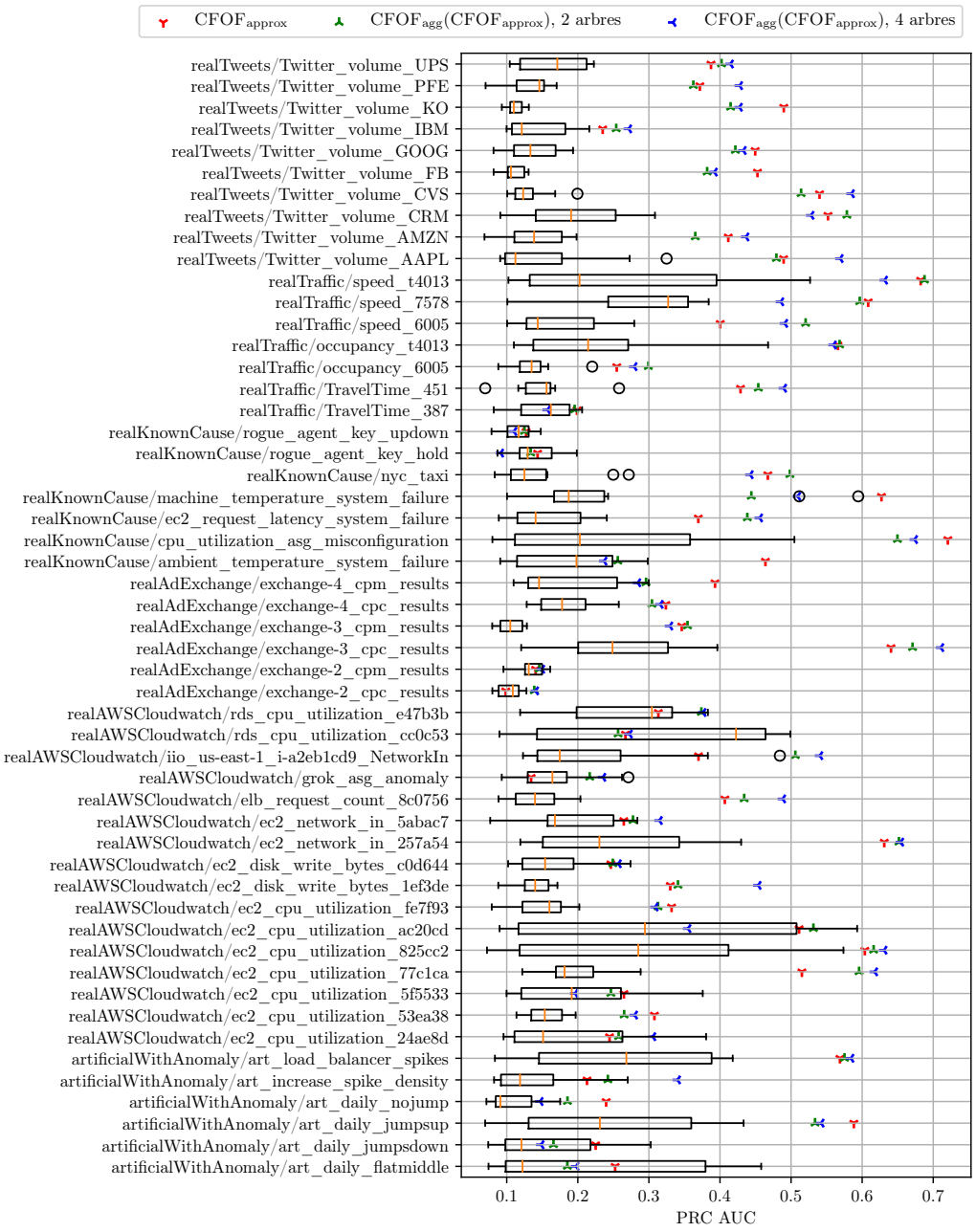


FIGURE 4.22 – Comparaison des résultats PRC AUC sur les séries du benchmark NAB pour $\text{CFOF}_{\text{approx}}$ et $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ avec 2 et 4 arbres. Les autres méthodes évaluées dans le benchmark NAB sont présentées sous forme de boîtes à moustache.

TABLE 4.5 – Moyenne et écart-type des scores ROC AUC pour l'ensemble des méthodes de détection ainsi que leur classement.

	moyenne	écart-type	class.
bayesChangePt	0.503225	0.008663	18
$\text{CFOF}_{\text{approx}}$	0.712901	0.103345	3
CFOF_{agg}(*v-rang*), 2 arbres	0.703918	0.113742	5
$\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$, 2 arbres	**0.716321**	**0.104541**	1
CFOF_{agg}(*v-rang*), 4 arbres	0.703635	0.113335	6
$\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$, 4 arbres	0.715952	0.116041	2
$\text{CFOF}_{\text{exact}}$	0.708183	0.100501	4
contextOSE	0.347730	0.136428	21
earthgeckoSkyline	0.501772	0.002408	19
expose	0.562708	0.113633	10
htmjava	0.578539	0.138069	7
knncad	0.536839	0.092588	14
null	0.500000	0.000000	20
numenta	0.558001	0.127363	12
numentaTM	0.560009	0.146204	11
random	0.504226	0.016633	16
randomCutForest	0.573264	0.106666	8
relativeEntropy	0.504263	0.004539	15
skyline	0.564601	0.076908	9
twitterADVec	0.503917	0.003585	17
windowedGaussian	0.540573	0.126220	13

CFOF. Pour rappel, comme expliqué dans la section 3.6.2.3, nous indiquons pour chaque méthode le nombre de séries ayant au moins 10 points pour tracer la courbe PRC, et les méthodes (sauf *baseline*) qui ne respectent jamais ce critère (c'est-à-dire pour l'ensemble des séries du benchmark NAB) n'apparaissent donc pas dans le tableau. C'est la méthode $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ avec 4 arbres qui donne la meilleure moyenne de score. $\text{CFOF}_{\text{approx}}$ est en seconde position et $\text{CFOF}_{\text{exact}}$ en troisième.

Quelle que soit la méthode d'approximation employée, la qualité du score se maintient. Nous constatons tout de même une légère baisse de qualité avec CFOF_{agg}(*v-rang*), mais sans réellement compromettre les performances vis-à-vis des autres méthodes de détection.

Au regard des résultats obtenus avec ROC AUC et PRC AUC, les scores $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ semblent meilleurs que les scores CFOF_{agg}(*v-rang*).

TABLE 4.6 – Moyenne et écart-type des scores PRC AUC pour l'ensemble des méthodes de détection ainsi que leur classement. La dernière colonne indique, pour chaque méthode de détection, le nombre de séries temporelles ayant au moins 10 points sur la courbe PRC.

	moyenne	écart-type	class.	nbr. pts ≥ 10
baseline	0.113063	0.007496	17	0
bayesChangePt	0.129687	0.043641	14	50
$\text{CFOF}_{\text{approx}}$	0.388696	0.159583	2	52
CFOF_{agg}(*v-rang*), 2 arbres	0.381192	0.158144	5	52
$\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$, 2 arbres	0.384078	0.159156	4	52
CFOF_{agg}(*v-rang*), 4 arbres	0.363731	0.160450	6	52
$\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$, 4 arbres	**0.390028**	**0.163454**	1	52
$\text{CFOF}_{\text{exact}}$	0.387604	0.157213	3	52
contextOSE	0.121671	0.071690	15	51
expose	0.169448	0.076705	12	52
htmjava	0.246111	0.110690	8	52
knncad	0.136964	0.038967	13	52
numenta	0.235370	0.103608	9	52
numentaTM	0.246172	0.117930	7	52
random	0.116469	0.011753	16	52
randomCutForest	0.221088	0.115606	10	52
windowedGaussian	0.203955	0.118911	11	52

Les écarts entre les résultats sont minces, mais les performances de l'approche $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ restent toujours meilleures, nous faisons donc le choix de ne travailler qu'avec ce score pour la suite des expérimentations.

4.4.3.2 Qualité de l'approximation $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$

Nous souhaitons à présent observer la qualité de l'approximation des scores $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$.

Les résultats de corrélation de SPEARMAN vis-à-vis des scores $\text{CFOF}_{\text{exact}}$, présentés sur la figure 4.23, montrent clairement une dégradation de la qualité de l'approximation lorsque le nombre d'arbres augmente. Aucune série n'est épargnée par le phénomène, bien que certaines soient plus touchées que d'autres. Par exemple, la famille de séries "realTweets" semble peu affectée par cette dégradation, sauf la série "Twitter_volume_UPS". Cette dernière subit une baisse de corrélation avec l'approximation utilisant 4 arbres de

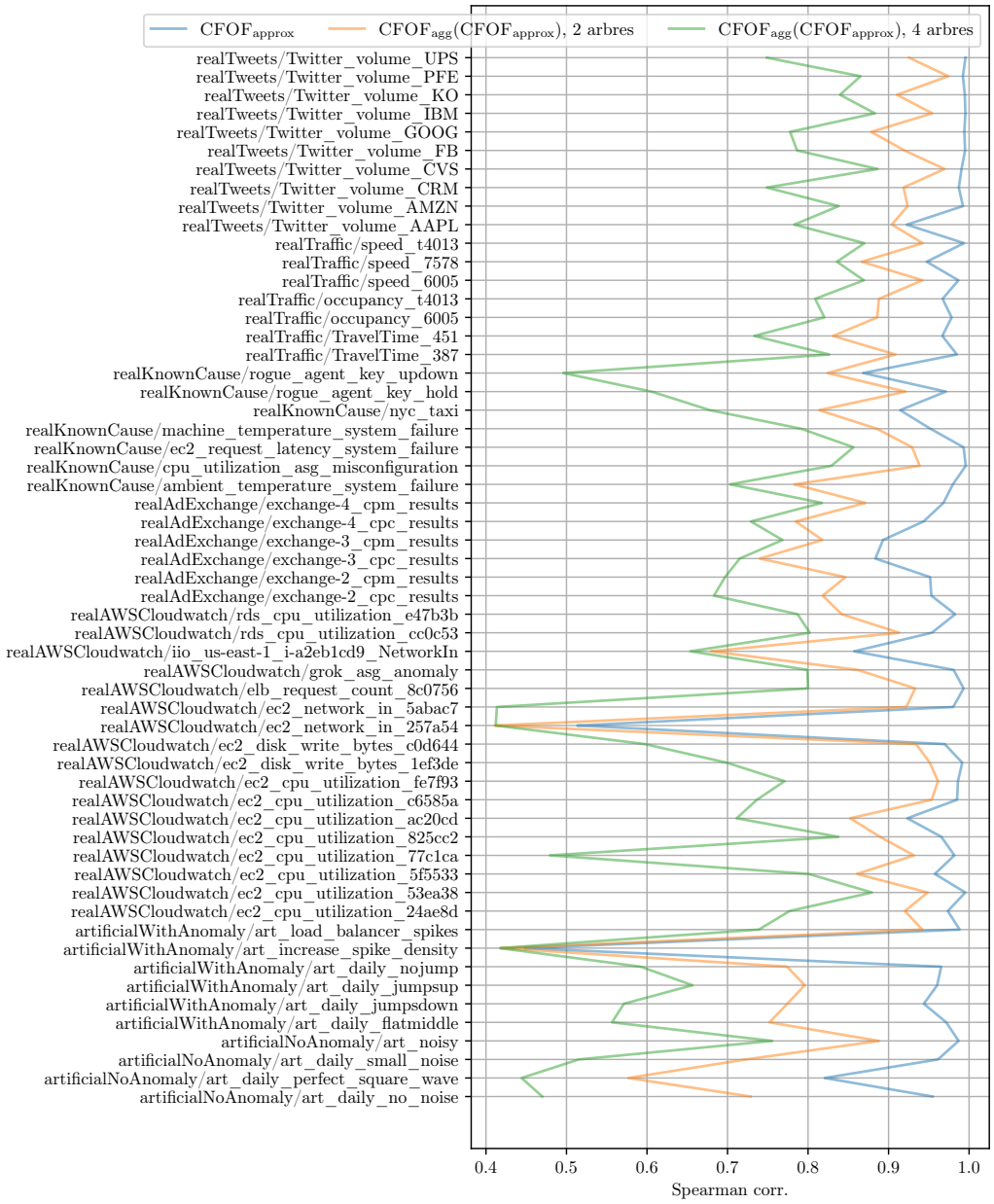


FIGURE 4.23 – Corrélations de SPEARMAN entre CFOF$_{exact}$ et CFOF approximé avec 1, 2 et 4 arbres, pour toutes les séries temporelles du benchmark NAB.

plus de 0.15 par rapport à celle utilisant 2 arbres, et de 0.25 avec celle n'utilisant qu'un seul arbre. Ceci signifie que les scores CFOF$_{agg}$(CFOF$_{approx}$) conduisent à des classements (rang des séquences selon le score) qui s'écartent

de plus en plus de celui de $\text{CFOF}_{\text{exact}}$ lorsque le nombre d'arbres augmente. Toutefois, ces différences doivent impacter modérément les séquences ayant un fort score (i.e., les anomalies) car comme nous l'avons vu dans la section précédente la qualité de détection est quant à elle maintenue.

4.4.3.3 Gain sur le coût de calcul

Nous évaluons à présent le gain obtenu concernant le nombre de nœuds parcourus, grâce aux méthodes CFOF_{agg} [6]. Comme expliqué précédemment, les dimensions d'origine des séquences du benchmark NAB ne sont pas réellement adaptées pour l'obtention d'un gain considérable lors de l'utilisation de plusieurs arbres, car les dimensions des séquences des différents jeux sont comprises entre 8 et 12 (cf. Section 3.6.2, et Section 4.4.3 pour le découpage en sous-espaces).

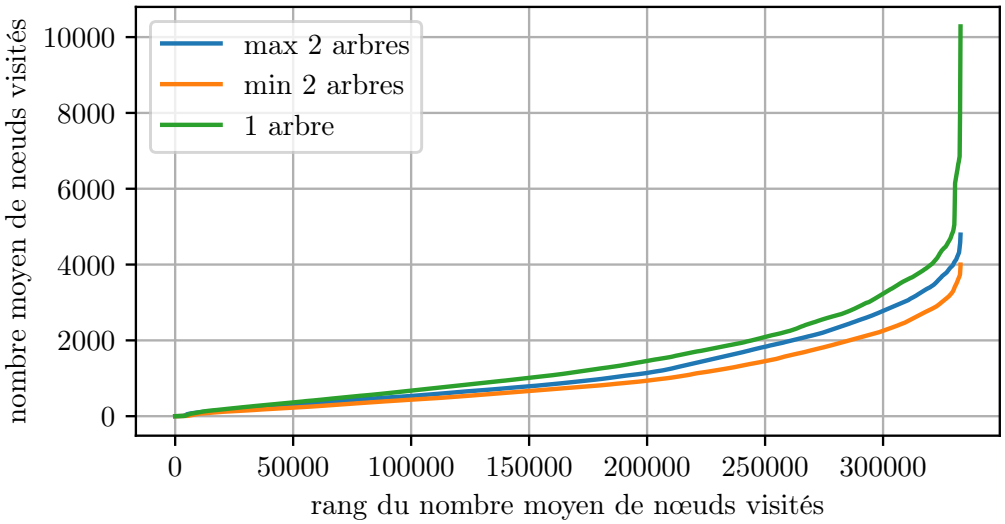


FIGURE 4.24 – Comparaison du nombre moyen de nœuds parcourus pour chaque séquence évaluée à partir des séries du benchmark NAB. La courbe bleue et la courbe orange représentent respectivement le maximum et le minimum des deux mesures *nombre moyen de nœuds visités* sur chacun des deux arbres.

Dans le cas de 2 arbres, nous obtenons des sous-espaces dont les dimensions sont comprises entre 4 et 6. Sur la figure 4.24, la courbe ayant les

6. Pour rappel, le gain est le même pour $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et pour $\text{CFOF}_{\text{agg}}(v\text{-}rang)$.

valeurs les plus hautes représente le nombre moyen de nœuds visités pour la méthode d'approximation avec 1 seul arbre. Sur cette même figure, la courbe la plus basse montre, pour la méthode d'approximation avec 2 arbres, le nombre moyen le plus faible entre les 2 arbres parcourus, et la courbe du milieu indique le nombre moyen le plus élevé entre ces 2 arbres. Le gain n'est certes pas très élevé, mais la méthode avec 2 arbres reste moins coûteuse si l'utilisateur a la possibilité d'effectuer en parallèle les deux parcours d'arbres.

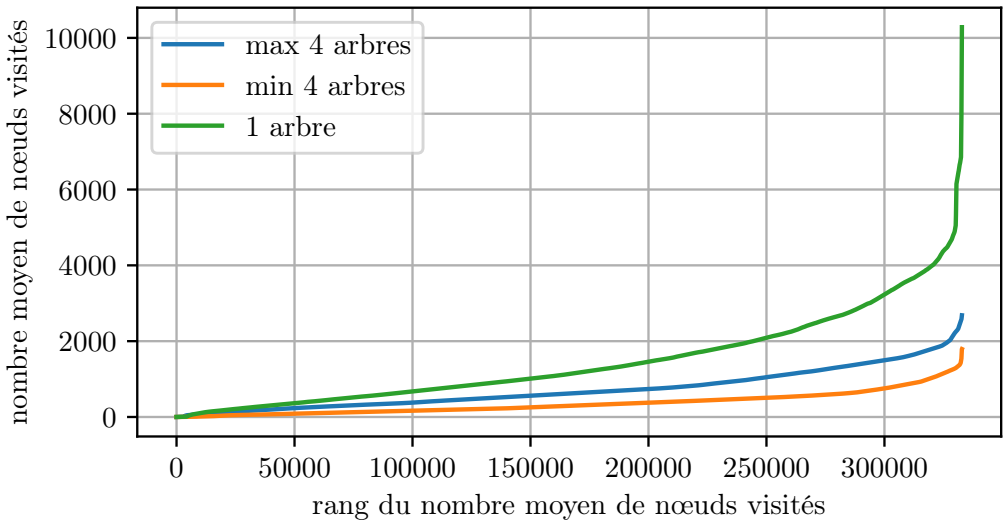


FIGURE 4.25 – Comparaison du nombre moyen de nœuds parcourus pour chaque séquence évaluée à partir des séries du benchmark NAB. La courbe bleue et la courbe orange représentent respectivement le maximum et le minimum des quatre mesures *nombre moyen de nœuds visités* sur chacun des quatre arbres.

Avec l'utilisation de 4 arbres d'indexation, les dimensions des sous-espaces sont de 2 ou 3. Nous pouvons observer alors Figure 4.25 une diminution plus importante du nombre moyen de nœuds visités, tant pour le nombre moyen le plus faible entre les 4 arbres parcourus que pour le plus élevé, comparativement à la méthode d'approximation avec 1 arbre. Le nombre maximum reste inférieur à 3000 tandis que le nombre minimum ne dépasse pas 2000. Le gain obtenu en parcourant en parallèle les arbres serait alors appréciable au vu du nombre moyen de nœuds visités avec 1 seul arbre qui peut dépasser les 10000 nœuds.

4.5 Conclusion

Nous avons constaté les limites du gain en coût de calcul apporté par l'élagage lors des parcours de l'arbre d'indexation dans les cas où le niveau 1 de l'arbre contient un nombre de nœuds du même ordre de grandeur que le nombre de séquences. Pour résoudre ce problème nous avons projeté les séquences dans des sous-espaces plus petits afin de les indexer dans des arbres ayant des nombres de nœuds de niveau 1 plus restreints. Nous avons alors proposé deux méthodes d'approximation du score CFOF par agrégation de scores calculés sur les arbres correspondant à ces sous-espaces : $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et $\text{CFOF}_{\text{agg}}(v\text{-}rang)$.

Nous avons démontré de façon formelle le bien-fondé de l'approximation $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ à partir d'un théorème de [Angiulli, 2020]. L'approximation a ensuite été validée sur le plan empirique dans le cas idéal de $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{exact}})$ avec des jeux contenant un seul groupe d'objets (famille de jeux *Clust1*). En comparant ces résultats aux scores CFOF exacts, nous avons obtenu des scores de corrélation de Spearman supérieurs à 0.99 en dimensions 20, 50, 100 et 200.

Les deux approximations $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et $\text{CFOF}_{\text{agg}}(v\text{-}rang)$ ont ensuite été évaluées sur les jeux *Clust2* de dimensions 20 et 200 pour différentes dimensions de sous-espaces. Les résultats de corrélation de Spearman sont satisfaisants pour les deux approximations sauf dans le cas particulier où la dimension des sous-espaces est de 1. Les meilleurs compromis entre qualité de l'approximation et réduction du nombre moyen de nœuds parcourus sont obtenus pour des sous-espaces de dimension de 2 à 4.

Enfin, nous avons vérifié sur le benchmark NAB que la qualité de détection était préservée par nos deux méthodes d'approximation. Les résultats montrent qu'elles restent parmi les meilleures méthodes, avec un léger avantage pour $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ avec 2 arbres dans le classement ROC AUC, et avec 4 arbres dans le classement PRC AUC. Ces résultats sont prometteurs, et si l'utilisateur a la possibilité d'effectuer les parcours d'arbres en parallèle il peut obtenir une réduction significative des temps de calcul.

Chapitre 5

Mise en œuvre et résultats sur les données SNCF

Les travaux détaillés dans ce chapitre ont été présentés en partie dans [Foulon *et al.*, 2019b] et dans le poster [Foulon *et al.*, 2019a].

5.1 Contexte organisationnel de la SNCF

La Société nationale des chemins de fer français (SNCF) est l'entreprise ferroviaire publique française, créé le 1er janvier 1938. Elle est notamment présente dans les domaines du transport de voyageurs, du transport de marchandises et réalise la gestion, l'exploitation et la maintenance du réseau ferré national.

SNCF regroupe trois sociétés anonymes, dont SNCF Voyageurs. Cette dernière se compose des activités Voyage, qui gère les Trains à Grande Vitesse (TGV), Intercité, TER pour les trains régionaux, et Transilien, qui s'occupe de trains de la région d'Île de France. Chacune d'elle comprend sa propre Direction des Systèmes d'Information, abrégées DSI, avec par exemple la DSI Transilien, mais aussi la DSI Voyage. Les activités de SNCF Voyageurs proposent différentes prestations de transport et interviennent également dans des services dits "au porte à porte" tel que iDVROOM pour le covoiturage ou encore OUICAR pour la location de véhicule entre particuliers. En plus de ces activités s'ajoute la Direction Industrielle qui comprend la Direction du Matériel chargé de l'achat et de l'entretien du matériel roulant, et la Production Ferroviaire, qui gère l'ensemble des services transverses de production des trains, c'est-à-dire de réservation des ressources matérielles, du réseau, et humaines pour faire circuler les trains. La Production Ferroviaire gère également la constitution du plan de transport effectif à partir de l'offre de

transport demandée par les activités. Ces deux dernières entités comprennent également leurs propres DSI.

Le projet s'inscrit, pour notre part, dans la Direction des Systèmes d'Information de la Production Ferroviaire, abrégée DSI PF. La DSI PF est transverse et interagit beaucoup avec les autres DSI de SNCF Voyageurs, dont la DSI Matériel, la DSI Voyageur, la DSI Transilien, mais aussi la DSI TER, ou encore la DSI Fret, ce dernier s'occupant du transport de marchandise. Dans la DSI PF, se trouve le programme Train Communicant, qui gère tous les flux de communication temps réel entre les trains et le Système d'Information de la SNCF à travers l'offre de supervision de bout-en-bout.

5.2 Le programme Train Communicant

Le programme Train Communicant a pour objectif de gérer la communication entre les trains et le Système d'Information de la SNCF. Ce système permet d'assurer la transmission des messages entre les trains constituant le parc matériel appelé "le bord", et les différents services "de sol" qui consomment les messages. Cette transmission se fait également, pour certains services, du sol vers le bord. Ces messages contiennent les informations métiers de différentes natures, comme la position géographique d'un train, le nombre de passagers, l'information-voyageur à afficher ou à communiquer oralement dans un train, ... Tous ces messages sont nécessaires, certains indispensables, pour suivre et maintenir le trafic ferroviaire de façon stable et sécurisée.

C'est au sein de ce programme que s'est déroulé notre travail. Nous étudions plus en détail l'architecture du Système d'Information du programme Train Communicant dans la section suivante.

5.3 Description de l'architecture du Système d'Information Train Communicant

Le Système d'Information (SI) Train Communicant permet la communication des messages sol-bord dans les deux sens : depuis le bord vers le sol, et inversement. Mais ce sont majoritairement des messages du bord vers le sol qui sont envoyés. Il existe aussi un faible nombre de messages transmis depuis des balises fixées au sol qui transitent dans le Système d'Information. Bien que, d'un point de vue métier, la distinction est possible entre les messages provenant directement des trains et ceux qui viennent du réseau ferré, ces

derniers contiennent aussi des informations sur les trains, c'est pourquoi dans ce travail nous les considérons comme provenant du bord, sans distinction.

Au centre de l'architecture de ce Système d'Information se trouve la plateforme CanalTrain. Cette dernière, initiée par l'équipe Train Communicant, répond à trois grandes problématiques :

— réduire les coûts d'acquisition et de maintien du service de communication sol-bord par la mutualisation du Système d'Information et du matériel à bord (notamment les équipements communicants),
— rendre la communication sol-bord plus efficace, fiable et plus intelligente en offrant des solutions hybrides (multi-bandes, multi-technologies, ...),
— uniformiser les services sol-bord directement liés au SI, et masquer la forte hétérogénéité des équipements et des systèmes constructeurs.

L'architecture de CanalTrain est visible sur la figure 5.1. La plateforme est adaptée pour recevoir des messages de différents formats et provenant de différents protocoles de communication. La partie en amont de CanalTrain, appelée Frontal, a la charge de réceptionner ces messages (avec l'envoi d'un accusé pour certains types de messages), et de les stocker avant traitement. Sur la partie en aval, abrégée PDS pour "Plateforme de Service", les messages sont transformés sous un format dit "pivot" afin d'être lisibles par tous. Ces derniers sont ensuite transmis par l'intermédiaire d'abonnements, auxquels les différentes plateformes de la SNCF peuvent adhérer.

CanalTrain se trouve au centre du système de communication, entre les trains et le reste du SI SNCF. Le moindre défaut de fonctionnement de cette plateforme peut affecter toutes les chaînes de traitement et de suivi opérationnel, par le biais d'un "effet domino". C'est plus précisément sur cette plateforme que nous allons évaluer notre méthode.

Nous présentons le Système d'Information sous forme de graphe sur la figure 5.1. Ce dernier ne montre pas la grande complexité du SI, mais permet de mieux comprendre le rôle de CanalTrain pour la transmission des messages. Durant leurs transmissions, les messages passent par différentes plateformes, de telle sorte que l'acheminement des messages peut être vu comme un travail à la chaîne effectué par différentes plateformes les unes à la suite des autres. Par exemple, la plateforme Geomobile visible sur la figure 5.1 est directement abonnée à CanalTrain, mais elle redistribue les messages de géolocalisation seulement après les avoir enrichis, projeté les coordonnées géographiques sur le réseau ferré (pour les convertir en coordonnées linéaires), et s'être assurée que les données reçues étaient cohérentes. Elle fournit une information plus précise à Géopulse qui elle-même fournit les cartes en temps-réel des positions

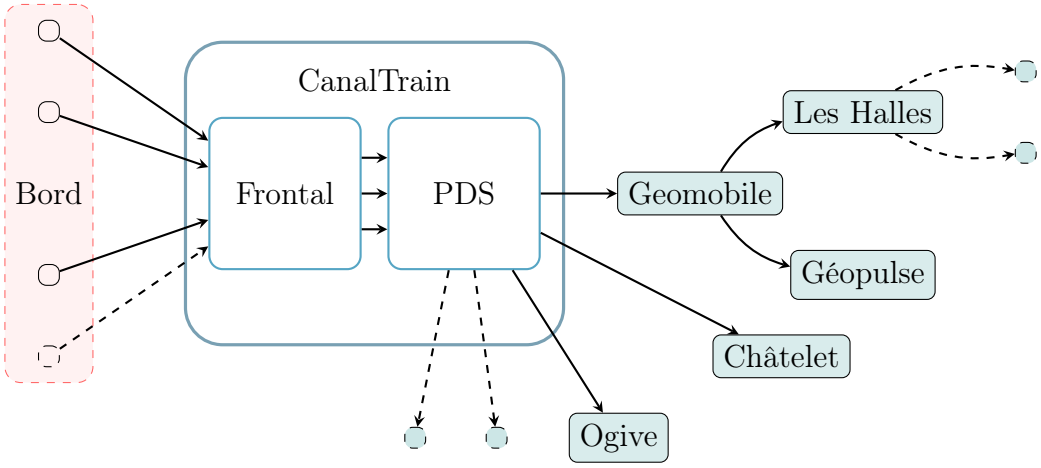


FIGURE 5.1 – Le Système d'Information SNCF, représenté de façon simplifiée permettant de mettre en avant le rôle de CanalTrain. Seulement quelques services sont présents sur cette figure. Nous présentons le SI dans le sens d'envoi des messages depuis le bord vers le sol.

des trains.

Les messages sont émis depuis les trains, à gauche de la figure 5.1, réceptionnés par CanalTrain, puis redistribués aux différents consommateurs. Nous constatons également que le Système d'Information Trains Communicants, contribuant au bon fonctionnement du système ferroviaire, notamment pour le suivi opérationnel des trains et de l'information voyageurs, est entièrement dépendant de la plateforme CanalTrain.

Nous venons de voir la structure principale du SI Train Communicant. Nous allons à présent aborder la problématique de supervision de ce SI.

5.4 Présentation de la problématique

Les messages envoyés depuis les trains sont très hétérogènes. Cela s'explique par plusieurs points :

— un large spectre d'informations différentes, envoyées depuis le bord et le sol : géolocalisation, comptage voyageur, télé-maintenance, ... ,
— les différents types d'engins qui, selon le constructeur, mais aussi selon la génération de mise en service, sont amenés à partager une grande variété de formats de messages,
— la diversité des protocoles de communication, directement liée aux

deux points précédemment cités.

Comme développé précédemment, CanalTrain a la charge de recevoir et d'insérer dans un modèle pivot les messages, puis de les mettre à disposition des autres plateformes par le biais d'abonnements. Environ 5 millions de messages sont envoyés chaque jour depuis CanalTrain. Plus le nombre d'abonnés et de trains augmente, plus la charge s'avère importante. Les nouveaux trains étant de plus en plus communicant, il est prévu que cette charge soit multipliée par 2 ou 3 dans les cinq prochaines années.

Cependant, lorsque la plateforme fonctionne de façon dégradée ou qu'elle est hors-service, c'est tout le Système d'Information qui est impacté. Dans ce type de situation non-désirée, le suivi opérationnel du trafic ferroviaire se retrouve également en difficulté, de même que l'information voyageur à bord des trains et au sol.

Les mises à jour des différents services, y compris CanalTrain, passent par plusieurs phases de recettes et de préproduction avant d'être réellement exploitées par le métier en production. Mais il arrive que certains scénarios inattendus se produisent en production, menant parfois à une ou plusieurs anomalies. Certaines anomalies surviennent également sur des systèmes qui ont toujours bien fonctionné jusque-là. Si elles ne sont pas détectées et prises en compte rapidement, le Système d'Information peut se retrouver avec un ou plusieurs services défaillants et momentanément indisponibles.

Pour prévenir les pannes sur le Système d'Information, nous proposons d'utiliser notre méthode de détection présentée dans la section 3.3 afin d'anticiper d'éventuels incidents, et maintenir en condition opérationnelle le Système d'Information.

5.4.1 Projet de supervision de bout-en-bout

Le projet de supervision de bout-en-bout consiste à mettre en place et à maintenir fonctionnellement un outil de surveillance, en temps-réel, de l'acheminement des messages sur le Système d'Information. Le suivi commence depuis l'envoi et se termine jusqu'à la réception chez le consommateur final. Pour s'assurer que ces derniers circulent correctement, une trace est générée puis stockée à chaque fois qu'un message franchit une nouvelle étape de transmission, étape délimitée par deux points de collecte de traces, généralement en entrée et en sortie de chaque plateforme.

Les traces générées permettent ainsi de décrire les différentes étapes de transmission des messages. Ces dernières ne donnent pas d'informations métiers, mais les informations sur le contexte de réception ou d'émission du message. Lors d'un dysfonctionnement, par exemple la non-conformité de

messages attendus par un ou plusieurs services, ces traces permettent de pister ces messages à travers le Système d'Information, et de retrouver le ou les composants à l'origine du dysfonctionnement.

5.4.2 Comportement des messages

Grâce à ces traces, nous pouvons relever plusieurs indicateurs tels que le débit de messages, c'est-à-dire le nombre de messages transmis par unité de temps. Les traces fournissent également des données de contexte comme le type d'équipement bord communicant, de train commercial (par exemple le TGV 6650), de service métier (géolocalisation, télé-diagnostic, ...), mais aussi l'horodatage de l'événement bord lorsque le message représente un événement (par exemple l'horodatage de la prise de mesure GPS), et le type ou la série d'engin (Régiolis, Régio2N, TGV 2N2, ...).

Cependant, quatre facteurs compliquent significativement la prédiction. D'abord, l'évolution du système : certains services amplifient la transmission (en augmentant le nombre de messages échangés), d'autres l'estompent. Ensuite, les journées ne sont jamais réellement identiques : le rythme du trafic ferroviaire étant souvent modifié ou perturbé, le nombre de messages transmis varie. De plus, le nombre de combinaisons "service fonctionnel/type d'engins" à analyser est très grand (plusieurs dizaines) et tous ont leurs propres spécificités, ce qui ajoute de la difficulté dans la prédiction. Et enfin, certains services comme la géolocalisation dominent largement tous les autres flux réunis en matière de volumétrie, pouvant donner l'illusion que tout fonctionne lorsque ce service est stable.

Ainsi, le nombre important d'indicateurs, leur hétérogénéité et leur variation d'une journée à l'autre sont autant d'obstacles à une analyse correcte de l'état de santé du Système d'Information. De plus, le périmètre d'analyse d'une anomalie sur le SI Train Communicant est très variable en fonction de la localisation au sein du SI et du type d'anomalie.

Pour faire le bilan sur ces points de difficultés, nous concluons qu'il est difficile d'analyser tous les indicateurs caractérisant la santé du Système d'Information, car ils sont trop nombreux, trop hétérogènes et varient d'une journée à l'autre.

5.5 Supervision actuelle

Pour le traitement des traces, la SNCF utilise la suite logicielle ELK [1]. Cette suite est composée de trois outils : Elasticsearch, Logstash et Kibana [2]. Elle est utilisée pour surveiller les messages et s'assurer qu'ils circulent correctement sur le Système d'Information. L'architecture globale est illustrée sur la figure 5.2.

Logstash collecte et envoie une trace pour chaque nouveau message arrivant sur un point de collecte (dans ce travail, nous nous intéressons seulement aux points de collecte se trouvant sur Canal Train).

Elasticsearch permet l'indexation et le stockage des traces au format JSON (JavaScript Object Notation). Il permet de construire et stocker les index mensuels de toutes les traces reçues de Logstash.

Kibana offre différents types de visualisation des traces récoltées, sous forme de tableaux de bord. Il permet aux superviseurs de la SNCF de visualiser des statistiques et des indicateurs (histogrammes, nuages de points comme présentés dans la section 2.2.3.1) élaborés à partir des traces. L'indicateur principal suivi par ces superviseurs est le nombre de messages reçus par minute, mesure sur laquelle nous effectuons la détection des anomalies. La latence moyenne des messages par unité de temps peut aussi constituer un indicateur intéressant pour la détection d'anomalies, toutefois elle n'est pas encore disponible facilement depuis ELK.

Dans la section suivante, nous allons présenter le procédé de création des séquences permettant l'intégration de notre méthode dans cette architecture de supervision.

5.6 Architecture de mise en œuvre de la détection *i*SAX-CFOF

La plateforme au cœur du Système d'Information Train Communicant — CanalTrain — gère l'échange de toutes les informations numériques entre le sol et les équipements mobiles. Lorsque le comportement de l'une des composantes impliquées dans ce service essentiel devient anormal, des dysfonctionnements graves peuvent non seulement perturber cette composante centrale, mais aussi avoir un impact indirect sur d'autres services. Il est donc

1. Elastic is a trademark of Elasticsearch BV
2. Elasticsearch, Logstash and Kibana are trademarks of Elasticsearch BV, registered in the U.S. and in other countries.

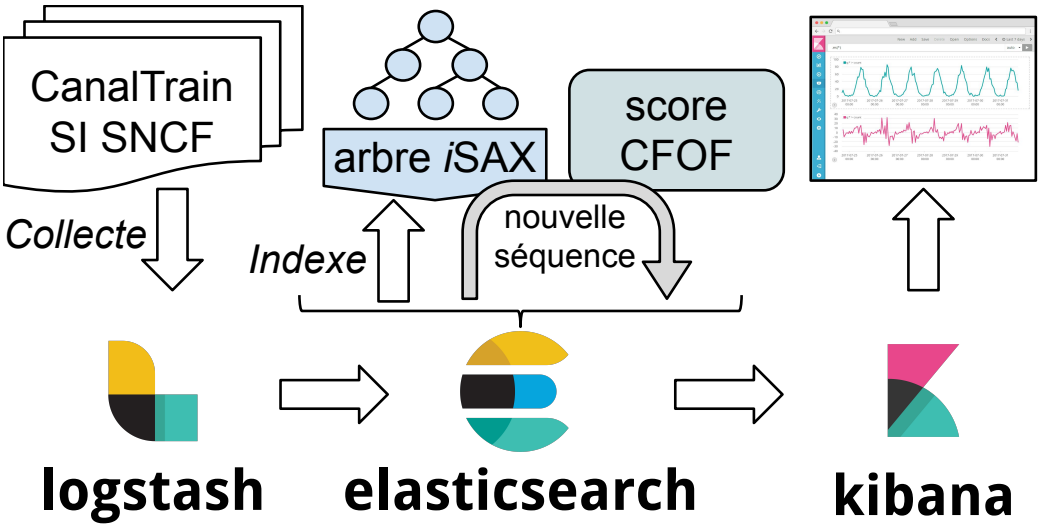


FIGURE 5.2 – Architecture du système supportant la méthode de détection des anomalies.

essentiel de veiller à ce que nous soyons en mesure de détecter les anomalies le plus rapidement possible. Pour cette raison, notre processus de détection se concentre sur les indicateurs liés à la qualité des communications entre les composants bord et sol plutôt que sur le contenu du message lui-même.

Le flux de données brutes est trop hétérogène et trop volumineux pour être directement analysé. Comme nous nous intéressons à la santé globale du Système d'Information, nous extrayons une description simplifiée de la dynamique du flux à partir des données de traces disponibles. Ce processus est décrit dans la figure 5.3.

Le flux de traces est régulièrement collecté à partir des différents points de collecte mis en place dans le système, et mis à disposition par le mécanisme de stockage et d'indexation des traces. Le contenu et la sémantique des messages bruts sont ignorés, car ils sont liés et trop spécifiques aux traitements internes de l'équipement qui les a générés. C'est pourquoi l'analyse se concentre sur le nombre de messages par minute, noté M dans la figure 5.3.

Toutes les minutes, une valeur moyenne de M est calculée. Une série de valeurs moyennes consécutives du débit dk $M_1, M_2, \ldots, M_{dk}$ est appelée une *séquence*, qui s'étend sur une durée $dh = dk \times dm$. Ces séquences sont générées toutes les dd minutes et peuvent se chevaucher dans le temps.

Le prétraitement des données est structuré en trois étapes et visible sur la figure 5.2 : collecte, agrégation et indexation. Les deux premières étapes sont le fruit de requêtes sur Elasticsearch, qui renvoient le résultat au for-

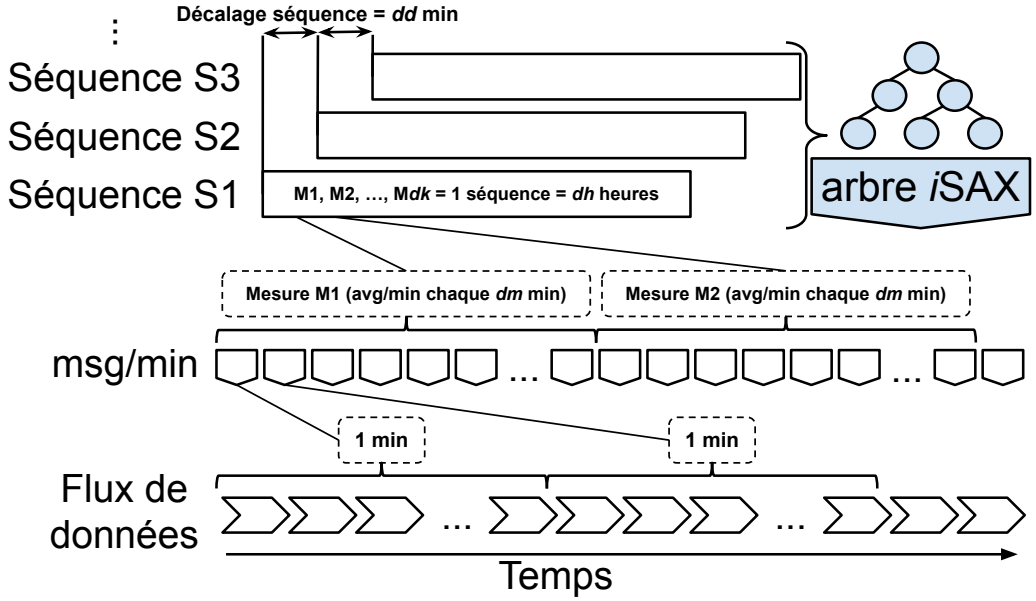


FIGURE 5.3 – Processus d'extraction, des données brutes aux séquences temporelles.

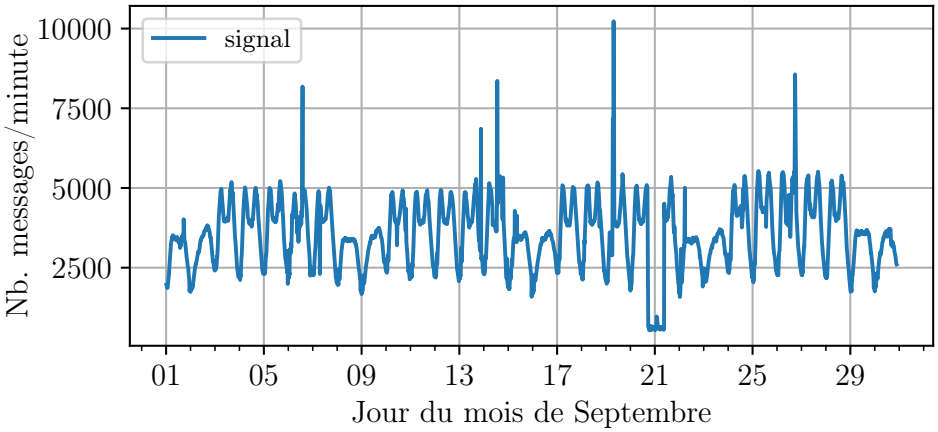
mat JSON. La dernière étape, quant à elle, consiste à insérer dans un arbre iSAX les séquences passées, afin de construire un historique des objets de référence pour la détection d'anomalie. Ainsi, pour chaque nouvelle séquence, une approximation du score d'anomalie CFOF est calculée avec la méthode décrite à la section 3.3, et stockée dans Elasticsearch, ce qui permet ensuite de visualiser dans Kibana l'ensemble des scores obtenus.

Une collection de séquences passées formée par les séquences de référence est insérée dans l'arbre iSAX, et un score $\text{CFOF}_{\text{approx}}$ est calculé pour chaque nouvelle séquence construite à partir du flux. Dans les résultats présentés dans la section 5.7, les séquences sont construites avec $dm = 15$ minutes, $dk = 24$ (et donc $dh = 6$ heures), et $dd = 30$ minutes.

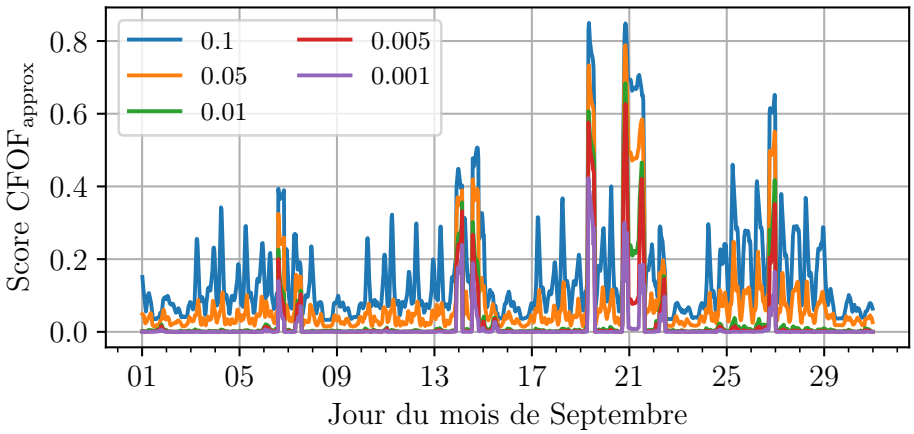
5.7 Résultats

Le signal analysé est présenté sur la figure 5.4a. À partir de ce signal, les séquences ont été générées en suivant les étapes décrites dans la section 5.6. Pour présenter les scores CFOF de ces séquences au sein de la SNCF, nous suivons le procédé décrit dans la section 5.6 et visible sur la figure 5.2, et nous utilisons l'algorithme 1 pour l'obtention des $\text{CFOF}_{\text{approx}}$.

Nous présentons sur la figure 5.4b les scores des séquences avec $\varrho \in$



(a) Nombre de messages reçus par minute.



(b) Score $\text{CFOF}_{\text{approx}}$ et $\text{CFOF}_{\text{exact}}$ avec $\varrho \in [0.001, 0.005, 0.01, 0.05, 0.1]$.

FIGURE 5.4 – Résultats de détection basée sur l'indicateur du nombre de messages reçus par minute durant le mois de Septembre 2018 sur le point de collecte se situant dans CanalTrain.

$[0.001, 0.005, 0.01, 0.05, 0.1]$. Pour rappel, l'obtention des scores selon différentes valeurs de ϱ s'effectue lors du parcours de la liste des *v-rang*, à la ligne 24 de l'algorithme 1. Le nombre de scores retournés, associés à différentes valeurs de rho, pour une même séquence n'impacte que très faiblement le coût de calcul et reste alors négligeable. Pour chaque valeur de ϱ choisie, les résultats sont présentés dans des figures distinctes en annexe.

Sur ces données SNCF, les scores de détection d'anomalies sont présentés avec différents niveaux de granularité sur la figure 5.4b. Pour une même séquence, plus la valeur de ϱ est grande, plus le score CFOF l'est aussi.

Cependant, certaines périodes, comme celle du 21 Septembre, sont clairement identifiées comme anormales par toutes les valeurs de ϱ proposées ici. Mais ce n'est pas toujours le cas : certains intervalles de la période comprise entre le 10 et le 13 Septembre peuvent paraître anormaux pour $\varrho = 0.1$ si le seuil de détection est défini à 0.2. Tandis que les autres valeurs de ϱ n'alertent pas sur cette même période. Ce paramètre ϱ nous permet de régler la sensibilité de détection que nous souhaitons obtenir sur ce flux de données et ainsi rendre la détection plus sensible en augmentant ϱ, ou limiter le nombre de faux positifs en le diminuant.

Grâce à l'expertise des superviseurs de la SNCF, nous avons choisi de présenter les résultats suivants avec une valeur de $\varrho = 0.05$.

5.7.1 Qualité de l'approximation

Avant d'évaluer la pertinence des détections obtenues par le score CFOF approximé, nous souhaitons les comparer avec les scores $\text{CFOF}_{\text{exact}}$.

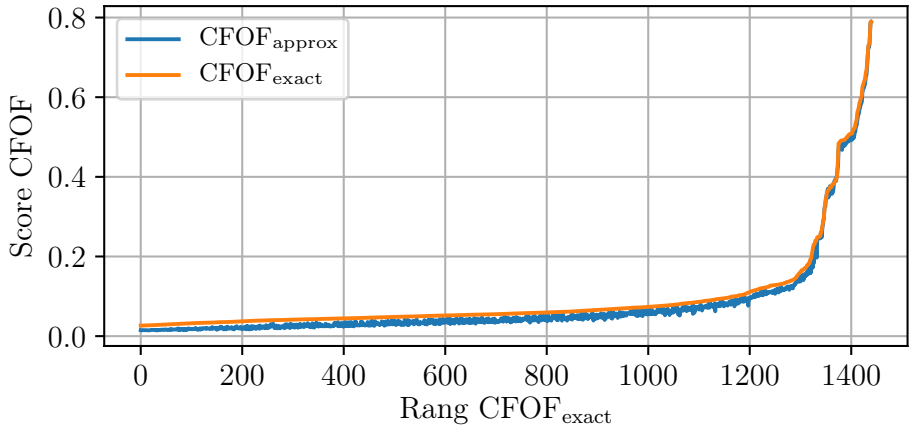


FIGURE 5.5 – Scores $\text{CFOF}_{\text{exact}}$ et $\text{CFOF}_{\text{approx}}$ des séquences du mois de Septembre, tous deux triés selon le rang des scores $\text{CFOF}_{\text{exact}}$.

Sur la figure 5.5, nous présentons les scores $\text{CFOF}_{\text{exact}}$ et les scores $\text{CFOF}_{\text{approx}}$ des séquences du mois de Septembre 2018. Ces deux types de scores ont été triés selon le rang obtenu à partir du score $\text{CFOF}_{\text{exact}}$ de chaque séquence. Nous observons que ces deux scores sont très proches, ce qui est confirmé par le résultat de corrélation de Spearman de 0.988.

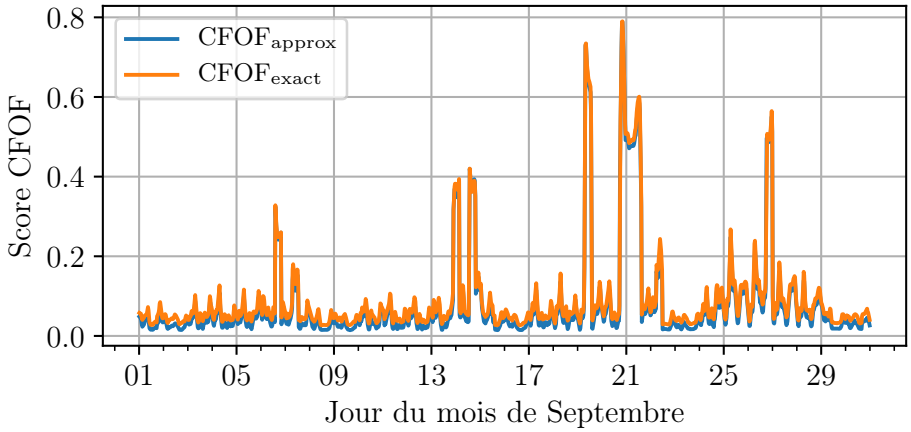


FIGURE 5.6 – Score CFOF$_{\text{approx}}$ et CFOF$_{\text{exact}}$ avec $\varrho = 0.05$. Résultats de détection basée sur l'indicateur du nombre de messages reçus par minute durant le mois de Septembre 2018 sur le point de collecte se situant dans CanalTrain.

De plus, la figure 5.6 permet de confirmer que les détections faites par CFOF$_{\text{exact}}$ et celles par CFOF$_{\text{approx}}$ sont sur les mêmes intervalles temporels.

5.7.2 Gain en coût de calcul

L'arbre d'indexation stocke un historique de 14592 séquences de dimension 12. L'arbre est constitué de 2405 nœuds ce qui signifie que le gain minimum obtenu à chaque calcul du score CFOF$_{\text{approx}}$ d'une nouvelle séquence est supérieur à 6 par rapport au calcul du score CFOF$_{\text{exact}}$.

Nous avons choisi de présenter les nombres moyens de nœuds visités, nécessaires aux calculs des CFOF$_{\text{approx}}$ de chacune des séquences, selon l'horodatage de celles-ci. Ces résultats sont présentés sur la figure 5.7.

En comparant ces résultats au signal de la figure 5.4a, nous observons que les scores des jours de week-end sont plus rapidement calculés que ceux de la semaine. Cela s'explique par le fait que les séquences générées à partir des mesures effectuées sur le week-end forment dans l'espace d'origine un plus petit groupe que celles générées durant la semaine.

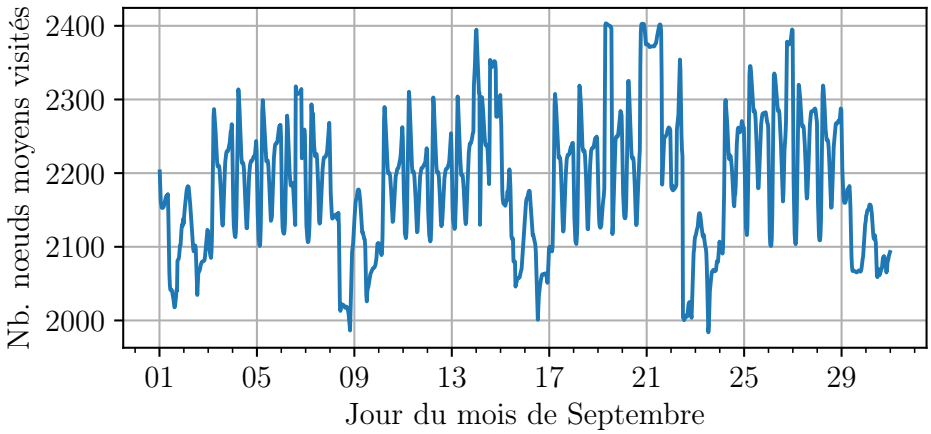


FIGURE 5.7 – Nombre de nœuds visités pour les calculs des scores CFOF approximés des séquences du mois de Septembre 2018.

5.7.3 Qualité de la détection

Dans cette section, nous analysons les détections obtenues avec $\text{CFOF}_{\text{approx}}$ puis nous comparons les résultats avec les détections de l'outil ELK.

Pour tester la qualité de détection d'anomalies avec $\text{CFOF}_{\text{approx}}$, nous utilisons une valeur de $\varrho = 0.05$. Ce paramètre, expliqué à la section 3.2, définit la taille du voisinage minimum dans lequel doit se trouver la séquence à évaluer pour obtenir son score CFOF.

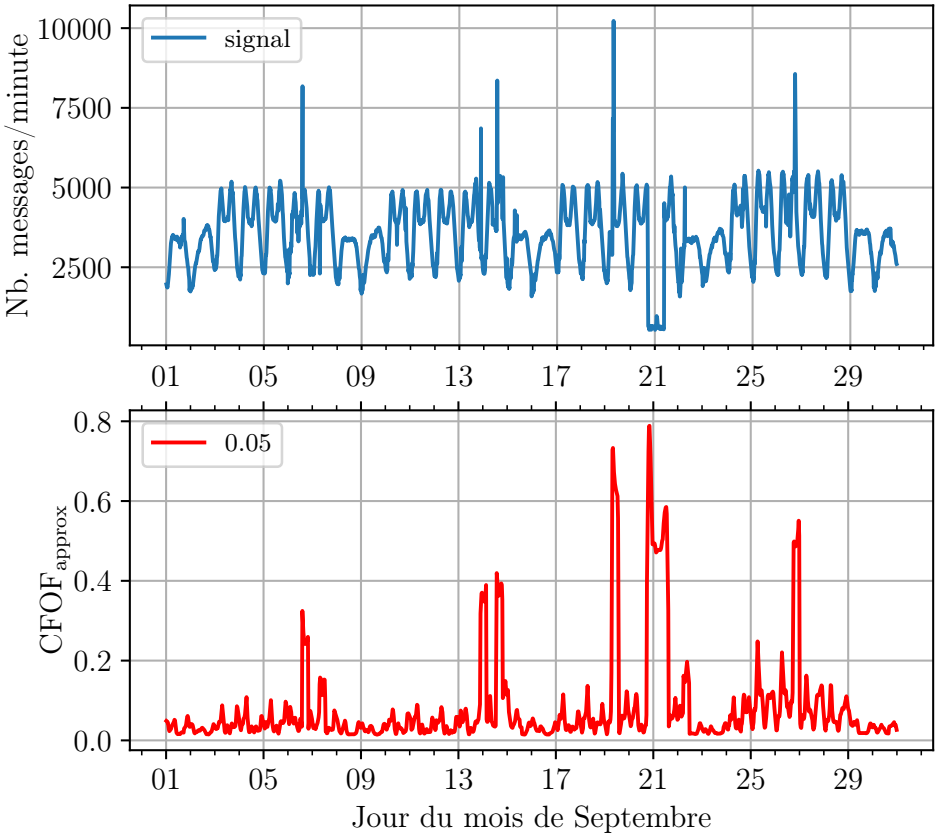
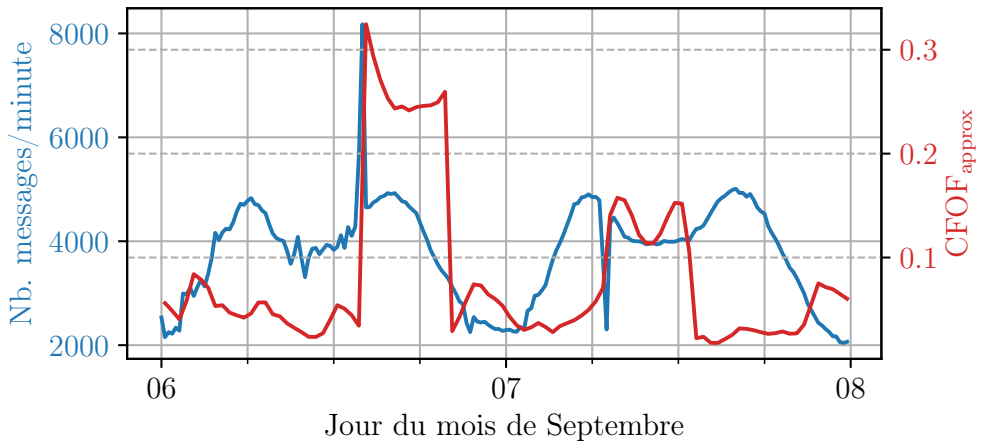


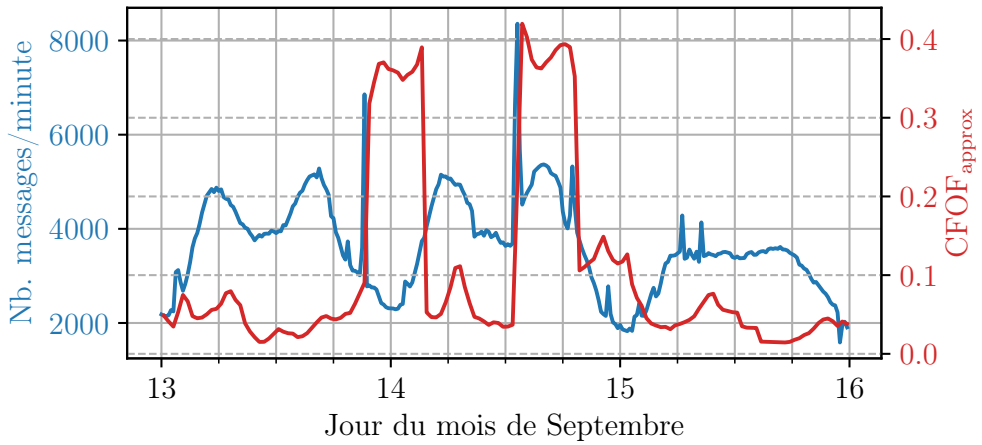
FIGURE 5.8 – Résultats de détection basée sur l'indicateur du nombre de messages reçus par minute durant le mois de Septembre 2018 sur le point de collecte se situant dans CanalTrain.

Nous comparons le signal du nombre de messages reçus par minute et les scores $\text{CFOF}_{\text{approx}}$ pour les séquences obtenues sur ces mêmes horodatages que les valeurs du signal.

Nous distinguons quatre zones anormales : du 6 au 7, du 13 au 15, du 19 au 22, et du 24 au 26 Septembre. Les deux premières sont visibles de manière plus détaillée sur la figure 5.9, et sur la figure 5.10 pour les deux dernières.



(a) Anomalie du 6 au 7.



(b) Anomalie du 13 au 15.

FIGURE 5.9 – Zoom sur les anomalies du 6 au 7 et du 13 au 15 Septembre.

Le 7 Septembre, la première anomalie — visible sur la figure 5.9a — est due à l'accumulation de messages provenant d'une plateforme en amont. La mesure $\text{CFOF}_{\text{approx}}$ est capable de détecter le pic anormal du nombre de messages entre le 6 et le 7 Septembre, mais également la baisse anormale le matin du 7.

La seconde anomalie présentée sur la figure 5.9b a lieu le 19 Septembre au

matin : elle est représentative des anomalies souvent détectées tardivement. Nous observons que le nombre de messages diminue légèrement de 3 à 6 heures du matin au lieu d'augmenter, avant de soudainement grimper à plus de 10000 messages par minute. Cette situation anormale est assez classique : à cause des difficultés de traitement de l'ensemble des messages, la plateforme se retrouve saturée. Plusieurs files d'attente, en amont de CanalTrain, déchargent massivement leurs messages accumulés, et inondent CanalTrain d'un grand nombre de messages. De façon plus amoindrie, un comportement similaire peut être observé le 6 Septembre autour de 14 heures. Les nœuds de la plateforme ont été alors purgés et redémarrés pour que CanalTrain retrouve un comportement normal, mais induisant alors une perte de messages importante. Mais avant cela, il était possible de constater techniquement une saturation de la mémoire vive suivie d'une saturation du CPU.

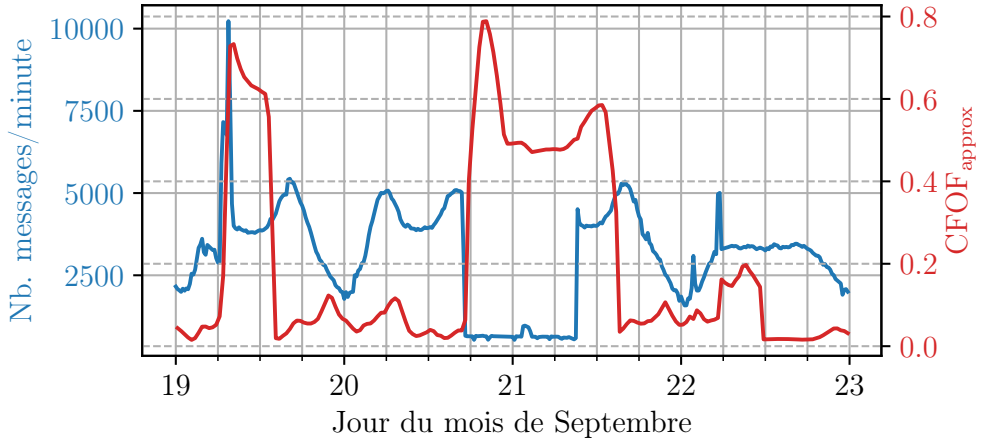
L'anomalie la plus critique se déroule dans la nuit du 20 au 21 Septembre : nous constatons que le nombre de messages s'est écroulé à zéro. Comme nous pouvons le voir sur la figure 5.10a, l'ensemble des flux de la plateforme CanaTrain sont stoppés. Un dysfonctionnement du système de gestion des messages (ActiveMQ) interne à CanalTrain est la cause de cette anomalie, qui a dans les faits été résolue très tardivement. Bien que la cause du comportement anormal du composant n'ait toujours pas pu être identifiée, un redémarrage s'est avéré nécessaire.

Nous observons qu'avec le paramètre $\varrho \in (0.001, 0.005, 0.01)$ l'anomalie est marquée en début et en fin par deux pics de scores élevés, tandis qu'avec $\varrho = 0.05$ et $\varrho = 0.1$, l'anomalie est signalé du début à la fin. En effet, une valeur de ϱ plus élevée rend le coefficient CFOF plus exigeant en terme de similarité.

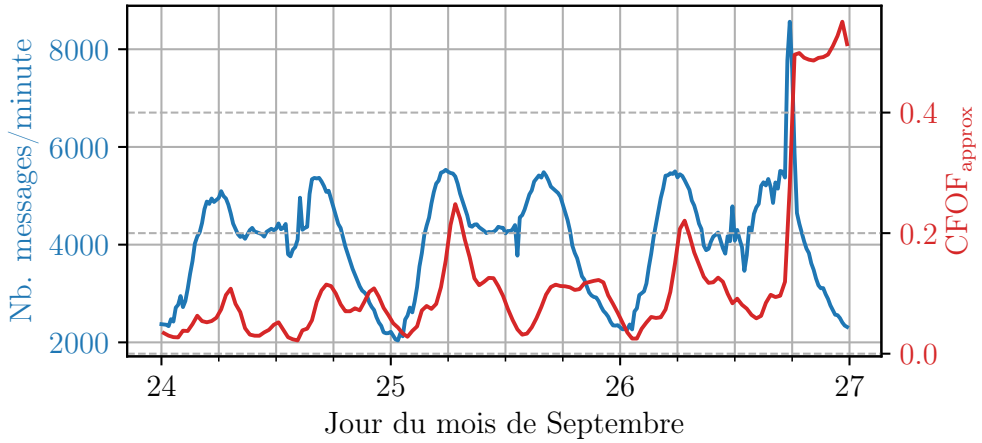
Après l'anomalie du 20-21 Septembre, un grand nombre d'analyses de services et de composants ont été faites pour éviter que ce type de situation ne se reproduise. Les jours qui ont suivi ont connu plusieurs perturbations dues au redémarrage de plusieurs services et composants, notamment durant le 26 Septembre où la PDS, un des principaux composants de CanalTrain, a dû être redémarrée pour éviter une nouvelle surcharge des messages.

La suite ELK propose également un outil, appelé "Machine Learning" (ML-ELK), permettant de superviser les flux indexés dans Elasticsearch. En plus d'être payant, cet outil n'est pas réellement paramétrable et fonctionne comme une boîte noire [Veasey et Dodson, 2014]. Outre la sensibilité des données mises à disposition par la SNCF et analysées par un outil externe, les traitements effectués sur ces données afin de fournir une prédiction, ne sont pas clairement décrits.

Dans l'objectif de pouvoir contrôler et paramétrer les détections, la me-



(a) Anomalie du 19 au 22.

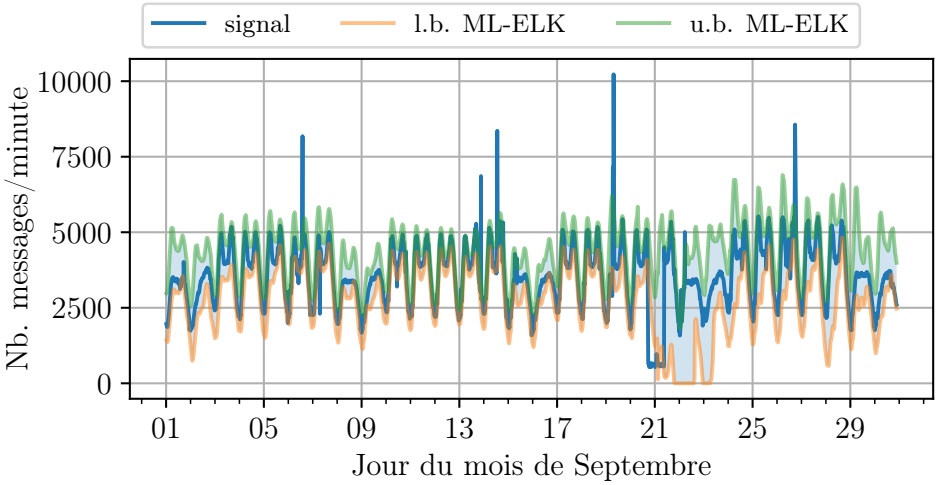


(b) Anomalie du 24 au 26.

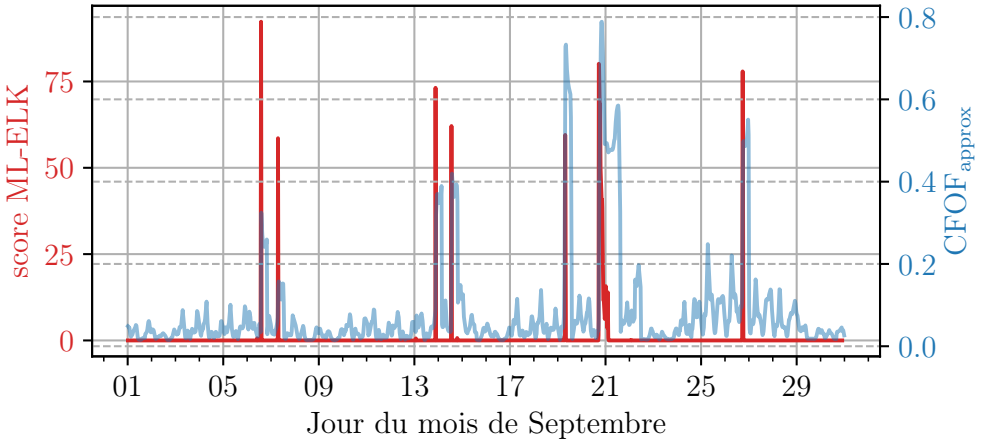
FIGURE 5.10 – Zoom sur les anomalies du 19 au 22 et du 24 au 26 Septembre.

sure CFOF est plus appropriée pour être utilisée sur différents types de flux. Comme expliqué dans la section 3.2, le score CFOF ne dépend que du paramètre ϱ. Mais le paramétrage de la construction des séquences permet aussi aux experts de la SNCF de pouvoir choisir ce qu'ils souhaitent analyser. Cela permet de gérer de façon adaptable l'outil d'analyse, paramétré spécifiquement pour chacun des besoins.

L'outil ML-ELK apprend sur un court historique de données et prédit les données ultérieures selon une borne minimale et une borne maximale. Les bornes prédites pour les données du mois de Septembre sont visibles sur



(a) Résultats de prédiction avec ML-ELK.



(b) Comparaison des résultats de détection d'anomalies avec ML-ELK et avec $\text{CFOF}_{\text{approx}}$ sur le mois de Septembre.

FIGURE 5.11 – Résultats de prédiction avec ML-ELK et comparaison des résultats de détection d'anomalies avec ML-ELK et avec $\text{CFOF}_{\text{approx}}$ sur le mois de Septembre.

la figure 5.11a avec "l.b." pour *lower bound* et "u.b." pour *upper bound*. Si le signal dépasse une des bornes, alors l'outil retourne un score d'anomalie élevé visible sur la figure 5.11b.

La borne de prédiction maximale obtenue sur le premier jour du mois, un samedi, est légèrement surévaluée et nous constatons que cela se reproduit tous les samedis. En effet, l'outil se met à jour continuellement et se réadapte

selon les dernières données obtenues.

En comparant les détections de $\text{CFOF}_{\text{approx}}$ avec celles de ML-ELK sur la figure 5.11b, nous constatons qu'elles sont similaires, avec cependant un point de nuance important : la détection de 21 Septembre perturbe les futures prédictions de ML-ELK. En effet, ce dernier considère cette anomalie comme normale après quelques heures, et réadapte sa prédiction en proposant des bornes plus lâches visibles sur la figure 5.11a sur les jours qui suivent. Tandis que $\text{CFOF}_{\text{approx}}$ n'intègre pas cette période anormale comme un évènement nouveau à apprendre rapidement. Le voisinage des séquences générées durant cette période n'est pas assez abondant pour être considérée.

Ces deux techniques, $\text{CFOF}_{\text{approx}}$ et ML-ELK, sont donc complémentaires en terme de détection, avec cependant plus de contrôle sur le paramétrage de la méthode $\text{CFOF}_{\text{approx}}$.

5.8 Conclusions

Dans ce chapitre, nous avons décrit plus en détail l'infrastructure du Système d'Information de la SNCF et les problématiques qui en découlent.

Nous proposons d'utiliser les traces récoltées dans ELK pour construire des séquences résumant l'état du Système d'Information sur un point de collecte. Nous calculons les scores $\text{CFOF}_{\text{approx}}$ de ces séquences à l'aide de la méthode détaillée dans le chapitre 3, puis nous rendons ces résultats accessibles aux superviseurs de la SNCF dans l'outil de visualisation Kibana.

Après plusieurs mois de récolte de séquences, permettant de construire une base de référence indexée dans un arbre iSAX, nous analysons les séquences construites durant le mois de Septembre 2018. En terme de coût de calcul et d'approximation du score $\text{CFOF}_{\text{exact}}$, notre méthode reste tout aussi robuste que dans les chapitres précédents. De plus, l'expertise des superviseurs SNCF a pu confirmer la cohérence des détections obtenues.

Chapitre 6

Conclusion et perspectives

6.1 Synthèse

Ce travail de thèse s'est principalement concentré sur la problématique de la détection d'anomalies dans les flux de données. Le domaine de la détection d'anomalies, tous types de données confondus, est particulièrement large, et nous avons présenté les principales définitions du terme "anomalie" ainsi que les principaux domaines d'application de leur détection. Nous avons également présenté les méthodes d'apprentissage et de fouille de données de la littérature qui ont été créées ou adaptées pour répondre aux particularités de la détection d'anomalies.

L'application visée est la détection d'anomalies pour des objets qui représentent des séquences de valeurs survenant dans un flux de données. Notre première préoccupation a donc été de proposer une méthode de détection qui ne serait pas impactée par les différents phénomènes liés à la malédiction de la dimensionnalité ; le but étant de lever les limitations portant sur la taille des séquences à analyser. Dans la première contribution, nous avons donc proposé d'utiliser la mesure de concentration nommée CFOF, initialement définie par [Angiulli, 2017] pour la détection d'anomalies, et permettant d'évaluer des données en haute dimension. En effet, [Angiulli, 2020] a montré formellement que, contrairement à d'autres méthodes d'évaluation non supervisées, la distribution des scores attribués par CFOF garde tout son sens même quand la dimensionnalité augmente. Cette mesure se base sur la notion de voisinage entre les différents objets : un objet est évalué selon son inclusion dans le voisinage des autres objets. Les scores obtenus ne sont pas non plus impactés par la différence de densité, ce qui est également un argument pour utiliser cette mesure sur différents types de données.

Cependant, nous avons montré que le coût de calcul du score d'un nouvel

objet à partir de la méthode classique, proposée par Angiulli, n'était pas adapté pour notre cas d'usage portant sur des flux de données. Nous avons donc proposé d'utiliser un arbre d'indexation permettant de résumer la distribution des voisinages des objets, et un algorithme permettant d'obtenir une approximation du score CFOF en un temps réduit à partir de cet arbre. Les travaux de [Shieh et Keogh, 2008, Shieh et Keogh, 2009] ont montré que la structure *i*SAX était particulièrement adaptée pour l'indexation des séries temporelles et la recherche des séries les plus similaires. Le choix de cette structure pour l'adaptation du score CFOF sur des flux de données a notamment été justifié par sa capacité d'indexation, pouvant aller jusqu'à plusieurs milliards de séries temporelles, comme expliqué dans [Camerra *et al.*, 2010]. Nous avons conservé le processus d'indexation de la méthode *i*SAX, et nous l'avons complété par des pré-traitements permettant ensuite d'obtenir rapidement, à la demande, une approximation de la distribution du voisinage de chaque séquence indexée dans l'arbre. Nous avons proposé une technique sûre d'élagage lors du parcours de l'arbre, basée sur le calcul de bornes sur les valeurs minimales et maximales des distances entre séquences. L'approximation des distributions de voisinage s'effectue alors sur les nœuds feuilles de l'arbre n'ayant pas pu être élagués sur des critères de distance. Un des avantages de notre méthode est la possibilité laissée à l'utilisateur de pouvoir proposer ses propres fonctions d'approximation des distributions.

À l'aide de notre méthode, nous obtenons une approximation $\text{CFOF}_{\text{approx}}$ du score CFOF proposé par Angiulli (noté $\text{CFOF}_{\text{exact}}$). Il a été important de s'assurer d'une part de la cohérence des scores $\text{CFOF}_{\text{approx}}$ au regard des scores $\text{CFOF}_{\text{exact}}$ et d'autre part de vérifier que les coûts de calcul des scores $\text{CFOF}_{\text{approx}}$ étaient plus faibles que ceux des scores $\text{CFOF}_{\text{exact}}$. Dans un premier temps, nous avons comparé ces deux scores sur la famille de jeux de données artificielles *Clust2* proposée et utilisée par Angiulli. Dans un second temps, nous avons effectué une autre évaluation sur un benchmark du domaine, le *Numenta Anomaly Benchmark* (NAB) proposé par [Lavin et Ahmad, 2015] [1]. Ceci nous a permis de montrer sur une large variété de séries de données que les anomalies détectées avec $\text{CFOF}_{\text{approx}}$ ou $\text{CFOF}_{\text{exact}}$ sont très pertinentes.

Sur le plan de l'approximation réalisée, elle s'est révélée de très bonne qualité : pour les jeux *Clust2* les coefficients de corrélation de Spearman sont très proches de 1 dans la grande majorité des configurations testées, et sur les 57 séries du benchmark NAB ces coefficients sont supérieurs à 0.9 pour 53 d'entre eux. En ce qui concerne les coûts de calcul, le nombre de nœuds à visiter dans l'arbre d'indexation pour l'analyse d'un voisinage reste la plu-

1. Disponible sur GitHub : https://github.com/numenta/NAB

part du temps inférieur au nombre d'objets. Pour une majorité de séquences extraites du benchmark NAB, une réduction d'un facteur supérieur à 2 est obtenue, et seul un petit nombre de séries ne permet pas de réduction, voire entraîne un surcoût. Pour les jeux artificiels *Clust2*, le gain quant à lui est toujours d'un facteur supérieur à 5. Enfin pour évaluer la qualité de la détection d'anomalies, nous avons comparé les scores ROC AUC et PRC AUC avec ceux des autres méthodes de détection présentes dans le benchmark NAB. Que ce soit pour $\text{CFOF}_{\text{approx}}$ et pour $\text{CFOF}_{\text{exact}}$, les scores sont pour chaque série presque toujours supérieurs à la moyenne, et la moyenne globale des scores obtenus toutes séries confondues est supérieure à celles des autres méthodes.

Notre seconde contribution a visé à améliorer la réduction du nombre de nœuds à visiter, pour limiter les risques de surcoût dans les cas où nous en avons observés dans les expériences précédentes. Pour cela nous avons souhaité dépasser une des limites de l'arbre d'indexation *i*SAX qui, d'après certaines études comme [Shieh et Keogh, 2009], semble être plutôt adapté pour l'indexation de séries temporelles ayant des dimensions proches de 7. Lorsque la dimension augmente au-delà, l'arbre tend à avoir trop de nœuds à visiter par rapport au nombre de séquences indexées, ce qui réduit son intérêt. Nous avons donc proposé deux méthodes permettant d'approximer le score CFOF d'un objet en combinant les calculs de voisinage effectués dans plusieurs espaces de plus petites dimensions que celui d'origine. Ces deux méthodes, $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et $\text{CFOF}_{\text{agg}}(v\text{-}rang)$, ont permis d'utiliser des arbres d'indexation en plus petite dimension et ainsi de réduire le nombre de nœuds visités.

De nouvelles expérimentations sur les jeux artificiels *Clust2* et sur le benchmark NAB ont montré la réduction du coût de l'approximation avec $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$ et $\text{CFOF}_{\text{agg}}(v\text{-}rang)$, tout en offrant une bonne conservation du rang vis-à-vis du score $\text{CFOF}_{\text{exact}}$, préservant ainsi la qualité de détection des anomalies.

Pour le troisième volet de notre travail, l'objectif était de vérifier la pertinence de notre approche de détection sur les données réelles en provenance du système d'information de la SNCF. Nous avons donc créé un flux de séquences, à partir de traces d'activités extraites de l'infrastructure industrielle de l'entreprise, et construit un historique de référence. Nous avons ensuite calculé le score $\text{CFOF}_{\text{approx}}$ des nouvelles séquences arrivant dans ce flux et mis ces scores à disposition des superviseurs de la SNCF au travers d'une interface dédiée.

Pour évaluer les résultats, nous avons analysé un mois de séquences gé-

nérées à partir des traces d'un point de collecte de la plateforme CanalTrain, avec un historique de référence de dix mois. Nous avons ainsi identifié quatre périodes temporelles ayant des scores $\mathrm{CFOF}_{\mathrm{approx}}$ plus élevés que la normale. Nous avons pu valider avec les experts de supervision de la SNCF que ces détections mettaient en évidence des anomalies avérées. Nous avons par ailleurs comparé les résultats de notre processus de détection avec ceux de l'outil ML-ELK, avec lequel nous avons confirmé la pertinence des détections réalisées. Sur le plan de la mise en œuvre, le principal atout de notre méthode par rapport à l'outil ML-ELK, est qu'elle est entièrement paramétrable, quel que soit le flux étudié. Elle s'avère ainsi satisfaisante et très adaptée à la détection d'anomalies au sein du système d'information d'une grande entreprise industrielle.

6.2 Perspectives

Les méthodes de détection d'anomalies que nous présentons dans ce travail aident à résoudre plusieurs difficultés posées par la question de la détection d'anomalies dans un flux de données industrielles. Nous avons cependant identifié plusieurs pistes d'améliorations qui pourraient être explorées dans le futur.

Mise à jour de l'historique de référence. Lorsqu'une nouvelle séquence arrive, il est nécessaire de mettre à jour l'arbre d'indexation utilisé pour la recherche de voisinage. Pour l'instant, cette mise à jour n'est pas effectuée incrémentalement, et nous recalculons périodiquement les distributions des voisinages pour toutes les séquences déjà indexées. La réalisation d'une mise à jour incrémentale doit tenir compte non seulement des suites d'insertions mais aussi du retrait possible de l'historique de référence de séquences considérées comme obsolètes. Ainsi, la structure de l'arbre, initialement équilibrée lors de la première construction à partir de l'historique d'origine, pourra se retrouver rapidement déséquilibrée en cas de mise à jour incrémentale. Pour éviter cette difficulté, il serait intéressant de proposer une analyse dynamique de la distribution des séquences à fournir à l'arbre iSAX et il serait aussi nécessaire d'intégrer des fonctions de rééquilibrage des séquences stockées tout en minimisant les coûts de calcul par rapport à une reconstruction totale d'un nouvel arbre.

Échantillonnage des séquences de référence pour l'approximation. Comme proposé par [Angiulli, 2017, Angiulli, 2020], il est possible d'utiliser un échantillon d'éléments de la base de référence pour approximer

le score CFOF. Ceci est tout à fait applicable conjointement à nos méthodes, et pourrait permettre d'approximer encore plus rapidement le score CFOF. De plus, dans notre cas, les nœuds de l'arbre nous informent sur la répartition des séquences dans l'espace, et il semble possible de mettre ces informations à profit pour orienter la sélection aléatoire des éléments afin de limiter la taille de l'échantillon tout en le gardant représentatif de la distribution d'ensemble.

Extension de l'application SNCF à de la détection multi-flux. Pour l'instant, notre méthode n'a été testée que sur un seul flux — le nombre de messages reçus par minute — et sur un seul point de collecte du système de communication. Nous pourrions analyser d'autres indicateurs, comme les temps de latence ou l'écart-type du nombre de messages reçus par minute (comme cela a été présenté sur les jeux du benchmark NAB dans le chapitre 3), et ainsi utiliser pour la détection des fluctuations très locales de transmissions ou de valeur du signal analysé. Une autre source d'information disponible est la ventilation du nombre de messages par minute selon deux attributs : le type de service (géolocalisation, télémaintenance, ...), et le type d'engin (TGV, TER, ...). En créant un flux par paire d'attributs $\langle$type de service, type d'engin$\rangle$ existante, nous aurions plusieurs dizaines de flux à analyser sur un seul point de collecte, mais nous pensons que cela permettrait d'identifier plus rapidement l'origine d'une anomalie. En effet, si plusieurs signaux ayant la même valeur pour l'un des attributs retournent des scores d'anomalie élevés, les superviseurs du système d'information pourraient inspecter en priorité le ou les composants correspondants à cette valeur commune.

Annexe A

Comparatifs ROC AUC et PRC AUC sur les séries du benchmark NAB

Nous présentons dans cet annexe de façon plus détaillée certains des résultats des expérimentations des chapitres 3 et 4.

Pour chacun des jeux de données du *Numenta Anomaly Benchmark* (NAB) proposé par [Lavin et Ahmad, 2015], et disponible sur GitHub [1], nous décrivons les résultats ROC AUC pour chacune des méthodes de détection présentes dans le benchmark NAB et pour les méthodes basées sur CFOF exact et approximé. Ensuite, nous donnerons les résultats PRC AUC correspondants. Pour chacun des deux scores ROC AUC et PRC AUC, les scores sont présentés par famille de jeux de données du benchmark NAB.

1. https://github.com/numenta/NAB

A.1 Comparatifs ROC AUC

A.1.1 Les séries "artificialWithAnomaly"

	art_daily_flatmiddle	art_daily_jumpsdown	art_daily_jumpsup	art_daily_nojump	art_increase_spike_density
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.667417	0.647451	0.754190	0.639654	0.646797
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.703178	0.615048	0.770470	0.572709	0.651498
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.669075	0.651315	0.759979	0.644331	0.646636
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.709009	0.625600	0.779632	0.592959	0.651268
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.643816	0.572103	0.724170	0.574422	0.672537
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.704996	0.581459	0.735494	0.574267	0.668738
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.645018	0.575048	0.728665	0.577855	0.672493
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.702950	0.588819	0.739419	0.580411	0.668647
$CFOF_{approx}$ 10-20	0.700584	0.688960	0.747495	0.657271	0.647596
$CFOF_{approx}$ 30	0.704122	0.693643	0.754379	0.663307	0.647350
$CFOF_{exact}$	0.709160	0.691639	0.749258	0.664960	0.724165
bayesChangePt	0.499670	0.499831	0.503193	0.497022	0.504860
contextOSE	0.219640	0.367301	0.147471	0.775320	0.280727
earthgeckoSkyline	0.500000	0.500000	0.501241	0.500000	0.501241
expose	0.385322	0.466131	0.558213	0.230080	0.672381
htmjava	0.769327	0.529008	0.540740	0.639824	0.623083
knncad	0.546368	0.434256	0.366843	0.330867	0.517667
null	0.500000	0.500000	0.500000	0.500000	0.500000
numenta	0.594832	0.519976	0.517261	0.459637	0.261442
numentaTM	0.603642	0.612667	0.493216	0.383173	0.347205
random	0.489510	0.500119	0.500119	0.500119	0.512260
randomCutForest	0.768629	0.548513	0.598035	0.352341	0.378170
relativeEntropy	0.511001	0.503392	0.513648	0.499504	0.500000
skyline	0.744802	0.484664	0.537935	0.442050	0.511224
twitterADVec	0.504302	0.501490	0.509926	0.501490	0.500414
windowedGaussian	0.310771	0.264718	0.527132	0.361468	0.524917

	art_load_balancer_spikes
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.729917
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.754596
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.734080
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.756673
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.740357
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.768961
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.742346
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.772184
$CFOF_{approx}$ 10-20	0.741022
$CFOF_{approx}$ 30	0.736980
$CFOF_{exact}$	0.742541
bayesChangePt	0.510923
contextOSE	0.324720
earthgeckoSkyline	0.499918
expose	0.761614
htmjava	0.726749
knncad	0.639016
null	0.500000
numenta	0.795353
numentaTM	0.786808
random	0.497120
randomCutForest	0.692667
relativeEntropy	0.503557
skyline	0.637658
twitterADVec	0.505873
windowedGaussian	0.503171

A.1.2 Les séries “realAWSCloudwatch”

	ec2_cpu_utilization_24ae8d	ec2_cpu_utilization_53ea38	ec2_cpu_utilization_5f5533	ec2_cpu_utilization_77c1ca	ec2_cpu_utilization_825cc2
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.780199	0.667929	0.723418	0.852916	0.844007
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.791845	0.671028	0.700102	0.855411	0.866799
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.787567	0.663581	0.695775	0.852458	0.850671
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.796610	0.666345	0.702625	0.855629	0.865445
$CFOF_{agg}$(*v-rang*) 10-20 2 arbres	0.787410	0.653540	0.782143	0.850865	0.868891
$CFOF_{agg}$(*v-rang*) 10-20 4 arbres	0.754288	0.647890	0.775542	0.866275	0.879870
$CFOF_{agg}$(*v-rang*) 30 2 arbres	0.791425	0.648295	0.776723	0.851028	0.873090
$CFOF_{agg}$(*v-rang*) 30 4 arbres	0.755657	0.645196	0.775683	0.865376	0.879450
$CFOF_{approx}$ 10-20	0.770016	0.669652	0.761796	0.834891	0.819457
$CFOF_{approx}$ 30	0.772573	0.662888	0.773313	0.833696	0.830354
$CFOF_{exact}$	0.779784	0.664886	0.763549	0.836240	0.820340
bayesChangePt	0.504380	0.496273	0.499002	0.503636	0.494085
contextOSE	0.399647	0.469245	0.275735	0.532243	0.121228
earthgeckoSkyline	0.499426	0.501244	0.501244	0.501324	0.502915
expose	0.670935	0.601216	0.686990	0.688029	0.534056
htmjava	0.908682	0.496487	0.550931	0.692269	0.533672
knncad	0.473693	0.510686	0.558789	0.577702	0.724537
null	0.500000	0.500000	0.500000	0.500000	0.500000
numenta	0.406767	0.585507	0.609032	0.709898	0.514186
numentaTM	0.848575	0.546179	0.545677	0.695428	0.515223
random	0.508666	0.500968	0.508372	0.507069	0.519870
randomCutForest	0.448745	0.526535	0.531929	0.592443	0.715323
relativeEntropy	0.501079	0.503566	0.508541	0.501159	0.511500
skyline	0.543392	0.521836	0.506998	0.562378	0.705335
twitterADVec	0.501661	0.501661	0.501244	0.500084	0.511662
windowedGaussian	0.353543	0.561351	0.501747	0.343725	0.809377

	ec2_cpu_utilization_ac20cd	ec2_cpu_utilization_c6585a	ec2_cpu_utilization_fe7f93	ec2_disk_write_bytes_1ef3de	ec2_disk_write_bytes_c0d644
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.630967	1.0	0.599250	0.663025	0.632935
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.603782	1.0	0.545022	0.680707	0.606821
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.624308	1.0	0.610995	0.663718	0.634759
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.628099	1.0	0.548560	0.680503	0.607884
$CFOF_{agg}$(*v-rang*) 10-20 2 arbres	0.717162	1.0	0.595397	0.641151	0.640529
$CFOF_{agg}$(*v-rang*) 10-20 4 arbres	0.662089	1.0	0.580334	0.649607	0.604905
$CFOF_{agg}$(*v-rang*) 30 2 arbres	0.712640	1.0	0.603570	0.640503	0.639088
$CFOF_{agg}$(*v-rang*) 30 4 arbres	0.668708	1.0	0.581758	0.649501	0.608003
$CFOF_{approx}$ 10-20	0.649129	1.0	0.607197	0.644156	0.662361
$CFOF_{approx}$ 30	0.652849	1.0	0.616537	0.647312	0.663931
$CFOF_{exact}$	0.649371	1.0	0.602080	0.647675	0.667318
bayesChangePt	0.503634	1.0	0.492514	0.500778	0.504716
contextOSE	0.376387	1.0	0.273654	0.328591	0.429207
earthgeckoSkyline	0.502481	1.0	0.500650	0.500423	0.501973
expose	0.405554	1.0	0.412926	0.509777	0.554079
htmjava	0.523070	1.0	0.456514	0.541354	0.601152
knncad	0.385222	1.0	0.567188	0.516346	0.481924
null	0.500000	1.0	0.500000	0.500000	0.500000
numenta	0.507445	1.0	0.363293	0.387155	0.624139
numentaTM	0.504681	1.0	0.377524	0.602141	0.590186
random	0.503102	1.0	0.509192	0.479504	0.498038
randomCutForest	0.691671	1.0	0.507231	0.577671	0.549912
relativeEntropy	0.507114	1.0	0.503208	0.502608	0.507815
skyline	0.715011	1.0	0.516077	0.531919	0.587504
twitterADVec	0.508520	1.0	0.501477	0.498732	0.501477
windowedGaussian	0.723987	1.0	0.588545	0.664985	0.465988

	ec2_network_in_257a54	ec2_network_in_5abac7	elb_request_count_8c0756	grok_asg_anomaly	iio_us-east-1_i-a2eb1cd9_NetworkIn
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.880881	0.598181	0.700429	0.597718	0.902204
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.879077	0.639941	0.745044	0.616780	0.866682
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.880760	0.598482	0.698551	0.599991	0.874141
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.881222	0.636095	0.746490	0.625366	0.863664
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.865444	0.590222	0.680111	0.577877	0.692492
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.873042	0.678684	0.670594	0.632997	0.708148
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.860206	0.588922	0.678874	0.577194	0.680758
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.874123	0.676473	0.669416	0.633705	0.688703
$CFOF_{approx}$ 10-20	0.853514	0.590316	0.675644	0.518362	0.846666
$CFOF_{approx}$ 30	0.854471	0.586888	0.679285	0.515113	0.845890
$CFOF_{exact}$	0.742006	0.598347	0.665037	0.535178	0.767280
bayesChangePt	0.522526	0.524637	0.492398	0.500359	0.510605
contextOSE	0.504276	0.239038	0.243721	0.310545	0.649391
earthgeckoSkyline	0.501241	0.501123	0.502322	0.502360	0.499463
expose	0.494259	0.595805	0.566827	0.624217	0.517800
htmjava	0.464467	0.754732	0.533736	0.535555	0.467904
knncad	0.713545	0.615310	0.492034	0.632174	0.558991
null	0.500000	0.500000	0.500000	0.500000	0.500000
numenta	0.470842	0.672545	0.437357	0.553274	0.233829
numentaTM	0.469622	0.724891	0.529300	0.554446	0.352531
random	0.524962	0.508692	0.536241	0.508256	0.531720
randomCutForest	0.749097	0.503494	0.521098	0.524111	0.584326
relativeEntropy	0.503557	0.499436	0.501244	0.505088	0.498926
skyline	0.635856	0.593788	0.516451	0.518551	0.455177
twitterADVec	0.507279	0.501123	0.504645	0.499920	0.500000
windowedGaussian	0.747929	0.536642	0.502069	0.480259	0.643846

	rds_cpu_utilization_cc0c53	rds_cpu_utilization_e47b3b
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.498254	0.690675
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.527673	0.702551
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.524446	0.675405
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.529331	0.699604
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.454919	0.702379
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.517084	0.692304
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.460386	0.694061
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.519596	0.691296
$CFOF_{approx}$ 10-20	0.588193	0.694905
$CFOF_{approx}$ 30	0.615195	0.679694
$CFOF_{exact}$	0.525336	0.712869
bayesChangePt	0.490088	0.519235
contextOSE	0.182693	0.363886
earthgeckoSkyline	0.502157	0.502322
expose	0.795504	0.641396
htmjava	0.820682	0.383351
knncad	0.526228	0.680086
null	0.500000	0.500000
numenta	0.792843	0.416062
numentaTM	0.793049	0.384993
random	0.510924	0.504082
randomCutForest	0.762899	0.651292
relativeEntropy	0.508541	0.508706
skyline	0.771127	0.652729
twitterADVec	0.505723	0.501496
windowedGaussian	0.778789	0.572891

A.1.3 Les séries “realAdExchange”

	exchange-2_cpc_results	exchange-2_cpm_results	exchange-3_cpc_results	exchange-3_cpm_results	exchange-4_cpc_results
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.591591	0.492749	0.895227	0.558917	0.697422
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.581593	0.554371	0.917607	0.578632	0.728266
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.595324	0.477481	0.902657	0.573234	0.706680
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.610293	0.568630	0.913856	0.574643	0.728382
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.445561	0.490434	0.885284	0.731873	0.695307
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.423902	0.506963	0.889930	0.668670	0.733102
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.452663	0.482925	0.888199	0.732405	0.703030
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.419827	0.511389	0.884288	0.660793	0.739940
$CFOF_{approx}$ 10-20	0.597066	0.495782	0.881063	0.597255	0.616736
$CFOF_{approx}$ 30	0.458445	0.509793	0.827966	0.631259	0.606341
$CFOF_{exact}$	0.497131	0.507165	0.881609	0.605124	0.611314
bayesChangePt	0.496167	0.510763	0.523430	0.495951	0.509231
contextOSE	0.236466	0.383119	0.236115	0.228056	0.397238
earthgeckoSkyline	0.500000	0.500000	0.509804	0.501536	0.504843
expose	0.500136	0.430413	0.801624	0.459429	0.521738
htmjava	0.305696	0.562341	0.448244	0.532617	0.504258
knncad	0.493334	0.568856	0.580726	0.324590	0.444562
null	0.500000	0.500000	0.500000	0.500000	0.500000
numenta	0.466021	0.500362	0.515811	0.457627	0.627007
numentaTM	0.253020	0.494546	0.479241	0.528973	0.456102
random	0.510381	0.515313	0.502255	0.468347	0.523549
randomCutForest	0.491392	0.503803	0.616162	0.511841	0.629295
relativeEntropy	0.502440	0.498359	0.508938	0.506103	0.508685
skyline	0.528506	0.488019	0.554155	0.492822	0.551370
twitterADVec	0.500000	0.503086	0.509804	0.503268	0.505655
windowedGaussian	0.482623	0.507727	0.623589	0.393300	0.596266

	exchange-4_cpm_results
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.617369
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.672003
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.601811
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.666830
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.630358
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.647504
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.634685
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.634586
$CFOF_{approx}$ 10-20	0.626036
$CFOF_{approx}$ 30	0.629973
$CFOF_{exact}$	0.638963
bayesChangePt	0.504864
contextOSE	0.373061
earthgeckoSkyline	0.508741
expose	0.513313
htmjava	0.612108
knncad	0.565246
null	0.500000
numenta	0.697718
numentaTM	0.583014
random	0.479962
randomCutForest	0.496855
relativeEntropy	0.505692
skyline	0.531465
twitterADVec	0.505692
windowedGaussian	0.472084

A.1.4 Les séries “realKnownCause”

	ambient_temperature_system_failure	cpu_utilization_asg_misconfiguration	ec2_request_latency_system_failure	machine_temperature_system_failure	nyc_taxi
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.712648	0.877271	0.706719	0.898852	0.751392
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.722864	0.851031	0.680240	0.901193	0.733255
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.715744	0.878131	0.716417	0.899288	0.771010
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.716243	0.852270	0.690612	0.899467	0.738637
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.690488	0.867018	0.670722	0.911119	0.744737
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.675347	0.845193	0.661281	0.924681	0.714795
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.691923	0.868489	0.675579	0.911456	0.755809
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.675911	0.846192	0.663991	0.924320	0.715980
$CFOF_{approx}$ 10-20	0.715603	0.869392	0.653545	0.934931	0.779548
$CFOF_{approx}$ 30	0.734788	0.872567	0.653168	0.938352	0.789508
$CFOF_{exact}$	0.715502	0.867198	0.676543	0.934682	0.767926
bayesChangePt	0.500310	0.497054	0.500098	0.507605	0.501149
contextOSE	0.415311	0.693397	0.323697	0.538062	0.363136
earthgeckoSkyline	0.502066	0.500748	0.502890	0.501221	0.500000
expose	0.619833	0.717223	0.525206	0.682522	0.506899
htmjava	0.691581	0.850286	0.555155	0.579340	0.588491
knncad	0.536310	0.489496	0.585314	0.486175	0.405864
null	0.500000	0.500000	0.500000	0.500000	0.500000
numenta	0.680344	0.731814	0.575618	0.627134	0.585997
numentaTM	0.718979	0.810579	0.564854	0.493084	0.342892
random	0.502183	0.500897	0.486442	0.498854	0.487104
randomCutForest	0.651737	0.802049	0.468688	0.876149	0.558373
relativeEntropy	0.501030	0.503844	0.507225	0.503790	0.503323
skyline	0.552103	0.727788	0.521988	0.736527	0.575121
twitterADVec	0.500000	0.508388	0.511561	0.509921	0.500000
windowedGaussian	0.721657	0.712303	0.484342	0.858003	0.504750

	rogue_agent_key_hold	rogue_agent_key_updown
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.570355	0.579385
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.357294	0.499863
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.557798	0.587148
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.366084	0.497958
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.493572	0.519294
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.442199	0.483609
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.490974	0.527612
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.446749	0.480123
$CFOF_{approx}$ 10-20	0.576603	0.592912
$CFOF_{approx}$ 30	0.576592	0.582514
$CFOF_{exact}$	0.558520	0.583638
bayesChangePt	0.499367	0.504890
contextOSE	0.473770	0.518895
earthgeckoSkyline	0.498936	0.499380
expose	0.299220	0.431068
htmjava	0.703432	0.420383
knncad	0.428828	0.542828
null	0.500000	0.500000
numenta	0.532100	0.520839
numentaTM	0.658666	0.583942
random	0.502113	0.510745
randomCutForest	0.458936	0.654495
relativeEntropy	0.499795	0.499380
skyline	0.503658	0.514890
twitterADVec	0.499291	0.499952
windowedGaussian	0.460642	0.272844

A.1.5 Les séries “realTraffic”

	TravelTime_387	TravelTime_451	occupancy_6005	occupancy_t4013	speed_6005
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.608671	0.829334	0.668389	0.729596	0.724353
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.631301	0.866752	0.713124	0.710276	0.721064
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.621867	0.841683	0.684628	0.739991	0.736277
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.582138	0.871468	0.712301	0.713229	0.694479
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.552894	0.740173	0.712315	0.686739	0.750407
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.650463	0.742134	0.662164	0.709967	0.764461
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.551126	0.751672	0.719109	0.689619	0.758647
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.627887	0.728901	0.664212	0.710607	0.754363
$CFOF_{approx}$ 10-20	0.579983	0.799538	0.646718	0.716448	0.766451
$CFOF_{approx}$ 30	0.592425	0.815645	0.644225	0.731465	0.777040
$CFOF_{exact}$	0.553901	0.767536	0.665438	0.698171	0.767465
bayesChangePt	0.500882	0.502071	0.499597	0.519100	0.508997
contextOSE	0.254809	0.130735	0.331387	0.349786	0.501141
earthgeckoSkyline	0.502950	0.502304	0.501532	0.504000	0.502092
expose	0.617286	0.549513	0.440496	0.647859	0.586262
htmjava	0.529170	0.345805	0.597915	0.774099	0.737629
knncad	0.532031	0.537269	0.396501	0.431532	0.457592
null	0.500000	0.500000	0.500000	0.500000	0.500000
numenta	0.536344	0.637864	0.450267	0.661087	0.770815
numentaTM	0.487536	0.349685	0.678556	0.853866	0.716512
random	0.506658	0.486663	0.521425	0.484789	0.504226
randomCutForest	0.551534	0.539099	0.533893	0.565675	0.519880
relativeEntropy	0.505225	0.501996	0.498729	0.515467	0.506011
skyline	0.539262	0.537391	0.502192	0.522929	0.523245
twitterADVec	0.500942	0.500000	0.501812	0.510000	0.506276
windowedGaussian	0.456994	0.523992	0.544123	0.580203	0.536477

	speed_7578	speed_t4013
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.820430	0.875141
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.809086	0.869251
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.806843	0.873005
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.786971	0.869535
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.799052	0.893105
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.757781	0.892430
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.796144	0.890916
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.757535	0.888870
$CFOF_{approx}$ 10-20	0.786725	0.868778
$CFOF_{approx}$ 30	0.793723	0.871885
$CFOF_{exact}$	0.789520	0.870311
bayesChangePt	0.525524	0.487573
contextOSE	0.232114	0.295068
earthgeckoSkyline	0.511743	0.504000
expose	0.661408	0.580639
htmjava	0.681541	0.850278
knncad	0.681296	0.609494
null	0.500000	0.500000
numenta	0.699581	0.756919
numentaTM	0.733936	0.816527
random	0.578702	0.486615
randomCutForest	0.657640	0.452988
relativeEntropy	0.520958	0.506000
skyline	0.670934	0.577824
twitterADVec	0.508621	0.508000
windowedGaussian	0.629889	0.513311

A.1.6 Les séries "realTweets"

	Twitter_volume_AAPL	Twitter_volume_AMZN	Twitter_volume_CRM	Twitter_volume_CVS	Twitter_volume_FB
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.819248	0.780202	0.833799	0.818046	0.728655
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.851908	0.814238	0.833698	0.827063	0.750711
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.815070	0.769459	0.833142	0.819101	0.729763
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.849318	0.804615	0.829998	0.826890	0.751113
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.801744	0.768633	0.805597	0.817880	0.710690
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.819234	0.794581	0.803279	0.830338	0.702326
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.799464	0.760299	0.804698	0.817664	0.713087
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.816922	0.777725	0.799653	0.830031	0.703491
$CFOF_{approx}$ 10-20	0.808732	0.766275	0.802565	0.824775	0.715824
$CFOF_{approx}$ 30	0.813242	0.763207	0.800072	0.825178	0.716918
$CFOF_{exact}$	0.791831	0.780303	0.803620	0.828531	0.722677
bayesChangePt	0.501045	0.499586	0.489247	0.500035	0.499444
contextOSE	0.239211	0.247256	0.292787	0.254005	0.353004
earthgeckoSkyline	0.501521	0.501101	0.500850	0.501016	0.500188
expose	0.671540	0.590430	0.664644	0.591152	0.461499
htmjava	0.286741	0.579887	0.580145	0.463766	0.418597
knncad	0.674967	0.593214	0.639095	0.567472	0.607103
null	0.500000	0.500000	0.500000	0.500000	0.500000
numenta	0.287552	0.576530	0.590197	0.632815	0.504456
numentaTM	0.295554	0.580464	0.576228	0.509524	0.505028
random	0.505959	0.504415	0.506228	0.508857	0.492684
randomCutForest	0.687922	0.526550	0.614327	0.497753	0.493974
relativeEntropy	0.501112	0.502215	0.501219	0.502220	0.500837
skyline	0.666503	0.531322	0.578286	0.507058	0.525471
twitterADVec	0.504133	0.502777	0.506573	0.506512	0.500951
windowedGaussian	0.677526	0.538978	0.558114	0.493308	0.530341

	Twitter_volume_GOOG	Twitter_volume_IBM	Twitter_volume_KO	Twitter_volume_PFE	Twitter_volume_UPS
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.761386	0.627123	0.728853	0.792517	0.748208
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.741164	0.598169	0.748887	0.816632	0.751669
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.760989	0.626896	0.730064	0.792250	0.743906
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.747385	0.601696	0.752263	0.815001	0.748596
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.798198	0.630745	0.747435	0.776604	0.766888
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.783109	0.616751	0.712689	0.814360	0.758232
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.798092	0.631021	0.748777	0.776806	0.762670
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.785925	0.618937	0.714492	0.812972	0.757350
$CFOF_{approx}$ 10-20	0.780209	0.624553	0.760065	0.772614	0.747224
$CFOF_{approx}$ 30	0.780857	0.626751	0.759931	0.772870	0.744361
$CFOF_{exact}$	0.781858	0.631728	0.763983	0.757280	0.747048
bayesChangePt	0.505344	0.495663	0.499933	0.504548	0.503852
contextOSE	0.293676	0.486985	0.339621	0.167899	0.317478
earthgeckoSkyline	0.500957	0.500299	0.500520	0.500927	0.500932
expose	0.548735	0.529847	0.540433	0.618862	0.578470
htmjava	0.504229	0.672711	0.481571	0.436344	0.696394
knncad	0.533415	0.565023	0.571518	0.657046	0.559459
null	0.500000	0.500000	0.500000	0.500000	0.500000
numenta	0.626106	0.675513	0.494657	0.546500	0.617759
numentaTM	0.527252	0.663436	0.493033	0.416736	0.687531
random	0.482424	0.505538	0.504307	0.493305	0.499892
randomCutForest	0.577354	0.452075	0.452999	0.508092	0.682083
relativeEntropy	0.500974	0.501073	0.500556	0.501112	0.504158
skyline	0.554724	0.520672	0.527982	0.524397	0.578200
twitterADVec	0.507499	0.504164	0.503115	0.503816	0.501674
windowedGaussian	0.624714	0.476789	0.516424	0.467974	0.582661

A.2 Comparatifs PRC AUC

A.2.1 Les séries "artificialWithAnomaly"

	art_daily_flatmiddle	art_daily_jumpsdown	art_daily_jumpsup	art_daily_nojump	art_increase_spike_density
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.184697	0.164365	0.531819	0.185229	0.242617
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.195817	0.145395	0.540499	0.141529	0.341148
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.185276	0.165750	0.533713	0.185467	0.242329
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.197599	0.148449	0.542400	0.147105	0.340472
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.170809	0.129193	0.541969	0.156542	0.227458
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.236356	0.129442	0.531277	0.143730	0.248555
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.171752	0.130067	0.543270	0.155189	0.227342
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.230159	0.131346	0.533812	0.143611	0.247722
$CFOF_{approx}$ 10-20	0.250464	0.224096	0.587420	0.238734	0.213126
$CFOF_{approx}$ 30	0.252534	0.225124	0.588330	0.239881	0.212926
$CFOF_{exact}$	0.259901	0.228234	0.589161	0.245002	0.238353
baseline	0.117561	0.117561	0.117561	0.117561	0.117561
bayesChangePt	0.114095	0.115669	0.145796	0.087122	0.144784
contextOSE	0.074407	0.089778	0.069673	0.163760	0.082192
earthgeckoSkyline	0.558781	0.558781	0.559875	0.558781	0.559875
expose	0.098489	0.105244	0.163048	0.071438	0.270208
htmjava	0.379671	0.245318	0.299058	0.175166	0.165580
knncad	0.116446	0.095607	0.085548	0.084611	0.118792
null	0.558781	0.558781	0.558781	0.558781	0.558781
numenta	0.166515	0.209821	0.352267	0.137941	0.089079
numentaTM	0.457982	0.220512	0.377068	0.094891	0.106479
random	0.122211	0.125790	0.125790	0.125790	0.125967
randomCutForest	0.388606	0.302607	0.433249	0.083960	0.091877
relativeEntropy	0.518634	0.362065	0.570824	0.058781	0.558781
skyline	0.342107	0.159042	0.376067	0.087842	0.153096
twitterADVec	0.313160	0.185970	0.567539	0.185970	0.143209
windowedGaussian	0.087210	0.074029	0.362110	0.084535	0.185741

	art_load_balancer_spikes
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.580659
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.584830
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.575198
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.584333
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.548222
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.684493
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.545026
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.686024
$CFOF_{approx}$ 10-20	0.568629
$CFOF_{approx}$ 30	0.568784
$CFOF_{exact}$	0.590772
baseline	0.117561
bayesChangePt	0.133465
contextOSE	0.083426
earthgeckoSkyline	0.115431
expose	0.393326
htmjava	0.417946
knncad	0.181188
null	0.558781
numenta	0.408355
numentaTM	0.373085
random	0.126713
randomCutForest	0.300781
relativeEntropy	0.437065
skyline	0.297265
twitterADVec	0.421398
windowedGaussian	0.235786

A.2.2 Les séries "realAWSCloudwatch"

	ec2_cpu_utilization_24ae8d	ec2_cpu_utilization_53ea38	ec2_cpu_utilization_5f5533	ec2_cpu_utilization_77c1ca	ec2_cpu_utilization_825cc2
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.257313	0.262818	0.240239	0.597288	0.612594
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.312204	0.293375	0.196144	0.619446	0.678450
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.257422	0.265210	0.246561	0.595687	0.616159
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.305658	0.280005	0.196202	0.617752	0.630913
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.254559	0.217566	0.323679	0.540180	0.635923
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.308243	0.245595	0.214730	0.539435	0.597629
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.255779	0.221510	0.321667	0.539078	0.629098
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.306465	0.244088	0.214974	0.537836	0.565987
$CFOF_{approx}$ 10-20	0.250253	0.315923	0.257792	0.516752	0.613222
$CFOF_{approx}$ 30	0.244640	0.307749	0.264887	0.515244	0.603772
$CFOF_{exact}$	0.256297	0.317728	0.261863	0.519323	0.627443
baseline	0.117270	0.117270	0.117270	0.117561	0.100058
bayesChangePt	0.214422	0.113733	0.119508	0.124018	0.078465
contextOSE	0.184294	0.151227	0.099777	0.173420	0.072224
earthgeckoSkyline	0.101399	0.559733	0.559733	0.172081	0.552653
expose	0.278976	0.167443	0.192181	0.179713	0.151002
htmjava	0.380515	0.188253	0.281112	0.222201	0.410936
knncad	0.106830	0.130137	0.119633	0.167787	0.169781
null	0.558635	0.558635	0.558635	0.558781	0.550029
numenta	0.116589	0.196931	0.375743	0.288368	0.400351
numentaTM	0.325977	0.181270	0.304959	0.239215	0.412359
random	0.118039	0.118357	0.123096	0.121911	0.107212
randomCutForest	0.109022	0.147771	0.199666	0.220304	0.416493
relativeEntropy	0.309733	0.436929	0.503820	0.160970	0.504969
skyline	0.166091	0.175116	0.223848	0.221721	0.505733
twitterADVec	0.203688	0.203688	0.559733	0.122375	0.560524
windowedGaussian	0.095531	0.155893	0.190652	0.182985	0.573697

	ec2_cpu_utilization_ac20cd	ec2_cpu_utilization_c6585a	ec2_cpu_utilization_fe7f93	ec2_disk_write_bytes_1ef3de	ec2_disk_write_bytes_c0d644
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.530477	1.000000	0.310665	0.347934	0.247660
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.366010	1.000000	0.310145	0.457184	0.254163
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.531431	1.000000	0.312827	0.340934	0.249287
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.355154	1.000000	0.309012	0.453854	0.257135
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.534945	1.000000	0.312047	0.319778	0.276384
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.507553	1.000000	0.294915	0.300935	0.310138
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.533786	1.000000	0.313311	0.312853	0.276921
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.508596	1.000000	0.294089	0.299745	0.312928
$CFOF_{approx}$ 10-20	0.502674	1.000000	0.330081	0.335837	0.245875
$CFOF_{approx}$ 30	0.511305	1.000000	0.331920	0.330100	0.246488
$CFOF_{exact}$	0.539788	1.000000	0.331276	0.314336	0.242864
baseline	0.117561	0.000000	0.118145	0.117632	0.118145
bayesChangePt	0.125598	1.000000	0.101207	0.121851	0.123472
contextOSE	0.116433	1.000000	0.078969	0.088145	0.101984
earthgeckoSkyline	0.560970	1.000000	0.138173	0.132110	0.261250
expose	0.113151	1.000000	0.121792	0.136247	0.161084
htmjava	0.523116	1.000000	0.187907	0.164256	0.274172
knncad	0.089996	1.000000	0.168283	0.139783	0.119232
null	0.558781	1.000000	0.559072	0.558816	0.559072
numenta	0.463272	1.000000	0.157065	0.140279	0.248503
numentaTM	0.469543	1.000000	0.178950	0.163593	0.201016
random	0.116932	1.000000	0.122996	0.115410	0.121495
randomCutForest	0.520552	1.000000	0.163069	0.144528	0.173305
relativeEntropy	0.440350	1.000000	0.312338	0.275900	0.358360
skyline	0.569092	1.000000	0.157020	0.161986	0.208194
twitterADVec	0.503944	1.000000	0.186250	0.058816	0.186250
windowedGaussian	0.592984	1.000000	0.202123	0.171327	0.146834

	ec2_network_in_257a54	ec2_network_in_5abac7	elb_request_count_8c0756	grok_asg_anomaly	iio_us-east-1_i-a2eb1cd9_NetworkIn
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.651085	0.278830	0.438145	0.215419	0.527348
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.652387	0.320928	0.485723	0.230608	0.538076
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.651770	0.277632	0.434037	0.216745	0.506025
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.654393	0.314881	0.488342	0.235319	0.540621
$CFOF_{agg}$(*v-rang*) 10-20 2 arbres	0.640729	0.264221	0.435090	0.236849	0.350736
$CFOF_{agg}$(*v-rang*) 10-20 4 arbres	0.608691	0.355615	0.469656	0.183580	0.325332
$CFOF_{agg}$(*v-rang*) 30 2 arbres	0.638292	0.262717	0.434099	0.253255	0.346124
$CFOF_{agg}$(*v-rang*) 30 4 arbres	0.608809	0.349819	0.468706	0.183134	0.320479
$CFOF_{approx}$ 10-20	0.630323	0.265382	0.406570	0.134986	0.334036
$CFOF_{approx}$ 30	0.631118	0.264696	0.406862	0.134324	0.369804
$CFOF_{exact}$	0.591451	0.271186	0.400830	0.238739	0.252123
baseline	0.117561	0.117881	0.117270	0.118381	0.119205
bayesChangePt	0.156060	0.169626	0.098054	0.123025	0.173069
contextOSE	0.360610	0.076760	0.088502	0.093023	0.484238
earthgeckoSkyline	0.559875	0.171913	0.394164	0.228701	0.059603
expose	0.119175	0.165892	0.137834	0.184209	0.144324
htmjava	0.150816	0.283819	0.203666	0.165708	0.382899
knncad	0.230391	0.245924	0.107209	0.149149	0.140649
null	0.558781	0.558941	0.558635	0.559190	0.559603
numenta	0.429757	0.251735	0.174393	0.162867	0.122919
numentaTM	0.429653	0.256265	0.199948	0.184714	0.242931
random	0.127217	0.118324	0.129115	0.118658	0.142709
randomCutForest	0.342686	0.159436	0.144735	0.271136	0.265985
relativeEntropy	0.437065	0.058941	0.559733	0.421073	0.059603
skyline	0.241111	0.224214	0.197108	0.172973	0.150624
twitterADVec	0.493921	0.171913	0.396360	0.115694	0.559603
windowedGaussian	0.233776	0.156880	0.141752	0.262174	0.176054

	rds_cpu_utilization_cc0c53	rds_cpu_utilization_e47b3b
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.252448	0.377047
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.272473	0.377375
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.256661	0.373808
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.272689	0.376381
$CFOF_{agg}$(*v-rang*) 10-20 2 arbres	0.212212	0.515230
$CFOF_{agg}$(*v-rang*) 10-20 4 arbres	0.191051	0.403989
$CFOF_{agg}$(*v-rang*) 30 2 arbres	0.216590	0.501290
$CFOF_{agg}$(*v-rang*) 30 4 arbres	0.191208	0.393465
$CFOF_{approx}$ 10-20	0.261281	0.311321
$CFOF_{approx}$ 30	0.267084	0.313319
$CFOF_{exact}$	0.257122	0.314608
baseline	0.117270	0.117270
bayesChangePt	0.089789	0.193986
contextOSE	0.131405	0.169109
earthgeckoSkyline	0.310831	0.394164
expose	0.422559	0.322548
htmjava	0.499034	0.290546
knncad	0.176289	0.211239
null	0.558635	0.558635
numenta	0.464380	0.336138
numentaTM	0.486417	0.317974
random	0.118829	0.118923
randomCutForest	0.422558	0.383020
relativeEntropy	0.503820	0.566320
skyline	0.417090	0.404890
twitterADVec	0.376624	0.185831
windowedGaussian	0.463337	0.336852

A.2.3 Les séries "realAdExchange"

	exchange-2_cpc_results	exchange-2_cpm_results	exchange-3_cpc_results	exchange-3_cpm_results	exchange-4_cpc_results
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.137335	0.148648	0.665434	0.352791	0.308352
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.133015	0.146195	0.717558	0.326331	0.355776
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.138823	0.145772	0.670918	0.354371	0.304265
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.140792	0.149400	0.710799	0.329571	0.316078
$CFOF_{agg}$(*v-rang*) 10-20 2 arbres	0.096487	0.151078	0.665444	0.362804	0.417764
$CFOF_{agg}$(*v-rang*) 10-20 4 arbres	0.093616	0.121346	0.694422	0.230541	0.291462
$CFOF_{agg}$(*v-rang*) 30 2 arbres	0.097737	0.149984	0.665057	0.362894	0.422145
$CFOF_{agg}$(*v-rang*) 30 4 arbres	0.093198	0.122409	0.696300	0.228957	0.295660
$CFOF_{approx}$ 10-20	0.128459	0.132874	0.660804	0.340812	0.324604
$CFOF_{approx}$ 30	0.098610	0.140625	0.640620	0.346199	0.323873
$CFOF_{exact}$	0.105680	0.140960	0.664506	0.344439	0.354194
baseline	0.118030	0.117306	0.116972	0.116972	0.118110
bayesChangePt	0.102288	0.155216	0.271816	0.087334	0.150365
contextOSE	0.081903	0.095416	0.125348	0.079633	0.147854
earthgeckoSkyline	0.559015	0.558653	0.567143	0.161372	0.264400
expose	0.121201	0.109095	0.396301	0.105234	0.183406
htmjava	0.084124	0.161139	0.334644	0.122112	0.220131
knncad	0.111061	0.160604	0.192921	0.081843	0.138102
null	0.559015	0.558653	0.558486	0.558486	0.559055
numenta	0.118248	0.133331	0.303919	0.128338	0.257619
numentaTM	0.079811	0.130686	0.344409	0.126710	0.215877
random	0.127716	0.126135	0.120367	0.104818	0.128131
randomCutForest	0.113429	0.131253	0.223357	0.121537	0.196059
relativeEntropy	0.155335	0.058653	0.367143	0.397591	0.442072
skyline	0.185163	0.102472	0.337633	0.101220	0.267165
twitterADVec	0.559015	0.561378	0.567143	0.561372	0.397733
windowedGaussian	0.106111	0.125619	0.225761	0.102309	0.172349

	exchange-4_cpm_results
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.298205
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.314404
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.296036
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.284557
$CFOF_{agg}$(*v-rang*) 10-20 2 arbres	0.395072
$CFOF_{agg}$(*v-rang*) 10-20 4 arbres	0.276208
$CFOF_{agg}$(*v-rang*) 30 2 arbres	0.397403
$CFOF_{agg}$(*v-rang*) 30 4 arbres	0.290750
$CFOF_{approx}$ 10-20	0.384603
$CFOF_{approx}$ 30	0.393038
$CFOF_{exact}$	0.384724
baseline	0.117394
bayesChangePt	0.134365
contextOSE	0.126620
earthgeckoSkyline	0.441770
expose	0.150484
htmjava	0.290257
knncad	0.149605
null	0.558697
numenta	0.299836
numentaTM	0.290235
random	0.109703
randomCutForest	0.141319
relativeEntropy	0.397412
skyline	0.235851
twitterADVec	0.397412
windowedGaussian	0.128896

A.2.4 Les séries "realKnownCause"

	ambient_temperature_system_failure	cpu_utilization_asg_misconfiguration	ec2_request_latency_system_failure	machine_temperature_system_failure	nyc_taxi
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.254284	0.649375	0.440276	0.443957	0.483810
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.239477	0.668667	0.349878	0.511327	0.438926
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.256243	0.649661	0.438460	0.444103	0.498338
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.237002	0.673902	0.455918	0.509497	0.442703
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.335573	0.631573	0.426237	0.426874	0.404615
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.307369	0.718775	0.335907	0.555739	0.422939
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.336743	0.633077	0.427465	0.427275	0.412494
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.308095	0.722921	0.331506	0.554695	0.423856
$CFOF_{approx}$ 10-20	0.460331	0.720095	0.370062	0.621967	0.461861
$CFOF_{approx}$ 30	0.463943	0.720561	0.369513	0.627177	0.467322
$CFOF_{exact}$	0.465793	0.717738	0.375702	0.621991	0.454176
baseline	0.111401	0.086647	0.100933	0.103349	0.108150
bayesChangePt	0.113813	0.079880	0.107244	0.164455	0.109979
contextOSE	0.090871	0.218806	0.088872	0.221404	0.083157
earthgeckoSkyline	0.557536	0.180601	0.553065	0.352861	0.554075
expose	0.177012	0.177194	0.138497	0.183504	0.110360
htmjava	0.219217	0.504669	0.240795	0.190843	0.271673
knncad	0.116626	0.089716	0.144224	0.100420	0.088951
null	0.555700	0.543324	0.550467	0.551675	0.554075
numenta	0.224111	0.362097	0.226192	0.242588	0.249908
numentaTM	0.257066	0.461248	0.223714	0.173063	0.157195
random	0.110849	0.087866	0.096815	0.103305	0.104521
randomCutForest	0.293898	0.345647	0.140282	0.594486	0.150336
relativeEntropy	0.157536	0.399921	0.556963	0.415233	0.494591
skyline	0.319899	0.175716	0.290830	0.562759	0.260077
twitterADVec	0.555700	0.422673	0.560861	0.560570	0.554075
windowedGaussian	0.298467	0.186913	0.141039	0.511542	0.139505

	rogue_agent_key_hold	rogue_agent_key_updown
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.136133	0.123593
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.089270	0.110812
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.133634	0.125029
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.091570	0.110217
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.118164	0.111254
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.108905	0.107395
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.119964	0.112570
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.111319	0.106372
$CFOF_{approx}$ 10-20	0.144050	0.129482
$CFOF_{approx}$ 30	0.143418	0.127260
$CFOF_{exact}$	0.140991	0.127809
baseline	0.118750	0.116101
bayesChangePt	0.122582	0.131563
contextOSE	0.086998	0.094761
earthgeckoSkyline	0.059375	0.058050
expose	0.107421	0.096833
htmjava	0.198574	0.115592
knncad	0.172721	0.114074
null	0.559375	0.558050
numenta	0.135018	0.117835
numentaTM	0.173937	0.131102
random	0.116752	0.130374
randomCutForest	0.128941	0.147889
relativeEntropy	0.117250	0.058050
skyline	0.150984	0.130677
twitterADVec	0.059375	0.114440
windowedGaussian	0.130124	0.078654

A.2.5 Les séries "realTraffic"

	TravelTime_387	TravelTime_451	occupancy_6005	occupancy_t4013	speed_6005
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.181993	0.439058	0.295596	0.564697	0.518597
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.166964	0.477781	0.278380	0.555367	0.495934
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.195550	0.453929	0.299233	0.568262	0.520498
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.157436	0.489705	0.279305	0.559689	0.491210
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.151117	0.458555	0.327710	0.559238	0.540232
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.198466	0.474319	0.286837	0.537890	0.479453
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.150222	0.465182	0.330032	0.559926	0.543032
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.192082	0.468852	0.288472	0.535131	0.472930
$CFOF_{approx}$ 10-20	0.196621	0.413489	0.254314	0.562085	0.401954
$CFOF_{approx}$ 30	0.198823	0.428894	0.254786	0.565992	0.400359
$CFOF_{exact}$	0.187860	0.370529	0.262825	0.557408	0.412135
baseline	0.117176	0.118063	0.118141	0.117647	0.112471
bayesChangePt	0.119822	0.124143	0.116553	0.186647	0.136368
contextOSE	0.081758	0.069758	0.088175	0.121011	0.125064
earthgeckoSkyline	0.228800	0.561064	0.227582	0.562353	0.558092
expose	0.147832	0.135126	0.124295	0.215869	0.142828
htmjava	0.206033	0.158888	0.158338	0.346179	0.279407
knncad	0.119488	0.168170	0.092731	0.110102	0.100561
null	0.558588	0.559032	0.559071	0.558824	0.556235
numenta	0.185578	0.258037	0.147770	0.286345	0.244211
numentaTM	0.189499	0.162524	0.220258	0.467649	0.247462
random	0.121813	0.116317	0.128246	0.114870	0.117732
randomCutForest	0.192723	0.155287	0.141754	0.213091	0.144530
relativeEntropy	0.313906	0.311064	0.099377	0.472941	0.436806
skyline	0.222699	0.185756	0.129689	0.229904	0.200630
twitterADVec	0.160361	0.559032	0.310916	0.567647	0.561806
windowedGaussian	0.176320	0.156802	0.147317	0.224622	0.158035

	speed_7578	speed_t4013
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.603189	0.686040
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.439213	0.631011
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.596602	0.687751
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.485664	0.631697
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.580386	0.694736
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.392948	0.590779
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.583531	0.693494
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.398524	0.589051
$CFOF_{approx}$ 10-20	0.617041	0.679733
$CFOF_{approx}$ 30	0.608519	0.682342
$CFOF_{exact}$	0.606493	0.685911
baseline	0.121086	0.117869
bayesChangePt	0.315385	0.109582
contextOSE	0.100913	0.102412
earthgeckoSkyline	0.371908	0.562463
expose	0.302871	0.186916
htmjava	0.338824	0.448503
knncad	0.222793	0.182703
null	0.560543	0.558934
numenta	0.341633	0.461136
numentaTM	0.366633	0.526741
random	0.149108	0.115559
randomCutForest	0.359609	0.218036
relativeEntropy	0.496152	0.564227
skyline	0.473437	0.432062
twitterADVec	0.568120	0.565992
windowedGaussian	0.384071	0.235225

A.2.6 Les séries “realTweets”

	Twitter_volume_AAPL	Twitter_volume_AMZN	Twitter_volume_CRM	Twitter_volume_CVS	Twitter_volume_FB
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.485403	0.366192	0.574706	0.516585	0.378090
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.583503	0.441196	0.532410	0.588519	0.394378
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.479382	0.365283	0.578523	0.514105	0.381966
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.569249	0.436690	0.528274	0.585031	0.392239
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.456402	0.346877	0.556712	0.489413	0.362090
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.449928	0.373738	0.513126	0.452301	0.281791
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.454273	0.346336	0.556284	0.484667	0.363923
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.450575	0.366238	0.511958	0.450272	0.282148
$CFOF_{approx}$ 10-20	0.485220	0.410848	0.551039	0.538172	0.451620
$CFOF_{approx}$ 30	0.489550	0.411669	0.551946	0.540061	0.452736
$CFOF_{exact}$	0.444512	0.412814	0.552446	0.530714	0.455104
baseline	0.104805	0.104768	0.105135	0.101040	0.104886
bayesChangePt	0.107321	0.104197	0.090982	0.100763	0.103526
contextOSE	0.117073	0.068701	0.161079	0.122819	0.081587
earthgeckoSkyline	0.241593	0.192689	0.187024	0.218365	0.124438
expose	0.187244	0.133302	0.175968	0.135748	0.102860
htmjava	0.090801	0.188389	0.283548	0.168179	0.085867
knncad	0.148949	0.125984	0.134115	0.108876	0.130706
null	0.552402	0.552384	0.552567	0.550520	0.552443
numenta	0.094828	0.192364	0.308541	0.199472	0.127304
numentaTM	0.091773	0.198246	0.260793	0.137229	0.127100
random	0.106317	0.105758	0.108877	0.103790	0.101182
randomCutForest	0.272944	0.145116	0.230941	0.121623	0.108254
relativeEntropy	0.303530	0.554367	0.453691	0.441470	0.303292
skyline	0.304574	0.167020	0.276400	0.131885	0.133570
twitterADVec	0.298566	0.238833	0.413585	0.395415	0.167494
windowedGaussian	0.324916	0.143719	0.205095	0.123373	0.117756

	Twitter_volume_GOOG	Twitter_volume_IBM	Twitter_volume_KO	Twitter_volume_PFE	Twitter_volume_UPS
$CFOF_{agg}(CFOF_{approx})$ 10-20 2 arbres	0.422619	0.256912	0.413596	0.359665	0.407427
$CFOF_{agg}(CFOF_{approx})$ 10-20 4 arbres	0.428465	0.271130	0.425388	0.430383	0.417559
$CFOF_{agg}(CFOF_{approx})$ 30 2 arbres	0.421672	0.254022	0.414954	0.362541	0.402201
$CFOF_{agg}(CFOF_{approx})$ 30 4 arbres	0.432868	0.271701	0.427461	0.427742	0.414842
$CFOF_{agg}(v\text{-}rang)$ 10-20 2 arbres	0.434563	0.280747	0.391778	0.379318	0.411820
$CFOF_{agg}(v\text{-}rang)$ 10-20 4 arbres	0.415670	0.221513	0.329295	0.428035	0.452981
$CFOF_{agg}(v\text{-}rang)$ 30 2 arbres	0.433671	0.281873	0.393731	0.379857	0.408112
$CFOF_{agg}(v\text{-}rang)$ 30 4 arbres	0.417379	0.221687	0.331106	0.427488	0.451097
$CFOF_{approx}$ 10-20	0.447540	0.237572	0.488468	0.370449	0.389141
$CFOF_{approx}$ 30	0.449540	0.235013	0.489785	0.371714	0.387390
$CFOF_{exact}$	0.440001	0.246427	0.487044	0.356110	0.359957
baseline	0.094885	0.104999	0.105092	0.105110	0.104856
bayesChangePt	0.109880	0.099826	0.104434	0.108063	0.108500
contextOSE	0.081447	0.153027	0.093125	0.070252	0.113688
earthgeckoSkyline	0.173706	0.120292	0.137007	0.207528	0.147872
expose	0.123275	0.122152	0.122102	0.150462	0.146007
htmjava	0.143329	0.192288	0.097354	0.148490	0.217909
knncad	0.111383	0.120191	0.126336	0.169421	0.134256
null	0.547442	0.552500	0.552546	0.552555	0.552428
numenta	0.193322	0.216132	0.110918	0.170240	0.209091
numentaTM	0.144467	0.204889	0.108927	0.143259	0.213657
random	0.090408	0.105231	0.106027	0.102051	0.104283
randomCutForest	0.176946	0.107329	0.118826	0.153203	0.222565
relativeEntropy	0.348390	0.275848	0.303110	0.303682	0.389715
skyline	0.182385	0.133655	0.151578	0.164515	0.208110
twitterADVec	0.461802	0.306721	0.255930	0.289833	0.183719
windowedGaussian	0.182480	0.107316	0.131085	0.131572	0.196393

Annexe B

Coefficients de corrélation de Spearman pour $\text{CFOF}_{\text{approx}}$ et pour CFOF_{agg} vis-à-vis de $\text{CFOF}_{\text{exact}}$ sur les séries du benchmark NAB

Nous présentons les scores de corrélation de Spearman entre les scores CFOF exacts et les différents scores CFOF approximés, utilisés dans les chapitres 3 et 4, pour chacun des jeux de données du *Numenta Anomaly Benchmark* (NAB) proposé par [Lavin et Ahmad, 2015], et disponible sur GitHub [1].

Pour chacun des tableaux ci-dessous, chacune des colonnes représente les scores de corrélation selon deux paramètres. Le paramètre "seuil" est introduit au chapitre 3 et représente la taille de stockage maximale d'un nœud feuille. Le second paramètre "arbre" représente le nombre d'arbres utilisés pour stocker les séquences de l'historique de référence, comme présenté dans le chapitre 4.

B.1 Avec la méthode $\text{CFOF}_{\text{approx}}$ et CFOF_{agg}(*v-rang*)

Remarque : la colonne $\text{CFOF}_{\text{approx}}$ présente dans les différents tableaux de cette section correspond à la méthode avec un seul arbre proposée dans

1. https://github.com/numenta/NAB

le chapitre 3.

B.1.1 Les séries "artificialNoAnomaly"

	seuil 10-20 $CFOF_{approx}$	seuil 30 $CFOF_{approx}$	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
art_daily_no_noise	0.973733	0.954703	0.706562	0.51006	0.7007	0.503701
art_daily_perfect_square_wave	0.824782	0.820708	0.619804	0.592407	0.617546	0.588667
art_daily_small_noise	0.9803	0.961311	0.725935	0.586266	0.723599	0.584885
art_noisy	0.99307	0.987016	0.932879	0.807117	0.923983	0.806118

B.1.2 Les séries "artificialWithAnomaly"

	seuil 10-20 $CFOF_{approx}$	seuil 30 $CFOF_{approx}$	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
art_daily_flatmiddle	0.983341	0.971619	0.751793	0.664436	0.752303	0.664816
art_daily_jumpsdown	0.963666	0.943794	0.752534	0.609382	0.751562	0.616883
art_daily_jumpsup	0.976894	0.960383	0.817461	0.69656	0.815184	0.697929
art_daily_nojump	0.981151	0.965331	0.771177	0.645288	0.764737	0.645631
art_increase_spike_density	0.447873	0.448061	0.490925	0.489087	0.490837	0.488959
art_load_balancer_spikes	0.993225	0.988208	0.927752	0.813859	0.927208	0.813417

B.1.3 Les séries "realAWSCloudwatch"

	seuil 10-20 $CFOF_{approx}$	seuil 30 $CFOF_{approx}$	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
ec2_cpu_utilization_24ae8d	0.978183	0.973569	0.891338	0.830376	0.896618	0.829746
ec2_cpu_utilization_53ea38	0.997325	0.995309	0.983989	0.934197	0.980262	0.932008
ec2_cpu_utilization_5f5533	0.992969	0.957481	0.854395	0.712678	0.884597	0.715921
ec2_cpu_utilization_77c1ca	0.984554	0.981736	0.901418	0.60844	0.903779	0.610165
ec2_cpu_utilization_825cc2	0.98758	0.965577	0.865055	0.819976	0.858775	0.820429
ec2_cpu_utilization_ac20cd	0.968028	0.923314	0.729245	0.655311	0.735436	0.649347
ec2_cpu_utilization_c6585a	0.990018	0.985119	0.939446	0.826257	0.942836	0.826226
ec2_cpu_utilization_fe7f93	0.990372	0.986139	0.952892	0.773429	0.951061	0.774836
ec2_disk_write_bytes_1ef3de	0.992809	0.991478	0.952864	0.812897	0.951925	0.81368
ec2_disk_write_bytes_c0d644	0.971843	0.969676	0.944829	0.738288	0.943688	0.73734
ec2_network_in_257a54	0.510738	0.51338	0.433014	0.440414	0.438334	0.441582
ec2_network_in_5abac7	0.983606	0.980331	0.897882	0.611495	0.900182	0.613093
elb_request_count_8c0756	0.994143	0.993205	0.972339	0.923519	0.972732	0.92259
grok_asg_anomaly	0.989474	0.980756	0.791212	0.734198	0.795623	0.732677
iio_us-east-1_i-a2eb1cd9_Net...	0.933339	0.857202	0.385395	0.361676	0.374621	0.363405
rds_cpu_utilization_cc0c53	0.975526	0.953733	0.909227	0.877541	0.912129	0.875892
rds_cpu_utilization_e47b3b	0.989189	0.982708	0.745001	0.748301	0.742098	0.744368

B.1.4 Les séries "realAdExchange"

	seuil 10-20 $CFOF_{approx}$	seuil 30 $CFOF_{approx}$	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
exchange-2_cpc_results	0.970638	0.953138	0.745494	0.827692	0.740203	0.822403
exchange-2_cpm_results	0.979537	0.951566	0.820147	0.832227	0.822448	0.829913
exchange-3_cpc_results	0.924742	0.88365	0.805378	0.696984	0.780135	0.697841
exchange-3_cpm_results	0.969836	0.893042	0.80717	0.781975	0.793541	0.780285
exchange-4_cpc_results	0.953683	0.943459	0.782803	0.638583	0.78358	0.640621
exchange-4_cpm_results	0.972331	0.968149	0.878309	0.796054	0.872425	0.795809

B.1.5 Les séries "realKnownCause"

	seuil 10-20 $\text{CFOF}_{\text{approx}}$	seuil 30 $\text{CFOF}_{\text{approx}}$	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
ambient_temperature_system...	0.995642	0.979839	0.751056	0.711462	0.752858	0.710174
cpu_utilization_asg_misconfig...	0.997046	0.995851	0.96095	0.868536	0.959447	0.868806
ec2_request_latency_system...	0.993242	0.993146	0.914342	0.844008	0.911497	0.848767
machine_temperature_system...	0.997088	0.951885	0.759989	0.752944	0.761061	0.751319
nyc_taxi	0.942902	0.914378	0.774594	0.679218	0.768309	0.678557
rogue_agent_key_hold	0.980018	0.971176	0.932234	0.798675	0.937984	0.807093
rogue_agent_key_updown	0.883516	0.868356	0.757737	0.669666	0.750571	0.672656

B.1.6 Les séries "realTraffic"

	seuil 10-20 $\text{CFOF}_{\text{approx}}$	seuil 30 $\text{CFOF}_{\text{approx}}$	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
TravelTime_387	0.988404	0.984838	0.945215	0.889118	0.94676	0.893497
TravelTime_451	0.97925	0.966774	0.926939	0.872681	0.932671	0.862664
occupancy_6005	0.987046	0.978288	0.866878	0.886173	0.867441	0.88473
occupancy_t4013	0.973818	0.967071	0.826843	0.83304	0.830084	0.832882
speed_6005	0.995438	0.986713	0.93503	0.949599	0.932921	0.948981
speed_7578	0.97217	0.947334	0.950267	0.906218	0.950666	0.902437
speed_t4013	0.99443	0.993667	0.931134	0.938248	0.933461	0.939671

B.1.7 Les séries "realTweets"

	seuil 10-20 $\text{CFOF}_{\text{approx}}$	seuil 30 $\text{CFOF}_{\text{approx}}$	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
Twitter_volume_AAPL	0.971204	0.922671	0.944728	0.873942	0.944725	0.873543
Twitter_volume_AMZN	0.996187	0.992379	0.892283	0.871835	0.890651	0.874854
Twitter_volume_CRM	0.989624	0.987239	0.952831	0.910568	0.953868	0.907969
Twitter_volume_CVS	0.991233	0.990619	0.982161	0.943109	0.983023	0.943169
Twitter_volume_FB	0.996045	0.995006	0.947712	0.93349	0.947351	0.933966
Twitter_volume_GOOG	0.995023	0.994104	0.934026	0.905072	0.931171	0.901809
Twitter_volume_IBM	0.995961	0.995379	0.964336	0.929559	0.964251	0.929123
Twitter_volume_KO	0.995969	0.994673	0.959795	0.900525	0.958635	0.900574
Twitter_volume_PFE	0.992345	0.992304	0.983263	0.93583	0.985092	0.938032
Twitter_volume_UPS	0.997331	0.995441	0.916664	0.686641	0.914151	0.685921

B.2 Avec la méthode $\text{CFOF}_{\text{agg}}(\text{CFOF}_{\text{approx}})$

B.2.1 Les séries "artificialNoAnomaly"

	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
art_daily_no_noise	0.736863	0.469196	0.7289	0.469922
art_daily_perfect_square_wave	0.585325	0.438935	0.57688	0.444263
art_daily_small_noise	0.734524	0.503631	0.722712	0.51516
art_noisy	0.895686	0.7548	0.888055	0.755626

B.2.2 Les séries "artificialWithAnomaly"

	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
art_daily_flatmiddle	0.75381	0.537212	0.752118	0.556496
art_daily_jumpsdown	0.776858	0.551777	0.774221	0.571279
art_daily_jumpsup	0.820839	0.650411	0.795777	0.656466
art_daily_nojump	0.782658	0.579536	0.773976	0.592087
art_increase_spike_density	0.419459	0.417525	0.419363	0.417476
art_load_balancer_spikes	0.946929	0.735279	0.941899	0.738856

B.2.3 Les séries "realAWSCloudwatch"

	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
ec2_cpu_utilization_24ae8d	0.923005	0.780314	0.920242	0.777082
ec2_cpu_utilization_53ea38	0.952857	0.876436	0.948514	0.879394
ec2_cpu_utilization_5f5533	0.907943	0.798591	0.860787	0.800712
ec2_cpu_utilization_77c1ca	0.929512	0.470225	0.931564	0.479493
ec2_cpu_utilization_825cc2	0.905395	0.843434	0.889391	0.837697
ec2_cpu_utilization_ac20cd	0.850038	0.736029	0.852307	0.71131
ec2_cpu_utilization_c6585a	0.956378	0.736757	0.953586	0.73657
ec2_cpu_utilization_fe7f93	0.966017	0.76941	0.961372	0.770815
ec2_disk_write_bytes_1ef3de	0.949551	0.697388	0.951126	0.698693
ec2_disk_write_bytes_c0d644	0.937149	0.595152	0.934267	0.597137
ec2_network_in_257a54	0.411218	0.409888	0.410912	0.412128
ec2_network_in_5abac7	0.919336	0.413876	0.922002	0.413557
elb_request_count_8c0756	0.936815	0.801062	0.933057	0.800015
grok_asg_anomaly	0.868124	0.799167	0.861562	0.799105
iio_us-east-1_i-a2eb1cd9_NetworkIn	0.708918	0.639665	0.679182	0.653889
rds_cpu_utilization_cc0c53	0.905917	0.801429	0.913241	0.802059
rds_cpu_utilization_e47b3b	0.84693	0.791466	0.841805	0.787036

B.2.4 Les séries "realAdExchange"

	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
exchange-2_cpc_results	0.811874	0.71511	0.818199	0.683095
exchange-2_cpm_results	0.838735	0.691788	0.84644	0.696921
exchange-3_cpc_results	0.805877	0.731685	0.740084	0.715182
exchange-3_cpm_results	0.858331	0.781642	0.81792	0.768248
exchange-4_cpc_results	0.800184	0.738224	0.784591	0.729054
exchange-4_cpm_results	0.874222	0.824841	0.871198	0.817302

B.2.5 Les séries "realKnownCause"

	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
ambient_temperature_system_failure	0.781887	0.704138	0.783189	0.703095
cpu_utilization_asg_misconfiguration	0.940324	0.829972	0.938522	0.828993
ec2_request_latency_system_failure	0.934616	0.849198	0.929257	0.856476
machine_temperature_system_failure	0.886387	0.79554	0.885396	0.791234
nyc_taxi	0.828059	0.67255	0.813869	0.678113
rogue_agent_key_hold	0.926537	0.595622	0.921123	0.605632
rogue_agent_key_updown	0.827165	0.495038	0.824249	0.496051

B.2.6 Les séries "realTraffic"

	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
TravelTime_387	0.909542	0.813375	0.908559	0.8263
TravelTime_451	0.840295	0.747716	0.830978	0.733183
occupancy_6005	0.890487	0.824629	0.885877	0.820275
occupancy_t4013	0.891973	0.818602	0.888014	0.808833
speed_6005	0.942912	0.876754	0.942071	0.86932
speed_7578	0.865783	0.852926	0.866718	0.835827
speed_t4013	0.944832	0.874912	0.942468	0.870005

B.2.7 Les séries "realTweets"

	seuil 10-20 2 arbres	seuil 10-20 4 arbres	seuil 30 2 arbres	seuil 30 4 arbres
Twitter_volume_AAPL	0.902852	0.779653	0.904079	0.782839
Twitter_volume_AMZN	0.919955	0.830259	0.923646	0.838042
Twitter_volume_CRM	0.920215	0.73828	0.918583	0.748684
Twitter_volume_CVS	0.967847	0.886107	0.969522	0.886358
Twitter_volume_FB	0.918784	0.772381	0.919714	0.786292
Twitter_volume_GOOG	0.881353	0.767595	0.878416	0.777741
Twitter_volume_IBM	0.954927	0.881323	0.954229	0.882978
Twitter_volume_KO	0.90906	0.838998	0.910489	0.839702
Twitter_volume_PFE	0.973054	0.866259	0.974076	0.865177
Twitter_volume_UPS	0.925959	0.751575	0.925397	0.74888

Annexe C

Résultats de détection sur les données SNCF

Nous donnons ici certains détails concernant les scores de détection obtenus, comme présenté dans le chapitre 5, pour les séquences du mois de Septembre. Chaque figure décrit les résultats CFOF approximés pour des valeurs de ϱ différentes.

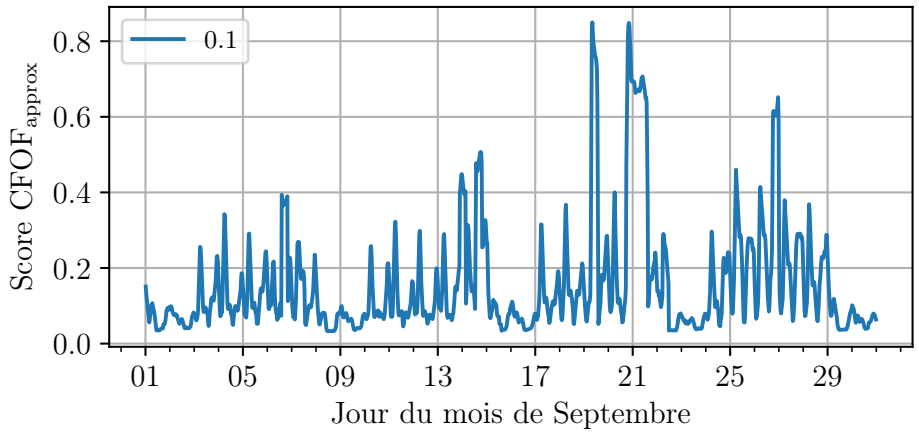


FIGURE C.1 – Résultats de détection par CFOF$_{\text{approx}}$ avec $\varrho = 0.1$ basée sur l'indicateur du nombre de messages reçus par minute durant le mois de Septembre 2018 sur le point de collecte se situant dans CanalTrain.

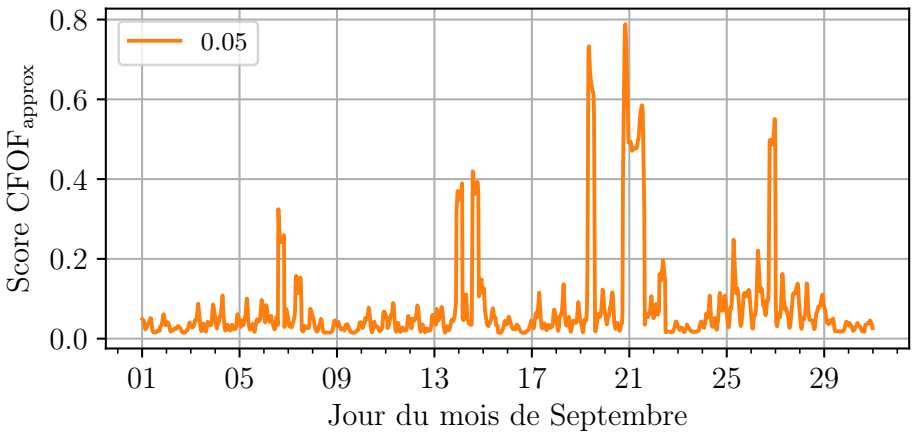


FIGURE C.2 – Résultats de détection par CFOF$_{\text{approx}}$ avec $\varrho = 0.05$ basée sur l'indicateur du nombre de messages reçus par minute durant le mois de Septembre 2018 sur le point de collecte se situant dans CanalTrain.

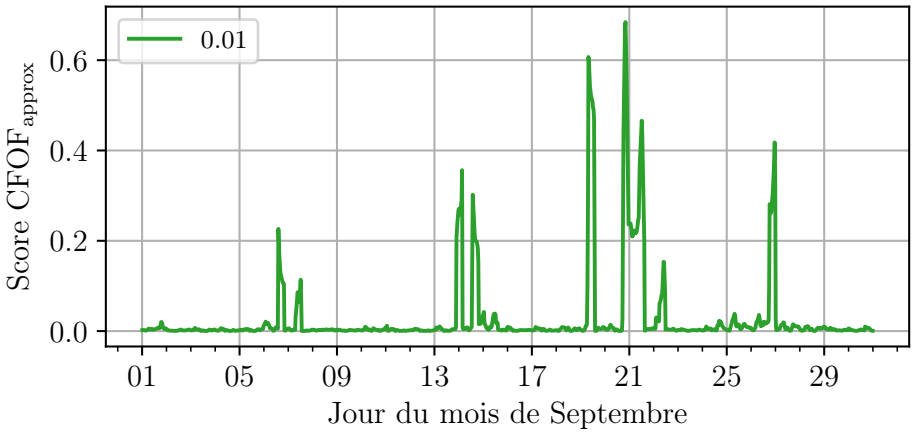


FIGURE C.3 – Résultats de détection par CFOF$_{\text{approx}}$ avec $\varrho = 0.01$ basée sur l'indicateur du nombre de messages reçus par minute durant le mois de Septembre 2018 sur le point de collecte se situant dans CanalTrain.

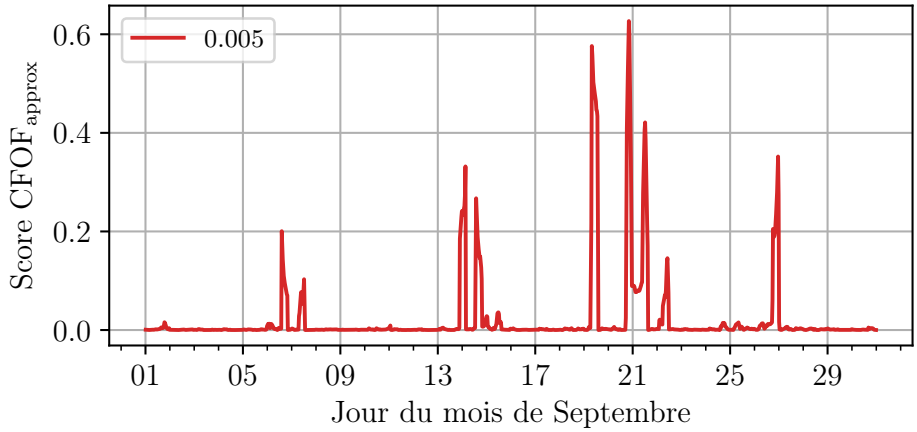


FIGURE C.4 – Résultats de détection par CFOF$_{\text{agg}}()$ avec $\varrho = 0.005$ basée sur l'indicateur du nombre de messages reçus par minute durant le mois de Septembre 2018 sur le point de collecte se situant dans CanalTrain.

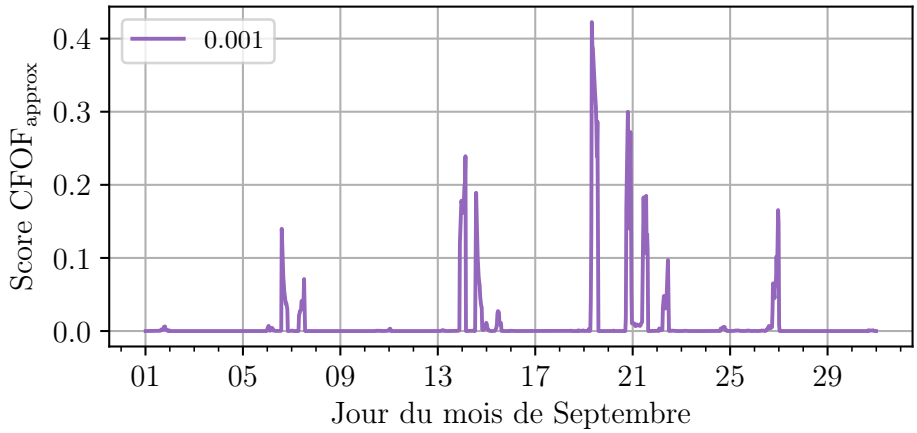


FIGURE C.5 – Résultats de détection par CFOF$_{\text{approx}}$ avec $\varrho = 0.001$ basée sur l'indicateur du nombre de messages reçus par minute durant le mois de Septembre 2018 sur le point de collecte se situant dans CanalTrain.

Bibliographie

[Aggarwal, 2007] Aggarwal, C. C. (2007). *Data Streams : Models and Algorithms*, volume 31. Springer US. 354 p.

[Aggarwal, 2017] Aggarwal, C. C. (2017). *Outlier Analysis*. Springer International Publishing. 466 p.

[Aggarwal *et al.*, 2001] Aggarwal, C. C., Hinneburg, A. et Keim, D. A. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. *In International Conference on Database Theory*, pages 420–434. Springer, Springer Berlin Heidelberg.

[Agrawal *et al.*, 1998] Agrawal, R., Gehrke, J., Gunopulos, D. et Raghavan, P. (1998). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 27, pages 94–105, New York, NY, USA. Association for Computing Machinery.

[Ahmad *et al.*, 2017] Ahmad, S., Lavin, A., Purdy, S. et Agha, Z. (2017). Unsupervised Real-Time Anomaly Detection for Streaming Data. *Neurocomputing*, 262:134–147.

[Ahmed *et al.*, 2016] Ahmed, M., Mahmood, A. N. et Hu, J. (2016). A Survey of Network Anomaly Detection Techniques. *Journal of Network and Computer Applications*, 60:19–31.

[Allan *et al.*, 1998] Allan, J., Carbonell, J. G., Doddington, G., Yamron, J. et Yang, Y. (1998). Topic Detection and Tracking Pilot Study Final Report. *In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218. Morgan Kaufmann.

[Ambwani, 2003] Ambwani, T. (2003). Multi Class Support Vector Machine Implementation to Intrusion Detection. *In Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 2300–2305. IEEE.

[Andrews, 1972] Andrews, D. F. (1972). Plots of High-Dimensional Data. *Biometrics*, 28(1):125–136.

[Angiulli, 2017] Angiulli, F. (2017). Concentration Free Outlier Detection. *In Machine Learning and Knowledge Discovery in Databases*, pages 3–19. Springer International Publishing.

[Angiulli, 2018] Angiulli, F. (2018). On the behavior of intrinsically high-dimensional spaces : Distances, direct and reverse nearest neighbors, and hubness. *Journal of Machine Learning Research*, 18(1):6209–6268.

[Angiulli, 2020] Angiulli, F. (2020). CFOF : A Concentration Free Measure for Anomaly Detection. *ACM Transactions on Knowledge Discovery from Data*, 14(1).

[Angiulli et Pizzuti, 2002] Angiulli, F. et Pizzuti, C. (2002). Fast Outlier Detection in High Dimensional Spaces. *In Principles of Data Mining and Knowledge Discovery*, pages 15–27. Springer.

[Augusteijn et Folkert, 2002] Augusteijn, M. F. et Folkert, B. A. (2002). Neural Network Classification and Novelty Detection. *International Journal of Remote Sensing*, 23(14):2891–2902.

[Basharat *et al.*, 2008] Basharat, A., Gritai, A. et Shah, M. (2008). Learning Object Motion Patterns for Anomaly Detection and Improved Object Detection. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[Baydogan et Runger, 2015] Baydogan, M. G. et Runger, G. (2015). Learning a Symbolic Representation for Multivariate Time Series Classification. *Data Mining and Knowledge Discovery*, 29(2):400–422.

[Bellmann, 1961] Bellmann, R. E. (1961). *Adaptive Control Processes : A Guided Tour*. Princeton University Press, New Jersey, USA. 255 p.

[Bergroth *et al.*, 2000] Bergroth, L., Hakonen, H. et Raita, T. (2000). A Survey of Longest Common Subsequence Algorithms. *In Proceedings Seventh International Symposium on String Processing and Information Retrieval*, pages 39–48. IEEE.

[Berndt et Clifford, 1994] Berndt, D. J. et Clifford, J. (1994). Using Dynamic Time Warping to Find Patterns in Time Series. *In Proceedings of AAAI Workshop on Knowledge Discovery in Databases*, pages 359–370.

[Beyer *et al.*, 1999] Beyer, K., Goldstein, J., Ramakrishnan, R. et Shaft, U. (1999). When Is "Nearest Neighbor" Meaningful ? *In International Conference on Database Theory*, pages 217–235. Springer Berlin Heidelberg.

[Bianco *et al.*, 2001] Bianco, A. M., Garcia Ben, M., Martinez, E. J. et Yohai, V. J. (2001). Outlier Detection in Regression Models with ARIMA Errors using Robust Estimates. *Journal of Forecasting*, 20(8):565–579.

[Böhm *et al.*, 2010] Böhm, C., Plant, C., Shao, J. et Yang, Q. (2010). Clustering by Synchronization. *In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 583–592. Association for Computing Machinery.

[Breslow et Aha, 1997] Breslow, L. A. et Aha, D. W. (1997). Simplifying Decision Trees : A Survey. *The Knowledge Engineering Review*, 12(1):1–40.

[Breunig *et al.*, 1999] Breunig, M. M., Kriegel, H.-P., Ng, R. T. et Sander, J. (1999). OPTICS-OF : Identifying Local Outliers. *In Principles of Data Mining and Knowledge Discovery*, pages 262–270. Springer, Springer Berlin Heidelberg.

[Breunig *et al.*, 2000] Breunig, M. M., Kriegel, H.-P., Ng, R. T. et Sander, J. (2000). LOF : Identifying Density-Based Local Outliers. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, page 93–104. Association for Computing Machinery.

[Budalakoti *et al.*, 2009] Budalakoti, S., Srivastava, A. N. et Otey, M. E. (2009). Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):101–113.

[Camerra *et al.*, 2010] Camerra, A., Palpanas, T., Shieh, J. et Keogh, E. (2010). *i*SAX 2.0 : Indexing and Mining One Billion Time Series. *In IEEE International Conference on Data Mining*, pages 58–67.

[Camerra *et al.*, 2014] Camerra, A., Shieh, J., Palpanas, T., Rakthanmanon, T. et Keogh, E. (2014). Beyond One Billion Time Series : Indexing and Mining Very Large Time Series Collections With *i*SAX2+. *Knowledge and Information Systems*, 39(1):123–151.

[Campos *et al.*, 2016] Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., Assent, I. et Houle, M. E. (2016). On the Evaluation of Unsupervised Outlier Detection : Measures, Datasets, and an Empirical Study. *Data Mining and Knowledge Discovery*, 30(4):891–927.

[Castro et Azevedo, 2010] Castro, N. et Azevedo, P. (2010). Multiresolution Motif Discovery in Time Series. *In Proceedings of the SIAM International Conference on Data Mining*, pages 665–676. SIAM.

[Chakrabarty, 2018] Chakrabarty, N. (2018). A Deep Learning Method for the Detection of Diabetic Retinopathy. *In 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering*, pages 1–5. IEEE.

[Chandola *et al.*, 2009] Chandola, V., Banerjee, A. et Kumar, V. (2009). Outlier Detection : A Survey. *ACM Computing Surveys*, 41(3).

[Chen et Shah, 2018] Chen, G. H. et Shah, D. (2018). Explaining the Success of Nearest Neighbor Methods in Prediction. *Foundations and Trends in Machine Learning*, 10(5-6):337–588.

[Chomsky, 1986] Chomsky, N. (1986). *Knowledge of Language : Its Nature, Origin, and Use*. Greenwood Publishing Group. 307 p.

[Cohen, 1995] Cohen, W. W. (1995). Fast Effective Rule Induction. *In Machine learning Proceedings*, pages 115–123. Elsevier.

[Collins *et al.*, 2000] Collins, R. T., Lipton, A. J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P. et Wixson, L. (2000). A System for Video Surveillance and Monitoring. *VSAM final report*, pages 1–68.

[Cornuéjols *et al.*, 2018] Cornuéjols, A., Miclet, L. et Barra, V. (2018). *Apprentissage Artificiel : Deep Learning, Concepts et Algorithmes*. Eyrolles. 912 p.

[Dasgupta et Majumdar, 2002] Dasgupta, D. et Majumdar, N. S. (2002). Anomaly Detection in Multidimensional Data Using Negative Selection Algorithm. *In Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1039–1044. IEEE.

[Datar et Motwani, 2007] Datar, M. et Motwani, R. (2007). *The Sliding-Window Computation Model and Results*, pages 149–167. Springer US.

[Deza et Deza, 2009] Deza, M. M. et Deza, E. (2009). *Encyclopedia of Distances*. Springer. 650 p.

[Domingos, 1999] Domingos, P. (1999). MetaCost : A General Method for Making Classifiers Cost-Sensitive. *In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 99, pages 155–164. Association for Computing Machinery.

[Donoho, 2004] Donoho, S. (2004). Early Detection of Insider Trading in Option Markets. *In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 420–429. Association for Computing Machinery.

[Drucker *et al.*, 1997] Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J. et Vapnik, V. (1997). Support Vector Regression Machines. *In Advances in Neural Information Processing Systems*, pages 155–161. MIT Press.

[Eskin *et al.*, 2002] Eskin, E., Arnold, A., Prerau, M., Portnoy, L. et Stolfo, S. (2002). A Geometric Framework for Unsupervised Anomaly

Detection. *In Applications of Data Mining in Computer Security*, pages 77–101. Springer.

[Ester *et al.*, 1996] Ester, M., Kriegel, H.-P., Sander, J. et Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *In Proceedings of Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press.

[Forrest *et al.*, 1996] Forrest, S., Hofmeyr, S. A., Somayaji, A. et Longstaff, T. A. (1996). A Sense of Self for Unix Processes. *In Proceedings IEEE Symposium on Security and Privacy*, pages 120–128. IEEE.

[Forrest *et al.*, 1994] Forrest, S., Perelson, A. S., Allen, L. et Cherukuri, R. (1994). Self-nonself Discrimination in a Computer. *In Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy*, pages 202–212. IEEE.

[Foulon *et al.*, 2019a] Foulon, L., Fenet, S., Rigotti, C. et Jouvin, D. (2019a). Detecting Anomalies over Message Streams in Railway Communication Systems. Poster : 4th Workshop on Advanced Analytics and Learning on Temporal Data, ECML/PKDD.

[Foulon *et al.*, 2019b] Foulon, L., Fenet, S., Rigotti, C. et Jouvin, D. (2019b). Scoring Message Stream Anomalies in Railway Communication Systems. *In Workshop on Learning and Mining with Industrial Data (International Conference on Data Mining Workshops)*, pages 769–776. IEEE.

[Foulon *et al.*, 2019c] Foulon, L., Rigotti, C., Fenet, S. et Jouvin, D. (2019c). Approximation du Score CFOF de Détection d'Anomalie dans un Arbre d'Indexation iSAX : Application au Contexte SI de la SNCF. *In EGC 2019 - 19ème Conférence francophone sur l'Extraction et la Gestion des Connaissances*, pages 165–176.

[Foulon *et al.*, 2020] Foulon, L., Rigotti, C., Fenet, S. et Jouvin, D. (2020). Anomaly Detection Based on Sequence Indexation and CFOF Score Approximation. *In Book : Advances in Knowledge Discovery and Management, vol.9*, pages 1–14. Springer International Publishing.

[Francois *et al.*, 2007] Francois, D., Wertz, V. et Verleysen, M. (2007). The Concentration of Fractional Distances. *Transactions on Knowledge and Data Engineering*, 19(7):873–886.

[Freund *et al.*, 2009] Freund, D. E., Bressler, N. et Burlina, P. (2009). Automated Detection of Drusen in the Macula. *In IEEE International Symposium on Biomedical Imaging : From Nano to Macro*, pages 61–64. IEEE.

[Friesem et Vest, 1969] Friesem, A. A. et Vest, C. M. (1969). Detection of Micro Fractures by Holographic Interferometry. *Applied Optics*, 8(6):1253–1254.

[Fujimaki *et al.*, 2005] Fujimaki, R., Yairi, T. et Machida, K. (2005). An Approach to Spacecraft Anomaly Detection Problem Using Kernel Feature Space. *In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 401–410. Association for Computing Machinery.

[Garcia-Teodoro *et al.*, 2009] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G. et Vázquez, E. (2009). Anomaly-Based Network Intrusion Detection : Techniques, Systems and Challenges. *Computers & Security*, 28(1):18–28.

[Garofalakis *et al.*, 2016] Garofalakis, M., Gehrke, J. et Rastogi, R. (2016). *Data Stream Management : Processing High-Speed Data Streams*. Springer Berlin Heidelberg. 537 p.

[Gupta et Sekar, 2003] Gupta, A. et Sekar, R. (2003). An Approach for Detecting Self-Propagating Email Using Anomaly Detection. *In Recent Advances in Intrusion Detection*, pages 55–72. Springer.

[Gupta *et al.*, 2014] Gupta, M., Gao, J., Aggarwal, C. C. et Han, J. (2014). Outlier Detection for Temporal Data : A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267.

[Hamilton *et al.*, 2017] Hamilton, W., Ying, Z. et Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *In Advances in Neural Information Processing Systems*, pages 1024–1034. Curran Associates, Inc.

[Hammerla *et al.*, 2013] Hammerla, N. Y., Kirkham, R., Andras, P. et Ploetz, T. (2013). On Preserving Statistical Characteristics of Accelerometry Data Using Their Empirical Cumulative Distribution. *In Proceedings of the International Symposium on Wearable Computers*, pages 65–68. Association for Computing Machinery.

[Han *et al.*, 2012] Han, J., Kamber, M. et Pei, J. (2012). *Data mining : Concepts and Techniques*. Elsevier. 744 p.

[Hanley et McNeil, 1982] Hanley, J. A. et McNeil, B. J. (1982). The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143(1):29–36.

[Hariri *et al.*, 2019] Hariri, S., Kind, M. C. et Brunner, R. J. (2019). Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering*.

[Hautamaki *et al.*, 2004] Hautamaki, V., Karkkainen, I. et Franti, P. (2004). Outlier Detection Using k-Nearest Neighbour Graph. *In Proceedings of the 17th International Conference on Pattern Recognition*, volume 3, pages 430–433. IEEE.

[Hawkins, 1980] Hawkins, D. M. (1980). *Identification of Outliers*, volume 11. Springer. 188 p.

[Hinton et Roweis, 2003] Hinton, G. E. et Roweis, S. T. (2003). Stochastic Neighbor Embedding. *In Advances in Neural Information Processing Systems*, pages 857–864. MIT Press.

[Hoglund *et al.*, 2000] Hoglund, A. J., Hatonen, K. et Sorvari, A. S. (2000). A Computer Host-Based User Anomaly Detection System Using the Self-Organizing Map. *In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. Neural Computing : New Challenges and Perspectives for the New Millennium*, volume 5, pages 411–416. IEEE.

[Hoque *et al.*, 2012] Hoque, M. S., Mukit, A. et Bikas, A. N. (2012). An Implementation of Intrusion Detection System Using Genetic Algorithm. *International Journal of Network Security & Its Applications*, 4(2):109–120.

[Jiang *et al.*, 2014] Jiang, D., Xu, Z., Zhang, P. et Zhu, T. (2014). A Transform Domain-Based Anomaly Detection Approach to Network-Wide Traffic. *Journal of Network and Computer Applications*, 40:292–306.

[Kasiviswanathan *et al.*, 2011] Kasiviswanathan, S. P., Melville, P., Banerjee, A. et Sindhwani, V. (2011). Emerging Topic Detection Using Dictionary Learning. *In Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 745–754. Association for Computing Machinery.

[Keogh *et al.*, 2005] Keogh, E., Lin, J. et Fu, A. (2005). HOT SAX : Efficiently Finding the Most Unusual Time Series Subsequence. *In Fifth IEEE International Conference on Data Mining*, page 226–233. IEEE Computer Society.

[Kim *et al.*, 2015] Kim, S., Choi, Y. et Lee, M. (2015). Deep Learning with Support Vector Data Description. *Neurocomputing*, 165:111–117.

[Kohonen, 1982] Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43(1):59–69.

[Kong *et al.*, 2018] Kong, X., Song, X., Xia, F., Guo, H., Wang, J. et Tolba, A. (2018). LoTAD : Long-Term Traffic Anomaly Detection Based on Crowdsourced Bus Trajectory Data. *World Wide Web*, 21(3):825–847.

[Korn et Muthukrishnan, 2000] Korn, F. et Muthukrishnan, S. (2000). Influence Sets Based on Reverse Nearest Neighbor Queries. *ACM SIGMOD Record*, 29(2):201–212.

[Kramer, 1991] Kramer, M. A. (1991). Nonlinear Principal Component Analysis Using Autoassociative Neural Networks. *AIChE journal*, 37(2): 233–243.

[Kriegel *et al.*, 2008] Kriegel, H.-P., Schubert, M. et Zimek, A. (2008). Angle-Based Outlier Detection in High-dimensional Data. *In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 444–452. Association for Computing Machinery.

[Kromanis et Kripakaran, 2013] Kromanis, R. et Kripakaran, P. (2013). Support Vector Regression for Anomaly Detection from Measurement Histories. *Advanced Engineering Informatics*, 27(4):486–495.

[Lavin et Ahmad, 2015] Lavin, A. et Ahmad, S. (2015). Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. *In IEEE 14th International Conference on Machine Learning and Applications*, pages 38–44. IEEE.

[Le Guen, 2002] Le Guen, M. (2002). La Boîte à Moustaches pour Sensibiliser à la Statistique. *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, 73(1):43–64.

[Leder, 2020] Leder, S. (2020). BlackRock, la Finance au Chevet des Retraités Français. *Le Monde Diplomatique*, (790):16–17.

[Lee *et al.*, 1997] Lee, W., Stolfo, S. J. et Chan, P. K. (1997). Learning Patterns from Unix Process Execution Traces for Intrusion Detection. *In AAAI Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 50–56. AAAI Press.

[Lin *et al.*, 2005] Lin, J., Keogh, E., Fu, A. et Van Herle, H. (2005). Approximations to Magic : Finding Unusual Medical Time Series. *In 18th IEEE Symposium on Computer-Based Medical Systems*, pages 329–334. IEEE.

[Lin *et al.*, 2007a] Lin, J., Keogh, E., Wei, L. et Lonardi, S. (2007a). Experiencing SAX : a Novel Symbolic Representation of Time Series. *Data Mining and knowledge discovery*, 15(2):107–144.

[Lin *et al.*, 2007b] Lin, J., Keogh, E., Wei, L. et Lonardi, S. (2007b). Experiencing SAX : a Novel Symbolic Representation of Time Series. *Data Mining and Knowledge Discovery*, 15(2):107–144.

[Liu *et al.*, 2008] Liu, F. T., Ting, K. M. et Zhou, Z. (2008). Isolation Forest. *In Eighth IEEE International Conference on Data Mining*, pages 413–422.

[McCulloch et Pitts, 1943] McCulloch, W. S. et Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

[McGovern *et al.*, 2011] McGovern, A., Rosendahl, D. H., Brown, R. A. et Droegemeier, K. K. (2011). Identifying Predictive Multi-Dimensional Time Series Motifs : an Application to Severe Weather Prediction. *Data Mining and Knowledge Discovery*, 22(1):232–258.

[Minsky et Papert, 1969] Minsky, M. et Papert, S. A. (1969). *Perceptrons : an Introduction to Computational Geometry*. MIT Press. 258 p.

[Muñoz-Marí *et al.*, 2010] Muñoz-Marí, J., Bovolo, F., Gómez-Chova, L., Bruzzone, L. et Camp-Valls, G. (2010). Semisupervised One-Class Support Vector Machines for Classification of Remote Sensing Data. *IEEE Transactions on Geoscience and Remote Sensing*, 48(8):3188–3197.

[Ng *et al.*, 2002] Ng, A. Y., Jordan, M. I. et Weiss, Y. (2002). On Spectral Clustering : Analysis and an Algorithm. *In Advances in Neural Information Processing Systems*, pages 849–856. MIT Press.

[Panda et Patra, 2007] Panda, M. et Patra, M. R. (2007). Network Intrusion Detection Using Naive Bayes. *International Journal of Computer Science and Network Security*, 7(12):258–263.

[Piciarelli et Foresti, 2006] Piciarelli, C. et Foresti, G. L. (2006). Online Trajectory Clustering for Anomalous Events Detection. *Pattern Recognition Letters*, 27(15):1835–1842.

[Pokrajac *et al.*, 2007] Pokrajac, D., Lazarevic, A. et Latecki, L. J. (2007). Incremental Local Outlier Detection for Data Streams. *In IEEE Symposium on Computational Intelligence and Data Mining*, pages 504–515. IEEE.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of Decision Trees. *Machine learning*, 1(1):81–106.

[Radovanović *et al.*, 2009] Radovanović, M., Nanopoulos, A. et Ivanović, M. (2009). Nearest Neighbors in High-Dimensional Data : The Emergence and Influence of Hubs. *In Proceedings of the 26th Annual International Conference on Machine Learning*, pages 865–872. Association for Computing Machinery.

[Radovanović *et al.*, 2010] Radovanović, M., Nanopoulos, A. et Ivanović, M. (2010). Hubs in Space : Popular Nearest Neighbors in High-Dimensional Data. *Journal of Machine Learning Research*, 11:2487–2531.

[Ramaswamy *et al.*, 2000] Ramaswamy, S., Rastogi, R. et Shim, K. (2000). Efficient Algorithms for Mining Outliers from Large Data Sets. *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 427–438. ACM Press.

[Ribeiro *et al.*, 2017] Ribeiro, M. H., Calais, P. H., Santos, Y. A., Almeida, V. A. F. et Meira Jr., W. (2017). "Like Sheep Among Wolves" : Characterizing Hateful Users on Twitter. *In Proceedings of the Workshop on Misinformation and Misbehavior Mining on the Web*, pages 1–8. Association for Computing Machinery.

[Saito et Rehmsmeier, 2015] Saito, T. et Rehmsmeier, M. (2015). The Precision-Recall Plot is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS One*, 10(3):1–21.

[Sakurada et Yairi, 2014] Sakurada, M. et Yairi, T. (2014). Anomaly Detection using Autoencoders with Nonlinear Dimensionality Reduction. *In Proceedings of the MLSDA 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11. Association for Computing Machinery.

[Salehi *et al.*, 2016] Salehi, M., Leckie, C., Bezdek, J. C., Vaithianathan, T. et Zhang, X. (2016). Fast Memory Efficient Local Outlier Detection in Data Streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3246–3260.

[Savage *et al.*, 2014] Savage, D., Zhang, X., Yu, X., Chou, P. et Wang, Q. (2014). Anomaly Detection in Online Social Networks. *Social Networks*, 39:62–70.

[Schneider *et al.*, 2016] Schneider, M., Ertel, W. et Ramos, F. (2016). Expected Similarity Estimation for Large-Scale Batch and Streaming Anomaly Detection. *Machine Learning*, 105(3):305–333.

[Schölkopf *et al.*, 2000] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J. et Platt, J. C. (2000). Support Vector Method for Novelty Detection. *In Advances in Neural Information Processing Systems*, pages 582–588. MIT Press.

[Schölkopf *et al.*, 2001] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J. et Williamson, R. C. (2001). Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, 13(7):1443–1471.

[Sebyala *et al.*, 2002] Sebyala, A. A., Olukemi, T., Sacks, L. et Sacks, L. (2002). Active Platform Security through Intrusion Detection Using Naïve Bayesian Network for Anomaly Detection. *In Proceedings of the London Communications Symposium*, pages 1–5. Citeseer.

[Shieh et Keogh, 2008] Shieh, J. et Keogh, E. (2008). *i*SAX : Indexing and Mining Terabyte Sized Time Series. *In Proceedings of the 14th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 623–631. Association for Computing Machinery.

[Shieh et Keogh, 2009] Shieh, J. et Keogh, E. (2009). *i*SAX : Disk-Aware Mining and Indexing of Massive Time Series Datasets. *Data Mining and Knowledge Discovery*, 19(1):24–57.

[Soni *et al.*, 2011] Soni, J., Ansari, U., Sharma, D. et Soni, S. (2011). Predictive Data Mining for Medical Diagnosis : An Overview of Heart Disease Prediction. *International Journal of Computer Applications*, 17(8): 43–48.

[Specht, 1991] Specht, D. F. (1991). A General Regression Neural Network. *IEEE Transactions on Neural Networks*, 2(6):568–576.

[Spence *et al.*, 2001] Spence, C., Parra, L. et Sajda, P. (2001). Detection, Synthesis and Compression in Mammographic Image Analysis with a Hierarchical Image Probability Model. *In Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 3–10. IEEE.

[Srivastava *et al.*, 2008] Srivastava, A., Kundu, A., Sural, S. et Majumdar, A. (2008). Credit Card Fraud Detection using Hidden Markov Model. *IEEE Transactions on Dependable and Secure Computing*, 5(1):37–48.

[Szegedy *et al.*, 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. et Rabinovich, A. (2015). Going Deeper with Convolutions. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[Thaseen et Kumar, 2013] Thaseen, S. et Kumar, C. A. (2013). An Analysis of Supervised Tree Based Classifiers for Intrusion Detection System. *In International Conference on Pattern Recognition, Informatics and Mobile Engineering*, pages 294–299. IEEE.

[Tibshirani, 1996] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society : Series B (Methodological)*, 58(1):267–288.

[Turing, 1950] Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, (49):433–460.

[Veasey et Dodson, 2014] Veasey, T. J. et Dodson, S. J. (2014). Anomaly Detection in Application Performance Monitoring Data. *International Journal of Machine Learning and Computing*, 4(2):120–126.

[Verbesselt *et al.*, 2012] Verbesselt, J., Zeileis, A. et Herold, M. (2012). Near Real-Time Disturbance Detection Using Satellite Image Time Series. *Remote Sensing of Environment*, 123:98–108.

[Wang *et al.*, 2013] Wang, Y., Wang, P., Pei, J., Wang, W. et Huang, S. (2013). A Data-Adaptive and Dynamic Segmentation Index for Whole Matching on Time Series. *Proceedings of the VLDB Endowment*, 6(10): 793–804.

[Wei *et al.*, 2008] Wei, L., Keogh, E., Xi, X. et Yoder, M. (2008). Efficiently Finding Unusual Shapes in Large Image Databases. *Data Mining and Knowledge Discovery*, 17(3):343–376.

[Weizenbaum, 1966] Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

[Woisetschläger *et al.*, 1994] Woisetschläger, J., Sheffer, D. B., Loughry, C. W., Somasundaram, K., Chawla, S. K. et Wesolowski, P. J. (1994). Phase-Shifting Holographic Interferometry for Breast Cancer Detection. *Applied Optics*, 33(22):5011–5015.

[Yamanishi *et al.*, 2004] Yamanishi, K., Takeuchi, J.-I., Williams, G. et Milne, P. (2004). On-Line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300.

[Ye, 2000] Ye, N. (2000). A Markov Chain Model of Temporal Behavior for Anomaly Detection. *In Proceedings of the IEEE Workshop on Information Assurance and Security*, pages 171–174.

[Zimek *et al.*, 2012] Zimek, A., Schubert, E. et Kriegel, H.-P. (2012). A Survey on Unsupervised Outlier Detection in High-Dimensional Numerical Data. *Statistical Analysis and Data Mining : The ASA Data Science Journal*, 5(5):363–387.

FOLIO ADMINISTRATIF

THÈSE DE L'UNIVERSITÉ DE LYON OPÉRÉE AU SEIN DE L'INSA LYON

NOM : FOULON
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 16/10/2020

Prénoms : Lucas

TITRE : Détection d'anomalies dans les flux de données par structure d'indexation et approximation. Application à l'analyse en continu des flux de messages du système d'information de la SNCF.

NATURE : Doctorat

Numéro d'ordre : 2020LYSEI082

Ecole doctorale : InfoMaths (ED 512)

Spécialité : Informatique

RESUME : Dans cette thèse, nous proposons des méthodes de calcul approchées d'un score d'anomalie, pouvant être mises en œuvre sur des flux de données pour détecter des portions anormales. La difficulté du problème est de deux ordres. D'une part, la haute dimensionnalité des objets manipulés pour décrire les séries temporelles extraites d'un flux brut, et d'autre part la nécessité de limiter le coût de détection afin de pouvoir la réaliser en continu au fil du flux. Concernant le premier aspect du problème, notre étude bibliographique a permis de sélectionner un score de détection d'anomalies proposé récemment, le score CFOF, qui est le seul pour lequel il existe des garanties formelles quant à son adéquation pour les données en haute dimensionnalité. Nos contributions ont alors porté sur la proposition de deux méthodes d'approximation du score CFOF pour permettre son usage en continu sur des flux. La première est une approche combinant élagage et approximation lors du parcours des voisinages dans l'espace de description des objets. Notre second apport est une approximation par agrégation de scores obtenus sur des sous-espaces, qui complète la première contribution et se combine avec elle. Nous avons montré sur une collection de jeux de données, utilisés comme cadre d'évaluation de référence dans le domaine, que nos méthodes permettaient des gains importants en temps de calcul, tout en fournissant des approximations qui préservent la qualité des détections. Enfin, nous présentons également l'application de ces approches au sein du système d'information de la SNCF dans lequel de nombreux flux sont collectés en temps réel, transformés et rediffusés. Dans ce contexte, nous avons étendu la supervision de bout-en-bout existante par la mise en œuvre d'un outil d'aide à la détection d'anomalies sur le flux de messages entrant d'une des principales plateformes de traitement.

MOTS-CLÉS : Détection d'Anomalies, Flux de Données, Arbre d'Indexation, Système d'Information

Laboratoire (s) de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)

Directeur de thèse :

Dr. Christophe RIGOTTI (INSA de Lyon)

Présidente de jury :

Pr. Élisa FROMONT (Université de Rennes 1) - Examinatrice

Composition du jury :

Pr. Alain COURNIER (Université de Picardie Jules Verne) - Rapporteur
Pr. Thierry CHARNOIS (Université Paris 13) - Rapporteur
Pr. Sylvie CALABRETTO (INSA de Lyon) - Examinatrice
Pr. Élisa FROMONT (Université de Rennes 1) - Examinatrice
Dr. Peter STURM (INRIA Grenoble Rhône-Alpes) - Examinateur
Dr. Serge FENET (Université Lyon 1) - Examinateur
Dr. Christophe RIGOTTI (INSA de Lyon) - Examinateur
Dr. Denis JOUVIN (SNCF Voyageurs – Direction Industrielle) – Invité