École doctorale : Science Pour l'Ingénieur

THÈSE

Pour obtenir le grade de docteur délivré par

Université Clermont Auvergne

Spécialité *informatique*

Présentée et soutenue publiquement par

Nicolas WAGNER

Le 27 octobre 2020

Détection des modifications de l'organisation circadienne des activités des animaux en relation avec des états pré-pathologiques, un stress, ou un événement de reproduction

Directeurs : **Jonas KOKO** & **Isabelle VEISSIER**
Co-encadrante : **Violaine ANTOINE**

Membres du jury

Rapporteur : **M. Nicolas LABROCHE**, *MCF HDR*
Rapporteur : **M. Alexandre TERMIER**, *Professeur*
Examinateur : **Mme Masoomeh TAGHIPOOR**, *Ingénieur de recherche*
Examinateur : **M. Engelbert MEPHU NGUIFO**, *Professeur*
Encadrante : **Mme Violaine ANTOINE**, *MCF*
Encadrante : **Mme Isabelle VEISSIER**, *DR*

LIMOS
Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes
UMR CNRS 6158, Clermont-Ferrand, France

Table des matières

Remerciements

Je tiens à adresser toute ma gratitude à Madame Violaine Antoine et Monsieur Jonas Koko, Maitres de conférences à l'Université Clermont Auvergne pour m'avoir encadré durant ces trois années de thèse.

Je veux également remercier Madame Isabelle Veissier, directrice de recherche à l'INRAE pour m'avoir co-encadré et pour avoir pris le temps de m'expliquer tous les principes biologiques nécessaires pour ma thèse.

Je remercie également Monsieur Romain Lardy et Madame Marie-Madeleine Mialon pour m'avoir suivi durant ces trois années.

Je remercie aussi Monsieur Nicolas Labroche, Maitre de conférences à l'université de Tours et Monsieur Alexandre Termier, Professeur à l'université de Rennes pour avoir accepté d'être rapporteurs de ma thèse.

Je tiens à remercier Madame Masoomeh Taghipoor ingénieur de recherche à l'INRAE et Monsieur Engelbert Mephu Nguifo, professeur à l'Université Clermont Auvergne pour avoir accepté de faire parti de mon jury de thèse.

Enfin, je remercie Madame karen Helle Sloth de la société GEA farms technologies pour m'avoir fourni un des jeux de données.

Résumé

L'élevage de précision consiste à enregistrer des paramètres sur les animaux ou leur environnement grâce à divers capteurs. Dans cette thèse, il s'agit de suivre le comportement de vaches laitières via un système de localisation en temps réel. Les données sont collectées en une suite de valeurs à intervalle régulier, c'est ce que l'on appelle une série temporelle. Les problèmes liés à l'utilisation de capteurs sont le grand nombre de données engendré et la qualité de ces données. Le Machine Learning (**ML**) permet d'atténuer ce problème.

Le but de cette thèse est de détecter les comportements anormaux de vaches. L'hypothèse de travail, étayée par la littérature en biologie, est que le rythme circadien d'activité d'une vache change si celle-ci passe d'un état normal à un état de maladie, stress ou encore un stade physiologique spécifique (œstrus, mise-bas) et ce, de manière très précoce. La détection d'une anomalie de comportement permettrait de prendre des décisions plus rapidement en élevage. Pour cela, il existe des outils de classification de séries temporelles ou Time Series Classification (**TSC**) en anglais.

Le problème avec les données de comportement est que le schéma comportemental dit *normal* de la vache varie selon les vaches, les jours, la ferme, la saison, etc. Trouver un schéma normal commun à toutes les vaches est donc impossible. Or, la plupart des outils de TSC se basent sur l'apprentissage d'un modèle global pour définir si un comportement donné est proche de ce modèle ou non.

Cette thèse s'articule autour de deux grandes contributions. La première consiste à l'élaboration d'une nouvelle méthode de TSC : FBAT. Elle se base sur les transformées de Fourier pour identifier un pattern d'activité sur 24 h et le comparer à celui d'une autre période de 24 h consécutive, afin de palier le problème de l'absence de schéma commun d'une vache normale. La deuxième contribution consiste à utiliser les étiquettes floues. En effet, autour des jours considérés comme anormaux, il est possible de définir une zone incertaine où la vache serait dans un état intermédiaire. Nous montrons que la logique floue permet d'améliorer les résultats quand les étiquettes sont incertaines et nous introduisons une variante floue de FBAT : F-FBAT.

Mots clés : série temporelle, classification de séries temporelles, transformées de Fourier, étiquettes floues, comportement animal, bien-être animal, santé animale.

Abstract

Precision livestock farming consists of recording parameters on the animals or their environment using various sensors. In this thesis, the aim is to monitor the behaviour of dairy cows via a real-time localisation system. The data are collected in a sequence of values at regular intervals, a so-called time series. The problems associated with the use of sensors are the large amount of data generated and the quality of this data. The Machine Learning (**ML**) helps to alleviate this problem.

The aim of this thesis is to detect abnormal cow behaviour. The working hypothesis, supported by the biological literature, is that the circadian rhythm of a cow's activity changes if it goes from a normal state to a state of disease, stress or a specific physiological stage (oestrus, farrowing) at a very early stage. The detection of a behavioural anomaly would allow decisions to be taken more quickly in breeding. To do this, there are Time Series Classification (**TSC**) tools.

The problem with behavioural data is that the so-called *normal* behavioural pattern of the cow varies from cow to cow, day to day, farm to farm, season to season, and so on. Finding a common normal pattern to all cows is therefore impossible. However, most TSC tools rely on learning a global model to define whether a given behaviour is close to this model or not.

This thesis is structured around two major contributions. The first one is the development of a new TSC method : FBAT. It is based on Fourier transforms to identify a pattern of activity over 24 hours and compare it to another consecutive 24-hour period, in order to overcome the problem of the lack of a common pattern in a normal cow. The second contribution is the use of fuzzy labels. Indeed, around the days considered abnormal, it is possible to define an uncertain area where the cow would be in an intermediate state. We show that fuzzy logic improves results when labels are uncertain and we introduce a fuzzy variant of FBAT : F-FBAT.

Keywords : time series, time series classification, Fourier transforms, fuzzy labels, animal behaviour, animal welfare, animal health.

Notations

$ARIMA(p, d, q)$ un modèle ARIMA d'ordre p, q de différentiation d. 38

$ARMA(p, q)$ un modèle ARMA d'ordre p et q avec ϕ et θ comme paramètres. 38

L^i l'opérateur retard avec $L^i x_t = x_{t-i}$. 38

$\hat{y}_p^i$ sortie d'un neurone p. 39

$\lambda_{MSE}(y)$ la fonction de coût Mean Square Error. 40, 41

$\mathcal{D}_{test}$ un jeu de données de test. 45, 57, 63, 67, 68, 76, 88, 93

$\mathcal{D}_{train}$ un jeu de données d'apprentissage. 45, 60, 63, 67, 68, 76, 89–91, 93, 100

$\mathcal{D}_{val}$ un jeu de données de validation. 45

$\nabla\lambda(y)$ la fonction de coût Mean Square Error. 41

$d_{BOSS}(\boldsymbol{b}_1, \boldsymbol{b}_2)$ est la distance BOSS entre deux histogrammes $\boldsymbol{b_1}$ et $\boldsymbol{b_2}$. 49, 50

$d_{DTW}(\boldsymbol{a}, \boldsymbol{b})$ est la mesure DTW entre les séries temporelles $\boldsymbol{a}$ et $\boldsymbol{b}$. 47

$d_{eucli}(\boldsymbol{x_i}, \boldsymbol{x_j})$ est la distance euclidienne entre les séries temporelles $\boldsymbol{x_i}$ et $\boldsymbol{x_j}$. 57

f_a une fonction d'activation. 39

$f_c : \mathcal{X} \to \{0, 1\}$ une fonction caractéristique avec $\mathcal{X}$ l'ensemble des séries temporelle du jeu de données et $c \in \mathcal{C}$ la classe correspondante.. 56

$u_c : \mathcal{X} \to [0, 1]$ une fonction d'appartenance avec $\mathcal{X}$ l'ensemble des séries temporelle du jeu de données et $c \in \mathcal{C}$ la classe correspondante.. 57

w_j^p poids sur l'entrée x_j^i d'un neurone p. 39

$|\mathcal{D}|$ la cardinalité de l'ensemble $\mathcal{D}$.. 57, 89

$\boldsymbol{x} = [x_1, x_2, ..., x_n]$ une série temporelle univariée de taille n. 32, 60

$X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m]$ une série temporelle multivariée de m dimensions. 32

$\mathcal{D} = \{(X_i, y_i)\}$ un jeu de données constitué d'une série temporelle X et d'une classe $y_i \in \mathcal{C}$. 32

$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ un jeu de données constitué d'une série temporelle $\boldsymbol{x}$ et d'une classe $y_i \in \mathcal{C}$. 32, 40, 56

$\mathcal{C}$ l'ensemble des classes possible d'un jeu de données $\mathcal{D}$, $\mathcal{C} \subset \mathbb{N}$. 32, 89, 90, 93

$\bar{\boldsymbol{x}}$ la moyenne du vecteur $\boldsymbol{x}$. 33

$Var(\boldsymbol{x})$ variance du vecteur $\boldsymbol{x}$. 33

$Cov(\boldsymbol{x}, \boldsymbol{t})$ covariance entre les vecteurs $\boldsymbol{x}$ et $\boldsymbol{t}$. 33

A_n l'amplitude de la composante cosinusoïdale n de la transformée de Fourier. 34

ρ_n la phase de la composante cosinusoïdale n de la transformée de Fourier. 34

h_n l'harmonique de rand n de la transformée de Fourier, $h_n \in \mathbb{C}$. 35

$arg(h_n)$ l'argument de h, n, $h_n \in \mathbb{C}$. 36

$|h_n|$ le module de h_n, $h_n \in \mathbb{C}$. 36

Introduction

Étymologiquement, le mot *informatique* signifie *science de l'information*, c'est-à-dire la science qui permet de traiter des données (parfois très volumineuses) afin d'exploiter l'information qu'elles contiennent. Cette science est souvent mise à contribution d'autres sciences. En effet, grâce à l'usage massif de capteurs et à la facilité de stockage de nos ordinateurs, la capitalisation de données est devenue à la fois aisée et automatique. Cela engendre une masse de données qui ne peut être traitée que par l'outil informatique. Dans ces données, se cache de l'information qui est souvent très précieuse mais qui demande à être trouvée et exploitée. C'est ainsi que des outils rassemblés sous le vocable : fouille de données, algorithmes de Machine Learning (**ML**), réseaux de neurones, etc. ont vu leur intérêt grandir ces dernières années. Ces outils ont pour particularité de n'être pas basés sur des règles prédéfinies mais sur un apprentissage basé sur un ensemble de données. C'est pour cela que l'on parle d'apprentissage : l'algorithme apprend lui-même le modèle via une expérience. Par exemple, le ML est utilisé par [94] pour prédire des tremblements de terre, [123] l'utilise pour prédire les radiations solaires ou encore [19] utilise le ML pour détecter des cancers.

Durant cette thèse, j'ai mis les outils informatiques et mathématiques au service de la biologie afin de détecter des comportements anormaux chez la vache laitière. Ces comportements anormaux peuvent être le signe d'un désordre comme une maladie, un stress ou encore un œstrus. Leur détection précoce pourrait grandement aider l'éleveur à gérer son élevage, à réduire la consommation de médicaments et à améliorer le bien-être de l'animal. Cette thèse traite donc à la fois d'informatique : ML, séries temporelles, classification supervisée, prédiction et à la fois de biologie : vaches laitières, comportement animal, capteurs de position, maladies, symptômes et rythme d'activité.

Le but de cette thèse est de trouver/créer une méthode qui permet de détecter des anomalies dans le rythme d'activité des vaches laitières. Cette activité, sous forme de série temporelle, a été mesurée à l'aide de capteurs de position.

Dans la suite de cette introduction, une première section traite des enjeux du bien-être animal, une deuxième section donne des détails sur l'élevage de précision

et une dernière section explique comment traiter ces données via les outils informatiques.

1 Les enjeux liés au bien-être animal et à la santé des animaux

Il est établi que les animaux sont des êtres sensibles, c'est-à-dire capables d'éprouver des émotions, du stress, de la douleur. A ce titre, les individus qui détiennent des animaux – dont les éleveurs — sont tenus de leur assurer de bonnes conditions de vie. Voir par exemple l'article 214 - 1 du code rural : *Tout animal étant un être sensible doit être placé par son propriétaire dans des conditions compatibles avec les impératifs biologiques de son espèce*[1].

A l'heure actuelle, le bien-être des animaux d'élevage fait l'objet de questionnements sociétaux importants. C'est devenu la première préoccupation des français au regard de l'élevage[2]. Le bien-être comprend à la fois des aspects physiques (bonne santé, confort) et psychologiques (absence de stress, de détresse, présence d'expériences positives). Il a été défini sur la base de 5 libertés par le Farm Animal Welfare Council [17] :

— l'absence de faim et de soif prolongée ;
— le confort physique ;
— l'absence de blessures, maladies ou douleurs ;
— l'absence de peur et détresse ;
— la possibilité d'exprimer les comportements normaux de l'espèce.

Dans cette définition le ressenti de l'animal est mis en avant. Ainsi ce n'est pas tant le fait que l'animal soit malade qui est problématique mais celui qu'il ressente qu'il est malade. Comme nous le verrons dans le chapitre 1, ce ressenti se traduit par un comportement de mal-être que nous allons chercher à détecter dans cette thèse.

Par ailleurs, la maitrise de la santé des animaux fait partie intégrante de l'élevage : pour produire (du lait, des œufs, de la viande, ...) les animaux doivent être en bonne santé. Par exemple, des pertes importantes de production de lait peuvent survenir lors de mammites ou de boiteries chez la vache laitière (jusqu'à 800 L de lait par lactation [102]). Il est donc important de détecter rapidement ces troubles afin de les traiter le plus rapidement possible. Par ailleurs, la maitrise de la reproduction est un élément crucial : l'éleveur doit détecter le moment propice où il faudra inséminer une femelle pour assurer la reproduction des animaux et dans le cas de femelles

1. https://www.legifrance.gouv.fr/affichCode.do?idSectionTA=LEGISCTA000006152208&cidTexte=LEGITEXT000006071367&dateTexte=20080531
2. Enquête auprès de 1083 élèves de Terminal ; https://www.ifip.asso.fr/sites/default/files/pdf-documentations/bpn2015-450-roguet.pdf

laitières, assurer la production de lait. Enfin, tout stress est susceptible d'affecter le fonctionnement de l'animal et donc la production.

Qu'il s'agisse de l'animal ou des impératifs de l'élevage, il est donc important de pouvoir détecter tout désordre survenant sur l'animal.

2 Outils d'élevage de précision

L'élevage de précision est apparu dans les années 2000. Il s'agit d'utiliser des capteurs et un traitement ad-hoc des données qu'ils fournissent pour suivre en continu les animaux ou leur environnement et être en mesure de détecter tout problème pour le corriger rapidement [40]. Plusieurs types de capteurs sont utilisés : localisation en temps réel ou Real Time Locating Systems (RTLS) en anglais, accéléromètres, vidéos, capteurs sonores, etc. pour suivre le comportement des animaux ou capteurs plus spécifiques pour suivre des troubles particuliers (capteurs de pH dans le rumen des vaches, analyseurs automatiques de lait,...). Ces systèmes fournissent une masse importante de données. Chacune de ces données a peu d'importance et est souvent bruitée : par exemple un capteur peut se retrouver momentanément dans une zone d'ombre et ne pas pouvoir envoyer d'information au récepteur avec lequel il est censé communiquer. C'est l'agrégation de l'ensemble des données qui leur donne leur signification. Ainsi ce n'est pas la position d'une vache au temps x qui est intéressante mais combien de temps elle passe dans certaines zones d'intérêt (à l'auge, au repos dans une logette, en train de pâturer,...) ou à quel moment de la journée elle le fait (voir chapitre 1). De plus il est indispensable d'interpréter les résultats en termes biologiques : comment détecter et interpréter le fait qu'une vache passe plus ou moins de temps couchée ou à manger ? Il est donc nécessaire de concevoir des algorithmes pour analyser les données, et ce en concertation entre informaticiens et biologistes.

3 Comment aider à prendre une décision à partir de ce grand nombre de données issues des outils d'élevage de précision ?

Les analyses statistiques (moyenne, variation, test d'hypothèse, etc.) permettent de mettre en évidence des effets d'une condition particulière sur l'animal. Ainsi, une baisse d'activité peut être observée chez des animaux malades en comparaison à des animaux sains (voir chapitre 1). Se pose la question de passer de la description, que permettent les approches statistiques, à la prédiction, par exemple : est-ce que ce comportement d'un animal relève d'une situation normale ou non ? Pour cela, il est

nécessaire de modéliser le comportement de l'animal afin de distinguer ce qui est normal de ce qui ne l'est pas. Des outils informatiques/mathématiques tel que le ML peuvent être d'une aide précieuse. De plus en plus de problèmes de biologie trouvent leur solution grâce au ML. Par exemple, [116] donne une liste d'applications qui utilisent le ML dans le domaine du comportement animal. Le ML est par exemple utilisé par [48] pour mesurer les comportements sociaux chez la souris sur la base de vidéos. De même, [55] utilise le ML pour directement annoter le comportement animal. Dans cette thèse nous explorons l'apport du ML pour détecter des anomalies de comportement chez des vaches, comportement que nous avons traité comme une série temporelle. Une série temporelle est une liste de valeurs ordonnées dans le temps représentant l'évolution d'une même observation. C'est un type de données que l'on retrouve souvant dans le monde de la science de données et doit être traité avec des méthodes spécifiques (voir chapitre 2).

Dans le premier chapitre, les connaissances en biologie du comportement et de ces anomalies sont décrites. Il est également expliqué comment les données de la thèse ont été acquises et quelles en sont leurs caractéristiques. Le deuxième chapitre fournit un état de l'art dans le domaine des séries temporelles. Le troisième chapitre présente une méthode (FBAT) de détection d'anomalies que nous avons développée et la compare à d'autres méthodes de la littérature. Le quatrième chapitre introduit la notion d'étiquettes floues et présente trois nouvelles méthodes, dont une version de FBAT, qui utilisent ce concept dans le domaine des séries temporelles. La thèse se finit par une conclusion et des perspectives.

Chapitre 1

État des connaissances en biologie et jeux de données

Ce chapitre traite dans une première section du comportement animal, en particulier des rythmes biologiques. Il aborde l'impact des maladies et du stress sur le comportement et son organisation selon le rythme circadien. Les données utilisées dans le cadre de cette thèse sont détaillées dans la seconde section du chapitre. Nous expliquons d'où les données proviennent et comment elles ont été collectées et nous exposons certaines propriétés de ces données qui seront utiles dans le cadre de la thèse.

1 Comportement, maladie et rythme circadien

1.1 Comportement animal : budget-temps

Le comportement d'un animal concerne toute activité qui se manifeste à un observateur extérieur. Chaque espèce possède un répertoire comportemental, c.-à-d. une façon qui lui est propre d'exprimer des activités, qu'il s'agisse d'activités liées à l'alimentation, au déplacement, aux relations sociales, à la reproduction, ... [23]. Parmi ces activités on distingue les activités dites « de base » comme manger, se reposer, se déplacer, parfois appelées « état » car elles sont exprimées avec une certaine durée par opposition à des événements ponctuels comme une interaction sociale (par exemple un coup entre deux animaux). Ces activités de base sont utilisées pour décrire le budget-temps de l'animal : combien de temps passe-t-il dans telle ou telle activité et à quel moment de la journée ? Ainsi, au pâturage (Figure 1.1) une vache passe en moyenne 8 h à pâturer, essentiellement entre le lever et le couché du jour [88]. À l'étable, le temps dédié à l'alimentation peut être réduit et les vaches passent alors plus de temps couchées ou debout immobile : 12 h couchées et près de

5 h debout immobiles [118]. Le temps passé et la répartition de ces activités peuvent varier en fonction de la disponibilité en nourriture au pâturage ou des conditions météorologiques [88].

FIGURE 1.1 – Image de vaches en pâturage. Crédit photo : Dominique Pomies.

1.2 Modifications du comportement

Les activités d'un animal peuvent varier sous l'effet d'une maladie (image d'une vache malade en Figure 1.2). La fièvre est souvent associée à un ensemble de modifications comportementales résumées sous le terme de *comportement de maladie* (sickness behaviour en anglais) [43]. Un état léthargique s'installe alors pendant lequel l'animal réduit son activité, dort davantage et à des moments où il est normalement éveillé, réduit sa consommation d'aliments et d'eau, et interagit moins avec les congénères ou avec les humains [11, 22, 42]. Le comportement de maladie peut commencer avant que les symptômes cliniques d'une maladie soient visibles. Par exemple, les vaches sont moins réactives à leur environnement et changent d'activité moins souvent quelques heures avant la fièvre due à une mammite [27].

Le comportement de maladie est observé chez de nombreuses espèces, y compris chez l'Homme. Il est induit par des cytokines libérées au début de l'infection, lesquelles agissent sur le cerveau. On pense que le comportement face à la maladie réduit la dépense énergétique pour maintenir le coût métabolique élevé de la réaction à la fièvre et faciliter ainsi le rétablissement après une infection [22, 42]. Par ailleurs, le comportement face à la maladie pourrait contribuer à réduire la propagation des maladies parmi les groupes d'animaux [104], mais cette hypothèse n'a pas encore été confirmée [74].

Les modifications comportementales peuvent être différentes lorsque la maladie s'accompagne de douleur. Ainsi, en cas de mammite (inflammation de la mamelle)

ou de boiterie, une vache mange moins et passe moins de temps couchée [36, 80, 87]. De même, atteintes de métrite (entrainant une douleur viscérale), les vaches passent moins de temps couchées mais plus de temps debout immobiles dans les aires de repos [72]. Là encore les modifications comportementales peuvent être observées plusieurs jours avant la détection des signes cliniques.

Des modifications du comportement sont observées également au cours de maladies non-infectieuses ou a priori non douloureuses comme des maladies métaboliques. Ainsi, en cas d'acidose ruminale (associée à un pH du contenu du rumen - pré-estomac principal des ruminants - qui diminue fortement du fait de l'ingestion d'aliments rapidement digérés par les microorganismes du rumen), les vaches sont moins actives après le repas du matin [105]. Des modifications sont également observées en cas de stress. Ainsi les animaux sont plus agités lorsqu'ils sont mélangés avec des congénères qu'ils ne connaissent pas [117, 122]. Il en va de même de certains états physiologiques de l'animal, comme le vêlage qui est précédé par une diminution du temps passé à ruminer [15] ou les chaleurs qui se manifestent par un temps passé à marcher plus important [75]. Ainsi le comportement peut révéler un état interne de l'animal, qu'il s'agisse d'une maladie, d'un stress ou d'un état physiologique lié au cycle de reproduction.

FIGURE 1.2 – Image d'une vache malade (à droite de la photo), elle est isolée des autres, ne bouge pas avec la tête baissée. Crédit photo : Alice De Boyer.

1.3 Rythme circadien

Les activités sont rythmées par l'alternance jour-nuit. Ainsi un rythme circadien, c.-à-d. d'environ 24 h, s'installe (voir Figure 1.3). Ce rythme est présent chez la plupart des animaux et également certains végétaux. Certaines espèces animales sont dites diurnes car les animaux sont actifs principalement le jour. C'est le cas de la plupart des espèces utilisées en élevage et bien-sûr de l'Homme. D'autres espèces sont au contraire nocturnes car les animaux sont actifs essentiellement la nuit. C'est le cas par exemple des chauves-souris, des chouettes ou encore du renard.

L'expression des activités et le métabolisme sont régulés par des horloges biologiques internes, à l'origine du rythme circadien. L'horloge principale se situe dans le cerveau (dans le noyau supra chiasmatique). Il existe également des horloges périphériques, dans la plupart des organes, lesquelles sont synchronisées avec l'horloge principale [61, 133]. En l'absence de signaux extérieurs, ces horloges internes induisent un rythme d'environ 24 h. L'horloge principale est elle-même régulée par des signaux extérieurs comme la lumière ou le moment des repas. Ainsi il est possible de réorganiser son activité après un décalage horaire, même si cela peut demander plusieurs jours.

Chez les bovins, le rythme circadien est visible au travers de l'activité de pâturage qui s'exprime souvent au lever du jour, en milieu de journée et à la tombée de la nuit [60, 88]. En étable, le rythme dépend non seulement de l'alternance jour-nuit mais aussi des heures de distribution du repas [120]. Le rythme de vaches laitières peut également dépendre des heures de traite, si elles sont traites à heures régulières.

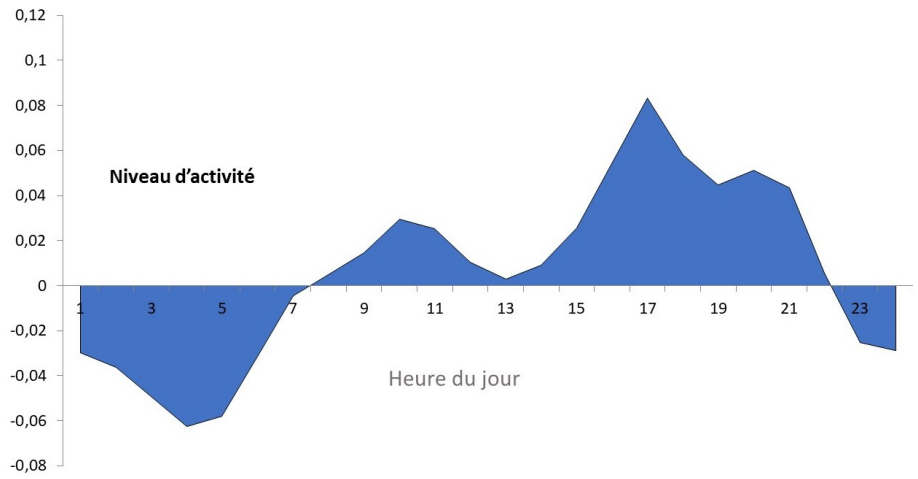


FIGURE 1.3 – Illustration du rythme d'activité d'une vache. Crédit : [92].

1.4 Perturbation du rythme circadien

Une perturbation des rythmes circadiens peut avoir des répercussions importantes sur la santé physique et mentale, par exemple en favorisant les cancers et la dépression [107, 133]. En retour, le rythme circadien peut être perturbé lorsque les animaux sont stressés ou malades, ce qui est un indicateur de tels troubles. Par exemple, les variations circadiennes de l'activité sont moins marquées chez les vaches malades [120] mais plus marquées lorsque les veaux sont mélangés à d'autres veaux, une procédure connue pour être stressante [117]. Ces effets peuvent impliquer des glucocorticoïdes, dont la libération est accrue en cas de stress ou de maladie. En effet, les glucocorticoïdes aident à coordonner les rythmes circadiens en remettant à zéro les horloges périphériques [31].

Les modifications du rythme circadien d'activité sont souvent des signes très précoces d'un trouble : le rythme peut être moins marqué deux jours avant que les symptômes cliniques d'une maladie infectieuse soient détectés par les soigneurs ou un jour avant que ceux-ci détectent les chaleurs d'une vache [119, 120]. Pour mettre en évidence de telles modifications de rythme, il est nécessaire d'enregistrer en continu les activités d'un animal. Ceci est maintenant possible grâce aux outils de monitoring utilisé en Élevage de Précision (voir introduction). Il est également nécessaire de disposer d'une mesure synthétique, reflétant le niveau d'activité de l'animal.

Pour mesurer les variations d'activité d'un animal certains utilisent des activités typiques comme l'utilisation d'une roue chez les hamsters ou les souris (dès qu'ils sont éveillés, ils vont dans la roue) [129]. D'autres se basent uniquement sur certaines activités comme le pâturage [88] ou le sommeil [114]. Cependant, les activités basiques peuvent changer en fréquence d'un jour à l'autre, ce qui peut biaiser la mesure du rythme, e.g. une activité avec une fréquence basse aura également de faibles variations durant la journée. Synthétiser les activités de base de l'animal en un niveau global d'activité permet de pallier cette difficulté. En effet, le niveau d'activité est exprimé dans des termes absolus, i.e. il n'a pas de fréquence. L'acquisition du niveau d'activité pourrait aider à détecter les différences entre un état *malade* ou *stressé* et un état *normal*. Cependant, il est toujours difficile de détecter exactement quand le rythme devient anormal. Des différences statistiques ont été relevées entre des animaux malades ou stressés et des animaux en bon état : par exemple, en moyenne des vaches atteintes de mammites ont un niveau d'activité plus élevé et plus variable dans la journée [120] (voire Figure 1.4) mais on ne peut pas dire précisément à partir de quand le niveau d'activité peut être considéré trop élevé ou variable pour un individu ou jour donné.

Le rythme peut également changer de manière naturelle, par exemple sous l'influence des saisons [62, 88, 98]. En effet, la température ou la durée d'ensoleillement peuvent jouer un rôle dans la modification du rythme d'activité, ce qui ajoute un

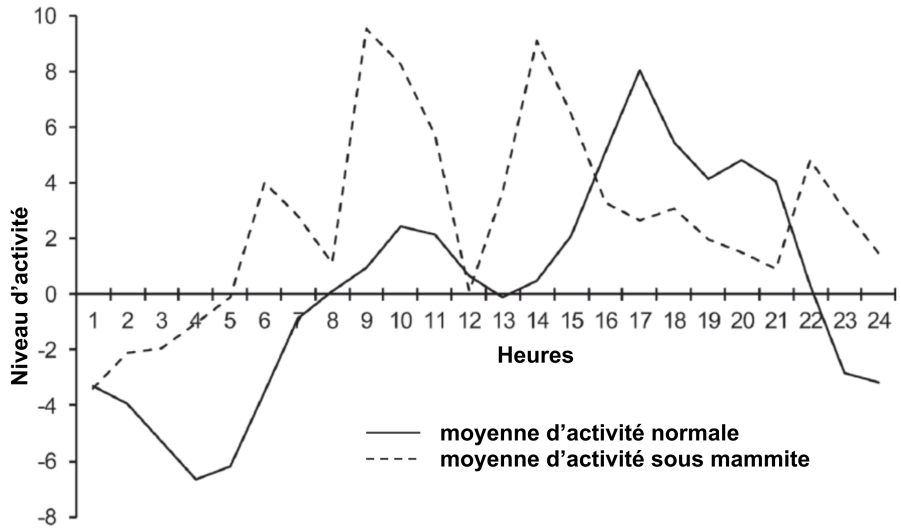


FIGURE 1.4 – Moyenne d'activité des jours sans perturbation et avec mammite, illustration provenant de [120].

difficulté supplément à la détection des états anormaux via l'activité.

2 Données de la thèse

Pour mener à bien cette thèse, je disposais de quatre jeux de données sur l'activité de vaches laitières dans une ferme expérimentale d'INRAE [1] et quatre fermes commerciales. Cette section explique dans un premier temps comment les données ont été collectées et traitées de sorte à obtenir des séries temporelles représentant le rythme d'activité des vaches. Dans un second temps, j'expose les principales caractéristiques des données afin de faire le parallèle avec les connaissances en biologie.

2.1 Acquisition des données

2.1.1 Caractéristiques des fermes

Dans les 5 fermes, les vaches étaient à l'étable au moins une partie de l'année. Les étables étaient des « stabulation à logettes », c'est-à-dire que la zone de repos des vaches était constituée de stalles individuelles, avec autant de stalles que de vaches (voir Figure 1.5 et 1.6).

1. Herbipôle, DOI : `https://doi.org/10.15454/1.5572318050509348E12`

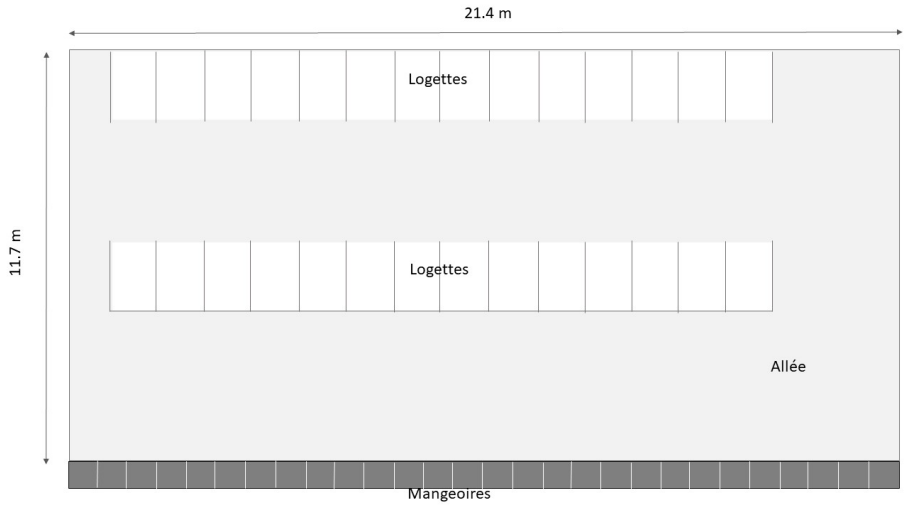


FIGURE 1.5 – Illustration d'une étable. Crédit : [126].

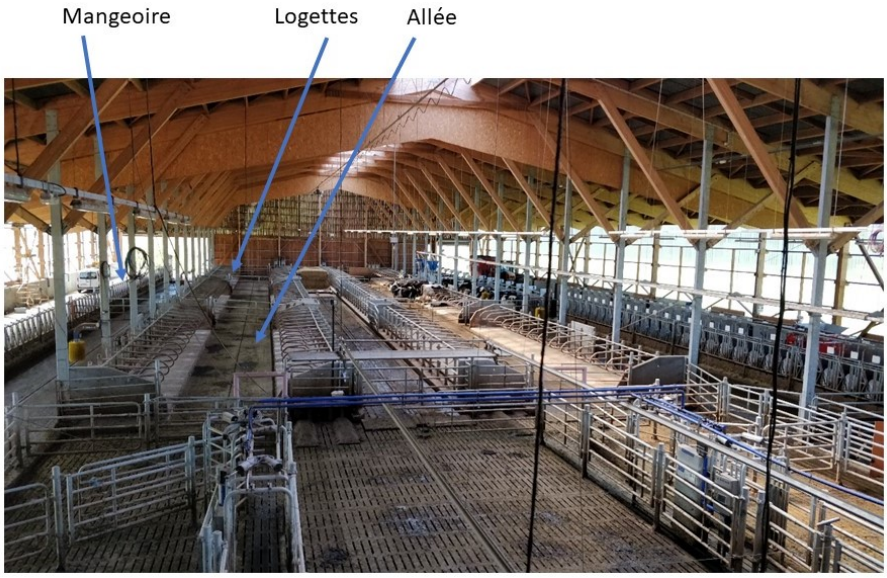


FIGURE 1.6 – Illustration d'une étable. Crédit photo : Nicolas Wagner.

Les éléments d'organisation des fermes sont synthétisés en Table 1.1. Dans la ferme expérimentale d'INRAE, les vaches étaient traites à heure fixe deux fois par jour (aux environs de 7 h et 16 h) ; l'aliment était distribué après la traite du matin et complété à 14 h (pour le jeu de données 1 décrit plus loin) ou simplement repoussé près de la barrière d'alimentation à 13 h, 16h30 et 20 h (pour le jeu de données 2 décrit plus loin).

Dans les fermes commerciales (A, B, C et D), les vaches étaient :

— Traites à heure fixe 2 fois (Ferme B) ou 3 fois par jour (ferme C) ou avec un robot de sorte qu'elles choisissaient elles-mêmes leur heure de traite (ferme A et D)

— Alimentées le matin puis l'aliment était repoussé régulièrement au cours de la journée (fermes A et D) ou repoussé seulement après les traites (ferme C), ou encore alimentées en deux fois dans la journée (ferme B)

Les étables étaient éclairées par la lumière naturelle exceptée dans la ferme A ou une lumière artificielle était utilisée avec une diminution de l'intensité de celle-ci pendant la nuit.

2.1.2 Suivi de l'activité des vaches

Les cinq fermes étaient équipées d'un système de positionnement en temps réel des animaux (RTLS). Il s'agissait du système CowView, commercialisé par GEA Farm Technologies [2]. Les vaches étaient munies d'un collier sur lequel est fixé un émetteur. L'émetteur (6 x 4,5 x 4 cm, 150 g) est maintenu sur le haut du cou de la vache grâce à un contrepoids de 7 x 5 x 3 cm, 400 g (voir Figure 1.7). Il émet des ondes radio dans la bande ultra large, lesquelles sont détectées par des antennes fixées sur le plafond de l'étable. La position des vaches est déterminée par triangulation avec une précision de 50 cm selon le fournisseur. Dans la ferme expérimentale d'INRAE, la précision est de 16 cm [81].

CowView permet d'obtenir la position de chaque vache toutes les secondes. Selon la localisation de la vache dans l'enclos, 3 activités sont déduites :

— *mange* si la vache est localisée près d'une mangeoire,

— *repos* si elle est localisée dans une logette,

— *dans l'allée* si la vache est localisée dans les couloirs (toute zone en dehors de la zone d'alimentation et de la zone de repos).

Pour chaque vache et pour chaque heure, le temps passé (exprimé en secondes) dans chacune de ces activités est calculé. Les données ont été enregistrées pendant 1 à 6 mois selon les fermes (table 1.1).

Pour calculer un niveau d'activité à partir des 3 activités de base, nous avons utilisé la méthode proposée par [120]. Nous avons réalisé une Analyse Factorielle des Correspondances (AFC) avec chaque heure du jour de chaque jour observé comme une observation et le temps passé par toutes les vaches dans chacune des 3 activités comme variables [117]. Par construction, le premier axe obtenu par l'AFC exprime le maximum de variabilité entre heures de la journée. Cette analyse a été conduite pour la ferme INRAE, A et D. A noter que pour la ferme A nous disposions d'un jeu de données antérieur obtenu sur 5 mois, qui a été utilisé pour l'AFC [120]. L'AFC

2. Bonen, Germany, `http://www.gea-cowview`. Corn

TABLE 1.1 – Caractéristiques des fermes dont sont issues les données utilisées dans la thèse.

Ferme	**Jeu de données**	**Eclairage**	**Distribution de l'aliment**	**Traites**	**Nombre de vaches**	**Durée du suivi (mois)**
Ferme INRAE	1	naturel	8 h puis repousse à 13h, 16h30, 20 h	2 fois / jour	28	3
Ferme INRAE	2	naturel	8 h et 14 h	2 fois / jour	28	6
Ferme A	3	artificiel avec diminution la nuit	le matin puis repoussée régulièrement	robot	10	1
Ferme B	3	naturel	2 fois / jour	2 fois / jour	15	1
Ferme C	3	naturel	le matin puis repoussée après chaque traite	3 fois / jour	15	1
Ferme D	4	naturel	le matin puis repoussée régulièrement	robot	>300	12

n'a pas été conduite pour les fermes B et C car nous disposions de trop petits jeux de données.

Les 3 activités de base obtiennent systématiquement des poids permettant de les classer par ordre de niveau d'activité qui serait attribué intuitivement : poids le plus faible pour l'activité *repos* et le plus élevé pour l'activité *mange* (Table 1.2). Les fermes B et C n'ont pas de poids car l'AFC n'a pas été conduite, donc la moyenne des poids a été calculée et appliquée pour tous les jeux de données. En attribuant

FIGURE 1.7 – Image d'une vache de la ferme INRAE avec son collier CowView. Crédit photo : Nicolas Wagner.

TABLE 1.2 – Poids des activités obtenus avec l'AFC selon les fermes.

Activité	**Repos**	**Dans une allée**	**Mange**
Ferme INRAE	- 0,34	0,29	0,52
Ferme A	- 0,15	0,12	0,34
Ferme D	- 0,19	0,06	0,41
Moyenne	- 0,23	0,16	0,42

le poids respectif obtenu par chacune des 3 activités au temps passé par une vache dans chaque activité pendant une heure donnée, on obtient son niveau d'activité au cours de cette heure. Ainsi, chaque vache est représentée par une série temporelle qui correspond à son niveau d'activité variant d'heure en heure (exemple illustré en Figure 1.8).

2.1.3 Suivi de l'état des vaches

Chaque jour les soigneurs notent dans un cahier d'élevage les évènements pouvant affecter les animaux : maladie, accident, événement spécifique lié à la reproduction (œstrus [3], vêlage), tout événement pouvant perturber les animaux (changement de parc, mélange avec d'autres animaux, incident mécanique pouvant affecter l'heure

3. l'œstrus – ou chaleurs - correspond à la période pendant laquelle la vache est fécondante. Elles se traduisent par un comportement d'agitation dans un premier temps puis d'acceptation d'être chevauché par un autre animal dans un second temps.

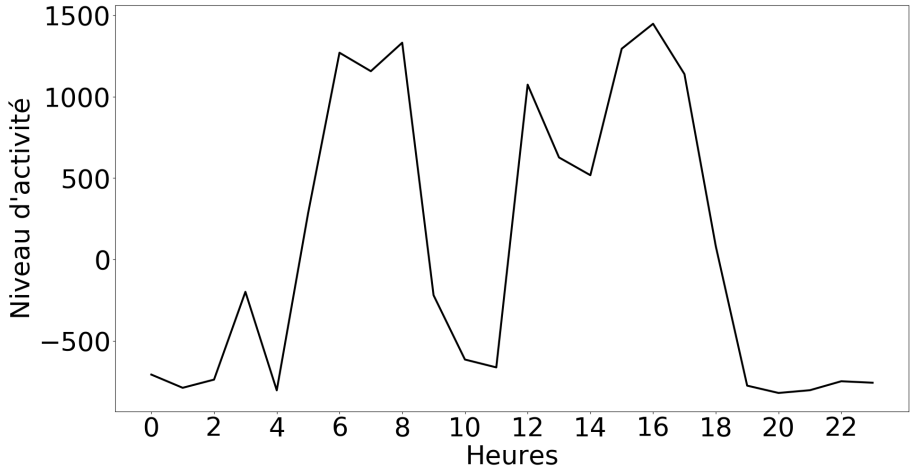


FIGURE 1.8 – Exemple d'une série temporelle représentant le rythme d'activité sur 24 h d'une vache de la ferme INRAE.

des repas ou de la traite, etc.). De plus, pour certains jeux de données, nous disposions d'éléments d'information supplémentaires (cf. infra) : température corporelle, pH du rumen, niveau de progestérone dans le lait. Nous avons groupé les événements par classe (voir table 1.3. Ainsi, pour chaque jour il est possible de définir l'état de l'animal comme *normal* ou au contraire affecté par un événement.

2.1.4 Jeux de données

Pour cette thèse j'ai eu accès à quatre jeux de données :

1. un premier jeu de données issu d'une expérimentation conduite sur la ferme d'INRAE incluant 28 vaches sur 3 mois. La moitié des vaches ont été nourries avec un régime riche en amidon pendant un mois dans le but d'engendrer une acidose ruminale subclinique (Sub-Acute Ruminal Acidosis en anglais, soit SARA)[4]. Un capteur de pH (eCow bolus, Exeter, UK) a été placé dans leur rumen afin de détecter les chutes de pH. Selon la méthode proposée par [121], nous avons normalisé les valeurs du pH ruminal de chaque vache pour prendre en compte la variabilité inter-individuelle, la dérive du capteur et le bruit associé aux données ; puis nous avons considéré qu'une vache était en acidose lorsque le pH ruminal normalisé (NpH) diminuait d'au moins 0,3 pendant plus de 50 min/j et que l'écart-type quotidien en NpH était supérieur à 0,2 ou que l'intervalle quotidien en NpH était supérieur à 0,8.

4. état caractérisé par un pH du contenu du rumen plus bas que la normale mais sans signes cliniques

TABLE 1.3 – Nombre d'évènements selon les jeux de données.

Évènement	**Jeu de données**			
	1	**2**	**3**	**4**
Accidents[1]	0	0	0	10
Vêlages	0	9	0	171
Œstrus[2]	7	41	29	257
Boiteries	16	4	0	114
Mammites	3	9	0	32
Autres maladies[3]	8	10	0	66
Injection LPS[4]	0	27	0	0
Acidoses[5]	271	0	0	0
Changement parc	0	63	0	0
Autres pertubations[6]	667	145	0	12070

[1] blessures, rétention placentaire, lacérations vaginales

[2] détectées par l'éleveur ou par le profile progestérone dans le jeu 3

[3] diarrhée, coliques, ingestion d'un corps étranger, avortement, kyste ovarien, baisse d'appétit, apathie, animal qui s'isole, perte de poids, chute de production laitière, fièvre de lait, acétonémie, affection respiratoire, autre maladie infectieuse (grippe, ...)

[4] injections LPS injectées dans la glande mammaire

[5] détectées par analyse du pH ruminal

[6] interventions, e.g. vaccinations, synchronisation des chaleurs, vermifugation, parage des onglons, changement de parc

2. un deuxième jeu de données issu d'une expérimentation conduite sur la ferme d'INRAE durant laquelle les vaches ont reçu une injection de lipopolysaccharides (LPS)[5] dans la mamelle afin de provoquer une inflammation. Le jeu de données est constitué de 28 vaches enregistrées sur 6 mois. La température corporelle a été enregistrée pendant 24 h lorsque les vaches ont reçu le LPS afin de vérifier qu'elles y réagissaient.
3. un troisième jeu de données issues des fermes commerciales A, B et C avec

5. molécules de la membrane externe des bactéries Gram-négatives, qui provoquent une inflammation

un total de 40 vaches sur un mois. Les œstrus y ont été détectés à partir du profil des concentrations de progestérone dans le lait des vaches. Une chute de la progestérone à des niveaux inférieurs à 5 vg/mL pendant plusieurs jours est concomitant de l'œstrus.

4. un jeu de données issue de la ferme commerciale D avec plus de 300 vaches sur 12 mois.

Ces jeux de données seront appelés jeu 1, 2, 3 et 4 par la suite. La liste des anomalies de chaque jeu de données est visible en Table 1.3.

2.2 Caractéristiques des jeux de données

L'hypothèse sous-jacente est que le rythme est modifié quand la vache est perturbée à cause d'une maladie, d'un événement lié à la reproduction (œstrus, vêlage) ou d'une situation stressante. Or il existe d'autres sources de variation du rythme circadien : variations dues à la saison, variations entre fermes du fait de routines différentes (heures de distribution des repas ou de traite en particulier), variations entre individus et variations entre jours. Aucun de nos jeux de données ne couvrent plusieurs années sur les mêmes animaux. Aussi ne nous est-il pas possible d'identifier la part des variations observées liée à la saison. Nous illustrerons ici uniquement les variations liées à la ferme, à l'animal ou au jour ainsi qu'à la nature de l'événement qui semble avoir provoqué l'anomalie.

2.2.1 Variations entre fermes

Les figures 1.9 et 1.10 illustrent les variations circadiennes d'activité moyenne des vaches des quatre jeux de données pour des jours où aucun événement n'a été noté (jours dits normaux). La première remarque est que, excepté pour le jeu de données 3, l'alternance jour/nuit se distingue aisément. Les courbes des jeux 1 et 2 sont très similaires, ce qui est normal car les deux jeux de données proviennent de la même ferme (ferme INRAE). Elles ont deux pics d'activité séparés de 9 heures, un premier de 05 :00 à 07 :00 et un second de 14 :00 à 16 :00. Ces 2 pics semblent correspondre aux heures de traite (06 :00 et 15 :00 l'hiver et 05 :00 et 14 :00 l'été). La courbe du jeu de données 4 est différente, les variations d'activité sont moins marquées. Cela peut provenir du fait que les vaches sont traites avec un robot et n'ont pas les mêmes heures de traite, chacune choisissant quand aller au robot. De plus l'alimentation est distribuée le matin mais repoussée près des vaches régulièrement dans la journée de sorte qu'elle est toujours facilement accessible. Cependant, on retrouve les deux pics d'activités séparés de neuf heures, un à 08 :00 (vraisemblablement lorsque l'aliment est distribué) et l'autre à 17 :00. Compte-tenu des conditions de cette ferme où les vaches sont plus libres de décider quand se faire traire et manger, elles semblent

retrouver un rythme proche de ce qui est observé dans des conditions naturelles [67, 88]. Les courbes moyennes d'activité des fermes A, B et C du jeu de données 3 sont différentes des autres et n'ont pas le même nombre de pic d'activité. La ferme A a un seul pic d'activité, la ferme B en a deux (dont un la nuit) et la ferme C en a trois. De plus, le rythme d'activité des fermes A, B et C est moins marqué que celui des autres fermes.

Il existe donc une forte variabilité entre les fermes, de sorte qu'il n'est pas possible de décrire un rythme circadien type quelle que soit la ferme.

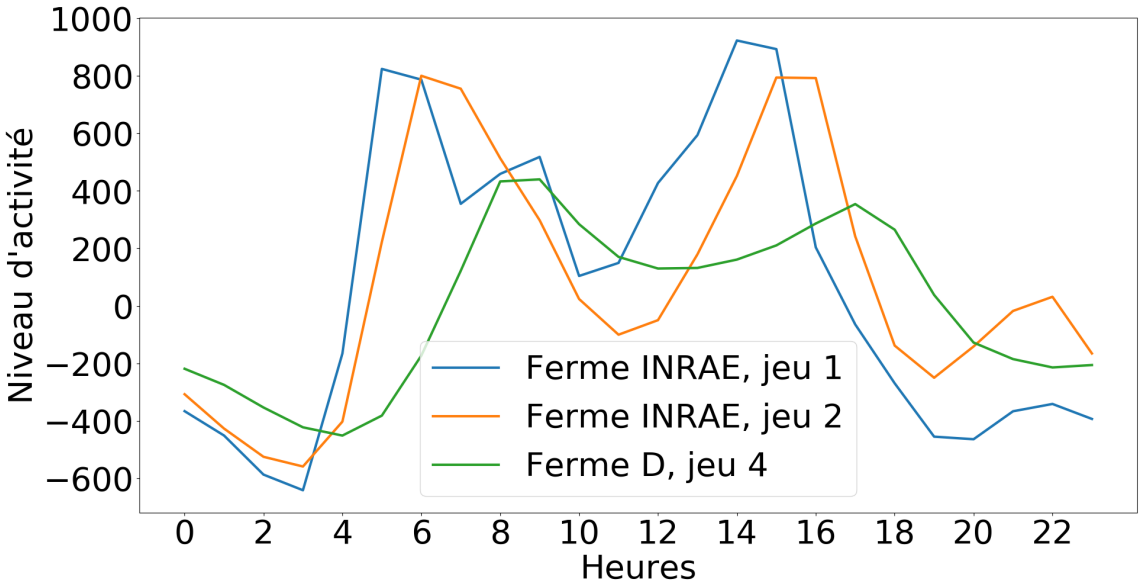


FIGURE 1.9 – Niveaux d'activités moyens des jours normaux des fermes INRAE et D

2.2.2 Variations entre individus

La figure 1.11 illustre les variations du niveau moyen d'activité de 3 vaches de la ferme D pour l'ensemble des jours où aucun événement n'a été noté pour ces vaches. L'alternance jour /nuit est visible pour les 3 vaches : pour une vache donnée, le niveau d'activité est plus élevé entre 07 :00 et 19 :00. Dans les 3 courbes on retrouve également les 2 pics d'activité de la journée. Cependant, les 3 courbes présentent certaines différences. La vache 1 a un rythme moyen très marqué : la différence entre les pics de la journée et l'activité faible de la nuit est très importante. Les vaches 2 et 3 ont une alternance jour/nuit moins marquée : le niveau d'activité de la vache 2 s'élève moins le jour et celui de la vache 3 diminue moins la nuit.

Ainsi, au sein d'une même ferme, il semble difficile de raisonner avec un rythme type quelles que soient les vaches.

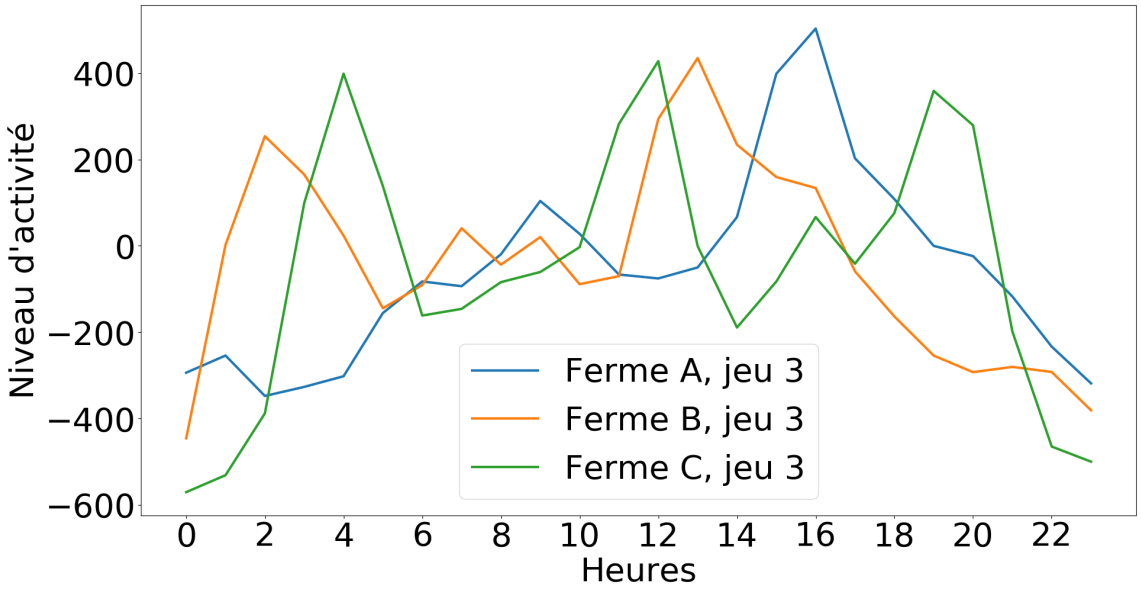


FIGURE 1.10 – Niveaux d'activités moyens des jours normaux des des fermes A, B et C du jeu de données 3

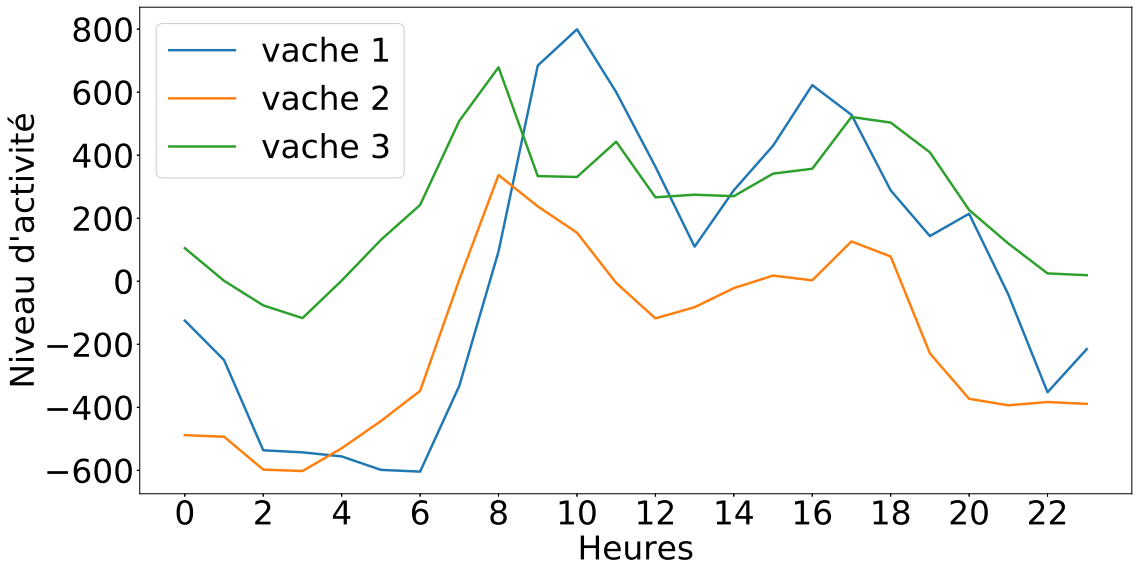


FIGURE 1.11 – Niveaux d'activités moyens des jours normaux de 3 vaches différentes de la ferme D

2.2.3 Variations entre jours

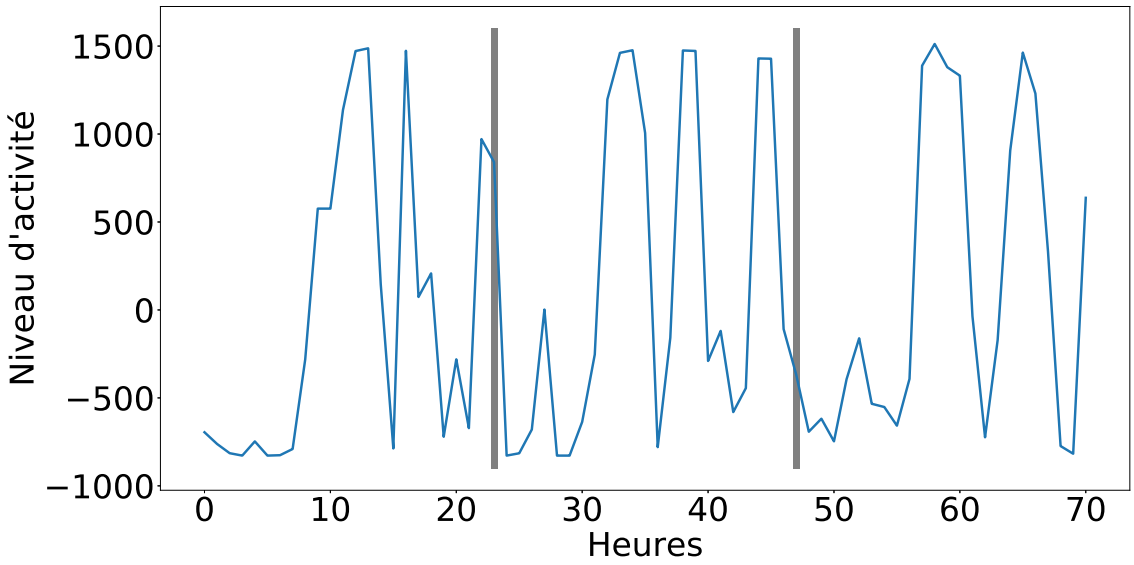
La figure figure 1.12 illustre les variations de niveau d'activité de deux vaches issues de la ferme D sur plusieurs jours où aucun événement n'a été noté. La vache représentée sur la figure (a) a un rythme globalement identique au fil des jours alors que la vache représentée sur la figure (b) peut voir son activité notablement changer. Par exemple, le jour 2 l'activité de cette vache ne ressemble ni au jour précédent, ni au jour suivant.

En l'absence d'événement extérieur – tout du moins en l'absence d'événement relevé par les soigneurs -, l'activité d'une vache peut varier spontanément d'un jour sur l'autre. Il nous faudra donc tenir compte de ce bruit qui ne semble lié à aucun facteur identifié.

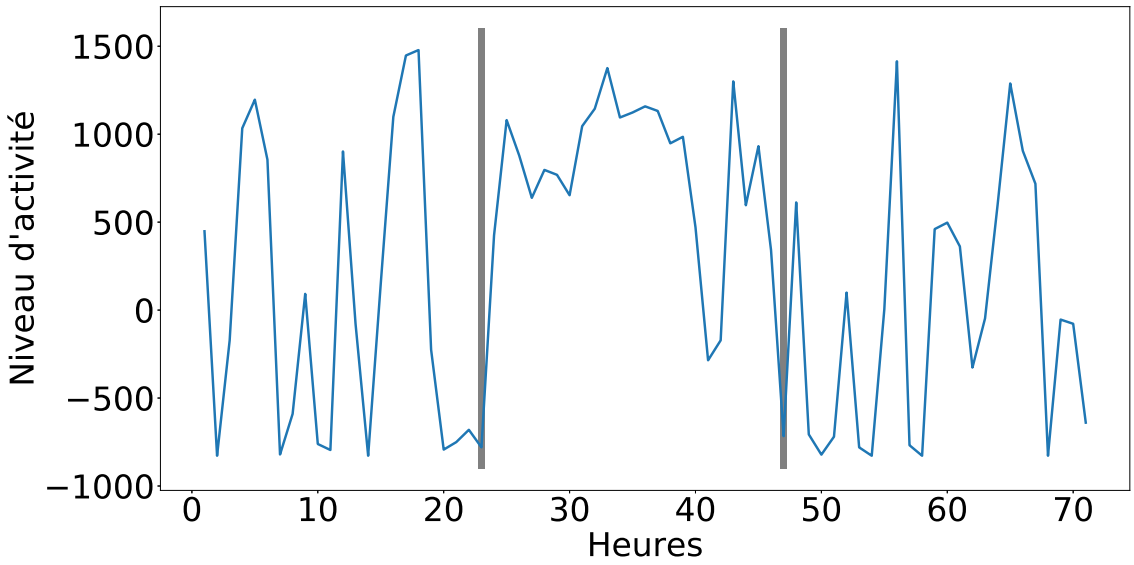
2.2.4 Variations entre états de l'animal

La figure 1.13 illustre les variations d'activité moyenne des vaches de la ferme D pour les principaux événements : œstrus, boiterie, mammite. A noter que l'on raisonne ici en jour * vache, une vache pouvant être atteinte de mammite un jour, un œstrus un autre et normale à d'autres moments. L'activité moyenne des vaches ayant une mammite se distingue de l'activité moyenne des vaches sans anomalie : les vaches atteintes de mammite sont globalement plus actives durant la période diurne. Les courbes d'activité des vaches en œstrus ressemblent à celles de vaches lors des jours dits normaux mais les vaches sont un peu plus agitées durant la nuit (en particulier entre 00 :00 et 06 :00). Les vaches avec boiterie ont un rythme proche de celui des vaches sans anomalie avec toutefois un niveau d'activité plus faible en milieu de journée. Selon le type de problème, le rythme d'activité change de manière plus ou moins importante.

Le problème auquel je fais face peut se résumer ainsi : comment identifier les anomalies du rythme circadien d'activité de vaches dues à des événements perturbateurs tels que une maladie, un œstrus, un vêlage ou un stress, des variations spontanées qu'il s'agisse de variations liées à la saison, à l'organisation du travail dans les fermes (heure d'alimentation, de traite et d'éclairage), à l'animal ou encore de variations aléatoires (bruit).



(a)



(b)

FIGURE 1.12 – Deux exemples de niveaux d'activité normaux sur 72 heures de deux vaches différentes du jeu de données 4 ; les lignes verticales délimitent les jours

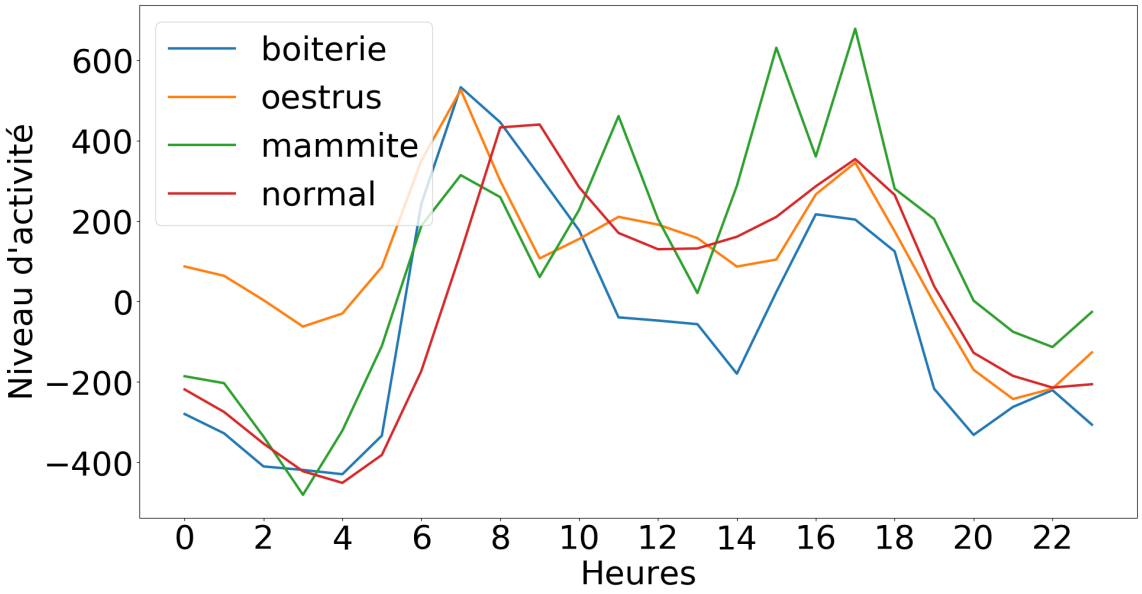


FIGURE 1.13 – Niveaux d'activités moyens normal, boiterie, mammite et œstrus de la ferme D

Chapitre 2

État de l'art

Ce chapitre a pour but d'expliquer les bases des problèmes liés aux séries temporelles. Il a aussi pour but d'introduire la logique floue qui ne concerne pas que les séries temporelles mais qui a été utilisée dans le cadre de cette thèse.

Séries temporelles Une série temporelle est une collection de valeurs temporellement ordonnées. Les séries temporelles sont présentes dans pratiquement tous les domaines scientifiques et sont acquises par la mesure d'un ou plusieurs phénomènes à travers le temps [33]. Elles sont utilisées par exemple en économie pour analyser un marché, en météorologie pour prédire le temps futur, en médecine pour détecter des anomalies dans le rythme cardiaque. Elles peuvent aussi être utilisées dans le domaine de l'audio, dans l'industrie en général (étude de capteurs) et plus particulièrement en biologie. Il est possible de regrouper toutes les problématiques liées aux séries temporelles en différentes grandes catégories :

- **l'analyse** qui consiste à étudier une série temporelle et comprendre toutes les caractéristiques qui la composent (tendance, cycles, fréquences, etc),
- **la prédiction** qui consiste à prévoir les futures valeurs que va prendre une série en se basant sur les anciennes valeurs,
- **la classification** qui consiste soit à regrouper les séries qui partagent les mêmes caractéristiques (classification non-supervisée ou clustering), soit à déterminer si une série appartient à un groupe ou à un autre (classification supervisée),
- **la détection d'anomalies** qui consiste à trouver des valeurs anormales au sein d'une série, soit à détecter si une série est anormale par rapport aux autres.

Selon la nature du problème à traiter, une ou plusieurs de ces catégories peuvent être utilisées.

Logique floue La logique floue est une extension de la logique classique, permettant de prendre en compte l'incertitude liée aux jeux de données. Dans une acquisition réelle de données, il arrive qu'il soit impossible de déterminer avec exactitude l'état des données. Par exemple, pour des données médicales, l'état du patient peut varier, le patient peut être plus ou moins malade. La logique floue permet de prendre ces états incertains en considération et ainsi améliorer les performances des algorithmes de décision.

Dans la suite de ce chapitre, après une section donnant les définitions de base des séries temporelles, chaque problématique liée au séries temporelles sera expliquée. Une dernière section sera dédiée à la logique floue.

1 Définitions

Une série temporelle est une collection de valeurs ordonnées avec le temps. Elle peut être univariée, c'est à dire qu'à chaque instant, il n'y a qu'une seule valeur, ou multivariée avec une séquence de valeurs données pour chaque instant.

Definition 1. Une série temporelle univariée $\boldsymbol{x} = [x_1, x_2, ..., x_n]$, $x_i \in \mathbb{R}$ est une collection de n valeurs réelles temporellement ordonnées.

Definition 2. Une série temporelle multivariée $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m]$ de dimension m est une séquence de m séries univariées $\boldsymbol{x}_i \in \mathbb{R}^n$.

Dans la suite de cette thèse, seules les séries temporelles univariées seront abordées.

Un jeu de données est un ensemble qui regroupe des données qui peuvent être de divers types comme par exemple des séries temporelles. Dans le cadre de la classification, chacune de ces données est associée à une étiquette, c'est à dire une classe d'appartenance. Par exemple, un jeu de données peut regrouper des séries temporelles qui représentent des électrocardiogrammes et chaque séries est associée à une étiquette, soit *cœur sain*, soit *cœur malade*.

Definition 3. Un jeu de données $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ (ou $\mathcal{D} = \{(X_i, y_i)\}$ si multivarié), est constitué d'un ensemble couples composés d'une séries temporelles $\boldsymbol{x}_i$ (ou X_i si multivarié) et d'une classe associée y_i.

Definition 4. Pour un jeu de données $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ (ou $\mathcal{D} = \{(X_i, y_i)\}$ si multivarié) est déterminé un ensemble $\mathcal{C} \subset \mathbb{N}$ de c classes possibles avec $y_i \in \mathcal{C}$.

2 Analyse

Souvent, le premier objectif d'une étude de séries temporelles est son analyse. L'analyse consiste à déterminer toutes les caractéristiques qui composent une série temporelle afin d'en extraire des connaissances telles que l'aspect cyclique (ou saisonnier) et la tendance. Une série est cyclique quand on observe un phénomène qui se répète à des intervalles réguliers. On dit qu'une série possède une tendance quand ses valeurs tendent à augmenter ou à diminuer avec le temps. Une série peut avoir les deux phénomènes de combinés ou aucun des deux. Il existe également les transformations de Fourier qui permettent d'extraire le spectre fréquentiel d'une séries temporelles. Ces caractéristiques peuvent être très utiles pour expliquer certains phénomènes scientifiques liés aux données.

2.1 Composante cyclique et tendance

Une série temporelle peut être composée de deux grands phénomènes : la tendance (orientation générale d'une série) et le cycle (schéma qui se répète). Ces phénomènes peuvent, par exemple, expliquer la tendance de valeurs boursières ou expliquer la variation des prix de l'électricité [86].

Pour extraire ces composantes il existe plusieurs techniques. Tout d'abord, une tendance peut être calculée par régression [3, 63]. Le plus simple est la régression linéaire.

Definition 5. Soit $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ une série temporelle. Une régression linéaire aura pour but de calculer une estimation linéaire de $\boldsymbol{x}$: $x_t \simeq at + b$ pour $t \in [1...n]$.

Pour résoudre cette équation, il existe la méthode des moindres carrés :

$$a = \frac{Cov(\boldsymbol{x}, \boldsymbol{t})}{Var(\boldsymbol{x}))} \quad \text{et} \quad b = \bar{\boldsymbol{x}} - a\bar{\boldsymbol{t}}, \tag{2.1}$$

avec $Cov(\boldsymbol{x}, \boldsymbol{t})$ la covariance entre $\boldsymbol{x}$ et $\boldsymbol{t}$, $Var(\boldsymbol{x})$ la variance de $\boldsymbol{x}$ ainsi que $\bar{\boldsymbol{x}}$ et $\bar{\boldsymbol{t}}$ respectivement la moyenne des $\boldsymbol{x}$ et la moyenne des $\boldsymbol{t}$.

Une tendance peut également être calculée en supprimant le cycle d'une série temporelle grâce à une moyenne mobile. Ensuite, il suffit de soustraire à la série originale le résultat de la moyenne mobile pour obtenir la composante saisonnière.

Definition 6. Soit $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ une série temporelle. Une moyenne mobile d'ordre $m_1 + m_2 + 1$ modifie chaque élément de $x_i \in \boldsymbol{x}$ avec la formule :

$$x_t^* = \frac{1}{m_1 + m_2 + 1} \sum_{i=-m_1}^{m_2} x_{t+i}. \tag{2.2}$$

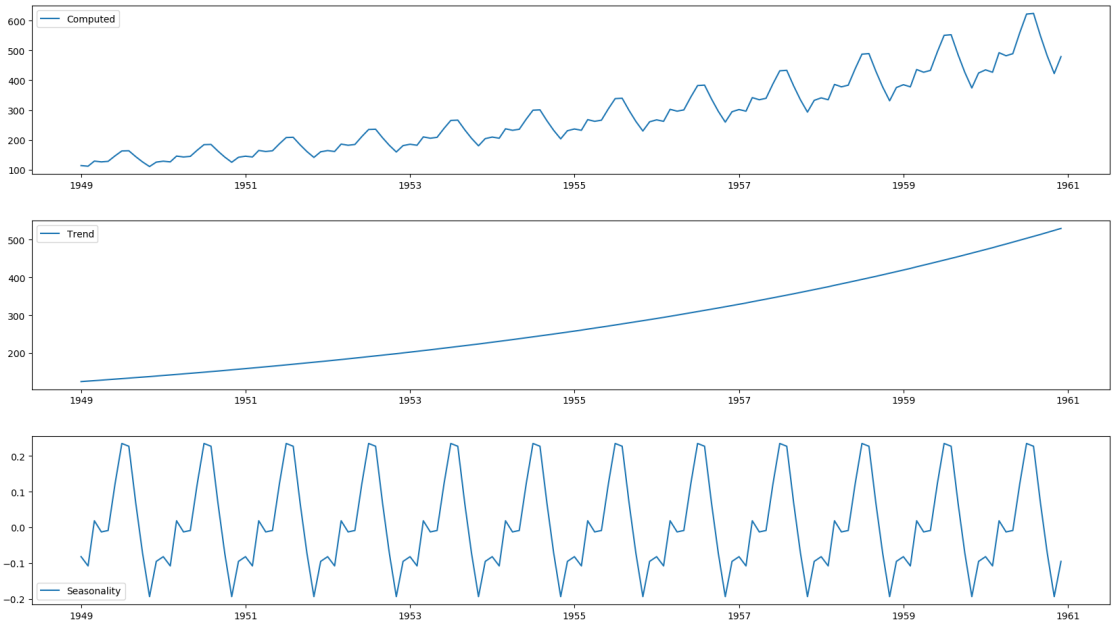


FIGURE 2.1 – Exemple d'une décomposition du phénomène cyclique et de la tendance sur le jeu de données Air Passengers.

Si on applique une moyenne mobile d'ordre $m_1 + m_2 + 1$ sur une série alors toute composante cyclique de périodicité $m_1 + m_2 + 1$ sera supprimée. Il faut également noter que la tendance est conservée.

Un exemple concret est montré en figure 2.1 où la tendance et la saisonnalité ont été extraite dans le jeu de donnée *Air Passengers*[1].

2.2 Transformées de Fourier

Selon le théorème des séries de Fourier [9], tout signal périodique $y(t)$ de fréquence f peut être décomposé en une somme infinie de fonctions cosinus. La transformation de Fourier permet de généraliser le concept à toute fonction qui peut être périodique ou non. Il est donc possible de définir un signal $y(t)$ comme :

$$y(t) = \sum_{n=-\infty}^{+\infty} A_n cos(2\pi n f t + \rho_n), \tag{2.3}$$

avec :

— n le rang de la composante cosinusoïdale,

1. `https://www.kaggle.com/rakannimer/air-passengers`

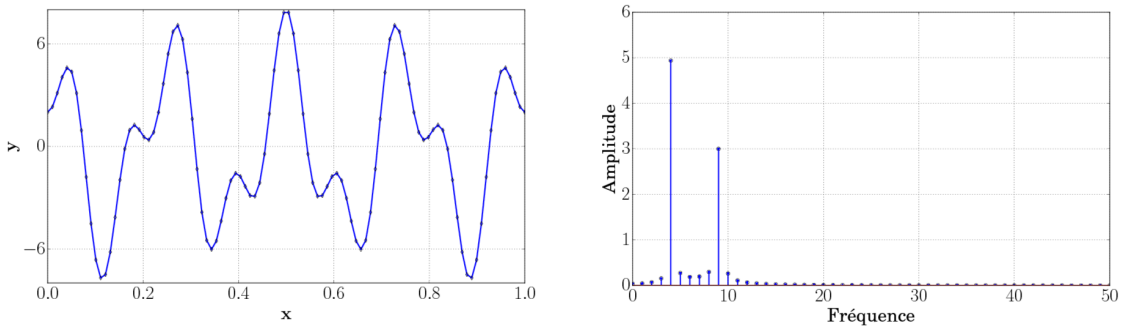


FIGURE 2.2 – Exemple d'une décomposition de Fourier ; crédit : `http://culturesciencesphysique.ens-lyon.fr/ressource/signal-reseau-velo.xml`.

— A_n l'amplitude de la composante cosinusoïdale n,
— ρ_n la phase de la composante cosinusoïdale n.

Il s'agit donc de transformer un signal temporel en un signal fréquentiel (voir Figure 2.2). Quand $n = 1$, il s'agit de la fondamentale, c'est à dire la composante principale qui a la même fréquence f que le signal $y(t)$ d'origine. Quand $n = 2$, il s'agit de la deuxième composante qui a une fréquence égale à $2f$ et ainsi de suite. Chaque composante est représentée par une harmonique complexe $h_n \in \mathbb{C}$ avec n le rang de l'harmonique. L'harmonique de rang 0 correspond à la valeur moyenne de la série et l'harmonique de rang 1 correspond à la fondamentale représentant la composante cosinus de fréquence f. Il faut noter que lorsque les valeurs de la série temporelle sont réelles, les harmoniques négatives sont symétriques avec les harmoniques positives.

Dans les jeux de données, les séries temporelles sont discrètes, c'est à dire qu'elles ne sont pas représentées par un signal continu mais par une suite de valeurs réelles appelées échantillons. Ces échantillons sont tous espacés par un même laps de temps Δt. On appelle fréquence d'échantillonnage, le nombre d'échantillons par unité de fréquence.

Pour calculer les harmoniques d'un signal, il existe un algorithme de transformation rapide des séries temporelles, le Fast Fourier Transform (FFT) [16]. FFT travaille de manière discrète et par conséquent il ne génère pas un nombre infini d'harmoniques. Selon le théorème de Nyquist-Shannon, pour pouvoir capter un signal de fréquence f, il faut au moins deux échantillons par période. Ceci implique que pour un signal réel avec e échantillons, il ne peut pas y avoir plus de $\lceil \frac{e-1}{2} \rceil + 1$ harmoniques (h_0 plus $\lceil \frac{e-1}{2} \rceil$ harmoniques positives qui sont égales aux harmoniques

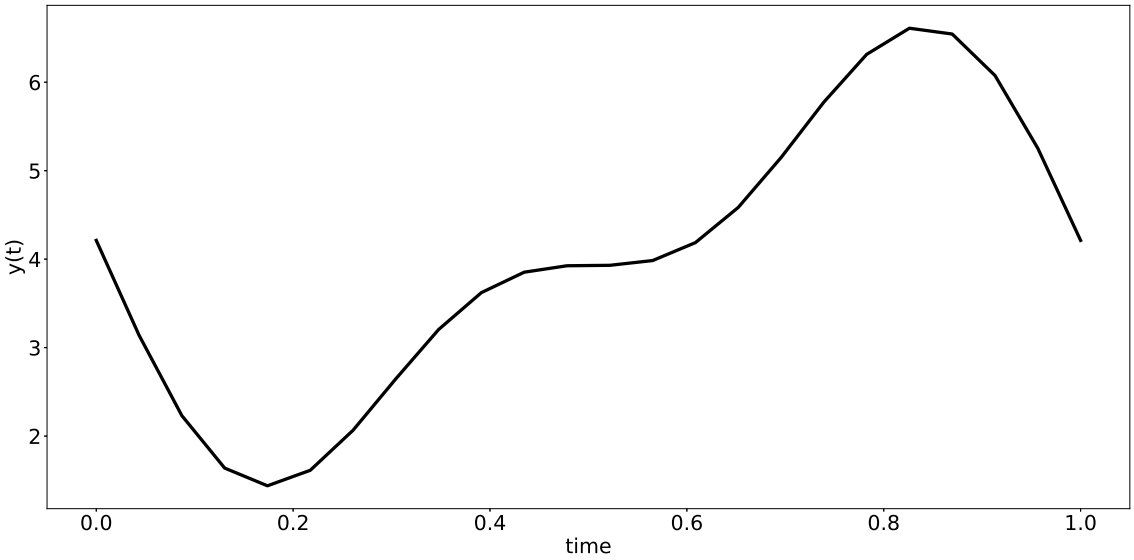


FIGURE 2.3 – Exemple d'un signal périodique $y(t) = 2cos(2\pi t + 1.5) + cos(2\pi 2t + 1.5) + 4$ de 24 échantillons.

négatives). La formule de $y(t)$ devient donc :

$$y(t) = \sum_{n=-\lceil \frac{e-1}{2} \rceil}^{+\lceil \frac{e-1}{2} \rceil} |h_n| cos(2\pi n f t + arg(h_n)), \tag{2.4}$$

avec :

— $|h_n|$ le module de l'harmonique de rang n,
— $arg(h_n)$ l'argument de l'harmonique de rang n.

Par exemple, pour un signal discret de 24 échantillons sur une période suivant la formule : $y(t) = 2cos(2\pi t + 1.5) + cos(2\pi 2t + 1.5) + 4$ (cf. figure 2.3), l'application de l'algorithm FFT donnera les harmoniques illustrées par la figure 2.4. L'harmonique 0 est à environ 4, ce qui correspond bien à la valeur moyenne du signal original. Les harmoniques 1 et 2 sont respectivement environ égales à 2 et 1, ce qui correspond aux deux composantes sinusoïdales de $y(t)$. Il est à noter que les harmoniques 0,1 et 2 ne sont pas tout à fait égales à 4,2 et 1 et que les harmoniques de rangs supérieurs ne sont pas égales à 0 alors que dans la formule originale, ces composantes n'existent pas. Ceci vient du fait de la discrétisation, elle crée un léger bruit dans les données et engendre une infime incertitude dans les résultats. Plus le taux d'échantillonnage est élevé, plus l'incertitude sera négligeable.

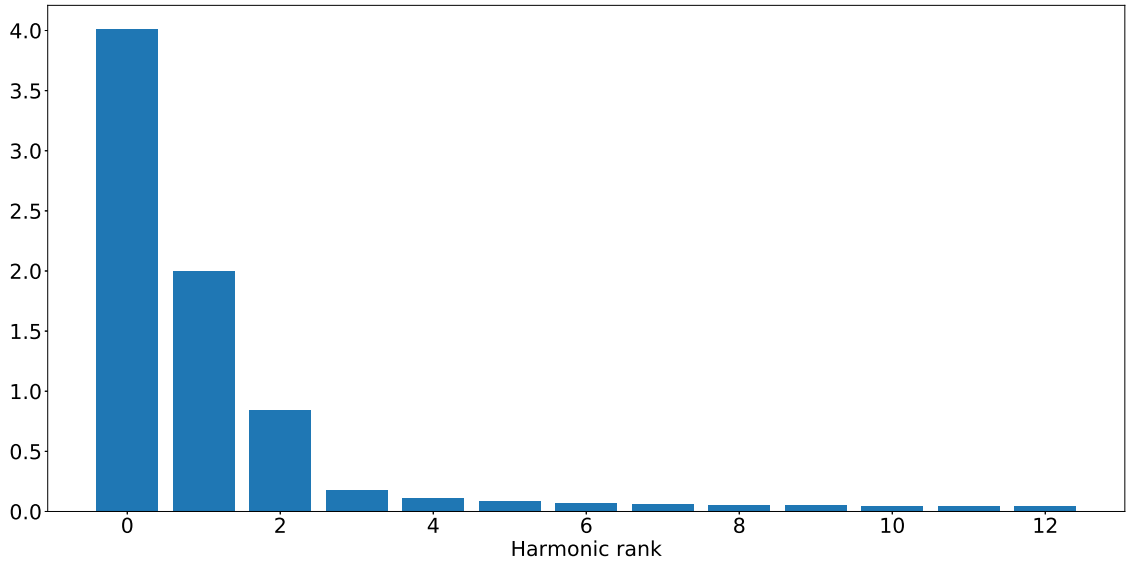


FIGURE 2.4 – Histogramme des harmoniques de la figure 2.3.

3 Prédiction

La nécessité de prédire les futures valeurs d'une série temporelle est très répandue. Par exemple, en analyse financière, il est impératif de prédire l'évolution des cours du marché, pour une entreprise, prévoir ses stocks en avance est crucial, etc.

Definition 7. La prédiction consiste à déterminer, à partir d'une série temporelle $\boldsymbol{x} = [x_1, x_2, ..., x_n]$, les l futures valeurs $x_{n+1}, x_{n+2}, ..., x_{n+l}$ que va prendre la séries aux instants $n+1, n+2, ..., n+l$.

Les méthodes de prédiction consistent à construire le modèle de la série temporelle afin d'être capable de calculer les futures valeurs. Il existe plusieurs méthodes dans la littérature. Les méthodes mathématiques se basent sur un principe de régression et les méthodes de machine learning construisent le modèle à partir d'une base de connaissances. Les principales méthodes sont décrites dans la suite cette section.

3.1 Modèles mathématiques

Les modèles mathématiques à régression partent du principe que la valeur d'un point à un instant t dépend des valeurs des points précédents. Ainsi les modèles mathématiques sont des modèles autorégressifs. Il en existe plusieurs mais les deux principaux sont le AutoRegressive Moving Average (ARMA) et le AutoRegressive Integrated Moving Average (ARIMA).

ARMA

Le modèle ARMA [39] est un modèle régressif qui est composé d'un processus AutoRégressif (AR) et d'un processus à moyenne mobile (MA) :

$$x_t = \epsilon_t + \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}, \tag{2.5}$$

avec ϵ_t le bruit à l'instant t. Les paramètres ϕ et θ sont respectivement les paramètres à prédire du modèle AR et du modèle MA, ils représentent les poids des p précédentes valeurs de x_t et les poids des q précédentes valeurs du poids ϵ_t. On dit qu'un modèle $ARMA(p, q)$ est d'ordre p et q. Un modèle $ARMA(p, 0)$ est équivalant à un modèle AR d'ordre p et un modèle $ARMA(0, q)$ est équivalent à un modèle MA d'ordre q.

Il faut noter que si on introduit l'opérateur retard L^i avec $L^i x_t = x_{t-i}$, alors on peut réécrire l'équation comme ceci :

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) x_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t. \tag{2.6}$$

Le modèle ARMA impose deux hypothèses sur les données. Il faut que la série temporelle suive un modèle linéaire et qu'elle soit stationnaire. Une série temporelle est dite stationnaire si ses valeurs peuvent être considérés comme des processus indépendants :

Definition 8. Soit $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ une série temporelle. Elle est dite stationnaire si pour tout $p > 0$, $[x_1, x_2, ..., x_p]$ suit la même loi que $[x_2, x_3, ..., x_{p+1}]$.

Une série qui possède par exemple une tendance n'est donc pas stationnaire. Pour cela, le modèle ARIMA tente de prendre cette tendance en compte.

ARIMA

Le modèle ARIMA prend en compte la non stationnarité de la série en appliquant une différentiation sur la série (une dérivée) :

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right) (1 - L)^d x_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t, \tag{2.7}$$

avec d l'ordre de différentiation et L l'opérateur de retard : $L^k x_t = x_{t-k}$. On dit qu'un modèle $ARIMA(p, d, q)$ est d'ordre $ARMA(p, q)$ et d'ordre de différentiation d.

Ce modèle permet de supprimer la tendance de la série. Cependant il reste limité à une régression linéaire et ne gère pas les séries périodiques (pour cela il existe les modèles SARIMA).

Le point faible de ce type de méthodes est qu'il est nécessaire de faire une étude minutieuse de la série afin de déterminer les paramètres p, q et éventuellement d. Il faut s'assurer que la série soit bien stationnaire, qu'il n'y plus ni cycle ni tendance et parfois cela demande d'utiliser des fonctions pré-transformation telles que des fonctions logarithmes par exemple.

3.2 K-NN pour la régression

La méthode des k plus proches voisins (k-NN) [18] est une méthode de classification simple et souvent utilisée. Elle est dite *lazy* car elle ne nécessite aucun apprentissage. Pour la prédiction il existe une variante adaptée pour la régression (k-NNR). Pour estimer la future valeur d'une série temporelle, l'algorithme cherche dans la base de connaissances les k plus proches séries. La prédiction est la moyenne des valeurs observées par les k voisins. L'algorithme est facilement généralisable pour prédire plusieurs valeurs. K-NNR est l'un des algorithmes de prédiction les plus utilisés [132]. Par exemple, [7] l'utilise pour réaliser des prédictions financières.

Cette méthode a deux gros avantages : elle est simple à mettre en œuvre et donne souvent de bons résultats. Cependant, elle a un défaut majeur : pour chaque instance du jeu de données à tester, l'algorithme doit calculer ses plus proches voisins. Si la taille du jeu de données d'apprentissage est importante, le résultat d'un test peut être relativement long.

3.3 Réseaux de neurones

Les réseaux de neurones sont de plus en plus utilisés pour résoudre des problèmes liés aux séries temporelles tels que la prédiction. On parle de deep learning quand la méthode consiste en un réseau de plusieurs couches successives de neurones.

3.3.1 Neurone

Un neurone (Figure 2.5) p est composé de n entrées $x_j^i \in \mathbb{R}$ et d'une sortie $\hat{y}_p^i$. Dans le cadre des séries temporelles, les entrées peuvent représenter les valeurs successives d'une série temporelle $\boldsymbol{x}_i = [x_1^i, x_2^i, ..., x_n^i]$. La sortie $\hat{y}^i$ peut représenter la classe de la série ou encore la valeur suivante de la série pour un problème de prédiction ($\hat{y}_p^i = x_{n+1}^i$). Le calcul de la sortie $\hat{y}_p^i$ en fonction des entrées s'effectue en deux étapes. Premièrement, les entrées sont additionnées entre elles après être pondérées par un poids w_j^p (qui est associé à l'entrée x_j^i). Pour éviter le phénomène de sur-apprentissage, un biais est ajouté à cette somme. Pour simplifier les calculs, le biais est ajouté en incorporant une entrée $x_0^i = 1$ avec un poids w_0^p. Le résultat de cette somme est envoyé à une fonction d'activation f_a. Le résultat d'un neurone

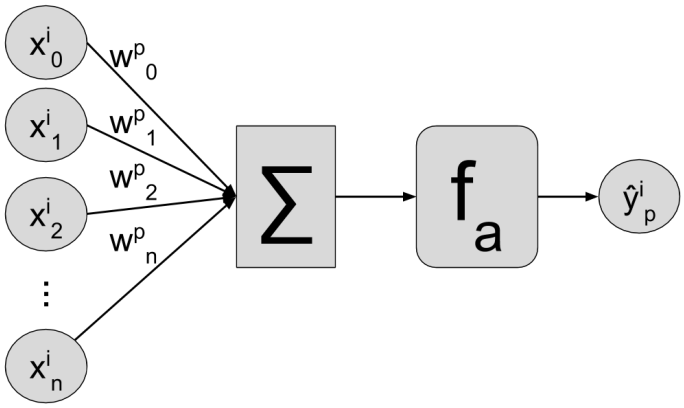


FIGURE 2.5 – Schéma d'un neurone à n entrées

p se résume donc par la formule :

$$\hat{y}_p^i = f_a(\sum_{j=0}^{n} x_j^i w_j^p). \tag{2.8}$$

La fonction d'activation est un paramètre à choisir et il en existe plusieurs avec différentes caractéristiques. Les plus connues sont la sigmoïde, la tangente, l'arc tangente, la tangente hyperbolique, la softmax, la marche ou encore l'identité. Le choix de celle-ci est déterminant pour l'apprentissage. Il est possible de créer sa propre fonction d'activation mais il est nécessaire que celle-ci soit facilement dérivable afin de mener à bien l'apprentissage (voir ci-dessous).

L'apprentissage d'un neurone consiste, à partir d'un jeu de données d'entrainement $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$, à calculer les meilleures valeurs des poids w_j^p afin que pour chaque série temporelle $\boldsymbol{x}_i$, la sortie du neurone soit la plus proche de la classe y_i ($\hat{y}_p^i \approx y_i$). Afin que l'apprentissage puisse s'effectuer, il est nécessaire d'introduire la notion de fonction coût (notée λ). Elle permet de quantifier l'erreur de classification (ou de prédiction) en mesurant l'écart entre les différentes sorties du neurone et les sorties espérées. Il en existe plusieurs mais la plus utilisée est la Mean Square Error (MSE) :

$$\lambda_{MSE}(y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_p^i)^2. \tag{2.9}$$

Le but de l'apprentissage est de trouver les meilleures valeurs des poids qui minimisent cette fonction de coût. Il est donc nécessaire de connaître l'impact de chacun des poids sur la fonction coût afin de les corriger. Pour mesurer l'impact d'un poids sur la fonction de coût, il faut calculer la dérivée partielle de la fonction de coût en

fonction de ce poids :

$$\frac{\partial \lambda_{MSE}(y)}{\partial w_j^p}. \tag{2.10}$$

L'ensemble de toutes les dérivées partielles de la fonction de coût en fonction de chacun des poids forment un vecteur appelé gradient $\nabla\lambda(y)$. Une fois le gradient calculé, il est possible d'apprendre les nouveaux poids avec la formule :

$$w_j^p = w_j^p - \alpha \frac{\partial \lambda_{MSE}(y)}{\partial w_j^p}, \tag{2.11}$$

avec α le coefficient d'apprentissage, appelé également learning rate. L'étape d'apprentissage est réitérée jusqu'à ce que la fonction de coût ait atteint une valeur minimale, jusqu'à ce que l'apprentissage ne donne plus aucune amélioration ou jusqu'à un certain nombre d'itérations prédéfinies. Le coefficient d'apprentissage est un paramètre très important. S'il est trop petit, l'apprentissage risque de trouver un minimum local, s'il est trop grand, l'apprentissage risque de ne jamais converger vers la solution. Une des solutions consiste à le réduire au fur et à mesure des itérations.

On parle de réseaux de neurones quand plusieurs neurones sont connectés en couche successives. Il en existe de plusieurs types. Dans la suite de cette section, le réseau de neurones MultiLayer Perceptron sera détaillé ainsi que les réseaux récurrents.

3.3.2 Perceptron multicouche

Le plus simple et le plus répandu des réseaux de neurones est le Perceptron multicouche (MultiLayer Perceptron (MLP) en anglais). C'est un réseau où toutes les couches sont totalement connectées, c'est à dire que les sorties des neurones d'une couches sont toutes connectées comme entrées des neurones de la couche suivante (Figure 2.6). Un MLP est composé d'une couche d'entrée, une couche de sortie et de plusieurs couches intermédiaires appelées couches cachées.

L'apprentissage est effectué de la même façon que pour le neurone. Cependant, pour calculer l'impact d'un poids sur la fonction de coût, il faut connaître son impact sur tous les neurones se trouvant sur le chemin jusqu'à la sortie. Donc il faut appliquer le principe de rétropropagation du gradient qui consiste à commencer par calculer l'impact des poids de la sortie pour ensuite remonter jusqu'à l'entrée.

Il a été prouvé qu'un MLP à une couche cachée peut approximer n'importe qu'elle fonction continue (théorème d'approximation universelle) [20, 21, 49, 66, 89]. Cependant, il faut parfois que la couche cachée soit composée de beaucoup de neurones ce qui entraine un très long temps d'apprentissage. Ce temps d'apprentissage a fait que les réseaux de neurones ont connu qu'un maigre succès dans les années 90. Avec

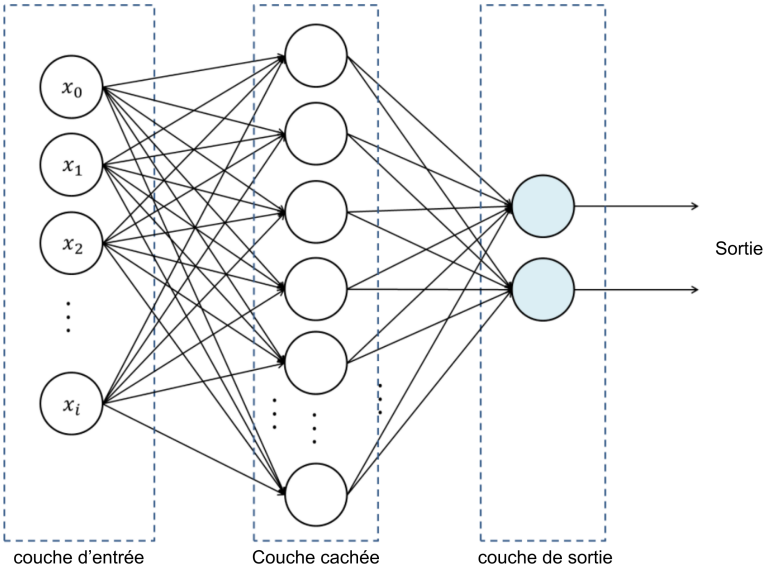


FIGURE 2.6 – Schéma d'un Perceptron multicouche à une couche cachée

l'arrivée du calcul GPU, les temps d'apprentissage se sont considérablement réduits ce qui a permis aux réseaux de neurones de finalement avoir le succès qu'ils ont aujourd'hui. Pour réduire davantage le temps de calcul, les neurones sont répartis sur plusieurs couches. En effet, le temps de calcul dépend du nombre de neurones et du nombre de couches mais la complexité est plus forte si le réseau possède moins de couches et plus de neurones par couche. Ainsi, quand un réseau de neurones possède un certain nombre de couches cachées, on parle de deep learning (ou apprentissage profond en français).

Cependant, certains problèmes nécessitent une longue période d'apprentissage qui se compte parfois en mois. Le réseaux MLP est donc très intéressant car il peut donner de très bonnes performances mais il nécessite du temps de calcul. Un autre point négatif est le paramétrage. En effet, il n'existe aucune règle qui permet de connaître, à partir d'un jeu de données, quels paramètres choisir (nombre de couches, nombre de neurones par couche, fonction de coût, fonction d'activation, etc.). Il existe un autre problème lié aux séries temporelles, le MLP n'est pas capable de gérer le fait que les données sont temporellement liées. C'est pour cela que les réseaux de neurones récurrents ont été créés.

3.3.3 Réseaux de neurones récurrents

Les réseaux de neurones récurrents ou Recurrent Neural Network (RNN) en anglais ont été inventés pour pouvoir traiter des entrées sous forme de séquences et sont donc plus adaptés pour traiter les problèmes liés aux séries temporelles. Le

principe est d'ajouter à chaque neurone une fonction mémoire afin que le calcul de la sortie se fasse par rapport aux entrées mais aussi par rapport à l'état précèdent (Fig 2.7). Soit une séquence d'entrée $[x_1^i, x_2^i, ...x_n^i]$, une sortie $\hat{y}_n^i$ est calculée en prenant comme entrée x_n^i mais aussi la sortie du neurone précédant $\hat{y}_{n-1}^i$. Ainsi, la prédiction de $\hat{y}_n^i$ se fait par rapport à l'entrée x_n^i mais aussi par rapport aux entrées précédentes. Concrètement, à l'intérieur d'un neurone récurrent, l'entrée x_n^i est concaténée avec la sortie $\hat{y}_{n-1}^i$, la somme pondérée est calculée ($\hat{y}_{n-1}^i$ possède son propre poids) et le résultat est envoyée dans une fonction d'activation tangente hyperbolique (tanh). C'est la fonction tanh qui est utilisée car elle est non-linéaire, dérivable et elle sa sortie est définie sur [-1,1] (on ajoute en mémoire quand c'est positif, on supprime quand c'est négatif).

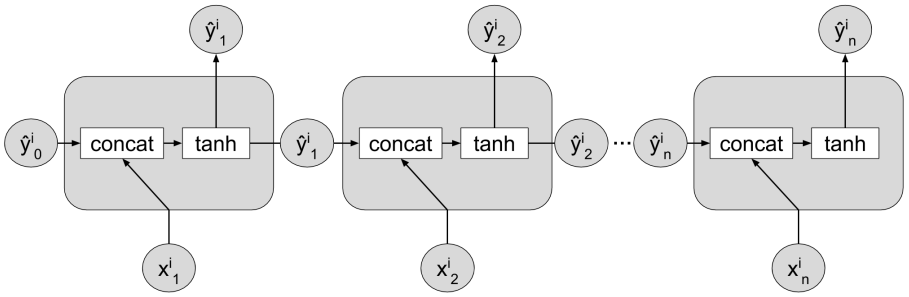


FIGURE 2.7 – Schéma d'un réseau de neurones récurrents

Pour augmenter la capacité mémoire d'un réseau récurrent, il suffit d'augmenter le nombre de neurones de la couche. Il est possible d'ajouter plusieurs couches au réseau. Cependant, ce type de réseau possède un défaut, il est affecté par le *problème de la disparition du gradient* (vanish gradient problem en anglais). En effet, pour entraîner un tel réseau, comme pour le MLP, il faut calculer le gradient. Par exemple, pour calculer l'impact du poids de $\hat{y}_{n-1}^i$ sur la sortie $\hat{y}_n^i$, il faut calculer l'impact du poids de $\hat{y}_{n-2}^i$, de $\hat{y}_{n-3}^i$, etc. Selon le théorème de dérivation des fonctions composées, tous les impacts intermédiaire (i.e. dérivées partielles) doivent être multipliées entre elles. Or, la dérivée de tanh est définie entre 0 et 1 et la multiplication de plusieurs nombres entre 0 et 1 converge vers 0. Autrement dit, même s'il est en théorie possible d'avoir une mémoire aussi grande que possible en ajoutant des neurones, en pratique la mémoire reste limitée et donc à court terme. Pour pallier ce problème, les réseaux de neurones Long Short-Term Memory (LSTM) ont été inventés.

Les réseaux LSTM ont une mémoire à court terme, comme les RNN, mais aussi une mémoire à long terme. Cette mémoire à long terme est transmise d'un neurone à un autre tout au long de la couche. Chaque neurone calcule la partie qui sera oubliée par la mémoire à long terme et il calcule également la partie qui sera conservée dans la mémoire. Les parties à oublier et à ajouter sont calculées à partir de l'entrée courante ainsi qu'à partir de la mémoire à court terme. La sortie du neurone est

quant à elle calculée à partir de l'entrée courante, de la mémoire à court terme et de la mémoire à long terme. Ce principe permet au réseau de pouvoir sélectionner des informations importantes dans toute la séquence d'entrée et pas seulement entre deux moments successifs.

Une cellule LSTM est détaillée par la Figure 2.8. Elle est constituée d'une mémoire à long terme, C_n et de trois parties :

— la partie *porte d'oubli* (forget gate en anglais) qui est chargée de calculer les informations que doit oublier la mémoire à long terme. Il s'agit de multiplier chaque valeur de C_{n-1} par un coefficient. Ces coefficients sont calculés en appliquant une fonction d'activation f_a sur l'entrée et la mémoire à court terme $\hat{y}^i_{n-1}$. Ainsi, en multipliant une valeur de C_{n-1} par un coefficients proche de 0, cette valeur (ou information) sera oubliée.

— La partie *porte d'entrée* (input gate en anglais) est la partie qui détermine les informations à ajouter à C_{n-1}. Les vecteurs $\hat{y}^i_{n-1}$ et x^i_n sont concaténés et passé dans une fonction d'activation f_a et dans une fonction *tanh*. Les deux vecteurs sont multipliés et ajoutés à C_{n-1} qui devient la nouvelle mémoire C_n.

— La partie *porte de sortie* (output gate en anglais) est chargée de calculer la sortie du neurone en fonction de $\hat{y}^i_{n-1}$, x^i_n et C_n. Les vecteur $\hat{y}^i_{n-1}$ et x^i_n sont concaténés et passés dans une fonction d'activation f_a, une fonction *tanh* est appliquée sur le vecteur C_n et les deux vecteurs ainsi calculés sont multipliés pour avoir $\hat{y}^i_n$.

La sortie $\hat{y}^i_n$ (mémoire court terme) ainsi que C_n (mémoire long terme) sont envoyés au neurone suivant, c'est ainsi qu'est formé une couche LSTM.

Ainsi, les réseaux LSTM sont moins sensibles au *problème de la disparition du gradient* car la mémoire à long terme est mise à jour tout au long de la couche LSTM. Ce type de réseau est très utilisé pour des problèmes liés aux séries temporelles et sont aussi très adaptées pour prédire des mots dans une phrase car le mot suivant dépend du mot précédent mais aussi très certainement d'autres mots dans la phrase.

Les réseaux LSTM ont les mêmes inconvénients que ceux des réseaux de neurones en général, c'est à dire qu'ils ont besoin d'un jeu de données d'apprentissage qui doit être assez conséquent et représentatif des conditions de test.

4 Classification

La classification de séries temporelles, ou Time Series Classification (TSC) en anglais, consiste à attribuer de manière automatique une classe à chaque série temporelle. Grâce à l'augmentation des données disponibles de nombreux algorithmes de TSC ont vu le jour [5]. Depuis les années 2000, la TSC est considérée comme une des problèmes les plus difficiles en data mining [33, 35, 134] et on compte des

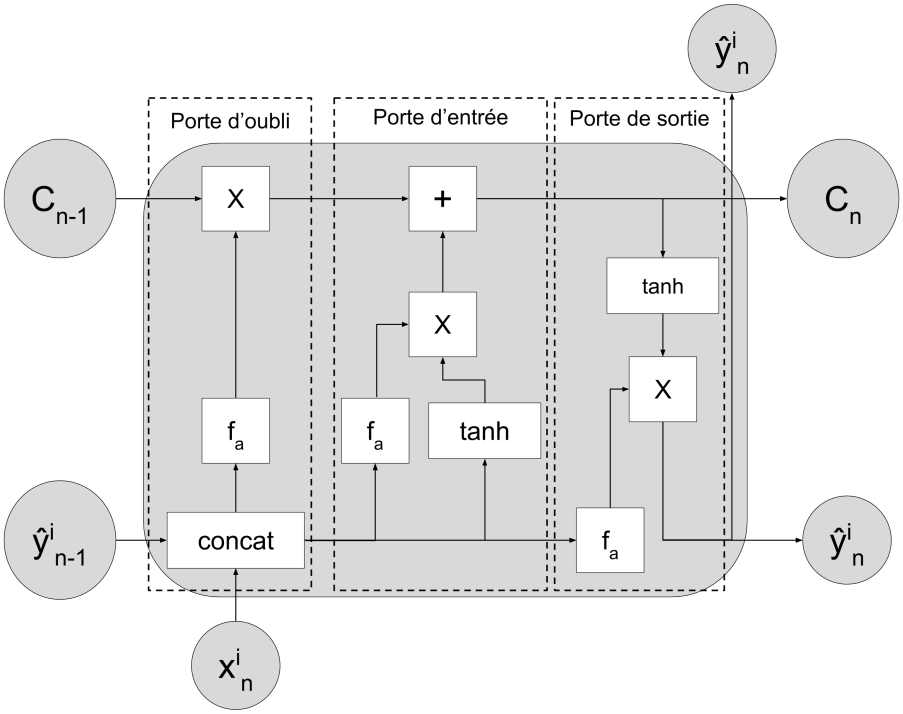


FIGURE 2.8 – Schéma d'une cellule LSTM, X : multiplication de deux vecteurs, $+$: addition de deux vecteurs et f_a une fonction d'activation

centaines d'articles proposant des solutions de TSC [5].

Les trois paradigmes *supervisé*, *non-supervisé* et *semi-supervisé* sont envisageables selon les situations. Si les séries temporelles sont bien étiquetées, utiliser une méthode dite supervisée est possible. Cependant, les étiquettes ne sont pas toujours disponibles, dans ce cas, il faut utiliser des méthodes dites non-supervisées (clustering). Parfois, il arrive de trouver une situation entre les deux, c'est à dire que la classe la plus représentée soit étiquetée mais pas les autres. Dans ce cas, une méthode semi-supervisée est plus adaptée. Dans la suite de cette section, seul le paradigme *supervisé* sera abordé car c'est celui qui est utilisé dans cette thèse.

La classification supervisée consiste donc, à partir d'une série temporelle $\boldsymbol{x}_i = [x_1, x_2, ..., x_n]$, à déterminer sa classe (ou étiquette) y_i. Pour cela, deux jeux de données sont généralement utilisés. Un jeu d'entraînement $\mathcal{D}_{train}$ pour lequel les étiquettes sont déjà connues est utilisé pour entraîner l'algorithme à reconnaître l'étiquette d'une série. Le second jeu de données est un jeu de test $\mathcal{D}_{test}$ qui contient des séries dont la bonne étiquette doit être déterminée par l'algorithme. Parfois, un troisième jeu de données de validation $\mathcal{D}_{val}$ est utilisé, il est également étiqueté et sert notamment à optimiser des paramètres de certains algorithmes.

Il existe de nombreuses méthodes de TSC dans la littérature. Dans la suite de

cette section, les méthodes Dynamic Time Warping, Bag Of SFA Symbols, Cote et deux réseaux de neurones deep learning (Fully Convolutional Networks et ResNet) seront présentées. Ces méthodes sont considérées comme étant les meilleures méthodes de TSC par la littérature récente [5, 35, 130]. Un rappel de la méthode des k plus proches voisins sera donné avant les autres méthodes.

4.1 K plus proches voisins

La méthode des k plus proches voisins [18] ou k Nearest Neighbor (k-NN) en anglais est une méthode incontournable en classification (voir également sous-section 3.2). C'est un algorithme simple et intuitif qui donne souvent de très bons résultats.

Pour classifier une série $\boldsymbol{x}_i \in \mathcal{D}_{test}$, la méthode consiste à trouver les k plus proches séries temporelles de $\boldsymbol{x}_i$ se trouvant dans $\mathcal{D}_{train}$. La classe prédite pour $\boldsymbol{x}_i$ est la classe la plus représentée parmi les k plus proches séries. La notion de plus proche est calculée par une fonction de distance qui est généralement la distance euclidienne.

Cette méthode présente l'avantage d'être simple à programmer et de ne nécessiter aucune période d'apprentissage. Son principal défaut est qu'elle n'est pas très adaptée pour la classification en temps réel, surtout si le jeu de d'entraînement est grand. En effet, pour chaque série $\boldsymbol{x}_i$ à classifier, la matrice des distances entre $\boldsymbol{x}_i$ et toutes les séries de $\mathcal{D}_{train}$ doit être calculée. Ce processus peut être très couteux en temps et dépend donc de la taille de $\mathcal{D}_{train}$. Cependant, le jeu d'entraînement $\mathcal{D}_{train}$ peut être modifié facilement sans aucun apprentissage, cela peut être utile si par exemple la nature des données tente à changer avec le temps (cela peut aussi modifier la performance de la méthode).

4.2 Dynamic Time Warping

Dynamic Time Warping (DTW) [96, 97] a été inventée pour le domaine de la reconnaissance vocale. DTW est une mesure élastique de comparaison entre deux séries temporelles qui est capable de prendre en compte les déphasages et les dilatations temporelles entres deux séries. Le problème avec les distances classiques (e.g. distance euclidienne) est que chaque point de la première série est comparé avec le point qui lui correspond dans la seconde série (Figure 2.9). Cependant, deux séries temporelles peuvent être très similaires mais l'une peut être légèrement déphasée par rapport à l'autre ou bien l'une peut avoir les mêmes schémas périodiques que l'autre mais dilatés dans le temps. Les distances standards ne prennent pas en compte ces caractéristiques et peuvent retourner une faible similarité entre les deux séries qui ne sont pourtant pas si différentes. DTW permet de prendre en compte ces ca-

ractéristiques en trouvant l'alignement global optimal entre deux séries temporelles (Figure 2.9).

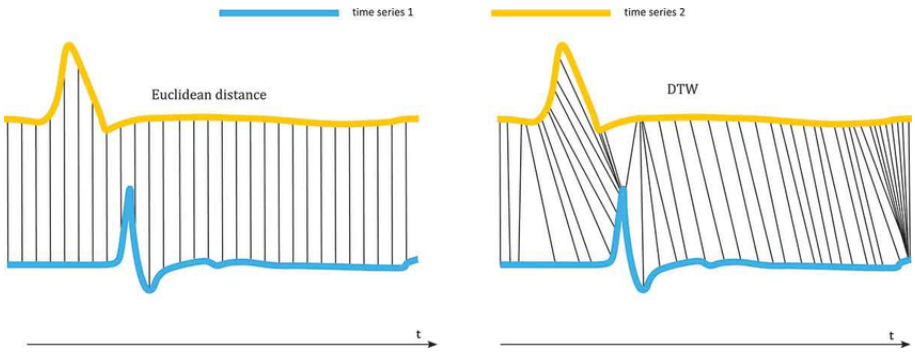


FIGURE 2.9 – Comparaison entre distance euclidienne et DTW

L'algorithme de DTW (algorithme 1) prend en entrée deux séries temporelles $\boldsymbol{a} = [a_1, a_2, ..., a_n]$ et $\boldsymbol{b} = [b_1, b_2, ..., b_m]$ et consiste à calculer la matrice des distances DTW entre chaque élément de $\boldsymbol{a}$ et de $\boldsymbol{b}$. La distance la plus utilisée est la distance euclidienne. La mesure DTW est donnée par la taille du plus court chemin entre $DTW_{1,1}$ et $DTW_{n,m}$ et est notée $d_{DTW}(\boldsymbol{a}, \boldsymbol{b})$.

Algorithme 1 : DTW

Entrées : $\boldsymbol{a} = [a_1, a_2, ..., a_n]$, $\boldsymbol{b} = [b_1, b_2, ..., b_m]$
Résultat : La mesure $d_{DTW}(\boldsymbol{a}, \boldsymbol{b})$
$DTW \in \mathbb{R}^{n+1*m+1}$;
pour $i = 1$ **à** n **faire**
  pour $j = 1$ **à** m **faire**
    $DTW_{i,j} = \infty$;
  fin
fin
$DTW_{0,0} = 0$;
pour $i = 1$ **à** n **faire**
  pour $j = 1$ **à** m **faire**
    $cout = dist(a_i, b_j)$;
    $DTW_{i,j} = cout + min(DTW_{i-1,j}, DTW_{i,j-1}, DTW_{i-1,j-1})$;
  fin
fin
retourner $DTW_{n,m}$

Cette mesure présente l'avantage d'être simple à programmer et donne de très bons résultats en moyenne quand elle est associée avec l'algorithme 1-NN [5]. Cependant, son temps d'exécution est plus long que pour une distance euclidienne. Il

convient également de noter que DTW n'est pas une distance car elle ne respecte pas obligatoirement l'inégalité triangulaire [95] bien qu'en pratique elle la respecte très souvent.

4.3 Bag Of SFA Symbols

Bag Of SFA Symbols (BOSS) [100] est une méthode de transformation de séries temporelles. Les méthodes de TSC peuvent être classées en deux catégories : les méthodes basées sur la forme et les méthodes basées sur la structure de la série. Les méthodes basées sur la forme utilisent une mesure de similarité combinée avec l'algorithme 1-NN (e.g. DTW). Ces méthodes sont assez performantes sur des jeux données nettoyés [29,100]. Les méthodes basées sur la structure (BOSS) transforment les séries temporelles afin de les faire changer de repère. Cette transformation a pour but d'extraire certaines caractéristiques intéressantes (e.g. pattern spécifique) rendant les méthodes basées sur la structure plus performantes sur des données bruitées [50, 67, 82, 100, 136].

Pour illustrer les performances de BOSS, la figure 2.10 montre le résultat d'un clustering hiérarchique [54] selon la méthode utilisée (euclidienne, DTW et BOSS). Le jeu de données utilisé est *Cylinder-Bell-Funnel*, il est constitué de séries temporelles synthétiques de trois formes différente : *bell*, *cylinder* et *funnel*. Avec une distance euclidienne, le clustering hiérarchique n'arrive pas à regrouper ni les *cylinders*n ni les *bells*. DTW donne de meilleurs résultats mais n'arrive pas à classer les *bells* ensemble. Seul BOSS arrive à faire un regroupement juste. L'avantage de BOSS réside dans le fait qu'il ne compare pas directement les séries temporelles mais des sous-séquences qu'il extrait. Ainsi, ça le rend plus robuste au bruit que les méthodes basées sur la forme [100] et tout comme DTW, BOSS est aussi capable de gérer les déphasages et les dilatations temporelles.

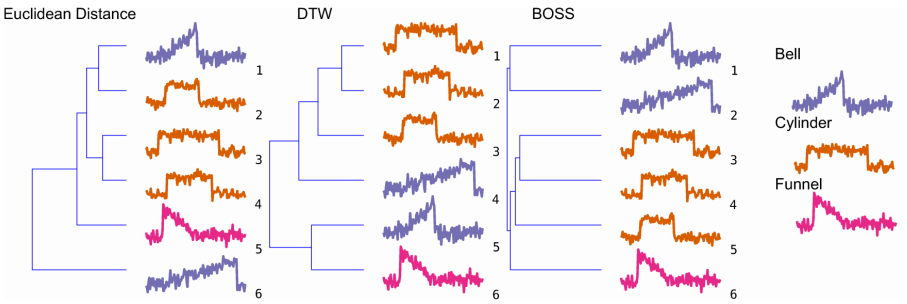


FIGURE 2.10 – Clustering hiérarchique sur le jeu de données *Cylinder-Bell-Funnel* basé sur trois mesures de similarité. Il y a trois types de courbes : *cylinder*, *bell* et *funnel* [100].

La méthode BOSS consiste pour chaque série du jeu de donnée à :

1. Appliquer une fenêtre glissante sur la série. La taille de cette fenêtre est un paramètre de la méthode. Il faut que la fenêtre soit au moins aussi grande la taille des sous-motifs intéressants de la série.
2. Appliquer une normalisation sur les valeurs de la fenêtre afin d'obtenir un écart-type de 1. C'est une étape optionnelle qui permet de réduire le problème des séries avec différentes valeurs moyennes.
3. Appliquer l'algorithme Symbolic Fourier Approximation (SFA) [101] sur chaque fenêtre (Fig.2.11). C'est un algorithme qui prend en entrée une séquence de valeurs (celle de la fenêtre glissante) et la transforme en un mot. Pour cela SFA échantillonne la série et discrétise chaque échantillon. L'algorithme a deux paramètres : la taille du mot et la taille de l'alphabet (précision de la discrétisation).
4. Chaque fenêtre donne un mot et donc pour chaque série il en résulte une séquence de mots. Les mots successifs qui se répètent sont agrégés afin d'éviter que des séquences de mêmes natures qui se suivent soient sur-représentées. Les occurrences des différents mots sont comptées et forment un histogramme. Il faut noter que tous les histogrammes du jeu de données sont de la même taille (une occurrence peut être à 0) et sont ordonnés de la même manière.
5. Chaque série est maintenant convertie en un histogramme de mots. La méthode de classification consiste à appliquer l'algorithme 1-NN sur les histogrammes ainsi calculés. La distance utilisée pour mesurer la similarité entre deux histogrammes $\boldsymbol{b}_1$ et $\boldsymbol{b}_2$ est la distance $d_{BOSS}(\boldsymbol{b}_1, \boldsymbol{b}_2)$ avec $\boldsymbol{b}_i(a)$ la fréquence du mot a dans l'histogramme $\boldsymbol{b}_i$ (eq. 2.12).

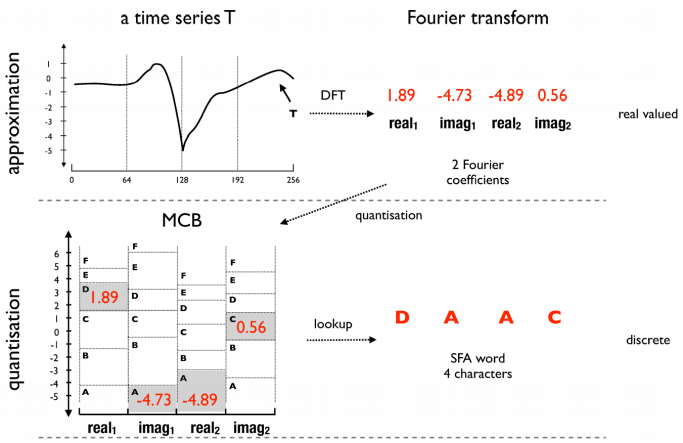


FIGURE 2.11 – Schéma illustrant l'algorithme SFA [100]

$$d_{BOSS}(\boldsymbol{b}_1, \boldsymbol{b}_2) = \sum_{a \in B_1; B_1(a) > 0} [B_1(a) - B_2(a)]^2. \tag{2.12}$$

4.4 COTE et Hive-COTE

4.4.1 COTE

L'algorithme Collective of Transformation-based Ensembles (COTE) [6], également appelé flat-COTE, a la particularité d'utiliser un ensemble de 35 classifieurs basés sur l'extraction de features depuis les domaines temporels et fréquentiels.

Les auteurs ont réfléchi à comment améliorer les algorithmes de TSC. Ils ont d'abord démontré que les méthodes de transformations basées sur le power spectrum (PS) et les fonctions d'autocorrélation (ACF) sont plus performants que les autres méthodes [4]. Les auteurs ont aussi montré que les méthodes qui utilisent les shapelets comme transformation sont plus performantes que les autres [45].

Les auteurs ont aussi fait l'hypothèse que d'utiliser des méthodes ensemblistes hétérogènes permettraient d'améliorer les résultats des classifieurs. Ils ont démontré que les mesures élastiques récemment proposées [53, 79, 108] ne sont pas forcément meilleures que DTW mais quand elles sont combinées en un ensemble élastique (EE) donnent de meilleures résultats [69].

L'idée de COTE est donc de combiner les méthodes ensemblistes et les méthodes de transformations. Il est composé de 11 classifieurs de type EE, 8 classifieurs basés sur les transformations en shapelets, 8 classifieurs basés sur des ACF et 8 classifieurs basés sur des PS. Ces 35 classifieurs forment un unique ensemble appelé flat-COTE. Le résultat final est donné par un vote pondéré des 35 résultats selon les performances des classifieurs sur une cross-validation.

Il a été démontré que COTE est en moyenne supérieur aux autres techniques de TSC [5]. Il a aussi été démontré que COTE est meilleur que les réseaux deep learning à convolution [71]. Cependant, même si flat-COTE donne de bonnes performances, il peut encore être amélioré. En effet, son principal défaut est son système de vote. Tous les classifieurs votent au même niveau alors que certains sont sur-représentés. Par exemple, les algorithmes EE sont au nombre de 11 et sont tous plus ou moins basés sur la mesure DTW alors que par exemple, les méthodes ACF sont au nombre de 8. Il est donc évident de conclure que les méthodes EE ont un plus gros poids dans le vote final. C'est principalement pour répondre à cette question que la méthode Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) a été étudiée.

4.4.2 Hive-COTE

Hive-COTE [70] est une amélioration de flat-COTE. Au lieu de faire voter à plat tous les classifieurs, Hive-COTE utilise une seule prédiction probabiliste par groupe de méthode (appelé module) évitant ainsi le biais induit par les méthodes sur-représentées.

Le principe de Hive-Cote est le suivant. Les classifieurs de chaque module sont soumis à un vote qui est pondéré par le résultat d'une cross-validation. Cela donne, pour chaque module et pour chaque classe du jeu de données, une probabilité. La classe finale qui est prédite est la classe qui a obtenu la plus grande probabilité. Grâce à cette méthode, peu importe si un module est composé d'un seul classifieur ou d'une centaine, il aura le même poids dans la décision finale.

Hive-COTE est constitué de 5 modules pour un total de 37 classifieurs :

— Elastic Ensemble (EE) qui est constitué des mêmes classifieurs que COTE (excepté que maintenant ils sont dans un module séparé),
— Shapelet Transform Ensemble (ST) qui regroupe toutes les méthodes de transformation par shapelets de flat-COTE en un seul module,
— Bag-of-SFA-Symbols qui est un module utilisant l'algorithme BOSS,
— Time Series Forest (TSF) [25] est un module qui emploie une approche random forest,
— Random Interval Features (RIF) regroupe les méthodes ACF et PS de flat-COTE. Ces deux types de méthodes ont été regroupées en un seul module car elles constituent la majorité des méthodes de flat-COTE (16 des 35 classifieurs) alors que [5] a montré que ces méthodes étaient en général moins performantes que les autres méthodes. Donc pour leur donner moins d'importance, elles ont été regroupées dans Hive-COTE.

Il a été démontré que Hive-COTE donne de meilleures performances que flat-COTE [71]. Cependant, cette méthode ne peut pas vraiment être utilisée comme classifieur définitif. En effet, lancer tous les algorithmes qui constituent COTE et Hive-COTE demande beaucoup de temps en calculs ce qui peut les rendre impossible à entrainer pour certains problèmes [76] et le rendant aussi inapproprié pour de la classification en temps réelle. Hive-COTE est en revanche un très bon point de départ pour la TSC [71]. En effet, regarder quels sont les modules qui ont les meilleures performances permet de faire le tri parmi tous les classifieurs possibles.

4.5 Deep learning

Le deep learning rencontre de plus en plus de succès en ce qui concerne les tâches de classification [65]. Cet engouement pour le deep learning a récemment touché le monde de la TSC et permet d'obtenir de très bonnes performances. Les meilleurs réseaux deep learning actuels pour la TSC sont le Fully Convolutional Networks

(FCN) et le Residual Network (ResNet) [34, 130] (Figure 2.12).

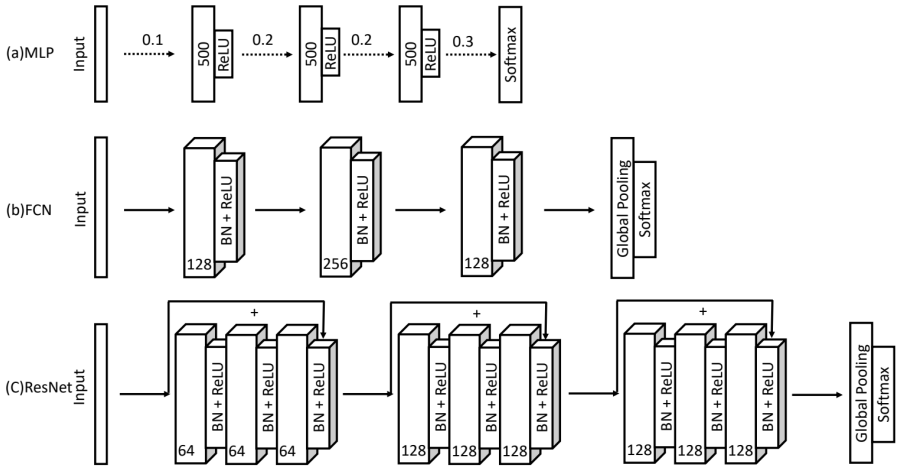


FIGURE 2.12 – Schéma des réseaux deep learning utilisés par [130] et [34].

4.5.1 FCN

Le réseau FCN est surtout utilisé pour la segmentation d'images [73]. Le principe est que chaque pixel de sortie correspond à un pixel d'entrée. Cela permet de faire une classification pixel par pixel. Par exemple, imaginons que le problème consiste à détecter des chats dans des images, avec un classifieur classique, la réponse est une classe : oui il y a un chat ou non il n'y en a pas. Avec un FCN la classification se fait par pixel, donc à l'endroit où il a le chat, tous les pixels seront activés, ça permet de localiser le chat dans l'image. Le FCN utilisé par [130] et [34] est employé comme une extracteur de features. Le réseau est constitué de trois blocs. Chaque bloc est composé d'une couche à convolutions, d'une batch normalisation [51] et d'une couche d'activation ReLU. Les trois couches à convolution ont respectivement un noyau de taille égal à 8, 5 et 3 et ils ont respectivement un filtre égal à 128 ,256 et 128. Après les trois blocs à convolutions, le dernier bloc est composé d'une couche *global average pooling* [68] et les labels finaux sont donnés par une dernière couche softmax.

4.5.2 ResNet

Les réseaux ResNet sont la nouvelle référence pour les problèmes de reconnaissance d'objets dans les images et dans tout ce qui touche à la vision en général [44]. Le principe des réseaux deep learning est de multiplier les couches de neurones ce

qui les rend plus particulièrement sensibles au problème du *vanishing gradient effect* (voir section 3.3.3). Le principe de ResNet est d'ajouter des connections entre les blocs à convolutions en réinjectant l'entrée à la sortie. Cela permet remettre de l'information potentiellement perdue par le bloc.

Le réseau utilisé par [130] et [34] est composé de 3 blocs à convolutions. Chacun de ces blocs enchaine 3 couches à convolutions suivies d'une couche batch normalisation et d'une couche d'activation ReLU. Les blocs utilisent respectivement 64, 128 et 128 filtres.

Selon [130], le réseaux FCN seraient plus performants que le réseau ResNet alors que pour [34] c'est l'inverse. La raison donnée par [34] est qu'il utilise plus de jeux de données tests (97 contre 44) et donc les résultats seraient plus précis. On peut en conclure que les réseaux FCN et ResNet donnent de très bonnes performances avec parfois un léger avantage pour ResNet. L'avantage d'utiliser de tels réseaux est qu'une fois l'apprentissage effectué, le test s'effectue en un temps très court. Ils peuvent être donc utilisés pour des applications en temps réel.

5 Détection d'anomalies

La détection d'anomalies est un large domaine qui touche un grand nombres d'applications [1, 13, 41, 47, 137]. Elle est utilisée dans la finance, par exemple pour détecter les anomalies afin de prédire les crashes boursiers, dans les diagnostiques de systèmes informatisés avec la détection de fraudes (CB ou piratage internet), dans la navigation automatique des avions, dans les systèmes de production industrielle... La détection d'anomalies est aussi utilisée en biologie ou encore dans tout ce qui touche aux actions utilisateurs (actions navigation web, transactions marchandes, détection fraude, etc.).

Dans le domaine des séries temporelles, la détection d'anomalies a commencé avec les modèles paramétriques [37]. D'autres modèles ont ensuite vu le jour Auto-Regressive Moving Average (ARMA), AutoRegressive Integrated Moving Average (ARIMA), Vector AutoRegression Moving Average(VARMA), CUmulative SUM Statistics (CUSUM), etc.

Aujourd'hui il existe deux grandes catégories de détection d'anomalies dans de domaines des séries temporelles : la détection de points anormaux (ou de séquences) au sein d'une série temporelle (approche série temporelle) et la détection de séries anormales dans un jeu de données (approche base de données) [41].

5.1 Approche série temporelle

Le but est de détecter toutes les séquences (ou points) anormales dans une seule série temporelle. Il existe plusieurs solutions : les modèles prédictifs, les profils types,

les déviants et toutes les méthodes de détection de sous-séries anormales. Le choix de l'approche à utiliser dépend de la nature des données, du problème et de la manière dont une anomalie se manifeste.

5.1.1 Modèles prédictifs

Le principe est d'abord de modéliser la série afin de pouvoir prédire plusieurs valeurs futures. Ensuite, les valeurs prédites et les valeurs réelles sont comparées, si la prédiction s'éloigne de la réalité, alors les points sont considérés comme anormaux. Par exemple, [77] propose un nouveau modèle basé sur un Support Vector Regression (SVM) [30], [64] créé la Mixture transition distribution (MTD) pour détecter les anomalies dans les séries temporelles non-gaussiennes ou encore [115] propose un modèle basé sur ARIMA.

Les réseaux de neurones sont aussi très utilisés. Par exemple [78] utilise un modèle basé sur un réseau LSTM. Cette méthode est composée de trois étapes (cf. Figure 2.13). La première étape consiste à apprendre un modèle LSTM sur un jeu de données d'apprentissage sans anomalie. Ensuite, les erreurs de prédictions sont analysées sur un jeu de données de validation qui contient des anomalies. Ces erreurs sont modélisées par une loi gaussienne. Un seuil est ensuite est calculé par maximisation du $F_\beta - score$ [99]. Si la valeur de la loi normale, pour une erreur donnée, est inférieure à ce seuil, alors l'observation est classée comme normale. Cette méthode prend comme paramètre une taille de prédiction l.

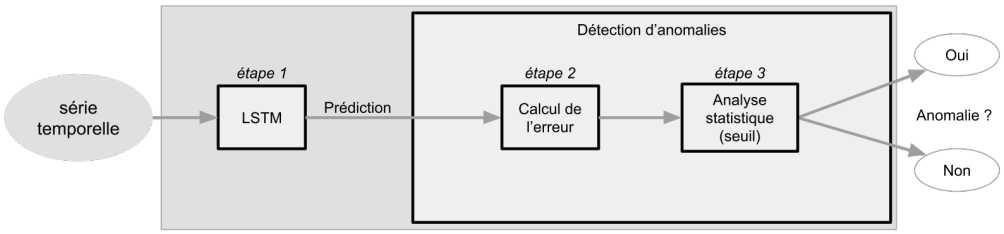


FIGURE 2.13 – Schéma illustrant la méthode de Malhotra et al. [78]

5.1.2 Détections basées sur un profil type

Cette approche est basée sur le principe que les anomalies engendrent des comportements instables sur les séries temporelles [131]. Ces instabilités peuvent être détectées en analysant les variations de certaines caractéristiques comme la tendance, la moyenne, etc. qui forment le profil type [32]. Par exemple, [131] analyse les performances de systèmes d'exploitations avec plusieurs métriques afin de détecter

des erreurs. Ou alors [106] utilise un réseau de neurones pour analyser les erreurs dans un système de contrôle de vol d'avion.

L'avantage de ce type de méthodes est qu'il est possible de l'adapter pour une détection en temps réel.

5.1.3 Déviant

Les déviants sont plus des points extrêmes que des anomalies. Il s'agit de modéliser la série par un histogramme et si en enlevant un point, le nouvel histogramme calculé est différent, alors le point est considéré comme déviant [52, 84]. Le principe de l'histogramme est de partitionner la série temporelle en plusieurs parties (appelées *buckets*). Chaque point de la série est modélisé par la moyenne de son bucket. Les buckets n'ont pas forcément la même taille et tous les histogrammes générés n'ont pas forcément le même nombre de buckets. Tout le problème consiste à pouvoir générer ces histogrammes le plus efficacement possible.

5.1.4 Sous-séries anormales

Une sous-série est dite anormale si sa distance avec les autres sous-séries voisines non recouvrantes est élevées [59]. Une solution simple consiste à comparer toutes les sous-séries avec toutes les autres (qui sont non recouvrantes) cependant, cette solution n'est pas très efficace d'un point de vue temps de calculs. Certaines solutions plus efficientes existent comme par exemple [58] qui utilise un algorithme heuristique ou encore [38] qui utilise l'ondelette de Haar [12].

D'autre variantes ont vu le jour comme par exemple [14] qui utilise également l'ondelette de Haar pour trouver les sous-séries anormales de différentes échelles.

5.2 Approche base de données

Cette approche consiste à déterminer, parmi un ensemble de séries (ou base de données), lesquelles sont anormales. Il existe deux possibilités, soit le jeu de données n'est pas étiqueté et dans ce cas on parle de techniques non-supervisées, soit le jeu de données est totalement étiquetée et dans ce cas on parle de techniques supervisées. Dans le premier cas de figure, il est supposé que la majorité des séries sont normales et que seulement une minorité sont anormales. Il existe plusieurs manières de procéder notamment l'approche par cluster ou l'approche par modèle paramétrique. Par exemple [10] utilise l'algorithme de clustering k-médoïdes [56] afin de grouper les séries temporelles. Un certain pourcentage de séries aberrantes est déterminé pour chaque cluster ce qui donne les séries anormales. Les algorithmes de clustering peuvent varier selon le type de problème. Par exemples, [85] utilise l'algorithme k-Means, [103] utilise le clustering dynamique ou encore [111] utilise une

Support-Vector Machine (SVM) à une classe pour clustériser ses données. L'approche par modèle paramétrique consiste à créer un modèle des séries temporelles. La série est dite anormale si la probabilité qu'elle soit générée par le modèle est faible. Par exemple, [110, 135] proposent des modèles basés sur des chaines de Markov.

En approche supervisée, la solution la plus utilisée consiste à utiliser des outils de TSC (voir section 4).

6 Étiquettes floues et classification supervisée

Dans le monde de la classification supervisée, il est essentiel d'avoir des données étiquetées. Cependant, l'étiquetage est un procédé qui engendre souvent du bruit et des erreurs [112] qui sont causés, par exemple par des erreurs humaines ou une mauvaise précision des capteurs. Ces erreurs dans les étiquettes font que les algorithmes sont biaisés dans leur apprentissage, ainsi les résultats sont moins précis, voir inexploitables dans les cas les plus extrêmes. Quand on n'est pas certain d'une étiquette, on parle d'étiquettes incertaines, floues ou soft/fuzzy en anglais, en opposition aux étiquettes dures (ou hard en anglais). Il existe des algorithmes dits flous (ou fuzzy/soft en anglais) qui prennent en compte des données avec des étiquettes floues. Plus les étiquettes sont bruitées, plus ils sont efficaces face aux algorithmes durs [2, 112]. Il y a deux avantages à utiliser un algorithme flou. Le premier est que l'apprentissage est amélioré. En effet, les étiquettes avec une forte incertitude sont moins prises en compte que les étiquettes les plus certaines. Le second avantage est que le résultat d'un tel algorithme donne une information supplémentaire. En effet, un algorithme dur ne donne qu'une information : la classe d'appartenance de chaque instance. Un algorithme flou, donne une information supplémentaire qui est en pratique souvent intéressante : la probabilité d'appartenance de chaque classe pour chaque instance. Cette information permet donc de connaître le degré d'incertitude de l'algorithme.

6.1 Fonction d'appartenance

La fonction d'appartenance (ou membership function en anglais) est définie pour les données floues, elle est en opposition à la fonction caractéristique qui définit les données dures.

Soit $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}$ un jeu de données étiqueté. Dans le cadre d'étiquettes dures, on définit la fonction caractéristique $f_c : \mathcal{X} \to \{0, 1\}$ qui pour chaque série $\boldsymbol{x}_i \in \mathcal{X}$ lui associe une classe $c \in \mathcal{C}$:

$$f_c(\boldsymbol{x}_i) = \begin{cases} 1, & c = y_i, \\ 0, & c \neq y_i. \end{cases} \tag{2.13}$$

Contrairement aux étiquettes dures, les étiquettes floues sont associées à un degré de confiance, c'est-à-dire une probabilité d'appartenir à une classe. Dans ce cas, on parle de fonction d'appartenance $u_c : \mathcal{X} \to [0, 1]$:

$$u_c(\boldsymbol{x}_i) = \mathcal{P}(y_i = c), \tag{2.14}$$

tel que :

$$\sum_{c \in \mathcal{C}} u_c(\boldsymbol{x}_i) = 1, \tag{2.15}$$

$$0 < \sum_{\boldsymbol{x} \in \mathcal{X}} u_c(\boldsymbol{x}) < |\mathcal{D}|, \, \forall c \in \mathcal{C}. \tag{2.16}$$

6.2 Algorithme k-NN flou

Il existe plusieurs algorithmes flous. Souvent, ce sont des algorithmes durs qui ont été convertis pour la logique floue. Parmi les tous les algorithmes utilisés pour la logique flous, le plus courant est l'algorithme k-NN et il existe plusieurs variantes floues [28, 46, 109]. Parmi ces variantes, il y a l'algorithme fuzzy k-NN [57] qui est l'une des variantes les plus populaires [26].

Le but de l'algorithme fuzzy k-NN est de calculer la fonction d'appartenance du jeu $\mathcal{D}_{test}$. C'est un algorithme en deux étapes. La première est similaire à la version dure, elle consiste à trouver les k plus proches voisins de chaque série $\boldsymbol{x}_i \in \mathcal{D}_{test}$. L'ensemble des k plus proches voisins est noté $\mathcal{K}$. La seconde étape consiste à calculer la fonction d'appartenance de chaque série $\boldsymbol{x}_i$ via la formule :

$$u_c(\boldsymbol{x}_i) = \frac{\sum_{\boldsymbol{x}_j \in \mathcal{K}} u_c(\boldsymbol{x}_j) d_{eucli}(\boldsymbol{x_i}, \boldsymbol{x_j})^{-2/(m-1)}}{\sum_{\boldsymbol{x}_j \in \mathcal{K}} d_{eucli}(\boldsymbol{x_i}, \boldsymbol{x_j})^{-2/(m-1)}}, \, \forall c \in \mathcal{C}, \tag{2.17}$$

avec m un coefficient qui contrôle le degré de flou dans la prédiction et $d_{eucli}(\boldsymbol{x_i}, \boldsymbol{x_j})$ la distance euclidienne entre les séries $\boldsymbol{x}_i$ et $\boldsymbol{x}_j$. Dans la littérature, le coefficient m est souvent fixé à $m = 2$.

Chapitre 3

Classification de séries temporelles avec les transformées de Fourier

Ce chapitre a pour but d'introduire, d'expliquer et de tester une nouvelle méthode de détection d'anomalies dans les séries temporelles : la Fourier Based Method with Thresholding (FBAT) méthode. La méthode FBAT a été spécialement créés pour détecter les anomalies dans l'activité de l'animal qui peuvent être liées à l'état de l'animal (maladie, stress, œstrus, vêlage)). Cette section explique dans un premier temps le fonctionnement de l'algorithme FBAT puis décrit les données utilisées ainsi que le protocole expérimental mis en place pour tester FBAT. Enfin, une dernière section donne les résultats ainsi qu'une discussion sur les points forts et points faibles de FBAT par rapport aux autres méthodes testées.

1 FBAT

1.1 Motivations

Il est montré dans le chapitre 1 sous-section 2.2 17, que les vaches laitières ont un rythme d'activité journalier et que ce rythme est variable et cette variabilité a plusieurs origines : la saison, la ferme, les jours, les individus ou l'état de ces individus (maladie, stress, etc.). Le fait qu'il existe une différence entre les jours où un animal est en état normal (dits *jours normaux*) et les jours où il est malade, stressé, en œstrus ou encore en vêlage (dits *jours anormaux*) montre qu'il est possible de détecter ces états spécifiques. Cependant, les autres sources de variation rendent cette détection difficile La méthode de détection doit donc respecter certains critères. Tout d'abord, la méthode doit être robuste au bruit. En effet, les variabilités entre jours et entre vaches introduisent un certain niveau de bruit, d'autant plus que parfois les variations entres jours peuvent être plus importantes que les variations liées à l'état

de l'animal (voir Figures 1.12 et 1.13). La plupart du temps, les méthodes de machine learning nécessitent un apprentissage. L'idéal serait d'avoir un jeu $\mathcal{D}_{train}$ pour chaque vache et que le modèle soit capable d'apprendre sur une courte durée afin d'éliminer la variabilité entre vache et entre jours. Cependant, les outils de machine learning ont souvent besoin d'un jeu de données conséquent pour apprendre, donc soit la méthode utilise un modèle par vache mais sur une période plutôt longue, soit la méthode utilise un modèle unique pour toutes les vaches mais appris sur des données acquises sur une courte période. Dans ces deux cas, les variations liées à l'état de l'animal (ce que l'on cherche) sont confondues soit avec les variations entre jours, soit avec les variations entre individus.

De plus, il doit être facile de mettre la méthode choisie en production. Les Figures 1.9 et 1.10 montrent qu'il existe une variabilité entre fermes, ce qui signifie que la problématique de mise en production risque d'être complexe. En effet, les données dans un jeu d'entraînement doivent être le plus proche possible des données test. Or, une solution simple pour mettre en production facilement aurait été d'utiliser un jeu d'entraînement générique et d'appliquer le même modèle à chaque ferme, ce qui n'est bien sûr pas possible à cause de la variabilité entre fermes (et aussi entre individus et entre jours). Il faut ajouter que le modèle mis en production devra vivre avec la ferme. Que se passera-t-il quand une nouvelle vache arrivera dans la ferme ? Il est également fort probable qu'il existe une variabilité saisonnière (cf. chapitre 1 sous-section 1.4, 21), donc soit les données doivent être acquises sur au moins une année, soit le modèle doit être réappris plusieurs fois par an, dans les deux cas, ce n'est pas acceptable pour une production.

L'idéal serait donc une méthode qui nécessite que très peu de données d'apprentissage (une seule vache sur quelques jours maximum) et qui s'adapte facilement à une nouvelle ferme. Il faudrait aussi que la méthode soit capable de s'adapter aux changements de saisons et à l'arrivée de nouvelles vaches. La méthode doit être également assez rapide pour détecter une anomalie, afin d'aider les éleveurs à prendre une décision rapidement pour corriger les problèmes. C'est pour répondre à toutes ces exigences que la méthode FBAT a été créée.

1.2 Présentation de la méthode

Le principe de FBAT est de comparer les variations de cycles dans les séries temporelles. L'hypothèse est que si un animal est malade, stressé, ou dans un état physiologique particulier (oestrus, vélage), alors son cycle circadien sera modifié. Pour cela, FBAT construit deux modèles de deux sous-séries proches l'une de l'autre (voir Figure 3.1) grâce aux décomposées de Fourier et les compare. Si les modèles sont trop différents, FBAT considère les deux sous-série comme anormales (Figure 3.2).

Chaque série temporelle $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ du jeu de données est d'abord dé-

composée en deux sous-séries de taille p. Les deux deux sous-séries sont décalées de q valeurs, elles peuvent être recouvrantes ou non, ainsi, $\boldsymbol{a} = [x_1, x_2, ..., x_p]$ et $\boldsymbol{b} = [x_{1+q}, x_{2+q}, ..., x_{p+q}]$ avec $p + q < n$.

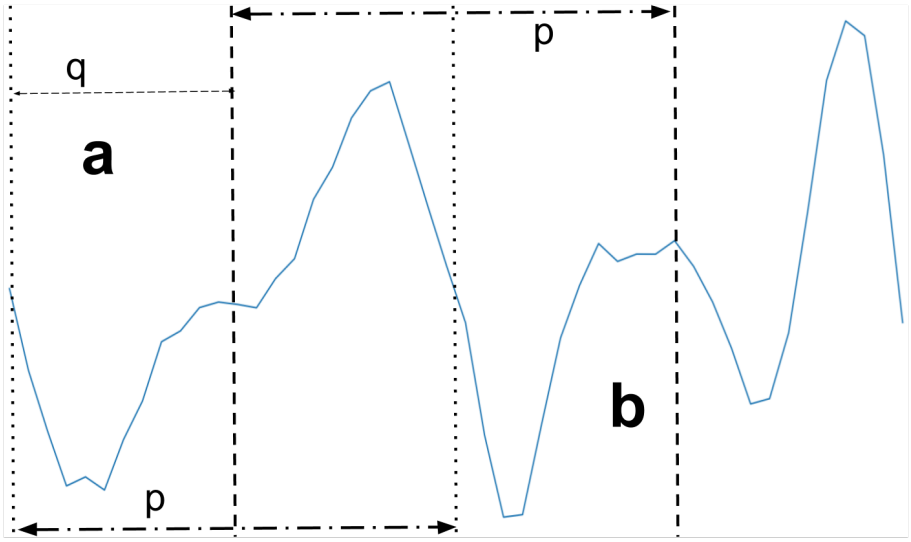


FIGURE 3.1 – Extraction des deux sous-séries $\boldsymbol{a}$ et $\boldsymbol{b}$ de taille p et décalées de q valeurs.

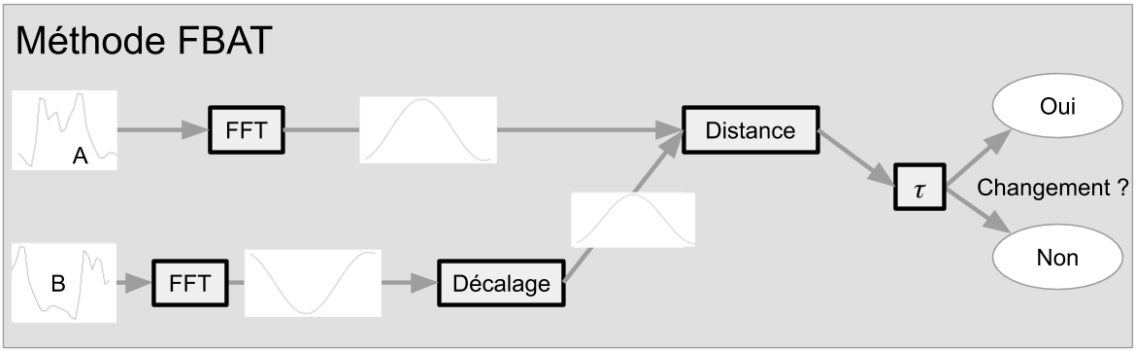


FIGURE 3.2 – Schéma global de la méthode FBAT.

Ensuite, les transformées de Fourier sont calculées pour chacune des deux sous-séries $\boldsymbol{a}$ et $\boldsymbol{b}$ grâce à l'algorithme FFT (voir chapitre 2, sous-section 2.2). Chaque sous-série obtient donc son vecteur d'harmoniques, $\boldsymbol{h_a}$ et $\boldsymbol{h_b}$:

$$\boldsymbol{h} = \{h_{-\lceil \frac{p-1}{2} \rceil}, h_{-\lfloor \frac{p-1}{2} \rfloor + 1}, ..., h_{-1}, h_0, h_{+1}, ..., h_{\lfloor \frac{p-1}{2} \rfloor}\}, \quad h_i \in \mathbb{C}, \tag{3.1}$$

avec h_0 l'harmonique de rang 0 qui correspond à la composante de fréquence $f = 0Hz$, h_1 l'harmonique de rang 1, c'est-à-dire la fondamentale qui représente la période p. Les harmoniques h_i de rangs supérieurs représentent les composantes de période $\frac{p}{i}$. Les séries temporelles utilisée dans cette thèse sont définies dans $\mathbb{R}$, donc les harmoniques sont symétriques ($h_i = h_{-i}$).

À partir de ces harmoniques, deux fonctions temporelles d'approximation des sous-séries $\boldsymbol{a}$ et $\boldsymbol{b}$ peuvent être calculées elles sont notées respectivement $m_{\boldsymbol{a}}(t)$ et $m_{\boldsymbol{b}}(t)$:

$$m_{\boldsymbol{a}}(t) = \sum_{f=-z}^{z} |h_f| cos\left(2\pi f \frac{t}{p} + arg(h_f)\right),\ h_f \in \boldsymbol{h_a},\ z \in \{0...\lceil\frac{p-1}{2}\rceil\}, \tag{3.2}$$

avec z le nombre d'harmoniques à garder pour la fonction. Il détermine la précision d'approximation de la fonction et donc détermine la précision du modèles (voir ci-dessous). Si z est petit, la fonction génère un modèle grossier de la série, si z est grand, l'approximation est plus précise. Cependant, un des intérêts de cette méthode et de cette modélisation est de pouvoir éliminer une certaine quantité de bruit liée aux données. Prendre une petite valeur de z permet d'éliminer les harmoniques de rangs élevés et donc d'éliminer le bruit. C'est donc un paramètre important de la méthode et doit être fixé judicieusement afin d'éliminer le bruit superflu tout en gardant le maximum d'informations. Les résultats présentés en section 4.2 donnent des détails sur le choix de z. La fonction $m_{\boldsymbol{b}}(t)$ est un peu différente. En effet, pour pouvoir être comparées, les deux sous-séries doivent être synchronisées temporellement. Or, les sous-séries $\boldsymbol{a}$ et $\boldsymbol{b}$ sont décalées de q valeurs, il faut donc modifier la formule en ajoutant le délai $-\frac{q}{p}2\pi$ à $m_{\boldsymbol{b}}(t)$ afin d'éliminer ce décalage :

$$m_{\boldsymbol{b}}(t) = \sum_{f=-z}^{z} |h_f| cos\left(2\pi f \frac{t}{p} + arg(h_f) - \frac{q}{p}2\pi\right),\ z \in \{0...\lceil\frac{p-1}{2}\rceil\}. \tag{3.3}$$

À partir de ces deux fonctions, il possible de calculer deux nouvelles séries temporelles $\boldsymbol{m_a}$ et $\boldsymbol{m_b}$:

$$\boldsymbol{m_i} = [m_i(0), m_i(1), ..., m_i(p-1)],\ i \in \{\boldsymbol{a}, \boldsymbol{b}\}. \tag{3.4}$$

La dernière étape consiste à calculer la distance de norme 2 (distance euclidienne) entre $\boldsymbol{m_a}$ et $\boldsymbol{m_b}$:

$$d_2(\boldsymbol{m_a}, \boldsymbol{m_b}) = \sqrt{\sum_{t=0}^{p-1}(m_{\boldsymbol{b}}(t) - m_{\boldsymbol{a}}(t))^2}. \tag{3.5}$$

La détection est faite sur cette distance. Si la distance entre les deux modèles dépasse un certain seuil τ, alors la série temporelle est considérée comme anormale. Le seuil τ se calcule par optimisation, les détails sont données en sous-section 3.2.

2 Données utilisées

Pour tester FBAT, plusieurs jeux de données ont été utilisés. Des jeux de données issus des vaches laitières mais aussi des jeux de données d'autres natures disponibles librement sur internet.

2.1 Données des vaches laitières

Je disposais de quatre jeux de données de vaches laitières (voir chapitre 1, sous-sous-section 2.1.4). Chaque vache de chaque jeu de données est représentée par une série temporelle représentant son niveau d'activité par heure et les jours sont notés normaux ou anormaux selon qu'une maladie ou autre perturbation a été notée par les soigneurs.

Le but est d'avoir pour chaque jeu de données un ensemble $\mathcal{D}_{train}$ et $\mathcal{D}_{test}$. La première étape consiste à décider de la taille de chaque série n. L'hypothèse est que le rythme circadien de l'animal change sous l'effet d'une anomalie. Donc, il est judicieux de prendre $n > 24$ afin d'avoir plus d'un jour d'activité. Il faut également que n ne soit pas trop grand pour pouvoir localiser l'anomalie le plus précisément possible (et donc le plus rapidement possible pour l'éleveur). J'ai choisi une valeur de $n = 36$, soit un jour et demi.

Pour extraire ces séries, une fenêtre glissante de 36 h est appliquée sur chaque série temporelle représentant une vache. Cette fenêtre se déplace d'heure en heure et à chaque passage, une série de 36 h est extraite. Pour étiqueter ces séries, il faut définir une zone potentiellement anormale - c'est-à-dire où une maladie ou autre perturbation a été relevée - dont la taille dépend de la nature de la perturbation. En effet, les jours anormaux ont été notés mais il se peut que le comportement de l'animal ait commencé à changer un peu avant l'apparition des symptômes cliniques et il se peut également que le comportement mette un peu de temps à revenir à la normale. Pour prendre en compte cela, selon le type de perturbation, certains jours avant et après le jour noté anormal ont aussi été notés comme anormaux. (voir Table 3.1). Certains jours après la perturbation ont été supprimés du jeu de données car ils sont considérés comme trop incertains, c'est-à-dire que l'on ne peut pas préjuger d'un retour à la normale de l'activité des vaches. Ces décisions ont été prises à partir de la bibliographie [27, 113, 119]. Si une série de 36 h contient plus de 12 h de jours considérés comme anormaux, alors la série est notée anormale, sinon,

elle est notée normale. Le nombre de séries temporelles ainsi extraites par jeu de données est donnée en Table 3.2.

TABLE 3.1 – Jours étiquetés anormaux en fonction de l'anomalie ; j-j représente le jour noté par l'éleveur, A signifie que le jour a été étiqueté comme anormal, X signifie que le jour a été supprimé et une case vide signifie que le jour est étiqueté comme *normal*.

Jours	**j-3**	**j-2**	**j-1**	**j-j**	**j+1**	**j+2**	**j+3**	**j+4**	**j+5**	**j+6**	**j+7**	**j+8**
Accidents				**A**	A	X	X	X	X	X	X	
Vêlages		A	A	**A**	A	X	X	X	X	X	X	
Oestrus			A	**A**	A	X						
Boiteries		A	A	**A**	A	X	X	X	X	X	X	
Mammites		A	A	**A**	A	X	X	X	X	X	X	
Autres maladies		A	A	**A**	A	X	X	X	X	X	X	
Injections LPS				**A**	A	X	X	X	X	X	X	
Acidoses				**A**	A							
Changement de parc				**A**	A	X	X	X	X	X	X	
Autres perturbations				**A**								

TABLE 3.2 – Nombre de séries temporelles par jeu de données.

Jeu de données	**nb nomales**	**nb anormales**	**total**
1	8349	29151	37500
2	68032	10273	78305
3	21013	1555	22568
4	1397736	474752	1872488

2.2 Données de l'UCR

Pour tester et comparer FBAT, j'ai également utilisé certains jeux de données disponibles gratuitement et facilement sur l'archive de l'University of California Riverside (UCR) [24]. Onze jeux de données sont utilisés. Ils ont tous des caractéristiques différentes mais ils sont tous composés de seulement deux classes pour pouvoir

être utilisés par FBAT. Leurs caractéristiques sont détaillées en Table 3.3. La particularité est qu'ils sont déjà découpés en deux jeux de données d'apprentissage et de test.

TABLE 3.3 – Caractéristiques des jeux de données de l'UCR.

Jeu de données	**Taille apprentissage**	**Taille test**	**Taille série**	**Type**
Earthquakes	322	139	512	SENSOR
FreezerSmallTrain	28	2850	301	SENSOR
ItalyPowerDemand	67	1029	24	SENSOR
MoteStrain	20	1252	84	SENSOR
SonyAIBORobotSurface1	20	601	70	SENSOR
ECG200	100	100	96	ECG
ECGFiveDays	23	861	136	ECG
TwoLeadECG	23	1139	82	ECG
WormsTwoClass	181	77	900	Motion
Lightning2	60	61	637	SENSOR
Yoga	300	3000	426	IMAGE

3 Protocole

3.1 Mesures d'évaluation

Pour évaluer une méthode, il faut définir plusieurs mesures. Dans un premier temps, si la classe *normale* des jeux de données 1, 2, 3 et 4 est dite *positif* (P) et la classe *anormale* est dite *négatif* (N), 4 termes peuvent être utilisés : Vrais positifs (VP), Vrais Négatifs (VN), Faux Positif (FP) et Faux Négatifs (FN) (voir Table 3.4). Plusieurs mesures en découlent :

$$precision = \frac{VP + VN}{VP + VN + FP + FN}, \tag{3.6}$$

$$precision_{-} = \frac{VN}{VN + FN} \quad \text{et} \quad rappel_{-} = \frac{VN}{VN + FP}, \tag{3.7}$$

$$precision_{+} = \frac{VP}{VP + FP} \quad \text{et} \quad rappel_{+} = \frac{VP}{VP + FN}. \tag{3.8}$$

La *precision* est la mesure la plus communément utilisée pour comparer les méthodes entre elles [35]. La *precision* et le *rappel* de chaque classe permet d'affiner la comparaison. En effet, ces mesures sont très appréciées pour savoir si une méthode est plus à l'aise avec une classe en particulier et permet également de savoir si la *precision* générale n'est pas faussée par une classe sur-représentée. Dans cette thèse, la mesure du temps de calcul est aussi utilisée. Il y en a deux, le temps CPU (Central Processing Unit) et le temps GPU (Graphics Processing Unit).

TABLE 3.4 – Définition des termes VP, VN, FP, FN.

		De classe	
		P	**N**
Détecté comme	**P**	VP	FP
	N	FN	VN

3.2 Algorithmes de test

Afin d'évaluer les performances de FBAT, d'autres algorithmes de prédiction ont été étudiés. Il s'agit des algorithmes les plus utilisés et les plus performants selon la littérature la plus récente [5, 35, 130] : BOSS, Hive-Cote, DTW et les deux réseaux de neurones FCN et ResNet. Les codes sources originaux ont été récupérés [1] [2]. Les architectures des réseaux FCN et ResNet qui ont été utilisées sont les mêmes que celles proposées par [130]. Pour les autres méthodes, comme proposé dans les articles originaux, le nombre de voisins de l'algorithme k-NN a été fixé à 1. Enfin, les paramètres de la méthode BOSS ont été optimisés selon la procédure utilisée par [5].

Les méthodes non-déterministes (i.e. réseaux de neurones) ont été exécutées 10 fois sur chaque jeu de données. Les résultats affichés sont la moyenne de chacune de ces exécutions.

Pour compléter les tests une méthode de type *approche série temporelle* a été utilisée (voir chapitre 2, sous-section 5.1). La méthode testée est la méthode à modèle prédictif LSTM présentée chapitre 2, sous-sous-section 5.1.1. La méthode sera désignée dans la suite de ce rapport par : LSTM. Le paramètre à fixer est la taille

1. `https://github.com/uea-machine-learning/tsml`
2. `https://github.com/hfawaz/dl-4-tsc`

de la fenêtre d'entrée et le nombre l de futures valeurs à prédire. Cette méthode est testée sur les données des vaches laitières. La taille du vecteur d'entrée est fixée à 24 (pour une journée) et le nombre de prédictions varie entre 1 et 12. Les résultats présentés sont ceux obtenus avec la meilleure valeur de l possible.

Sur les données des vaches laitières, les paramètres de FBAT ont été fixés selon les règles suivantes :

— Chaque instance du jeu de données a une taille de $n = 36$ heures, donc il faut fixer les paramètres p et q de telle sorte que $p + q = 36$.
— La fenêtre p est fixée à $p = 24$ afin d'avoir 24 h par fenêtre, soit une journée complète.
— Le décalage q est fixé à $q = 12$, ainsi les deux fenêtres sont décalées d'une demi-journée.
— Le nombre d'harmoniques z est fixé à $z = 1$ afin d'avoir la composante de période $24h$ (rythme circadien). D'autres valeurs ont été testées, $z = 1$ s'est avéré être la meilleure valeur possible.
— Le seuil est calculé en optimisant la moyenne : $\frac{rappel_- + rappel_+}{2}$. Ce n'est pas la valeur de la *precision* qui a été utilisée car les jeux de données ont des classes très déséquilibrées, la valeur du *rappel* est moins sensible à ce déséquilibre.

Sur les données UCR, les paramètres de FBAT ont été fixés selon les règles suivantes :

— Plusieurs valeurs de p et de q sont testées pour chacun des jeux de données. Soit un vecteur $\boldsymbol{p} = [\frac{n}{2}, ..., n]$ avec n la taille de la série. Pour éviter de faire trop de calculs quand les séries sont très grandes, la taille de $\boldsymbol{p}$ est fixée à 10. Pour chaque valeur de p, q est fixé à $q = n - p$.
— Pour chaque jeu de données et pour chaque valeur de p, 11 valeurs de z sont testées : $z \in [0, ..., 10]$.
— les résultats affichés sont ceux obtenus avec la configuration qui donne les meilleurs résultats.
— le seuil est calculé en optimisant la moyenne : $\frac{rappel_- + rappel_+}{2}$.

3.3 Jeux de données

Les jeux de données 1, 2, 3 et 4 obtenus en ferme ont été découpés aléatoirement en deux jeux $\mathcal{D}_{train}$ et $\mathcal{D}_{test}$ dans les proportions respectives de $\frac{2}{3}$ et $\frac{1}{3}$. Pour se rapprocher des conditions réelles, les zones anormales autours de chaque jours noté anormal (voir sous-section 2.1) n'ont pas été découpées, ainsi une anomalie est entièrement soit dans $\mathcal{D}_{train}$, soit dans $\mathcal{D}_{test}$ mais jamais dans les deux à la fois. Pour minimiser l'impact aléatoire, les jeux ont été découpés 10 fois aléatoirement et les mesures affichées dans les résultats sont la moyenne des 10 différents résultats obtenus. Les jeux de données issus de l'UCR sont déjà découpés et ce sont ces

découpages qui ont été utilisés.

Le jeu $\mathcal{D}_{train}$ est utilisé pour apprendre les réseaux FCN et ResNet, pour optimiser les paramètres de BOSS et Hive-Cote et pour calculer le seuil de FBAT.

3.4 Configuration matériel

Les expériences nécessitant un calcul CPU ont été lancées sur des machines composées d'un processeur Intel Xeon E5-2670 v2 cadencé à 2,5 GHz avec 25 Mo de cache, 62,5 Go de RAM et de 20 cœurs logiques. Les calculs GPU ont été lancés sur une machine disposant d'un GPU NVIDIA GP104GL Quatro P5000 (utilisé en simple précision) et d'un processeur Intel Xeon E5-2640 v4 condensé à 2,4 GHz avec 25 Mo de cache et 62,5 Go de mémoire RAM.

3.5 Plan expérimental

Pour étudier la méthode FBAT et la comparer aux autres méthodes, un plan expérimental en plusieurs étapes a été conçu :

1. L'algorithme FBAT ainsi que BOSS, DTW, Hive-Cote, FCN, ResNet et LSTM sont exécutés sur les jeux de données 1, 2 et 3. Les méthodes autres que FBAT n'ont pas été exécutées sur le jeu de données 4 car celui-ci est très volumineux et l'exécution de méthodes comme Hive-Cote aurait été trop chronophage. Cette étape a pour but de comparer FBAT avec les meilleures méthodes de classification sur les données de vaches laitières.

2. L'algorithme FBAT ainsi que BOSS, DTW, Hive-Cote, FCN et ResNet sont testés sur les données de l'UCR. Cette étape a pour but de tester FBAT sur d'autres jeux de données et d'étudier ses caractéristiques. La méthode LSTM n'est pas testée ici car elle n'appartient à la catégorie TSC mais plutôt à la prédiction.

3. Un seuil commun de 2000 (justification dans la sous-section 4.3) pour FBAT est fixé et FBAT est exécuté sur les jeux 1, 2, 3 et 4 sur l'intégralité des données ($\mathcal{D}_{train}$ et $\mathcal{D}_{test}$). Cette étape a pour but de déterminer quelles sont les anomalies les mieux détecter afin de prédire le comportement de FBAT en production.

4 Résultats

4.1 Comparaison de FBAT et les autres algorithmes sur les données de vaches laitières

Les mesures de *precision*, $rappel_+$, $rappel_-$ et temps de calcul pour les jeux de données 1, 2 et 3 sont illustrées respectivement par les figures 3.3, 3.4, 3.5 et 3.6. Le détail pour chacun des jeux est donné dans les sections suivantes.

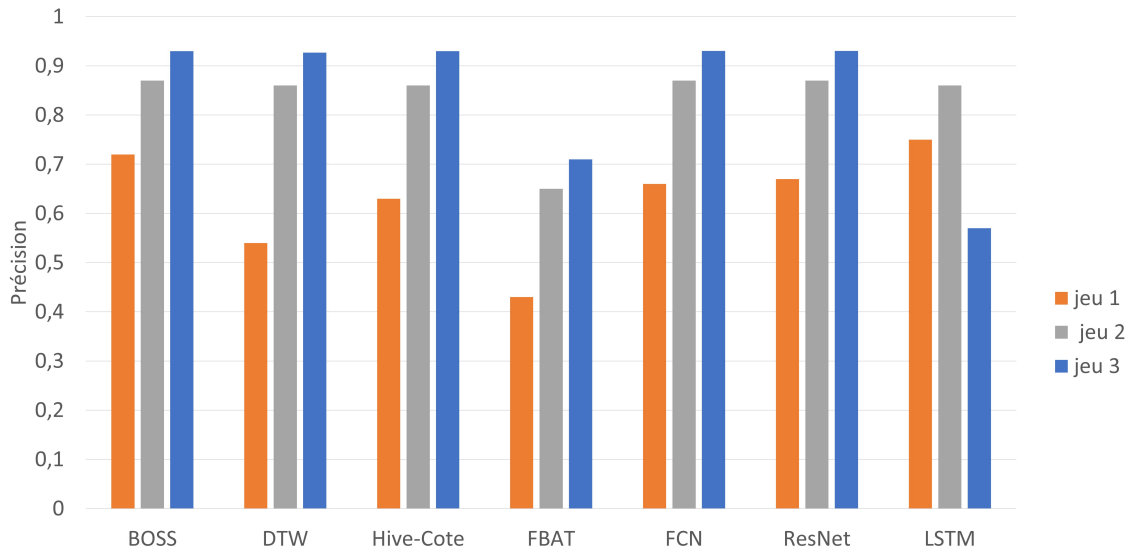


FIGURE 3.3 – Histogramme représentant la *precision* obtenue par chacune des méthodes sur les jeu de données 1, 2 et 3.

4.1.1 Jeu de données 1

Les résultats du jeu de données 1 sont décrits en Table 3.5. En terme de *precision*, c'est LSTM qui obtient le meilleure score (0.75) avec $l = 1$ (les résultats avec les autres valeurs de l sont en Annexe A, Tableau A.2). Cependant, ses valeurs moyennes $rappel_-$ et $rappel_+$ sont respectivement à 0 et 0.99. Cela signifie que que cette méthode a tendance à détecter toutes les séries comme anormales. Le jeu de données est déséquilibré, c'est à dire qu'il contient environ 3,5 fois plus de séries anormales que normales, c'est pour cela que la méthode obtient une valeur de *precision* correcte. C'est pourquoi dans le cadre de ces jeux de données, il est préférable de regarder les mesure $rappel_-$ et $rappel_+$ qui sont moins sensibles au déséquilibre des classes. D'un point de vu $rappel_-$ et $rappel_+$, Hive-Cote, FCN et ResNet ont de meilleurs

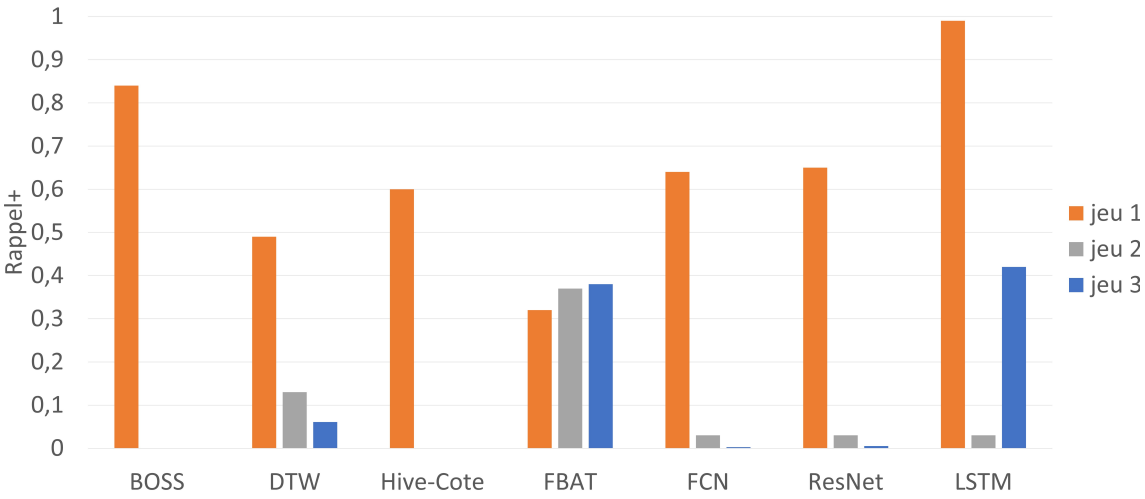


FIGURE 3.4 – Histogramme représentant le $rappel_+$ obtenu par chacune des méthodes sur les jeu de données 1, 2 et 3.

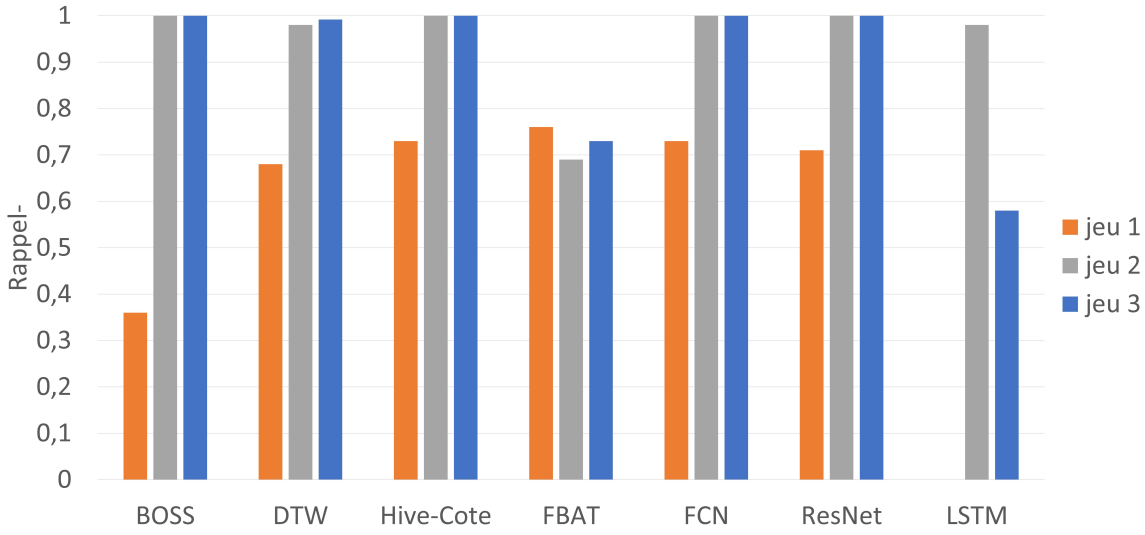


FIGURE 3.5 – Histogramme représentant le $rappel_-$ obtenu par chacune des méthodes sur les jeu de données 1, 2 et 3.

valeurs avec environ 0.72 pour le $rappel_-$ et 0.62 pour le $rappel_+$. Cependant ces trois méthodes ne sont pas les meilleurs d'un point de vue temps de calcul, notamment Hive-Cote avec 28 heures de calculs. FBAT propose un meilleur $rappel_-$ de 0.76 et un $rappel_+$ plus faible de 0.32. Cependant, FBAT a le meilleur temps de calcul avec 6 minutes. Les performances de DTW sont situées entre celles de FBAT et des réseaux de neurones. BOSS n'est pas très intéressant car malgré son bon $rappel_+$

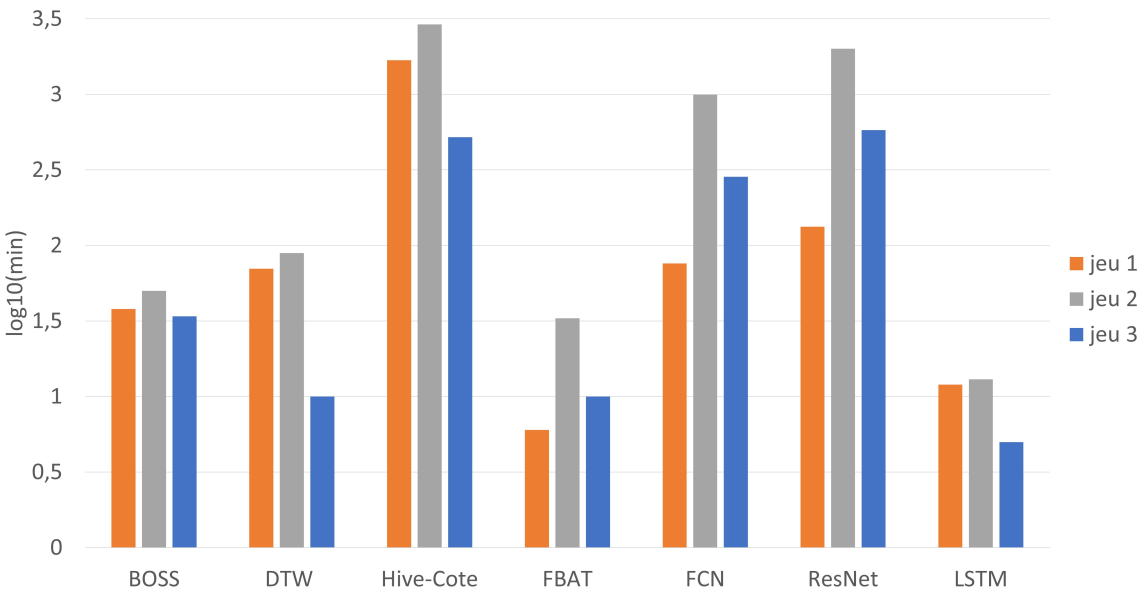


FIGURE 3.6 – Histogramme représentant le temps de calcul de chaque méthode sur les jeu de données 1, 2 et 3 ; le temps est exprimé en minutes et est illustré via une échelle logarithmique.

de 0.84, il a un assez mauvais *rappel*$_-$ de 0.36. Donc il a un taux de fausses alertes assez important.

Pour le moment, les meilleurs algorithmes d'un point de vue performance sont les réseaux de neurones ainsi que Hive-Cote et FBAT est le meilleur d'un point de vue temps de calcul.

TABLE 3.5 – Résultats de FBAT et des autres algorithmes sur le jeu de données 1 (pour LSTM, $l = 1$).

		DTW	**Hive-Cote**	**BOSS**	**FBAT**	**FCN**	**ResNet**	**LSTM**
precision		0.54	0.63	0.72	0.43	0.66	0.67	**0.75**
***precision*$_-$**		0.31	0.38	**0.43**	0.32	0.40	0.40	0.05
***rappel*$_-$**		0.68	0.73	0.36	**0.76**	0.73	0.71	0
***precision*$_+$**		0.82	0.87	0.80	0.80	**0.88**	0.87	0.75
***rappel*$_+$**		0.49	0.60	0.84	0.32	0.64	0.65	**0.99**
temps	CPU	1h10	28h	0h38	**0h06**	19h28	16h36	0h12
	GPU	-	-	-	-	1h16	2h13	-

4.1.2 Jeu de données 2

Les résultats du jeu de données 2 sont décrits en Table 3.6. Les meilleurs résultats de LSTM sont obtenus avec $l = 1$ et les résultats avec les autres valeurs de l sont en Annexe A, Tableau A.4). Toutes les méthodes ont une *precision* de 0.86 ou 0.87, excepté FBAT qui a une *precision* de 0.65. Cependant, ce jeu de données est aussi déséquilibré avec environ 6,6 fois plus de séries normales que d'anormales. Donc un classifieur qui classerait toute série comme normale, aurait quand même une très bonne *precision*. C'est pourquoi pour ce jeu de données il faut également regarder les valeurs du $rappel_-$ et $rappel_+$. Les méthodes Hive-Cote, BOSS, FCN, ResNet et LSTM ont un $rappel_-$ égal ou proche de 1 mais ont un $rappel_+$ égal ou proche de 0. Autrement dit, elles ne détectent aucune anomalie et classent toutes les séries comme normales. On peut donc conclure que ces cinq méthodes sont inefficaces pour ce jeu de données. DTW a un $rappel_+$ un peu plus élevé de 0.13. FBAT garde sensiblement les mêmes performances que pour le jeu de données 1, à savoir un $rappel_+$ de 0.37 et un $rappel_-$ de 0.69. Du point de vue du temps de calcul, LSTM a le meilleur temps avec 13 minutes.

Par rapport à ses performances, FBAT est la meilleure méthode pour ce jeu de données. Concernant le temps de calcul, LSTM est la meilleure méthode, bien que FBAT soit juste derrière avec 33 minutes.

TABLE 3.6 – Résultats de FBAT et des autres algorithmes sur le jeu de données 2 (pour LSTM, $l = 1$).

		DTW	**Hive-Cote**	**BOSS**	**FBAT**	**FCN**	**ResNet**	**LSTM**
precision		0.86	0.86	**0.87**	0.65	**0.87**	**0.87**	0.86
$\boldsymbol{precision_-}$		**0.88**	0.87	0.87	**0.88**	0.87	0.87	0.87
$\boldsymbol{rappel_-}$		0.98	**1**	**1**	0.69	**1**	**1**	0.98
$\boldsymbol{precision_+}$		0.46	0.67	0.22	0.15	0.99	**1**	0.21
$\boldsymbol{rappel_+}$		0.13	0	0	**0.37**	0.03	0.03	0.03
temps	CPU	1h29	48h20	0h50	0h33	-	-	**0h13**
	GPU	-	-	-	-	16h35	33h22	-

4.1.3 Jeu de données 3

Les résultats du jeu de données 3 sont décrits en Table 3.7. Les meilleurs résultats de LSTM sont obtenus avec $l = 1$ et les résultats avec les autres valeurs de l sont en Annexe A, Tableau A.6). En termes de *precision*, toutes les méthodes sauf LSTM ont d'excellents chiffres, au minimum 0.71 pour FBAT et jusqu'à 0.93 pour DTW,

Hive-Cote, BOSS, FCN et ResNet. Cependant, comme pour les jeux de données 1 et 2, les classes sont déséquilibrées : 13,5 fois plus de séries normales que d'anormales. Comme pour le jeu de données 2, la plupart des méthodes gèrent assez mal ce déséquilibre, la plupart des méthodes ont un $rappel_-$ proche ou égal à 1 et un $rappel_+$ proche ou égal à 0 sauf pour LSTM et FBAT. Les méthodes LSTM et FBAT ont un $rappel_+$ respectif de 0.42 et 0.38. La différence entre LSTM et FBAT se fait sur le $rappel_-$, 0.73 pour FBAT et seulement 0.58 pour LSTM. Donc FBAT et LSTM ont un taux de détection similaire mais LSTM produit plus de faux positifs que FBAT. Les autres méthodes n'arrivent pas à détecter les anomalies ($rappel_+$ égal ou proche de 0). Concernant le temps de calculs, LSTM et FBAT sont toujours les meilleurs avec respectivement 5 et 10 minutes.

Niveau performances, FBAT reste la meilleure méthode pour ce jeu de données et est l'une des méthodes les plus rapides.

TABLE 3.7 – Résultats de FBAT et des autres algorithmes sur le jeu de données 3 (pour LSTM, $l = 1$).

		DTW	**Hive-Cote**	**BOSS**	**FBAT**	**FCN**	**ResNet**	**LSTM**
precision		**0.93**	**0.93**	**0.93**	0.71	**0.93**	**0.93**	0.57
***precision*$_-$**		0.93	0.93	0.93	**0.94**	0.93	0.93	**0.94**
***rappel*$_-$**		0.99	**1**	**1**	0.73	**1**	1	0.58
***precision*$_+$**		**0.37**	0	0	0,1	0.20	0.25	0.06
***rappel*$_+$**		0.06	0	**0**	**0.38**	0.003	0.005	0.42
temps	CPU	0h10	8h40	0h34	0h10	-	-	**0h5**
	GPU	-	-	-	-	4h45	9h40	-

4.1.4 Bilan

Le bilan des méthodes varie selon le jeu de données et seule FBAT a des performances identiques pour les 3 jeux de données en terme de $rappel_-$ et $rappel_+$. Pour les jeux de données 2 et 3 FBAT s'est montré plus efficace, cependant a été battu pour le jeu 1. Il est également apparu que les méthodes (sauf FBAT) sont relativement sensibles au déséquilibre des classes. Les modèles que FBAT calcule sont basés sur une seule vache et sur une très courte durée et seul le calcul du seuil nécessite tout le jeu de données. C'est sans doute pour cette raison que FBAT n'est pas sensible au déséquilibre des classes et que ses performances varient très peu d'un jeu de données à l'autre.

Donc, pour détecter des anomalies chez les vaches laitières, FBAT semble être la meilleure pour plusieurs raisons :

1. elle a les meilleures performances en terme de $rappel_-$ et $rappel_+$ sur deux des trois jeux de données,
2. elle a des performances stables sur les trois jeux de données, ce qui est un avantage pour la mise en production (comportement prévisible),
3. c'est une méthode très rapide, ce qui rend possible les détections en temps réel.

FBAT a un bon $rappel_-$ (environ 0.70) mais un $rappel_+$ plutôt faible (environ 0.35). D'un autre côté, dans les jeux de données, chaque jour noté anormal a donné lieu à une zone anormale plus ou moins grande selon le type de l'anomalie, ceci pour être certain de capturer l'anomalie (voir sous-section 2.1). Cependant, il est probable que la vache n'ait pas un comportement anormal tout au long de cette période anormale et donc cela créerait automatiquement un nombre important de faux négatif et donc un $rappel_+$ plutôt bas. La question est donc de savoir si, même avec son $rappel_+$ assez faible, FBAT est en mesure de détecter la plupart des anomalies. Une autre question est importante : est-ce que les anomalies sont détectées de manière précoce ou non ? Répondre à ces questions est primordial pour évaluer l'efficacité de FBAT, elles sont étudiées en sous-section 4.3.

Par ailleurs, en amont de cette expérimentation, plusieurs valeurs de z ont été testées. Il est apparu que, pour tous les jeux de données, $z = 1$ soit la meilleure valeur. Cela correspond à l'harmonique de période 24 h et donc au cycle circadien. L'expérimentation de cette méthode a donc validé les observations des biologistes sur le rythme circadien et de son importance.

Dans la section suivante, FBAT est comparé aux autres méthodes sur d'autres données qui ne sont pas issues de vaches laitières, afin d'évaluer son comportement selon les caractéristiques des données.

4.2 Comparaison de FBAT et les autres algorithmes sur les données de l'UCR

Dans cette section, FBAT est comparé avec d'autres algorithmes sur des jeux de données issus de l'UCR. Il y en a de plusieurs types et les résultats sont présentés en Table 3.8. Sur l'ensemble des jeux de données, FBAT obtient une précision de $0,65$ ou plus ; pour 9 des 11 jeux de données, elle obtient une précision de plus de $0,83$ et pour plus de la moitié des jeux de données, elle obtient une précision de 90%. Avec une précision de $0,83$, FBAT obtient les meilleurs performances sur le jeu de données *Earthquakes*. Sur les jeux de données *MoteStrain* et *SonyAIBORobotSurface1* FBAT obtient respectivement $0,90$ et $0,95$ ce qui lui permet d'être meilleur que les méthodes BOSS, Hive-Cote et DTW, les réseaux de de neurones FCN et ResNet obtiennent des résultats comparables voir légèrement au-dessus, FCN obtient

deux points supplémentaires sur *MoteStrain* (0, 92) et seulement un point de plus sur *SonyAIBORobotSurface1* (0, 96). Sur le jeu *ItalyPowerDemand*, FBAT obtient 0, 93 ce qui lui permet d'être meilleur que BOSS mais moins bon que les autres qui obtiennent une précision entre 0, 95 et 0, 97. Sur le jeu *ECG200* FBAT obtient une précision de 0, 85 ce qui lui permet d'être meilleur que DTW (0, 77) moins bon que les autres méthodes qui obtiennent jusqu'à 0, 89 de précision. FBAT obtient un très bon score de 0, 96 sur le jeu *ECGFiveDays*, ça lui permet de battre DTW (0, 77), cependant BOSS et Hive-Cote arrivent à obtenir une précision parfaite de 1. FBAT obtient également une excellente précision de 0, 96 sur le jeu *TwoLeadECG*, ça lui permet également d'être meilleur de DTW (0, 90) mais les réseaux de neurones obtiennent une précision de 1. Sur le jeu *Lightning2*, FBAT obtient une précision de 0, 83 ce qui lui permet d'être meilleur que toutes les autres méthodes, excepté DTW qui obtient une précision de 0, 87. Sur les jeux de données *WormsTwoClass* et *Yoga* FBAT n'arrive pas à être compétitif avec une précision respectivement de 0, 65 et 0, 67.

FBAT obtient des résultats décevants sur deux jeux de données mais produit des résultats honorables sur les autres jeux, en particulier sur un jeu où elle est très performante. Pour les autres jeux de données, FBAT arrive à rester dans la moyenne ou même être proche des meilleures méthodes. De plus, FBAT est une méthode relativement simple à mettre en œuvre avec peu de paramètres à optimiser et un temps de calculs faible (voir section précédente). Cependant, FBAT possède deux limites. La première est qu'il est préférable que le jeu de données soit construit de telle manière que chaque série possède plus d'une période. En effet, FBAT se focalise sur le changement de cycle ; si les séries ne possèdent qu'un seul cycle, FBAT ne pourra pas détecter ce changement. Ceci peut en partie expliquer certains mauvais résultats obtenus : les données de l'UCR ne sont pas construites selon ce paramètre. La seconde limite est qu'il faut que les données aient un cycle et que les anomalies se manifestent via un changement de cycle. Aussi FBAT n'est-elle pas adaptée à tous les jeux de données.

La Table 3.9 affiche les paramètres obtenus de FBAT avec les meilleurs résultats. La première observation à noter est que la taille de p et de q dépend du jeu de données, aucune règle ne semble se dégager. Cela est plutôt logique dans le sens où chaque jeu de données possède des périodes de cycles différentes. Concernant le nombre d'harmoniques z, là aussi cela dépend du jeu de données. Pour rappel, ce nombre influe sur la précision du modèle, plus z est grand, plus le modèle est proche de l'original mais plus il est petit, plus le bruit est atténué. Il faut quand même noter que seuls deux jeux de données nécessitent un z élevé (9 ou 10), la plupart des jeux de données nécessitent un z inférieur ou égal à 4. Cela signifie que la plupart des jeux de données sont bruités et que supprimer plusieurs harmoniques est nécessaire.

Ce qui est le plus important c'est que les meilleures valeurs de paramètres dé-

TABLE 3.8 – Précision des différentes méthodes selon le jeu de données.

Dataset	FBAT	BOSS	Hive-Cote	DTW	FCN	ResNet
Earthquakes	**0,83**	0,75	0,75	0,72	0,73	0,71
FreezerSmall-Train	0,93	0,95	**0,98**	0,76	0,69	0,90
ItalyPower-Demand	0,93	0,86	**0,97**	0,95	0,96	**0,97**
MoteStrain	0,90	0,84	0,87	0,84	**0,92**	0,91
SonyAIBORobot-Surface1	0,95	0,61	0,74	0,73	**0,96**	0,95
ECG200	0,85	0,86	0,86	0,77	**0,89**	**0,89**
ECGFiveDays	0,96	**1,00**	**1,00**	0,77	0,98	0,98
TwoLeadECG	0,96	0,98	0,99	0,90	**1,00**	**1,00**
WormsTwoClass	0,65	**0,81**	0,77	0,62	0,75	0,77
Lightning2	0,83	0,80	0,80	**0,87**	0,74	0,75
Yoga	0,67	**0,91**	**0,91**	0,84	0,84	0,88

pendent du jeu de données et que la meilleure valeur de z a une grande chance d'être inférieure à 5. L'idéal est donc de tester plusieurs valeurs. Le fait de faire une analyse des séries afin de connaître leurs cycles est un plus.

4.3 Test FBAT

Le but de cette section est d'approfondir les résultats de FBAT sur les jeux de données issus des vaches laitières. Cette section répond à deux grandes questions :

1. Est-ce que malgré son $rappel_+$ plutôt bas, FBAT est en mesure de détecter la plupart des perturbations ?
2. Est-ce que FBAT détecte les anomalies plutôt au début de le période jugée anormale ou plutôt à la fin ?

Pour cela, FBAT a été lancé sur les quatre jeux de données avec un seuil commun $z = 2000$. Ce choix se justifie pour deux raisons. La première est que sur l'ensemble des quatre jeux de données, les seuils optimaux sont presque les mêmes (voir Table 3.10) et que la moyenne des quatre seuils donne un chiffre proche de 2000 (1986). La deuxième raison est que s'il est possible d'utiliser le même seuil pour chaque ferme, cela donne un avantage de plus à FBAT pour la mise en production. En effet, l'étape d'apprentissage pourrait possiblement devenir superflue et donc la mise en production pourrait gagner en simplicité et rapidité.

Les résultats avec le seuil $z = 2000$ ont été calculés sur l'ensemble de chaque jeu de données ($\mathcal{D}_{train} \cup \mathcal{D}_{test}$) afin d'avoir toutes les anomalies présentent pour les tests.

TABLE 3.9 – Paramètres p, q et z des meilleurs résultats de FBAT sur les différents jeux de données de l'UCR.

Jeux de données	p	q	z
Earthquakes	406	106	1
FreezerSmallTrain	301	1	3
ItalyPowerDemand	17	7	1
MoteStrain	58	26	4
SonyAIBORobotSurface1	65	5	3
ECG200	56	40	9
ECGFiveDays	134	2	10
TwoLeadECG	61	21	3
WormsTwoClass	765	135	1
Lightning2	442	195	0
Yoga	255	171	1

TABLE 3.10 – Valeurs du seuil z de FBAT en fonction du jeu de données.

Jeu de données	**Seuil z**
1	1947
2	2216
3	1894
4	1886
Moyenne	1986

TABLE 3.11 – Résultats obtenues par FBAT pour sur les jeux de données 1, 2, 3 et 4 avec un seuil $z = 2000$.

Jeu de données	**1**	**2**	**3**	**4**
temps test	11s	26s	7s	8min46
precision	42,5	65,6	75,7	66,5
***precision*$_+$**	80,8	15,3	9,6	28,9
***precision*$_-$**	21,7	87,8	93,8	75,5
***rappel*$_+$**	30,8	35,7	29,8	21,9
***rappel*$_-$**	77,9	70,1	79,1	81,7

4.3.1 Nombre d'anomalies détectées

Le but est de savoir si FBAT est en mesure de détecter toutes les perturbations et à partir de quel moment elle les détecte. Pour cela FBAT a été lancé sur les 4 jeux de données issus des vaches laitières avec un seuil commun de $z = 2000$. Les résultat sont affiché en Table 3.11. La première chose à noter est le temps relativement très court du test, seulement quelques secondes, excepté pour le jeu de données 4 qui est composé de 300 vaches sur une année. Ainsi pour une ferme de 1000 vaches (ce qui est bien au-delà de la taille moyenne des fermes françaises), il faut faire 1000 tests par heure, soit beaucoup moins que dans le plus petit jeu testé ici (jeu 3, 22568 tests) qui n'a demandé que 7 s. Donc, pour utiliser FBAT, une ferme n'aurait pas besoin d'un super ordinateur, un ordinateur personnel suffira, ce qui est un avantage pour FBAT. Les valeur de *precision*, *precision*$_+$ et *precision*$_-$ dépendent fortement du jeu de données et surtout de la répartition du nombre de séries normales et anormales du jeu de données. La valeur du *rappel*$_-$ oscille entre $0,70$ et 0.82. La valeur du *rappel*$_+$ oscille entre 0.22 et 0.36.

Pour savoir si les faibles valeurs de *rappel*$_+$ sont suffisantes ou non (c'est-à-dire si FBAT détecte en pratique la plupart des anomalies), la Table 3.12 regroupe le pourcentage d'anomalies détectées par jeu de données. Chaque jour noté anormal donne lieu à une zone de plusieurs jours anormaux (Table 3.1) qui elle-même donne lieu à plusieurs séries temporelles. La détection se fait si FBAT arrive à détecter au moins une anomalie de comportement au sein d'une zone définie autour d'une perturbation

Concernant les accidents, seulement présents dans le jeu de données 4, FBAT arrive à tous les détecter, ce qui est assez logique. En effet, un accident vient perturber de manière abrupte l'activité de l'animal ce qui rend leur détection plus facile.

Les vêlages sont aussi très bien détectés que ça soit dans le jeu 2 et le jeu 4. Ce résultat est également assez évident, un vêlage perturbe énormément l'activité de l'animal : la vache reste couchée et s'alimente moins. De plus il est possible qu'elle soit retirée de son groupe pour être placée dans un parc de vêlage, ce qui est

également susceptible de modifier son activité.

Concernant les œstrus, ceux des jeux de données 1, 2 et 4 sont très bien détectés avec un taux de détection supérieur à 85%. En revanche, ceux du jeu de données 3 sont moins bien détectés, seulement $69,2\%$, ce qui peut paraitre étonnant car c'est un jeu de données spécifiquement créé pour détecter les œstrus via le profil de progestérone. Néanmoins, ce chiffre peut s'expliquer par les chaleurs silencieuses. En effet certaines chaleurs se répercutent sur la progestérone sans impacter l'activité de l'animal. Il est donc fort probable que plusieurs œstrus détectés par la progestérone ne le soient pas par FBAT car cette méthode ne se base que sur le comportement animal.

Les boiteries sont très bien détectées à plus de 93%. Les autres maladies sont aussi très bien détectées à plus de 75%.

Aucune mammite n'est détectée pour le jeu de données 1. Cependant, il faut noter que le jeu 1 ne contient que 3 mammites donc on ne peut pas être certain que cela soit significatif. D'autant plus que pour le jeu 2, FBAT arrive a détecter les 9 mammites et FBAT arrive également à détecter $87,5\%$ des 32 mammites du jeu 4.

Les injections LPS sont assez bien détectées ($81,5\%$). L'administration de LPS semble donc fortement perturber l'animal. En effet, le LPS induit une inflammation générale qui entraine de la fièvre et une augmentation du temps passé couché [8].

Les acidoses sont moyennement bien détectées (69%). Les acidoses ont été étudiées uniquement dans le jeu de données 1 où le pH du rumen était enregistré à l'aide d'un capteur (pour détecter l'acidose) et certaines vaches recevaient un régime acidogène. Chez les vaches recevant le régime acidogène, les épisodes d'acidose étaient fréquents et pouvaient durer plusieurs jours. Or FBAT détecte les changements de rythmes. Si un rythme anormal se prolonge, FBAT ne le détectera pas. On peut faire l'hypothèse que les premiers jours où l'animal est en acidose sont détectés mais ensuite l'activité de l'animal est anormale mais stable et donc FBAT ne détectera plus de changement de rythme.

Concernant les changements de parc et les autres perturbations, leur détection n'est pas autant importante que les autres. En effet, ce sont des perturbations qui sont déjà connues par l'éleveur. Cependant leur détection peut être intéressante dans le sens où elle peut aider l'éleveur à établir l'état de bien-être de l'animal, plus son rythme s'agite, plus l'évènement est marquant pour l'animal. Les changements de parc ainsi que les autres perturbations sont moyennement bien détectés par FBAT avec une détection qui oscille entre 68% et 72% voire seulement $59,3\%$ pour les autres perturbations du jeu de données 4. La catégorie "autres perturbations" regroupe des événements de nature variée, tous ne se prêtent pas à un changement de rythme important.

Pour conclure, la plupart des évènements sont très bien détectés par FBAT. Pour savoir si FBAT est une méthode qui peut vraiment être utile en production,

TABLE 3.12 – Pourcentage d'anomalies détectées par FBAT selon le jeu de données et le type de l'anomalie. Le caractère "-" signifie que l'anomalie n'était pas présente dans le jeu de données.

Évènement	**Jeu de données**			
	1	**2**	**3**	**4**
Accidents[1]	-	-	-	100
Vêlages	-	100	-	99,4
Œstrus[2]	85,7	95,1	69,2	91,4
Boiteries	93,8	100	-	98,2
Mammites	0	100	-	87,5
Autres maladies[3]	75	80	-	90,9
Injection LPS[4]	-	81,5	-	-
Acidoses[5]	69	-	-	-
Changement parc	-	68,3	-	-
Autres pertubations[6]	71,7	69	-	59,3

[1] blessures, rétention placentaire, lacérations vaginales

[2] détectées par l'éleveur ou par le profile progestérone dans le jeu 3

[3] diarrhée, coliques, ingestion d'un corps étranger, avortement, kyste ovarien, baisse d'appétit, apathie, animal qui s'isole, perte de poids, chute de production laitière, fièvre de lait, acétonémie, affection respiratoire, autre maladie infectieuse (grippe, ...)

[4] injections LPS injectées dans la glande mammaire

[5] détectées par analyse du pH ruminal

[6] interventions, e.g. vaccinations, synchronisation des chaleurs, vermifugation, parage des onglons, changement de parc

il faut savoir si elle permet une détection précoce de l'anomalie ou au contraire une détection tardive.

4.3.2 Période de détection

Les Figures 3.7, 3.8, 3.9, 3.10, 3.11 et 3.12 représentent le taux de détection (parmi les anomalies que FBAT arrive à détecter) des principales anomalies en fonction de l'heure. Le taux est cumulé, c'est-à-dire qu'une fois l'anomalie détectée,

elle est comptabilisée jusqu'à la fin, ainsi la courbe est forcément croissante. Pour être dans des conditions réelles, l'heure détectée correspond à la dernière heure de la série de 36 heures, par exemple une série détectée comme anormale allant de l'heure 0 à l'heure 35 sera comptabilisée pour l'heure 35. L'heure 0 correspond à la première heure du jour détectée comme anormal par l'éleveur. Toutes courbes sont calculées par rapport au jeu de données 4 car c'est le plus volumineux de tous et qu'il est issu d'une ferme commerciale, ainsi les résultats obtenus reflètent au mieux les conditions réelles. Les courbes sur les injections LPS et les acidoses sont calculées à partir des jeux 1 et 2 car ces anomalies ne sont pas présentes dans le jeu 4.

La Figure 3.7 représente les détections des boiteries du jeu de données 4. Les résultats sont plutôt encourageants, la plupart des anomalies sont détectées très tôt. En effet, environ 80% des boiteries sont détectées 15 h avant le jour noté par l'éleveur et plus de 65% sont détectées 24 h avant.

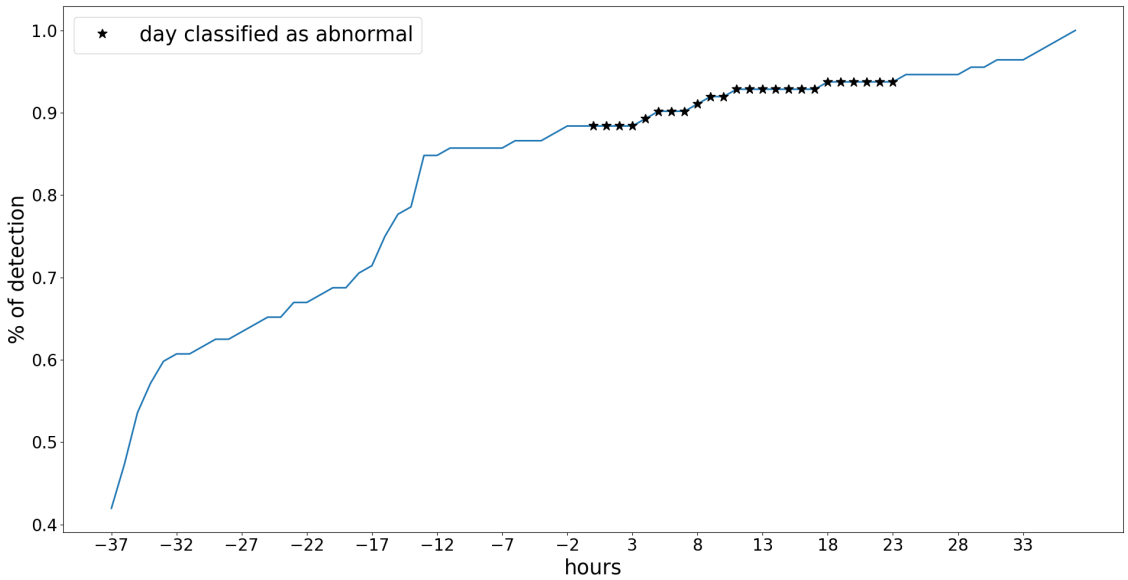


FIGURE 3.7 – Détection des boiteries du jeu de données 4 avec le seuil à 2000 (TRAIN + TEST)

La Figure 3.8 représente les détections des mammites du jeu de données 4. Les résultats sont également très bons. Plus de 70% des mammites détectées par FBAT le sont 24 h avant le jour noté par l'éleveur.

La Figure 3.9 représente les détections des autres maladies du jeu de données 4. Les résultats également bons. Plus de 70% des autres maladies détectées par FBAT le sont 24 h avant le jour noté par l'éleveur et 90% environ 8h avant.

La Figure 3.10 représente les détections des œstrus du jeu de données 4. Les résultats sont moins bons. Le taux de détection est plus progressif. Cela dit, plus de

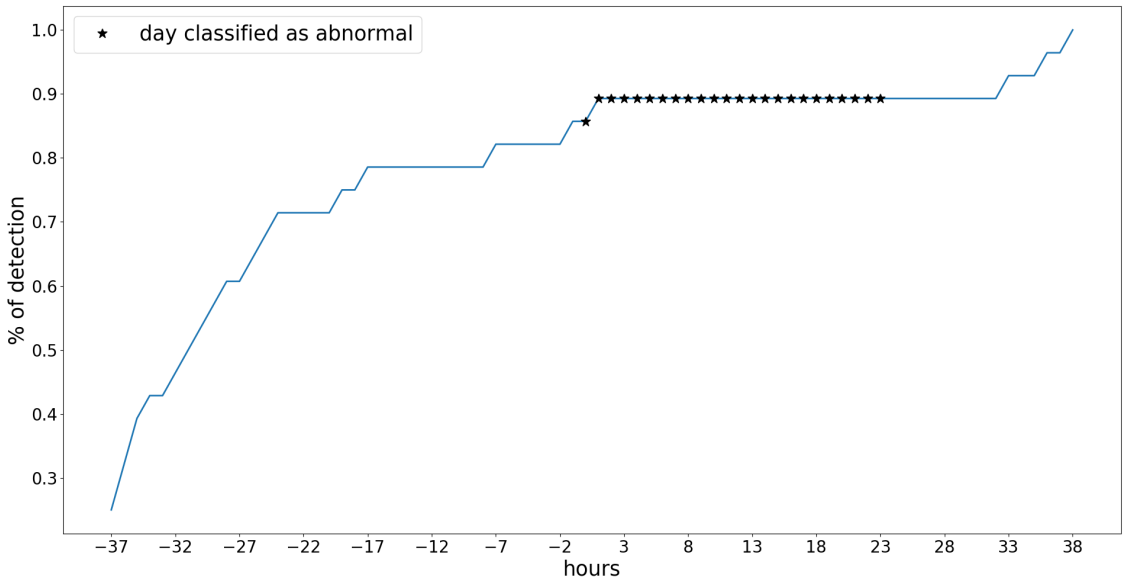


FIGURE 3.8 – Détection des mammites du jeu de données 4 avec le seuil à 2000 (TRAIN + TEST)

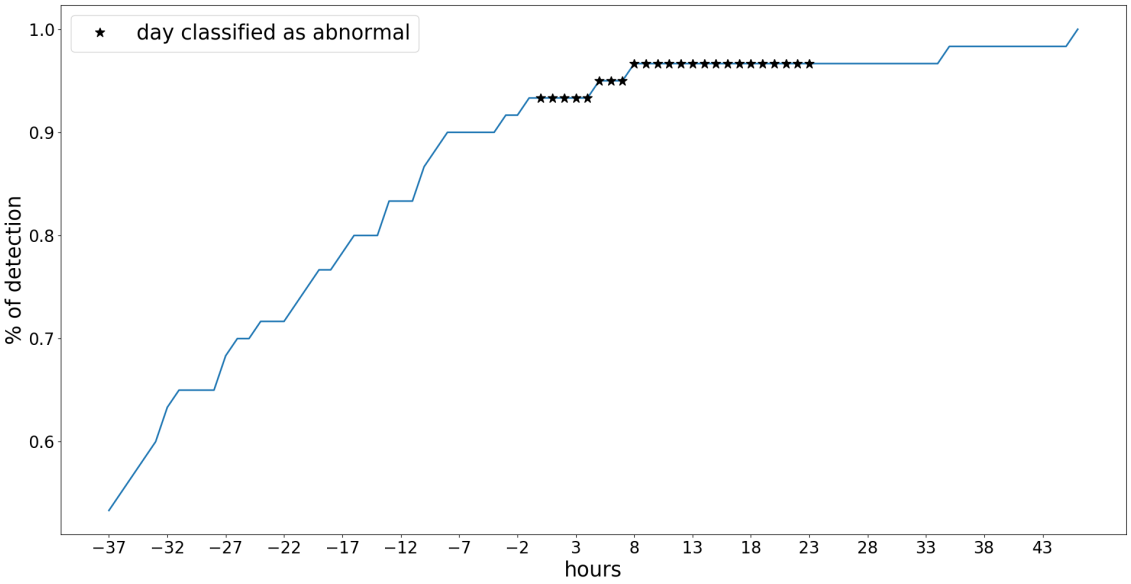


FIGURE 3.9 – Détection des autres maladies du jeu de données 4 avec le seuil à 2000 (TRAIN + TEST)

70% des œstrus détectés par FBAT le sont avant midi du jour noté par l'éleveur.

La Figure 3.11 représente les détections des acidoses du jeu de données 1. On

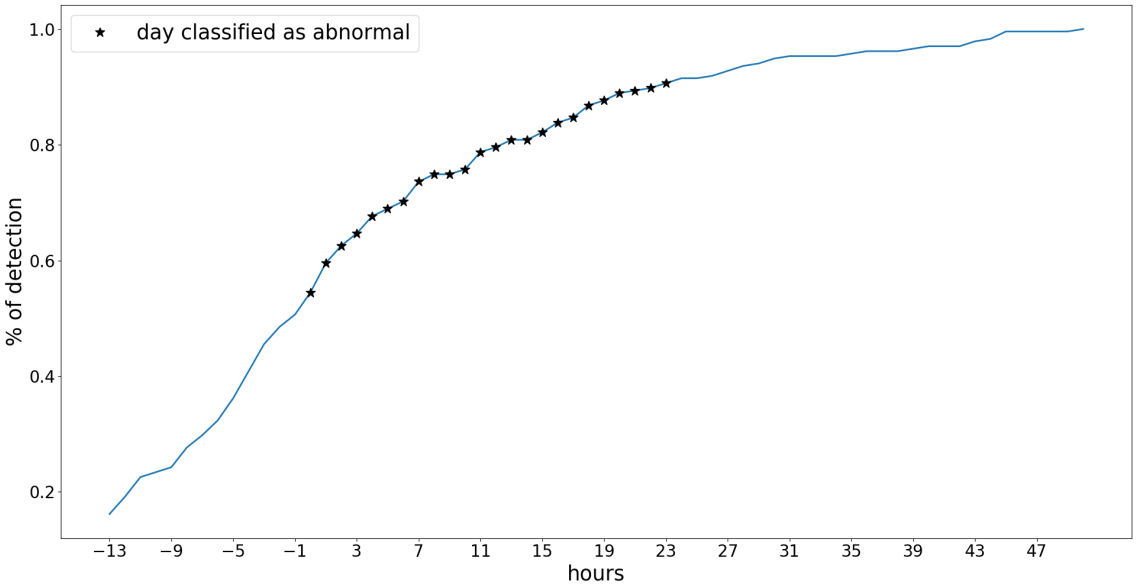


FIGURE 3.10 – Détection des chaleurs du jeu de données 4 avec le seuil à 2000 (TRAIN + TEST)

voit que le rythme est détecté anormal à 11 h, soit juste après le repas du matin. Or l'acidose provient d'une trop grande ingestion de concentré. Cela voudrait dire que les signes sont très précoces et que FBAT arrive à les détecter. Il a été observé par ailleurs que les jours où elles sont détectées en acidose, les vaches mangent moins longtemps mais plus de concentré et cette modification de comportement est vraisemblablement à l'origine de l'acidose (Silberberg et al., en préparation). Il est probable que FBAT détecte les modifications de comportement qui cause l'acidose.

La Figure 3.12 représente les détections des injections LPS du jeu de données 2. La courbe ressemble à ce que l'on pouvait s'attendre : pas de détections jusqu'à l'heure de l'injection (visiblement 11h) puis une hausse rapide de détections.

4.3.3 Bilan

Dans cette partie il a été démontré que FBAT est capable de détecter la plupart des anomalies et ce de manière précoce (sauf pour les œstrus qui sont bien détectées mais pas de manière précoce). Le nombre de détections est relativement stable d'un jeu de données à l'autre sauf pour quelques cas qui peuvent s'expliquer. Les anomalies les plus importantes sont souvent détectées plusieurs heures avant l'observation des symptômes cliniques, sauf les œstrus qui ont du mal à être détectées très tôt.

Les résultats les plus représentatifs de la réalité sont ceux obtenus par le jeu de données 4 car il s'agit d'une ferme commerciale réelle. Toutes les anomalies pré-

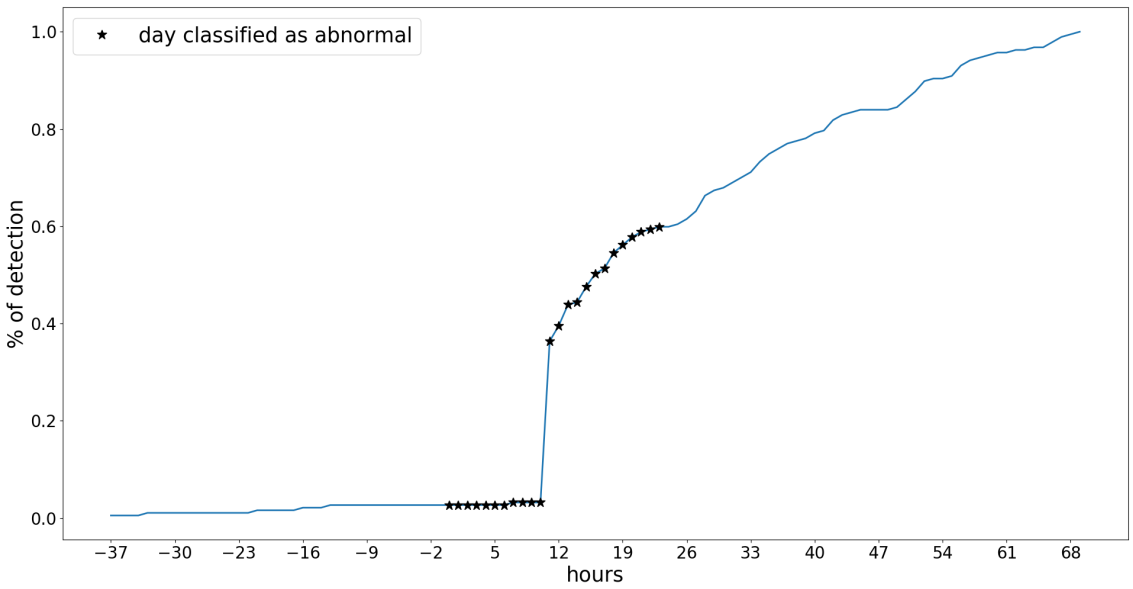


FIGURE 3.11 – Détection des acidoses du jeu de données 1 avec le seuil à 2000 (TRAIN + TEST)

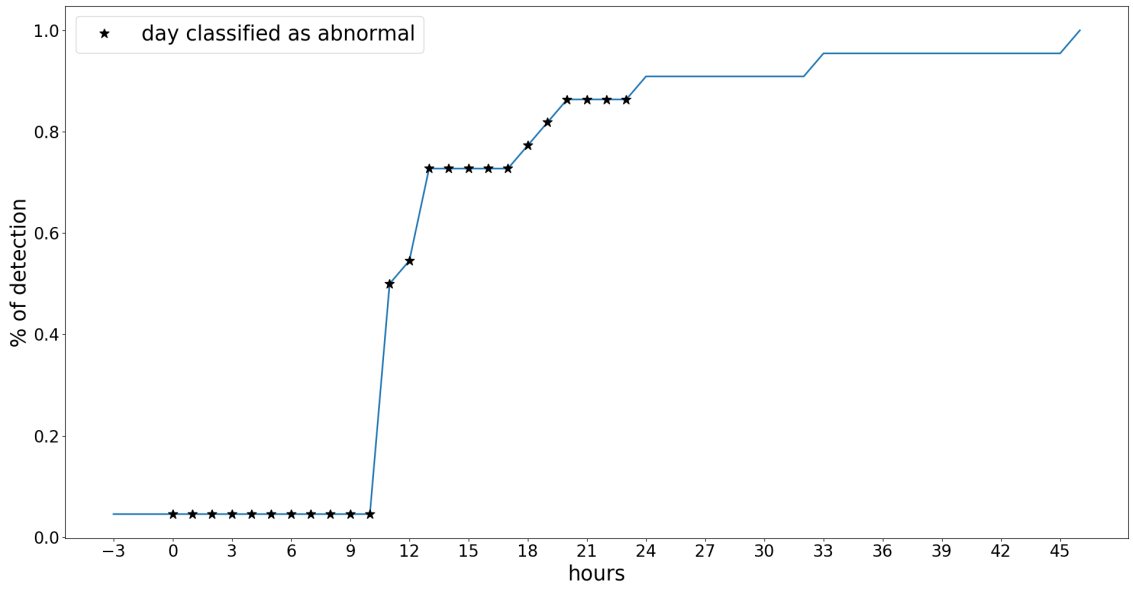


FIGURE 3.12 – Détection des injections LPS du jeu de données 2 avec le seuil à 2000 (TRAIN + TEST)

sentent dans ce jeu de données ont un taux de détection supérieur à 87%, sauf pour la catégorie des autres perturbations qui obtient un score de 59, 3% de détection.

Cependant cette catégorie correspond à des évènements liés au parc et donc leur bonne détection n'est pas très importante car l'éleveur est déjà informé de ce type d'évènements.

5 Conclusion

Dans ce chapitre, il a été démontré que FBAT propose de bonnes performances pour détecter des anomalies dans l'activité de vaches laitière. La méthode FBAT a été comparée avec les meilleures méthodes de TSC de la littérature récente. Ces tests ont montré que les performances de FBAT sont meilleures que celles des autres méthodes. Cependant, lorsqu'il s'agit de classifier d'autres types de données, FBAT se montre moins performantes.

FBAT a plusieurs avantages :

— Elle est rapide et pour tester les animaux, elle ne demande que peu de ressources. C'est un avantage pour la mise en production : il n'y a pas besoin d'un serveur pour faire les calculs.
— Elle a un taux de fausses alertes bien inférieur aux autres méthodes.
— Ses performances restent globalement identiques, quelque-soit le jeu de données. Cela est un avantage pour la mise en production : les résultats obtenus dans une nouvelle ferme sont prévisibles.
— Son taux de détection des anomalies est très élevé, notamment sur la ferme commerciale où le taux de détection selon les anomalies varie entre 87% et 100% (sauf pour les évènements liés au parc).
— La plupart des anomalies sont détectées de manière précoce.
— Les variabilités entre jour et entre animaux n'impactent pas la détection (grâce aux modèles sur une vache et sur une courte période). Donc, les performances resteront stables dans le temps et ce même si de nouvelles vaches arrivent, pas besoin de refaire un apprentissage.
— Le seuil calculé est sensiblement le même pour toutes les fermes testées. Il semblerait qu'un seuil commun $z = 2000$ donne de très bons résultats. Cela voudrait dire que la mise en production serait encore plus facilitée : plus besoin de période d'apprentissage où l'éleveur serait obligé de noter les anomalies à la main.
— Les performances de la méthode peuvent s'adapter à la ferme et à la vache. En effet, avec le temps l'éleveur pourrait avoir la possibilité d'ajuster le seuil à la vache. Par exemple, si une vache est peu sensible aux anomalies, le seuil peut être baissé pour cet animal là (et inversement). Cela augmenterait le coté adaptatif de la méthode.

FBAT a beaucoup d'avantages mais aussi quelques défauts :

— La plupart des anomalies sont détectées de manière précoce mais pas les

œstrus.

— FBAT ne propose pas de multi-classes. En effet, la détection se fait de manière global (normal VS anormal) mais elle n'est pas capable de détecter de quelle anomalie il s'agit exactement.
— Il existe encore une variabilité de sensibilité entre vaches. En effet les modèles sont calculés à la vache mais le seuil lui est toujours calcule de manière globale. Les animaux ne sont pas tous égaux face à une anomalie, certains vont être facilement perturbés, d'autre beaucoup moins. FBAT ne prend pas en compte cela. Néanmoins, comme expliqué dans les points positifs, on peut tout à fait imaginer un système de seuil qui pourrait être ajusté à la vache par l'éleveur. Cela résoudrait une partie du problème.

La méthode FBAT est, malgré ses quelques défauts, une très bonne méthode pour détecter des anomalies chez les vaches laitières. Non seulement elle propose de très bonnes performances théoriques mais elle a plusieurs avantages liés à la mise en production. Cependant, le travail peut être poursuivi et plusieurs perspectives peuvent être envisagées :

— Il faudrait tester FBAT en production et cela dans le plus de fermes possibles. Cela permettrait de valider les résultats théoriques et aussi de pointer les éventuels problèmes liés à la production.
— Il serait également bien d'ajouter le multi-classes. En effet, cela aiderait encore plus l'éleveur si la méthode était capable de dire de quelle anomalie il s'agit. Pour cela, une piste envisagée est de regarder plus en détails les décomposées de Fourier avant la création des modèles. On peut faire l'hypothèse que selon le type de l'anomalie, l'impact sur son activité sera différent. Donc en analysant les autres rythmes grâce aux harmoniques, cela pourrait être possible.
— Tous les animaux ne régissent pas de la même façon à une anomalie. Comme expliqué dans les points négatifs, FBAT ne prends pas en compte cela. Une piste envisagée dans les points positif consiste à autoriser l'éleveur à ajuster ce seuil de manière manuelle pour chacune des vaches. Néanmoins, il serait intéressant de mettre en place un système qui le fasse de manière automatique.
— Enfin, l'utilité de FBAT peut être élargie au bien-être animal. En effet, en plus de détecter des anomalies, FBAT pourrait donner une indication sur le bien-être de l'animal. Cela pourrait par exemple aider l'éleveur dans l'organisation de son élevage.
— Il pourrait également être intéressant de vérifier l'efficacité de la méthode FBAT avec des données représentant le niveau d'activité des animaux mais depuis des capteurs différents (par exemple des accéléromètres) mais également pour des contextes d'environnement différents (par exemple, des vaches sur parcelles).

Chapitre 4

Séries temporelles et logique floue

Ce chapitre traite de la logique floue et comment elle peut être appliquée à la classification supervisée de séries temporelles. La classification floue consiste le plus souvent à obtenir des prédictions floues à partir de données dures. Ainsi, une information supplémentaire est donnée : la probabilité que l'instance appartienne à ladite classe. Cela permet d'estimer le degré de confiance de la prédiction. Utiliser la prédiction floue à partir d'étiquettes floues est un principe qui n'est pas si courant. Cependant, dans le cadre de l'étude du comportement de vaches laitières, l'utilisation d'étiquettes floues est pertinente. En effet, il est tout à fait possible de faire l'hypothèse qu'une vache ne tombe pas malade brusquement mais plutôt de manière progressive et le même raisonnement s'applique pour son rétablissement. Dans ce chapitre, la classification floue pour les séries temporelles à partir d'étiquettes floues est introduite. Dans une première section, trois algorithmes flous développés durant cette thèse sont présentés. Deux d'entre eux sont des adaptations d'algorithmes existants : F-BOSS et F-DTW (F pour fuzzy, c'est-à-dire flou en français.) et le troisième est la version floue de FBAT (F-FBAT). La deuxième section décrit dans un premier temps la manière dont les données des vaches laitières sont exploitées afin d'obtenir des étiquettes floues. Dans un second temps, une explication est apportée sur le moyen de rendre flou des étiquettes dures provenant de données de l'archive en ligne UCR. Le protocole expérimental est détaillé dans une troisième section et la dernière section présente et analyse les résultats.

1 Algorithmes flous

J'ai développé trois algorithmes flous dont le but est de fournir une prédiction floue à partir d'étiquettes floues. Les deux premiers sont des versions floues des méthodes DTW et BOSS. Le troisième algorithme est FBAT en version floue. Il a été développé pour tester si la logique floue permet d'améliorer les résultats dans la

détection de comportements anormaux chez la vache laitière.

1.1 F-DTW et F-BOSS

L'algorithme F-DTW consiste à utiliser l'algorithme fuzzy k-NN (voir chapitre 2, sous-section 6.2) avec DTW comme distance (Figure 4.1). Il prend en entrée une série temporelle et cherche les k plus proches voisins via la mesure DTW. Ensuite, F-DTW utilise la formule de l'algorithme fuzzy k-NN (Équation 2.17, chapitre 2) pour calculer sa classe d'appartenance.

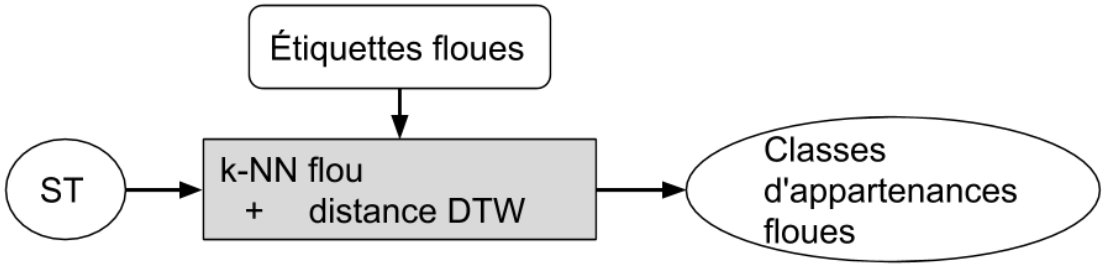


FIGURE 4.1 – Illustration de l'algorithme F-DTW ; ST : Série Temporelle.

L'algorithme F-BOSS (Figure 4.2) consiste d'abord à transformer la série en histogramme via l'algorithme BOSS. Ensuite, l'algorithme k-NN est utilisé avec la distance BOSS (Équation 2.12, chapitre 2) pour calculer les k plus proches voisins. Enfin, la formule de l'algorithme fuzzy k-NN est utilisée pour calculer la classe d'appartenance.

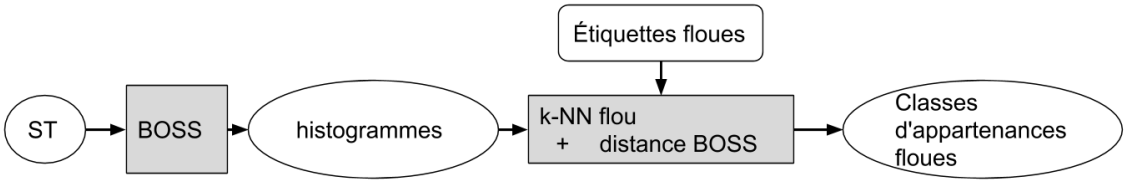


FIGURE 4.2 – Illustration de l'algorithme F-BOSS ; ST : Série Temporelle.

1.2 F-FBAT

Contrairement à la version dure de FBAT, F-FBAT prend en compte les étiquettes floues et cherche à calculer la fonction d'appartenance de chaque instance $\boldsymbol{x}_i \in \mathcal{D}_{test}$. Pour cela, la distance euclidienne entre les modèles des deux sous-séries $\boldsymbol{a}$ et $\boldsymbol{b}$ de $\boldsymbol{x}_i$ est calculée (voir chapitre 3, Équation 3.5), appelons-la $d_2(\boldsymbol{x}_i)$. Ainsi,

le but de F-FBAT est de calculer la probabilité que $\boldsymbol{x}_i$ appartienne à chaque classe $c \in \mathcal{C}$ sachant $d_2(\boldsymbol{x}_i)$: $\mathcal{P}(c|d_2(\boldsymbol{x}_i))$. F-FBAT utilise pour un ensemble de séries temporelles $\boldsymbol{x}_i \in \mathcal{D}_{train}$. Pour chacune de ces séries, F-FBAT dispose de la classe d'appartenance $u_c(\boldsymbol{x}_i)$, $\forall c \in \mathcal{C}$ (voir Équation 2.14, chapitre 2) qui correspond à la probabilité que la série $\boldsymbol{x}_i$ appartienne à la classe c.

Pour calculer $\mathcal{P}(c|d_2(\boldsymbol{x}_i))$, F-FBAT utilise la classification naïve bayésienne :

$$\mathcal{P}(c|d_2(\boldsymbol{x}_i)) = \frac{\mathcal{P}(c)\mathcal{P}(d_2(\boldsymbol{x}_i)|c)}{\mathcal{P}(d_2(\boldsymbol{x}_i))}, \tag{4.1}$$

les termes $\mathcal{P}(c)$, $\mathcal{P}(d_2(\boldsymbol{x}_i)|c)$ et $\mathcal{P}(d_2(\boldsymbol{x}_i))$ sont définis dans la suite de cette section.

1.2.1 Probabilité d'une classe

Dans le paradigme dur, la probabilité d'une classe $c \in \mathcal{C}$ est définie par une approche fréquentiste : $\mathcal{P}(c) = \frac{|\{\boldsymbol{x}_i \in \mathcal{D}_{train} : y_i = c\}|}{|\mathcal{D}_{train}|}$, avec $|\mathcal{D}|$ la cardinalité de l'ensemble $\mathcal{D}$. Dans le cadre de la logique floue la formule devient :

$$\mathcal{P}(c) = \frac{\sum_{i=1}^{|\mathcal{D}_{train}|} u_c(\boldsymbol{x}_i)}{|\mathcal{D}_{train}|}. \tag{4.2}$$

1.2.2 Probabilité d'une distance sachant la classe

Pour estimer la probabilité d'une distance $d_2(\boldsymbol{x}_i)$ par rapport à une classe c donnée, il faut faire une hypothèse de distribution. Ici, nous supposons que la distance suit une distribution gaussienne selon la classe c de moyenne $\bar{d}_c$ et de variance σ_c^2. Dans le cadre des étiquettes floues, nous utilisons la moyenne et la variance pondérées par les degrés d'appartenance aux classes :

$$\bar{d}_c = \frac{\sum_{i=1}^{|\mathcal{D}_{train}|} u_c(\boldsymbol{x}_i) d_2(\boldsymbol{x}_i)}{\sum_{i=1}^{|\mathcal{D}_{train}|} u_c(\boldsymbol{x}_i)}, \tag{4.3}$$

$$\sigma_c^2 = \frac{1}{\sum_{i=1}^{|\mathcal{D}_{train}|} u_c(\boldsymbol{x}_i)} \sum_{i=1}^{|\mathcal{D}_{train}|} u_c(\boldsymbol{x}_i)(d_2(\boldsymbol{x}_i) - \bar{d}_c)^2. \tag{4.4}$$

Grâce aux paramètres de la loi gaussienne, il possible de calculer la probabilité d'une distance $d_2(\boldsymbol{x}_i)$ sachant la classe c via la densité de probabilité gaussienne :

$$\mathcal{P}(d_2(\boldsymbol{x}_i)|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(d_2(\boldsymbol{x}_i) - \bar{d}_c)^2}{2\sigma_c^2}}. \tag{4.5}$$

1.2.3 Probabilité d'une distance

La probabilité d'une distance $\mathcal{P}(d_2(\boldsymbol{x}_i))$ ne dépend pas de la classe. Cela signifie qu'elle n'aide pas à décider à quelle classe $\boldsymbol{x}_i$ appartient si c'est la prédiction d'une étiquette dure qui est recherchée. Cependant, $\mathcal{P}(d_2(\boldsymbol{x}_i))$ permet de normaliser $\mathcal{P}(c|d_2(\boldsymbol{x}_i))$ afin que le résultat soit donné sous forme de probabilités :

$$\mathcal{P}(d_2(\boldsymbol{x}_i)) = \sum_{c=1}^{|\mathcal{C}|} \mathcal{P}(c)\mathcal{P}(d_2(\boldsymbol{x}_i)|c). \tag{4.6}$$

1.2.4 Probabilité des classes déséquilibrées

La détection de comportements anormaux chez la vache laitière est le principal problème qui a motivé le développement de F-FBAT. Les jeux de données issus des vaches laitières ont la particularité d'être plutôt déséquilibrés (voir chapitre 3, Table 3.2). Cependant, un déséquilibre de classe peut facilement biaiser la classification naïve bayésienne. Cette propriété est expliquée dans plusieurs article, notamment par [83] et [91]. Il existe dans la littérature plusieurs variante de la classification naïve bayésienne qui prennent en compte le déséquilibre des classes. La plus connue est CNB (Complement Naive Bayes) [91]. Cependant, cette solution n'est pas applicable ici car nous faisons l'hypothèse d'une distribution gaussienne.

Une solution simple [83] consiste à modifier le jeu d'apprentissage $\mathcal{D}_{train}$ pour le rendre équilibré. Les probabilités sont ensuite calculées en utilisant ce nouveau jeu de données :

$$\mathcal{P}_{eq}(c|d_2(\boldsymbol{x}_i)) = \frac{\mathcal{P}_{eq}(c)\mathcal{P}(d_2(\boldsymbol{x}_i)|c)}{\mathcal{P}(d_2(\boldsymbol{x}_i))}, \tag{4.7}$$

où les probabilités notées $\mathcal{P}_{eq}$ sont calculées avec le jeu de données $\mathcal{D}_{train}$ équilibré alors que les probabilités notées $\mathcal{P}$ sont calculées avec le jeu de données non-équilibré. Il faut noter que les probabilités $\mathcal{P}(d_2(\boldsymbol{x}_i)|c)$ et $\mathcal{P}(d_2(\boldsymbol{x}_i))$ ne dépendent pas de la répartition des classes dans le jeu de données. On a donc $\mathcal{P}_{eq}(d_2(\boldsymbol{x}_i)|c) = \mathcal{P}(d_2(\boldsymbol{x}_i)|c)$ et $\mathcal{P}_{eq}(d_2(\boldsymbol{x}_i)) = \mathcal{P}(d_2(\boldsymbol{x}_i))$.

Sachant que les données réelles sont déséquilibrées, la formule 4.7 ne peut pas être utilisée telle quelle pour une prédiction. Cependant, une relation existe entre 4.1 et 4.7. Cela passe par les probabilités $\mathcal{P}(d_2(\boldsymbol{x}_i)|c)$ et $\mathcal{P}(d_2(\boldsymbol{x}_i))$:

$$\frac{\mathcal{P}(d_2(\boldsymbol{x}_i)|c)}{\mathcal{P}(d_2(\boldsymbol{x}_i))} = \frac{\mathcal{P}_{eq}(c|d_2(\boldsymbol{x}_i))}{\mathcal{P}_{eq}(c)}, \tag{4.8}$$

En plaçant 4.8 dans 4.1 on obtient la formule suivante :

$$\mathcal{P}(c|d_2(\boldsymbol{x}_i)) = \mathcal{P}(c)\frac{\mathcal{P}_{eq}(c|d_2(\boldsymbol{x}_i))}{\mathcal{P}_{eq}(c)}, \tag{4.9}$$

C'est cette dernière formule qui est utilisée dans la suite.

2 Données

2.1 Données des vaches laitières

Dans le cadre des algorithmes durs, il a été convenu, grâce à un expert, de définir une zone anormale autour de chaque jour étiqueté comme anormal pour être certain que l'anomalie soit bien notée dans le jeu de donnée final (voir chapitre 3, sous-section 2.1). Cependant, toutes ces zones sont considérées comme totalement anormale alors qu'il est très probable qu'une vache ne tombe pas subitement malade ou en chaleur, mais plutôt qu'elle commence à être progressivement malade, puis est totalement malade et finit par se rétablir progressivement. C'est pour cette raison que nous avons introduit la logique floue pour ces jeux de données.

Le but étant de définir, pour chaque série $\boldsymbol{x}_i \in \mathcal{D}_{train}$ une probabilité d'être anormale, cette probabilité est égale à 0 quand la série est totalement normale et égale à 1 dans le cas contraire.

Comme expliqué dans le chapitre 3, sous-section 2.1, l'étiquetage des données se passe en deux temps. Dans un premier temps, chaque heure d'activité de chaque vache est labellisée. Ensuite, une fenêtre glissante de 36 h est appliquée sur chaque série temporelles de chaque vache afin d'extraire les séries qui composent les jeux de données.

Pour étiqueter les heures d'activité de chaque série, une zone floue est appliquée autour de chaque jour noté comme anormal (Tab 4.1). Ces zones floues ont été déterminées par un expert grâce aux résultats illustrés par les figures 3.7, 3.8, 3.9, 3.10, 3.11 et 3.12. La zone $\mathcal{P} = 1$ est la zone où la vache est considérée à 100% dans un état anormal. Dans la zone floue en amont, la probabilité d'être dans un état anormal commence à 0% pour arriver jusqu'à 100% en suivant une fonction affine croissante. Dans la zone floue d'après, la probabilité d'être dans un état anormal passe de 100% à 0% en suivant une fonction affine décroissante. L'heure 0 correspond à la première heure du jour notée comme anormale par l'éleveur. Si à une même heure, la vache est dans plusieurs états anormaux, alors il existe plusieurs probabilités supérieures à 0% pour cette même heure. La probabilité pour cette heure que la vache soit dans un état anormal, quel qu'il soit, est calculée selon la formule standard de l'union des

ensembles flous [93] :

$$\mathcal{P}(A_1 \cup A_2 \cup ... \cup A_n) = max(\mathcal{P}(A_1), \mathcal{P}(A_2), ..., \mathcal{P}(A_n)). \quad (4.10)$$

Ensuite, la fenêtre de 36 h est appliquée. L'étiquette donnée à chaque série extraite est la moyenne des 36 probabilités de chaque heure. Ainsi, chaque jeu de données est composé d'un certain nombre de séries de 36 heures de niveaux d'activité et chacune d'entre elle possède une probabilité d'être anormale et sa probabilité contraire.

TABLE 4.1 – Constructions des labels flous par heure et par type d'évènement (0 = première heure du jours labellisé comme anormal.

Type d'événement	**zone floue avant**	P = 1	**zone floue après**
rétentions placentaire	de -48 à 0	de 0 à 24	de 24 à 48
Velages	de -48 à 0	de 0 à 24	de 24 à 48
Oestrus	de -12 à 0	de 0 à 24	de 24 à 36
Boiteries	de -48 à -12	de -12 à 24	de 24 à 48
Mammites	de -48 à 0	de 0 à 24	de 24 à 48
Autres maladies	de -48 à 0	de 0 à 24	de 24 à 48
Injections LPS	p = 0	de 12 à 24	de 24 à 48
Acidose	de -12 à 12	de 12 à 24	de 24 à 60
Changement parc	p = 0	de 12 à 24	de 24 à 48
Autre perturbations	p = 0	de 12 à 24	de 24 à 48

2.2 Données de l'UCR

Pour vérifier que l'utilisation de la logique floue dans la classification de séries temporelles est prometteuse, j'ai utilisé des données provenant de l'archive de l'University of California Riverside (UCR) [24]. Les données choisies ont des caractéristiques différentes et sont les 11 mêmes que celles choisies dans le chapitre 3, sous-section 2.2.

3 Protocole

3.1 Mesures d'évaluation

Afin de pouvoir comparer les algorithmes durs avec les algorithmes flous, les mesures d'évaluation utilisées sont les mêmes que celles utilisées dans le chapitre 3, sous-section 3.1, c'est-à-dire : *precision*, $precision_-$, $rappel_-$, $precision_+$, $rappel_+$ ainsi que les temps de calculs CPU.

Cependant, le résultat obtenu à partir d'un algorithme de classification floue pour une série $\boldsymbol{x}_i$ n'est pas une classe y_i mais une fonction d'appartenance $u_c(\boldsymbol{x}_i)$ (voir chapitre 2, Équation 2.14). Pour pouvoir appliquer les mesures d'évaluation ci-dessus, il faut donc convertir la fonction d'appartenance en classe. Pour cela on désigne la classe dont la fonction d'appartenance est maximum : $max(u_c(\boldsymbol{x}_i)), \forall c \in \mathcal{C}$.

3.2 Jeux de données

Les jeux de données 1, 2, 3 et 4 obtenus en ferme ont été découpés aléatoirement en deux jeux $\mathcal{D}_{train}$ et $\mathcal{D}_{test}$ dans les proportions respectives de $\frac{2}{3}$ et $\frac{1}{3}$. L'équilibre initial des classes a été préservé. Pour se rapprocher des conditions réelles, les zones anormales autours de chaque jours noté anormal (voir chapitre 3, sous-section 2.1) n'ont pas été découpées, ainsi une anomalie est entièrement soit dans $\mathcal{D}_{train}$, soit dans $\mathcal{D}_{test}$ mais jamais dans les deux à la fois. Pour minimiser l'impact aléatoire, les jeux ont été découpés 10 fois aléatoirement et les mesures affichées dans les résultats sont la moyenne des 10 différents résultats obtenus.

Les jeux de données issus de l'UCR sont déjà découpés et ce sont ces découpages qui ont été utilisés. Cependant, des jeux de données avec étiquettes dures. Les étiquettes floues sont générées grâce à la méthode proposée par [90]. La méthode se déroule en deux temps. Dans un premier temps, l'étiquette de chaque instance est aléatoirement altérée. Ensuite, la fonction d'appartenance est générée. Pour chaque instance $\boldsymbol{x}_i$ du jeu de données, une probabilité p_i d'altérer l'étiquette y_i est aléatoirement générée en suivant une distribution bêta avec une variance de $\sigma = 0,04$ et d'espérance μ. L'espérance est un paramètre à fixer et sera détaillée dans le plan expérimental (voir sous-section 3.4). Plus l'espérance est grande, plus la probabilité d'altérer l'étiquette est grande. Une autre probabilité p'_i est également générée en suivant une distribution uniforme. Si $p_i > p'_i$, une nouvelle étiquette $y'_i \in \mathcal{C}$ avec $y'_i \neq y_i$ est attribuée aléatoirement à $\boldsymbol{x}_i$. Si $p_i \leq p'_i$, $y'_i = y_i$. Dans la deuxième étape, les étiquettes floues sont déduites à partir de p_i. Soit $\Pi_c : \mathcal{X} \rightarrow [0, 1]$ une fonction de possibilités avec $\mathcal{X}$ l'ensemble des séries $\boldsymbol{x}_i$:

$$\Pi_c(\boldsymbol{x}_i) = \begin{cases} 1, & c = y'_i, \\ p_i, & c \neq y'_i. \end{cases} \tag{4.11}$$

Cependant, les algorithmes employés dans cette thèse se basent sur des probabilités et non des possibilités. La différence entre les deux se tient dans la contrainte :

$$\sum_{c \in \mathcal{C}} \Pi_c(\boldsymbol{x}_i) = 1. \tag{4.12}$$

En effet, dans le domaine des probabilités cette somme doit être égale à 1, dans le cadre des possibilités, cette contrainte n'existe pas. Pour convertir la fonction de

possibilités en fonction de probabilité (ou fonction d'appartenance), il faut normaliser l'équation 4.11 :

$$u_c(\boldsymbol{x}_i) = \frac{\Pi_c(\boldsymbol{x}_i)}{\sum_{c \in \mathcal{C}} \Pi_c(\boldsymbol{x}_i)}. \tag{4.13}$$

Les jeux de données de l'UCR ont ainsi leurs étiquettes converties en logique floue.

3.3 Configuration matériel

Les expériences ont toutes été lancées sur une machine disposant d'un CPU Intel Xeon E7-8890 v3 disposant de 72 cœurs cadencés à 2.5 GHz. La machine dispose également de 3 To de RAM.

3.4 Plan expérimental

Le plan expérimental se divise en deux expériences, une première pour tester la logique floue pour la classification de séries temporelles en général et une seconde pour tester la logique floue sur les données des vaches laitières.

1. Pour tester la logique floue pour la classification de séries temporelles, la première expérience consiste à tester les algorithmes soft k-NN, F-DTW et F-BOSS sur les jeux de données de l'UCR présentés en sous-section 2.2. Trois stratégies sont proposées pour tester les algorithmes :
   — stratégie 1 : on considère que le bruit dans les étiquettes est inconnu et donc les algorithmes sont utilisés dans leur version dure. Pour chaque instance $\boldsymbol{x}_i$ l'étiquette choisie est y'_i (voir sous-section 3.2), c'est-à-dire le maximum de la fonction d'appartenance ;
   — stratégie 2 : elle consiste à supprimer les instances qui ont une étiquette trop incertaine et à utiliser les algorithmes dans leur version dure avec les instances restantes. Pour décider si une instance $\boldsymbol{x}_i$ est trop incertaines, nous calculons son entropie normalisée H_i :

$$H_i = \frac{1}{\log_2(|\mathcal{C}|)}(-\sum_{c \in \mathcal{C}} u_c(\boldsymbol{x}_i) \log_2(u_c(\boldsymbol{x}_i))). \tag{4.14}$$

   avec $H_i \in [0, 1]$. $H_i = 0$ correspond à l'état totalement certain et $H_i = 1$ à une distribution uniforme. Si $H_i > \theta$, l'étiquette y'_i est considérée trop incertaine et l'instance $\boldsymbol{x}_i$ correspondante est supprimée. Plusieurs valeurs de θ ont été testées et dans cette expérimentation, la valeur $\theta = 0.95$ est choisie ;

— stratégie 3 : elle consiste à utiliser les algorithmes dans leur version floue avec les étiquettes floues générées.

L'espérance μ de la loi bêta servant à générer les étiquettes floues (voir sous-section 3.2) est fixée à plusieurs valeurs : $\mu = [0,1; 0,2; ...; 0,7]$ (valeurs choisie selon [90]). Chacune de ces valeurs correspond à un niveau de bruit, plus la valeur est haute, plus le niveau de bruit est haut. Pour chaque algorithme, chaque stratégie, chaque valeur de μ, plusieurs valeurs de nombres de voisins k sont testées : $k = [1, 2, ..., 10]$.

2. La méthode F-FBAT est appliquée aux jeux de données 1, 2, 3 et 4 et est comparée avec les résultats obtenus par FBAT dans le chapitre 3.

4 Résultats

4.1 Logique floue pour la TSC

Dans cette section, il est question d'étudier l'impact de des étiquettes floues sur la classification de séries temporelles. Pour cela trois classifieurs durs (k-NN, DTW et BOSS) sont comparés à leur homologues flous (fuzzy k-NN, F-DTW et F-BOSS). Trois aspects sont étudiés : l'impact du nombre de voisins, l'impact de la stratégie (dure ou floue) et l'impact du bruit dans les étiquettes sur les algorithmes flous.

4.1.1 Influence du nombre de voisins

Habituellement, DTW et BOSS sont utilisés avec k-NN et le nombre de voisins k fixé à $k = 1$ (par exemple dans [35] DTW est comparé aux autres méthodes avec seulement un voisin). Cependant, on peut légitimement se demander si le fait d'utiliser un nombre de voisins supérieur peut engendrer de meilleurs résultats, notamment avec la version floue des algorithmes. C'est pour cela que dans cette section, les trois algorithmes sont testés avec un nombre de voisins allant de $k = 1$ à $k = 10$. Les algorithmes sont utilisés avec la stratégie 3 (c'est-à-dire la stratégie floue) avec un niveau de bruit dans les étiquettes modéré : $\mu = 0,3$.

La figure 4.3 représente l'évolution de la précision en fonction de k pour le jeu de données FreezerSmallTrain. La première remarque est que pour chaque valeur de k, l'algorithme fuzzy k-NN a une précision inférieure aux deux autres algorithmes. La précision des trois algorithmes augmente fortement jusqu'à $k = 5$ et ensuite augmente de manière plus douce. Jusqu'à $k = 6$, l'algorithme F-DTW est le meilleur et ensuite c'est F-BOSS qui devient meilleur. Donc pour ce jeu de données, les performances augmentent avec la valeur de k, le meilleur algorithme change en fonction de k et $k = 1$ n'est pas la meilleure solution.

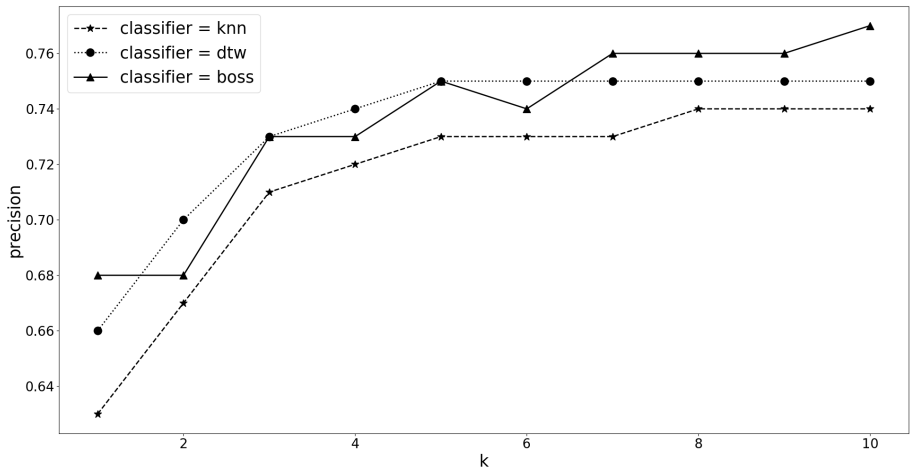


FIGURE 4.3 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données FreezerSmallTrain.

La figure 4.4 représente l'évolution de la précision en fonction de k pour le jeu de données Lightning2. Pour ce jeu de données, le meilleur classifieur est F-DTW et à partir de $k = 2$ les résultats pour ce classifieur sont stables et pour $k = 1$, les résultats sont moins bons que pour $k > 1$. Donc pour ce jeu de données, $k = 1$ n'est pas non plus la meilleure solution.

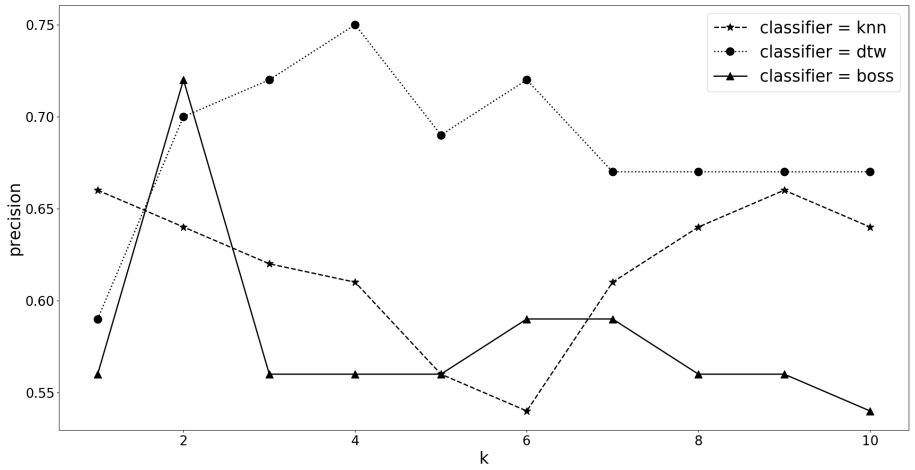


FIGURE 4.4 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données Lightning2.

La figure 4.5 représente l'évolution de la précision en fonction de k pour le jeu

de données WormsTwoClass. La meilleure précision est obtenue par F-BOSS pour $k = 6$. Selon la valeur de k, la précision de F-BOSS peut varier fortement. En revanche, la précision de F-DTW et fuzzy k-NN est moins impactée par la valeur de k.

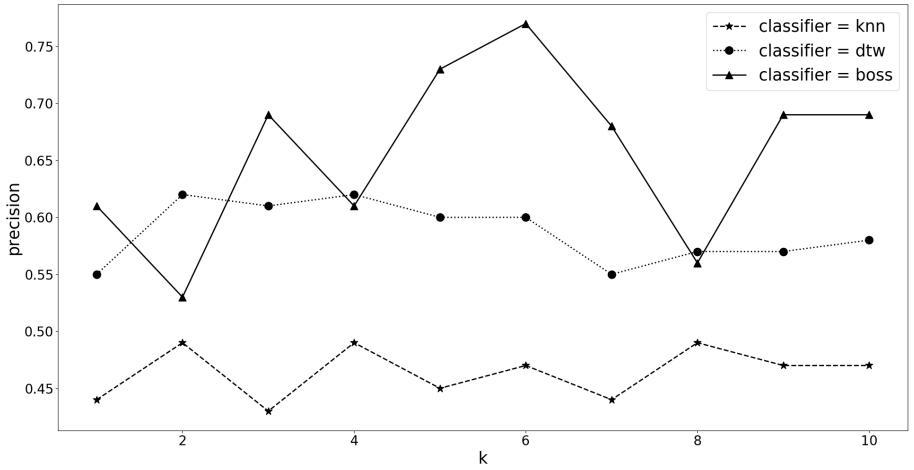


FIGURE 4.5 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données WormsTwoClass.

Il est donc difficile d'établir une règle simple pour le choix de k car cela dépend fortement du jeu de données et de l'algorithme. Cependant, sur les exemples montrés ci-dessus, $k = 1$ n'est jamais la meilleure solution. Pour éviter d'encombrer ce chapitre, les résultats pour les autres jeux de données n'ont pas été illustrés ici main en Appendice B, section 1.

Étant donné qu'il est difficile de prévoir quelle est la meilleure valeur de k et que $k = 1$ ne correspond pas à la meilleure solution trouvée pour les 11 jeux de données. Les prochains résultats sont présentés avec un nombre de voisins moyen de $k = 5$.

4.1.2 Comparaison des stratégies et des algorithmes

La table 4.2 regroupe la précision obtenue par les algorithmes fuzzy k-NN, F-DTW, F-BOSS en fonction des stratégies et du jeu données avec les paramètres $\mu = 0.3$ et $k = 5$. Les résultats dépendent beaucoup du jeu de données. Cependant, il est intéressant de noter qu'en terme de précision, F-BOSS est le meilleur dans une petite majorité des cas.

Concernant les stratégies, la stratégie 1 (c'est-à-dire avec des étiquettes dures) n'est la meilleure qu'une seule fois pour le jeu de données SonyAIBORobotSurface1

et elle partage la première place avec la stratégie 3 (c'est-à-dire floue) pour l'algorithme F-BOSS. La stratégie 2 est la meilleure pour 7 des jeux de données et la stratégie 3 est la meilleure pour 5 des jeux de données. Pour l'algorithme ECG200, les stratégies 2 et 3 sont toutes les deux meilleures avec l'algorithme fuzzy k-NN.

On peut conclure que le meilleur algorithme dépend essentiellement du jeu de données (même si F-BOSS a un léger avantage) et que la meilleure stratégie est celle qui prend en compte les étiquettes floues soit en supprimant les instances trop incertaines (stratégie 2) ou en utilisant l'algorithme dans sa version floue (stratégie 3).

4.1.3 Impact du bruit sur les algorithmes F-BOSS et F-DTW

Cette section décrit le comportement des algorithmes et des stratégies selon le degré de bruit (c'est-à-dire incertitude) dans les étiquettes qui est représenté par la valeur μ. Pour rappel, plus μ est élevé, plus le bruit ajouté est grand et plus les étiquettes sont floues. Dans chacune des figures, μ varie entre 0 et $0,7$ avec $\mu = 0$ représentant le jeu de données sans incertitude (c'est-à-dire le jeu de données original).

La figure 4.6 illustre les résultats obtenus pour le jeu de données FreezerSmallTrain. Pour $\mu = 0$ et $\mu = 0,1$, le classifieur F-BOSS est meilleur que F-DTW, indépendamment de la stratégie et pour $\mu = 0,1$, la stratégie 3 (c'est-à-dire floue) obtient le meilleur score. À partie de $\mu = 0,2$, la stratégie 2 (c'est-à-dire suppression des instances trop incertaines) est meilleure que la stratégie 3 indépendamment des classifieurs.

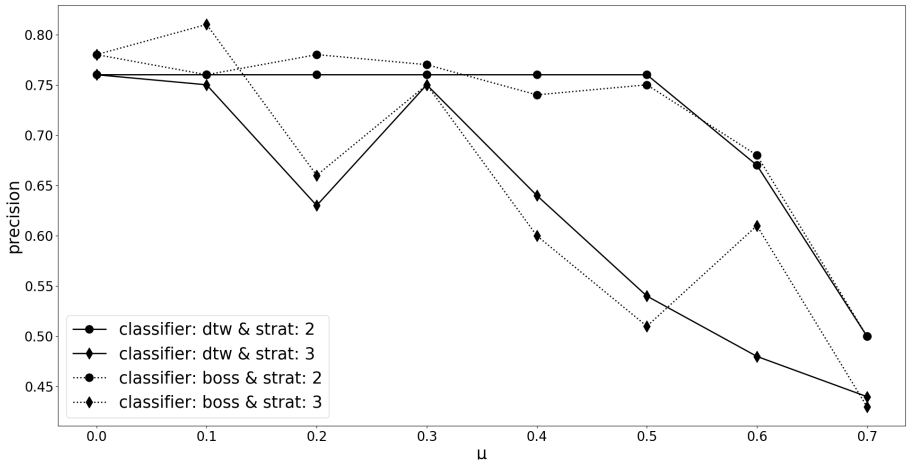


FIGURE 4.6 – Précision obtenue par les algorithme F-DTW et F-BOSS et les stratégies 2 et 3 en fonction de μ sur le jeu de données FreezerSmallTrain avec $k = 5$.

TABLE 4.2 – Précision obtenue par les algorithme fuzzy k-NN, F-DTW, F-BOSS en fonction des stratégies et du jeu données avec les paramètres $\mu = 0.3$ et $k = 5$.

	stra-tegy	fuzzy k-NN	F-DTW	F-BOSS
Earthquakes	1	0.53	0.65	0.66
	2	0.73	**0.74**	0.71
	3	0.63	0.70	0.71
ECG200	1	0.75	0.60	0.70
	2	**0.82**	0.69	0.63
	3	**0.82**	0.63	0.78
ECGFiveDays	1	0.51	0.56	0.50
	2	0.60	0.64	**0.94**
	3	0.60	0.58	0.72
FreezerSmallTrain	1	0.73	0.73	0.73
	2	**0.77**	0.76	**0.77**
	3	0.73	0.75	0.75
ItalyPowerDemand	1	0.79	0.73	0.73
	2	**0.95**	0.94	0.89
	3	0.84	0.78	0.77
Lightning2	1	0.43	0.67	0.56
	2	0.59	0.67	0.66
	3	0.56	**0.69**	0.56
MoteStrain	1	0.75	0.80	0.73
	2	0.85	**0.86**	0.87
	3	0.79	0.84	0.77
SonyAIBORobot.	1	0.54	0.62	**0.78**
	2	0.49	0.55	0.57
	3	0.58	0.67	**0.78**
TwoLeadECG	1	0.53	0.57	0.50
	2	0.60	0.77	**0.89**
	3	0.56	0.64	0.50
WormsTwoC.	1	0.44	0.56	0.70
	2	0.48	0.58	0.68
	3	0.45	0.60	**0.73**
Yoga	1	0.64	0.68	0.67
	2	0.68	0.72	0.71
	3	0.68	**0.73**	0.70

La figure 4.7 illustre les résultats obtenus pour le jeu de données ItalyPowerDemand. Les conclusions sont sensiblement les mêmes que pour le jeu FreezerSmallTrain. Un classifieur domine avant $\mu = 0,2$ (F-DTW pour ce jeu de données) indépendamment de la stratégie et à partir de $\mu = 0,2$, la stratégie 2 devient meilleure. La nuance pour ItalyPowerDemand est que F-DTW reste sensiblement meilleur que F-BOSS pour chacune des valeurs de μ. De plus, la combinaison F-DTW + stratégie 2 permet de garder des performances stables, peu importe le jeu de données.

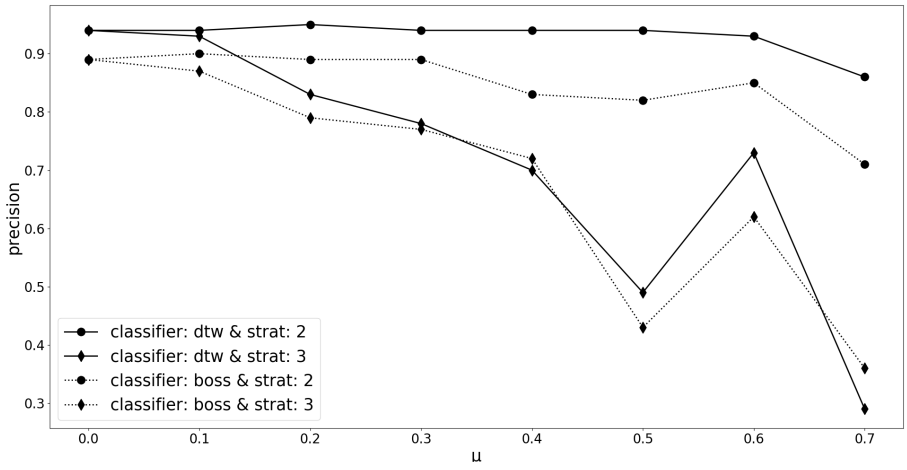


FIGURE 4.7 – Précision obtenue par les algorithme F-DTW et F-BOSS et les stratégies 2 et 3 en fonction de μ sur le jeu de données ItalyPowerDemand avec $k = 5$.

La figure 4.8 illustre les résultats obtenus pour le jeu de données SonyAIBORobotSurface1. Les résultats sont différents que ceux des deux jeux de données précédents. Dans un premier temps, il est important de constater que la valeur maximale de μ n'est que de $0,6$. Cela s'explique par le fait que SonyAIBORobotSurface1 est un petit jeu de données (20 instances dans $\mathcal{D}_{train}$). En effet, avec une valeur de $\mu = 0,7$, les étiquettes sont très incertaines et la stratégie 2 supprime trop d'instances dans $\mathcal{D}_{train}$ ce qui rend le calcul des plus proches voisins impossible. Le même phénomène est apparu avec le jeu de données MoteStrain qui lui aussi n'a que 20 instances dans son $\mathcal{D}_{train}$. Les résultats sont plus instables du fait du petit jeu de données. Par exemple, pour $\mu = 0,1$ et $\mu = 0,4$, la stratégie 2 semble mieux fonctionner (indépendamment du classifieur) alors que pour $\mu = 0,2$ et $\mu = 0,3$ c'est l'inverse. Le point le plus important est pour $\mu = 0,6$, contrairement aux autres jeux de données, c'est la stratégie 3 qui domine largement la stratégie 2. Cela vient du fait que la stratégie 2 supprime beaucoup trop d'éléments quand le degré d'incertitude est important, cela biaise les résultats sur les plus proches voisins.

La figure 4.9 illustre les résultats obtenus pour le jeu de données WormsTwoClass. Le classifieur F-BOSS est le meilleur pour chaque valeur de μ. Cependant,

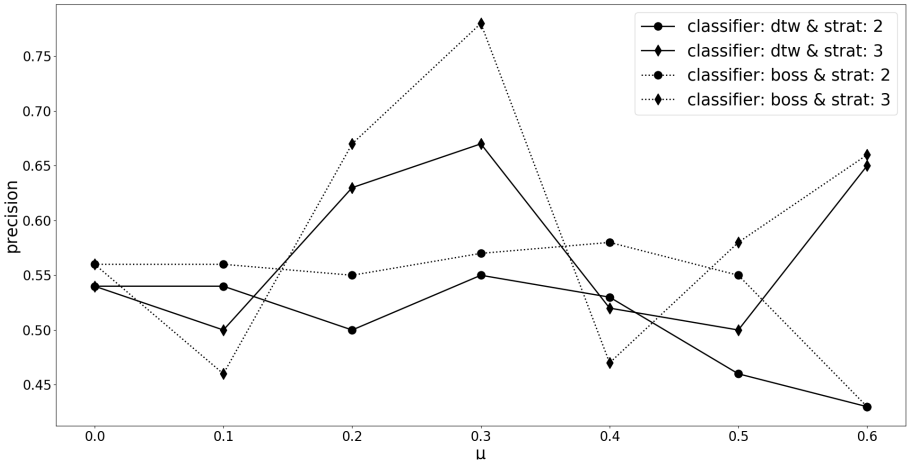


FIGURE 4.8 – Précision obtenue par les algorithme F-DTW et F-BOSS et les stratégies 2 et 3 en fonction de μ sur le jeu de données SonyAIBORobotSurface1 avec $k = 5$.

pour un niveau d'incertitude $\mu < 0,4$, la stratégie 3 est meilleure, pour un niveau d'incertitude supérieur, la stratégie 2 est meilleure.

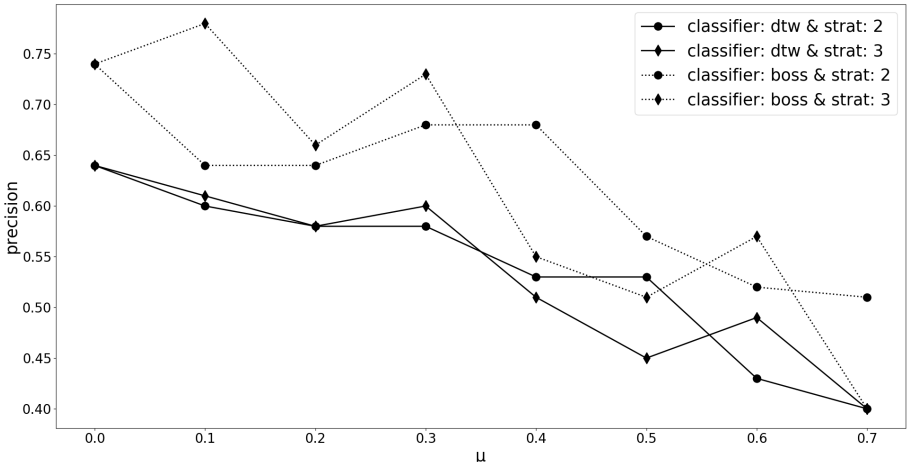


FIGURE 4.9 – Précision obtenue par les algorithme F-DTW et F-BOSS et les stratégies 2 et 3 en fonction de μ sur le jeu de données WormsTwoClass avec $k = 5$.

Ces résultats sont tous différents mais des conclusions communes peuvent être établies. Quand le niveau d'incertitude devient élevé, il vaut mieux privilégier la stratégie 2, sauf quand le jeu de données est trop petit car supprimer trop d'instances peut biaiser le calcul des plus proches voisins. Quand le niveau d'incertitude est modéré, on remarque que c'est le choix du classifieur qui est plutôt déterminant et

que la stratégie 3 est à privilégier dans la plupart des cas.

4.2 Logique floue sur les données de vaches laitières

Cette section traite de l'utilisation des étiquettes floues pour les données de comportement de vaches laitières. L'hypothèse sous-jacente est que l'apparition des symptômes dans le comportement de la vache ne se fait pas de manière brutale mais plutôt de manière progressive ainsi que le rétablissement. Pour vérifier cette hypothèse, les étiquettes des jeux de données 1, 2, 3 et 4 ont été converties en étiquettes floues (voirsous-section 2.1) et une nouvelle version de FBAT (F-FBAT) a été développée pour prendre en compte ces nouvelles étiquettes (voir sous-section 1.2).

Les résultats de F-FBAT comparés aux résultats de FBAT sont synthétisés dans la Table 4.3. La première remarque est que la version floue de FBAT met plus de temps pour le test. Cependant, ce temps reste faible (jusqu'à 13 min pour le jeu de données 4) donc pour une utilisation en production, l'augmentation du temps de calcul reste négligeable.

La deuxième remarque est que la *precision* a été augmentée de manière significative pour tous les jeux de données. Cependant, pour le jeu de données 1, le $rappel_-$ passe de $77,9\%$ pour FBAT à 0% pour F-FBAT et le $rappel_+$ passe de $30,8\%$ à 100%. Cela signifie que F-FBAT classe toutes les séries comme anormales et jamais comme normale. Or, dans le jeu de données 1, il y a beaucoup plus de séries anormales que normales. Pour les jeux de données 2 et 3 c'est l'inverse, ce sont les séries normales qui sont le plus représentées, le $rappel_+$ passe de $35,7\%$ à $5,0\%$ pour le jeu 2 et de $29,8\%$ à 0% pour le jeu 3. Le $rappel_-$ passe de $70,1\%$ à $98,4\%$ pour le jeu 2 et de $79,1\%$ à $99,9\%$ pour le jeu 3. Cela signifie que dans les jeux de données 2 et 3, F-FBAT classe la majorité des séries comme normales. En privilégiant la classe la plus représentée, F-FBAT augmente artificiellement la valeur *precision* (malgré l'utilisation du classifieur bayésien équilibré).

Pour le jeu de données 4 les conclusions sont différentes. En effet, la *precision* a également augmentée car le $rappel_-$ est passé de $81,7\%$ à $97,4\%$ sans pour autant impacter le $rappel_+$. Autrement dit, pour le jeu de données 4, F-FBAT permet de détecter autant de séries anormales que FBAT mais avec un taux moins élevé de fausses alertes. Donc F-FBAT est visiblement meilleur que FBAT pour le jeu de données 4.

Pourquoi F-FBAT fonctionne-t-elle pour le jeu de données 4 et pas pour les jeux 1, 2, 3 ? Nous pouvons avancer quelques hypothèses tenant à la nature des données qui constituent ces jeux :

— Dans le jeu de données 1, l'anomalie la plus représentée est l'acidose subclinique. Le jeu de données est en effet issu d'une expérimentation dont le but était de provoquer une acidose subclinique. Lorsqu'une vache mange trop

TABLE 4.3 – Résultats obtenues par FBAT et F-FBAT pour sur les jeux de données 1, 2, 3 et 4.

Jeu de données	**1**		**2**		**3**		**4**	
méthode	dure	floue	dure	floue	dure	floue	dure	floue
temps test	**11s**	74s	**26s**	1min31	**7s**	58s	**8min46**	13min43
precision	42,5	**62,8**	65,6	**87,7**	75,7	**93,2**	66,5	**95,2**
***precision*$_+$**	**80,8**	62,8	15,3	**28,9**	**9,6**	0	**28,9**	18,3
***precision*$_-$**	**21,7**	0	87,8	**88,9**	**93,8**	93,3	75,5	**97,7**
***rappel*$_+$**	30,8	**100**	**35,7**	5,0	**29,8**	0	**21,9**	20,4
***rappel*$_-$**	**77,9**	0	70,1	**98,4**	79,1	**99,9**	81,7	**97,4**

d'aliments concentrés, le pH de son rumen chute et devient plus variable. La vache réagit alors en mangeant moins de concentré. En conséquence, le pH redevient normal et la vache peut à nouveau manger trop de concentré. Dans cette expérimentation, les vaches passaient ainsi souvent du statut acidose un jour donné, au statut normal le lendemain puis acidose le surlendemain, etc. Elles se retrouvaient ainsi très souvent en zone floue, ce qui peut expliquer que F-FBAT considèrent que les vaches sont toujours en anomalie.

— Les données du jeu 2 proviennent au moins en partie d'une expérimentation qui consistait à injecter du LPS pour produire une inflammation. L'inflammation apparait rapidement le jour de l'administration et est passagère (environ 48 h). Les signes comportementaux apparaissent également brutalement et sont passagers, ce qui est cohérent avec le profil de détection décrit dans la figure 3.12 du chapitre 3 (détection uniquement le jour de l'administration du LPS). Une autre anomalie fréquemment observée dans le jeu 2 et qui est la seule anomalie relevée dans le jeu 3 est l'œstrus. Les signes comportementaux d'œstrus présentent une dynamique similaire à ceux de l'administration de LPS (apparition rapide et durée de 48 h environ). Pour ces anomalies, il n'y aurait pas de zone incertaine. Par conséquent la logique floue ne serait pas adaptée à ce type d'anomalies.

— Les données du jeu 4 proviennent d'une grande ferme commerciale. Les animaux n'étaient pas soumis à un traitement expérimental. Les anomalies notées par les soigneurs correspondent à ce qui est couramment rencontré de manière spontanée dans les élevages. Le jeu 4 contient des anomalies d'origine diverses : maladie, boiterie, et également accidents, vêlages, œstrus, et perturbations des animaux (par exemple manipulation pour un traitement). Le fait que F-FBAT ait des performances améliorées sur ce jeu de données est prometteur. La logique floue aurait un intérêt pour détecter des problèmes en élevage. Ceci demande à être confirmé sur d'autres jeux de données du même type.

5 Conclusion

Ce chapitre est consacré à l'utilisation d'étiquettes floues pour la classification de séries temporelles. Il montre que les algorithmes utilisant les étiquettes floues peuvent apporter de meilleurs résultats quand les étiquettes sont incertaines car ils prennent en compte plus d'informations pour calculer les prédictions.

Dans un premier temps il a été montré que l'utilisation d'algorithmes flous pour des étiquettes incertaines donne de meilleurs résultats que l'utilisation d'algorithmes durs. Pour cette expérimentation, deux nouveaux algorithmes ont été introduits : F-DTW et F-BOSS. Sur les jeux de l'UCR il a d'abord été montré que, pour les méthodes utilisant l'algorithme des plus proches voisins, choisir plus d'un voisin est préférable, que ce soit pour les versions dures ou floues des méthodes. Ensuite, plusieurs méthodes et plusieurs stratégies ont été comparées sur des étiquettes moyennement bruitées. Il s'est avéré que la stratégie de n'utiliser que les étiquettes dures n'est pas la bonne. Il est préférable d'utiliser des méthodes qui prennent en compte les étiquettes floues. Finalement, l'impact de la stratégie floue a été observée selon le degré d'incertitude des étiquettes. Les résultats montrent qu'en général il est préférable d'utiliser des méthodes qui prennent en compte les étiquettes floues quand le bruit est modéré. Quand le bruit est plus élevé, supprimer les instances avec une étiquette trop incertaine donne de meilleurs résultats, excepté pour les petits jeux de données pour lesquels supprimer trop d'instances peut considérablement dégrader les performances de la méthode.

Hormis le fait que les méthodes floues utilisent les étiquettes floues et donc peuvent donner de meilleurs résultats, cela permet d'avoir une information supplémentaire : la probabilité que l'instance appartienne à la classe. Cela peut s'avérer très profitables dans certaines applications, notamment pour la détection d'une maladie ou d'un stress.

Donc, dans un second temps, une nouvelle version de FBAT qui prend en compte les étiquettes floues a été proposée : F-FBAT. Elle a été testée et comparée avec sa version dure sur les jeux de données issus des vaches laitières (jeux 1, 2, 3 et 4). Les résultats dépendent des jeux de données. Pour les jeux 1, 2 et 3, FBAT reste la meilleure méthode. En revanche, F-FBAT donne de meilleurs résultats pour le jeu de données 4. La conclusion est que pour la détection d'anomalies chez les vaches laitière, la méthode F-FBAT obtient de mauvais résultats sur les données issues d'expérimentations mais de bons résultats sur les données issues de la ferme commerciale. L'hypothèse est que la ferme commerciale a permis d'avoir beaucoup plus de données que les autres (variété d'anomalie) et qu'elle n'est pas biaisée par une expérimentation. Cependant, il conviendrait de tester F-FBAT sur d'autres fermes commerciales à grande échelle afin de valider cette hypothèse. De plus, il faudrait vérifier que le $rappel_+$ obtenu par F-FBAT sur le jeu de données 4 permet de détecter

les différentes anomalies (comme cela a été fait dans le chapitre 3). Ce chapitre est consacré à la logique floue dans le monde de la classification de séries temporelles. Une dernière perspective serait donc de tester d'autres méthodes sur plus de jeux de données afin de valider les résultats obtenus.

Conclusion

Cette conclusion s'articule en trois sections. Une première section synthétise le bilan général de mon travail. La deuxième section présente les cinq productions (communications ou article) qui sont liées à cette thèse. La dernière section liste les perspectives qui permettraient d'approfondir mon travail.

1 Bilan général

La problématique de cette thèse consiste à développer une méthode pour détecter les anomalies dans le comportement d'animaux. Pour cela, j'ai dans ma première contribution développé une méthode, FBAT, qui permet de détecter la plupart des anomalies dans les séries temporelles. Comparée aux méthodes de référence dans la littérature, FBAT engendre moins de fausses alertes et nécessite moins de ressources matérielles. FBAT a été testée également sur d'autres jeux de données disponibles sur internet et qui ne sont pas en rapport avec le comportement animal. FBAT a réussi à être la meilleure méthode en termes de précision sur un des jeux de données et a montré des résultats globalement convaincants sur les autres jeux de données. Dans la pratique, des modifications de rythme d'activité peuvent survenir avant que les éleveurs détectent des signes cliniques de maladie ou d'états physiologiques spécifiques. La méthode FBAT permet d'identifier ces modifications avant que les signes cliniques ne soient détectés, ce qui est particulièrement important pour gérer finement les élevages (intervention précoce de l'éleveur pour isoler un animal, faire des observations complémentaires, administrer un traitement...)

Pour ma seconde contribution, je me suis intéressé aux étiquettes floues. En effet, en faisant l'hypothèse qu'une vache tombe malade et se rétablit progressivement, il est logique de s'intéresser aux étiquettes incertaines puisque les modifications de rythme vont vraisemblablement apparaître et disparaitre aussi progressivement. La logique floue en classification permet, en plus de prédire la classe d'appartenance, de donner une probabilité d'appartenance à chaque classe. Cela peut être très intéressant dans le cadre de la détection de comportements anormaux : l'éleveur pourrait avoir la probabilité qu'une vache ait besoin d'une attention particulière. Cependant, l'utilisation des étiquettes floues n'est pas courante dans le monde de la logique

floue, de même que son application aux séries temporelles. Dans un premier temps j'ai cherché à savoir si l'utilisation des étiquettes floues dans le cadre des séries temporelles permet d'obtenir de meilleurs résultats quand les étiquettes sont bruitées. Pour cela, j'ai converti deux méthodes issues de la classification de séries temporelles (DTW et BOSS) de manière qu'elles prennent en compte les étiquettes floues. J'ai également bruité et rendu incertaines les étiquettes de 11 jeux de données issus de l'UCR. Les résultats montrent que la logique floue permet d'obtenir de meilleurs résultats que la logique dure quand les étiquettes sont bruitées et cela pour tous les jeux de données. J'ai ensuite introduit une nouvelle méthode, F-FBAT qui est l'évolution de FBAT qui prend en compte les étiquettes floues. Les premiers résultats que j'ai obtenus sont prometteurs en ce qui concerne les données issues de la ferme commerciale.

2 Productions

Les diverses recherches effectuées durant cette thèse m'ont permis de réaliser cinq communications majeures :

- Durant ma première année, j'ai eu l'occasion de présenter mes premiers résultats durant la conférence Measuring Behavior de 2018 à Manchester [125]. Cela m'a permis de présenter à l'oral les premiers résultats que j'ai obtenus avec la méthode présentée dans la sous-sous-section 5.1.1, chapitre 2.
- Les résultats présentés à l'oral à Measuring Behavior ont fait l'objet d'un article publié dans le journal COMPAG [126].
- J'ai présenté la méthode FBAT ainsi que ses résultats dans un article publié dans la conférence ISMIS 2020 [128]. Cela a également donné lieu à une communication orale.
- Des résultats plus approfondis de la méthode FBAT ont donné lieu à un article accepté pour publication dans le journal Methods [127]. Les résultats présentés sont ceux qui sont décrits dans la sous-section 4.3, chapitre 3.
- Les résultats obtenus par F-DTW et F-BOSS sur les données UCR sont communiqués dans un article publié lors de la conférence IPMU 2020 [124]. Cet article a également donné lieu à une communication orale.

3 Perspectives

Le travail qui a été fait durant cette thèse a permis de répondre à la question qui m'a été posée. Cependant, je le vois comme un point de départ et non comme une fin en soi. En trois ans j'ai eu l'occasion de tester beaucoup d'hypothèses de travail et d'algorithmes cependant, cela reste tout de même assez court : on aimerait en

faire toujours plus, aller toujours plus loin. C'est pour cela que dans la suite de cette section, sont listées les pistes qui pourraient améliorer ou étendre mon travail.

— La première étape serait de tester FBAT en production, c'est-à-dire dans des élevages où on pourrait évaluer l'intérêt de la méthode pour l'éleveur. En effet, même si FBAT est évaluée sur plusieurs jeux de données, cela reste de la théorie. Pour valider définitivement les résultats, tester FBAT en production est indispensable.
— Une des principales limitations de FBAT est que sa classification est binaire : elle détecte les comportements anormaux sans pouvoir donner la cause. Ajouter le multi-classes à FBAT serait donc une évolution logique. Cependant, cela demande de collecter encore plus de jeux de données afin d'avoir un large panel d'anomalies représentées. Pour cela, il conviendrait d'aller au-delà d'un simple calcul de la distance euclidienne entre la fondamentale produite par la Transformée de Fourier sur 2 séries de 24 h successives et de sa comparaison à un seuil. Une première piste consisterait à caractériser cette distance (quelle est son ampleur ?). Une autre piste complémentaire serait d'explorer les harmoniques produites par la Transformée de Fourier ou encore d'utiliser des ondelettes pour capter les pics d'activité dans la journée (correspondant aux repas).
— Une autre limitation de FBAT est qu'elle ne prend pas en compte le comportement individuel de l'animal vis à vis d'une anomalie. En effet, le seuil est établi sur l'ensemble du jeu de données et non pas par individu. Bien entendu, le seuil peut tout à fait être ajusté a posteriori par l'éleveur. Néanmoins, prendre en compte cette variabilité de manière automatique serait sans doute apprécié par l'éleveur.
— Dans le monde de l'élevage de précision, il existe d'autres capteurs qui permettent de collecter des données sur le comportement animal (e.g. accéléromètres, vidéos, captures sonores, etc.). Toutes les fermes ne disposant pas de système de localisation, FBAT devrait être testée sur des données collectées à partir de ces autres capteurs.
— Il existe également des contextes d'environnement différents (par exemple, des vaches au pâturage). Mesurer les performances de FBAT sur d'autres types d'environnement serait également une valeur ajoutée à mon travail.
— Concernant F-FBAT, elle permet de prendre en compte les étiquettes incertaines et semble prometteuse en ce qui concerne les données issues de la ferme commerciale. Il faudrait cependant tester F-FBAT sur d'autres données issues d'autres fermes commerciales afin de valider les résultats. Il faudrait également tester F-FBAT sur d'autres données UCR afin d'étudier les avantages de F-FBAT à une échelle plus large.
— Les algorithmes F-BOSS et F-DTW ont permis d'améliorer les résultats sur

les 11 jeux de données UCR avec les étiquettes incertaines. Cela signifie donc que prendre en compte les étiquettes floues sur des données de type série temporelle est très prometteur et que les investigations devraient être poursuivies. Je pense notamment à convertir d'autres méthode (ex : Hive-Cote) et à tester ces méthodes sur la totalité des jeux présents dans UCR (i.e. plus de 100 jeux).

Bibliographie

[1] Charu C Aggarwal. Outlier analysis. In *Data mining*, pages 237–263. Springer, 2015.

[2] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4) :343–370, 1988.

[3] Yves Aragon. *Séries temporelles avec R*. EDP sciences, 2016.

[4] Anthony Bagnall, Luke Davis, Jon Hills, and Jason Lines. Transformation based ensembles for time series classification. In *Proceedings of the 2012 SIAM international conference on data mining*, pages 307–318. SIAM, 2012.

[5] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off : a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3) :606–660, 2017. Publisher : Springer.

[6] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with COTE : the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9) :2522–2535, 2015. Publisher : IEEE.

[7] Tao Ban, Ruibin Zhang, Shaoning Pang, Abdolhossein Sarrafzadeh, and Daisuke Inoue. Referential knn regression for financial time series forecasting. In *International Conference on Neural Information Processing*, pages 601–608. Springer, 2013.

[8] TF Borderas, AM De Passillé, and J Rushen. Behavior of dairy calves after a low dose of bacterial endotoxin. *Journal of animal science*, 86(11) :2920–2927, 2008.

[9] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[10] Suratna Budalakoti, Ashok N Srivastava, Ram Akella, and Eugene Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. 2006.

[11] C Byrd and D Lay. 229 can baseline heart rate variability be used as a predictor of the swine behavioral and febrile response to a sickness challenge ?. *Journal of Animal Science*, 96(Suppl 3) :9, 2018.

[12] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*, pages 126–133. IEEE, 1999.

[13] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection : A survey. *ACM computing surveys (CSUR)*, 41(3) :1–58, 2009.

[14] Xiao-yun Chen and Yan-yan Zhan. Multi-scale anomaly detection algorithm based on infrequent pattern of time series. *Journal of Computational and Applied Mathematics*, 214(1) :227–237, 2008.

[15] CEF Clark, NA Lyons, L Millapan, S Talukder, GM Cronin, KL Kerrisk, and SC Garcia. Rumination and activity levels as predictors of calving for dairy cows. *Animal*, 9(4) :691–695, 2015.

[16] James W. Cooley and John W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90) :297–301, 1965.

[17] Farm Animal Welfare Council. Fawc updates the five freedoms. *Vet. Rec.*, 131 :357, 1992.

[18] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1) :21–27, 1967.

[19] Joseph A Cruz and David S Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics*, 2 :117693510600200030, 2006.

[20] Balázs Csanád Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48) :7, 2001.

[21] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4) :303–314, 1989.

[22] Robert Dantzer and Keith W. Kelley. Twenty years of research on cytokine-induced sickness behavior. *Brain, Behavior, and Immunity*, 21(2) :153–160, February 2007.

[23] Anne-Sophie Darmaillacq and Frédéric Lévy. *Éthologie animale : une approche biologique du comportement*. De Boeck Supérieur, 2019.

[24] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[25] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239 :142–153, 2013. Publisher : Elsevier.

[26] Joaquín Derrac, Salvador García, and Francisco Herrera. Fuzzy nearest neighbor algorithms : Taxonomy, experimental analysis and prospects. *Information Sciences*, 260 :98–119, 2014. Publisher : Elsevier.

[27] Alice de Boyer Des Roches, Marion Faure, Alexandra Lussert, Vincent Herry, Pascal Rainard, Denys Durand, and Gilles Foucras. Behavioral and pathophysiological response as possible signs of pain in dairy cows during escherichia coli mastitis : A pilot study. *Journal of Dairy Science*, 100(10) :8385–8397, 2017.

[28] Sebastien Destercke. A k-nearest neighbours method based on imprecise probabilities. *Soft Computing*, 16(5) :833–844, 2012. Publisher : Springer.

[29] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data : experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2) :1542–1552, 2008.

[30] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[31] Rebecca Dumbell, Olga Matveeva, and Henrik Oster. Circadian clocks, stress, and immunity. *Frontiers in endocrinology*, 7 :37, 2016.

[32] Tudor Dumitraş and Priya Narasimhan. Fault-tolerant middleware and the magical 1%. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 431–441. Springer, 2005.

[33] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1) :1–34, 2012. Publisher : ACM New York, NY, USA.

[34] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification : a review. *Data Mining and Knowledge Discovery*, 33(4) :917–963, 2019. Publisher : Springer.

[35] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep neural network ensembles for time series classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2019.

[36] Katrine Kop Fogsgaard, Torben Werner Bennedsgaard, and Mette S Herskin. Behavioral changes in freestall-housed dairy cows with naturally occurring clinical mastitis. *Journal of dairy science*, 98(3) :1730–1738, 2015.

[37] Anthony J Fox. Outliers in time series. *Journal of the Royal Statistical Society : Series B (Methodological)*, 34(3) :350–363, 1972.

[38] Ada Wai-Chee Fu, Oscar Tat-Wing Leung, Eamonn Keogh, and Jessica Lin. Finding time series discords based on haar transform. In *International Conference on Advanced Data Mining and Applications*, pages 31–41. Springer, 2006.

[39] Clive William John Granger, GRANGER CWJ, and ANDERSEN AP. An introduction to bilinear time series models. 1978.

[40] M Guarino and D Berckmans. Precision livestock farming'15. *Edité par Guarino et Berckmans, EC-PLF. Milan, Italie, 871p*, 2015.

[41] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data : A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9) :2250–2267, 2013. Publisher : IEEE.

[42] Benjamin L Hart. Biological basis of the behavior of sick animals. *Neuroscience & Biobehavioral Reviews*, 12(2) :123–137, 1988.

[43] BL Hart. Beyond fever : comparative perspectives on sickness behavior. *Encyclopedia of animal behavior*, 1 :205–210, 2010.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[45] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4) :851–881, 2014. Publisher : Springer.

[46] Eyke Hüllermeier. Possibilistic instance-based learning. *Artificial Intelligence*, 148(1-2) :335–383, 2003. Publisher : Elsevier.

[47] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2) :85–126, 2004.

[48] Weizhe Hong, Ann Kennedy, Xavier P Burgos-Artizzu, Moriel Zelikowsky, Santiago G Navonne, Pietro Perona, and David J Anderson. Automated measurement of mouse social behaviors using depth sensing, video tracking, and machine learning. *Proceedings of the National Academy of Sciences*, 112(38) :E5351–E5360, 2015.

[49] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5) :359–366, 1989.

[50] Bing Hu, Yanping Chen, and Eamonn Keogh. Time series classification under more realistic assumptions. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 578–586. SIAM, 2013.

[51] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv :1502.03167*, 2015.

[52] HV Jagadish, Nick Koudas, and S Muthukrishnan. Mining deviants in a time series database. In *VLDB*, volume 99, pages 7–10, 1999.

[53] Young-Seon Jeong, Myong K Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern recognition*, 44(9) :2231–2240, 2011.

[54] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3) :241–254, 1967.

[55] Mayank Kabra, Alice A Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. Jaaba : interactive machine learning for automatic annotation of animal behavior. *Nature methods*, 10(1) :64, 2013.

[56] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data : an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[57] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, pages 580–585, 1985. Publisher : IEEE.

[58] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax : Efficiently finding the most unusual time series subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. Ieee, 2005.

[59] Eamonn Keogh, Jessica Lin, Sang-Hee Lee, and Helga Van Herle. Finding the most unusual time series subsequence : algorithms and applications. *Knowledge and Information Systems*, 11(1) :1–27, 2007.

[60] Robert J Kilgour, Katsuji Uetake, Toshie Ishiwata, and Gavin J Melville. The behaviour of beef cattle at pasture. *Applied Animal Behaviour Science*, 138(1-2) :12–17, 2012.

[61] CE Koch, B Leinweber, BC Drengberg, C Blaum, and H Oster. Interaction between circadian rhythms and stress. *Neurobiology of stress*, 6 :57–67, 2017.

[62] Vijay Kumar, Parveen Kumar, Satish Kumar, and BS Prakash. Seasonal alterations in plasma cortisol in peripartum murrah buffaloes (bubalus bubalis) as assayed by a sensitive enzymeimmunoassay in subtropical climate. *Biological Rhythm Research*, 51(2) :194–203, 2020.

[63] Agnes Lagnoux. Séries chronologiques. *ISMAG, Master1-MI00141X, Université de Toulouse Le Mirail, 53p*, 1996.

[64] Nhu D Le, R Douglas Martin, and Adrian E Raftery. Modeling flat stretches, bursts outliers in time series using mixture transition distribution models. *Journal of the American Statistical Association*, 91(436) :1504–1515, 1996.

[65] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553) :436–444, 2015.

[66] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6) :861–867, 1993.

[67] Jessica Lin, Rohan Khade, and Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39(2) :287–315, 2012.

[68] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv :1312.4400*, 2013.

[69] Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3) :565–592, 2015. Publisher : Springer.

[70] Jason Lines, Sarah Taylor, and Anthony Bagnall. Hive-cote : The hierarchical vote collective of transformation-based ensembles for time series classification. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1041–1046. IEEE, 2016.

[71] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with hive-cote : The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5) :52, 2018.

[72] J Lomb, DM Weary, KE Mills, and MAG von Keyserlingk. Effects of metritis on stall use and social behavior at the lying stall. *Journal of dairy science*, 101(8) :7471–7479, 2018.

[73] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[74] Patricia C Lopes, Per Block, Alice Pontiggia, Anna K Lindholm, and Barbara König. No evidence for kin protection in the expression of sickness behaviors in house mice. *Scientific reports*, 8(1) :1–9, 2018.

[75] F López-Gatius, P Santolaria, I Mundet, and JL Yániz. Walking activity at estrus and subsequent fertility in dairy cows. *Theriogenology*, 63(5) :1419–1429, 2005.

[76] Benjamin Lucas, Ahmed Shifaz, Charlotte Pelletier, Lachlan O'Neill, Nayyar Zaidi, Bart Goethals, François Petitjean, and Geoffrey I Webb. Proximity forest : an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, 33(3) :607–635, 2019.

[77] Junshui Ma and Simon Perkins. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618, 2003.

[78] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[79] Pierre-François Marteau. Time warp edit distance with stiffness adjustment for time series matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(2) :306–318, 2008.

[80] C Medrano-Galarza, J Gibbons, S Wagner, AM De Passillé, and J Rushen. Behavioral changes in dairy cows with mastitis. *Journal of dairy science*, 95(12) :6994–7002, 2012.

[81] Bruno Meunier, Philippe Pradel, Karen H Sloth, Carole Cirié, Eric Delval, Marie M Mialon, and Isabelle Veissier. Image analysis to refine measurements of dairy cow behaviour from a real-time location system. *Biosystems Engineering*, 173 :32–44, 2018.

[82] Abdullah Mueen, Eamonn Keogh, and Neal Young. Logical-shapelets : an expressive primitive for time series classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1154–1162, 2011.

[83] Kevin P Murphy et al. Naive bayes classifiers. *University of British Columbia*, 18 :60, 2006.

[84] Shan Muthukrishnan, Rahul Shah, and Jeffrey Scott Vitter. Mining deviants in time series data streams. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.*, pages 41–50. IEEE, 2004.

[85] Alexandre Nairac, Neil Townsend, Roy Carr, Steve King, Peter Cowley, and Lionel Tarassenko. A system for the analysis of jet engine vibration data. *Integrated Computer-Aided Engineering*, 6(1) :53–66, 1999.

[86] Francisco Javier Nogales, Javier Contreras, Antonio J Conejo, and Rosario Espínola. Forecasting next-day electricity prices by time series models. *IEEE Transactions on power systems*, 17(2) :342–348, 2002.

[87] M Norring, J Häggman, H Simojoki, P Tamminen, C Winckler, and M Pastell. Lameness impairs feeding behavior of dairy cows. *Journal of Dairy Science*, 97(7) :4317–4321, 2014.

[88] M Petit. Emploi du temps des troupeaux de vaches-mères et de leurs veaux sur les pâturages d'altitude de l'aubrac. 1972.

[89] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8(1) :143–195, 1999.

[90] Benjamin Quost, Thierry Denœux, and Shoumei Li. Parametric classification with soft labels using the evidential EM algorithm : linear discriminant analysis versus logistic regression. *Advances in Data Analysis and Classification*, 11(4) :659–690, 2017. Publisher : Springer.

[91] Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003.

[92] Marie Madeleine Richard, Karen Helle Sloth, and Isabelle Veissier. Real time positioning to detect early signs of welfare problems in cows. 2015.

[93] Timothy J Ross et al. *Fuzzy logic with engineering applications*, volume 2. Wiley Online Library, 2004.

[94] Bertrand Rouet-Leduc, Claudia Hulbert, Nicholas Lubbers, Kipton Barros, Colin J Humphreys, and Paul A Johnson. Machine learning predicts laboratory earthquakes. *Geophysical Research Letters*, 44(18) :9276–9282, 2017.

[95] Enrique Vidal Ruiz, Francisco Casacuberta Nolla, and Hector Rulot Segovia. Is the DTW "distance" really a metric ? An algorithm reducing the number of DTW comparisons in isolated word recognition. *Speech Communication*, 4(4) :333–344, 1985.

[96] Hiroaki Sakoe and Seibi Chiba. A dynamic programming approach to continuous speech recognition. In *Proceedings of the Seventh International Congress on Acoustics, Budapest*, volume 3, pages 65–69, Budapest, 1971. Akadémiai Kiadó.

[97] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1) :43–49, 1978.

[98] IJ Salfer, PA Bartell, CD Dechow, and KJ Harvatine. Annual rhythms of milk synthesis in dairy herds in 4 regions of the united states and their relationships to environmental indicators. *Journal of dairy science*, 103(4) :3696–3707, 2020.

[99] Yutaka Sasaki. The truth oh the f–measure. *Manchester : School of Computer Science, University of Manchester*, 2007.

[100] Patrick Schäfer. The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6) :1505–1530, 2015. Publisher : Springer.

[101] Patrick Schäfer and Mikael Högqvist. SFA : a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 516–527, 2012.

[102] H Seegers and F Sérieys. Umt maîtrise de la santé des troupeaux bovins. *Guide d'intervention pour la maîtrise des mammites dans les troupeaux laitiers*, 2011.

[103] Karlton Sequeira and Mohammed Zaki. Admit : anomaly-based data mining for intrusions. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 386–395, 2002.

[104] Keren Shakhar and Guy Shakhar. Why do we feel sick when infected—can altruism play a role ? *PLoS Biol*, 13(10) :e1002276, 2015.

[105] M Silberberg, B Meunier, I Veissier, and MM Mialon. Continuous monitoring of cow activity to detect sub-acute ruminal acidosis (sara). *8th, ECPLF. Nantes, France*, 2017.

[106] G Silvestri, F Bini Verona, Mario Innocenti, and M Napolitano. Fault detection using neural networks. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 6, pages 3796–3799. IEEE, 1994.

[107] Michael H Smolensky, Ramon C Hermida, Alain Reinberg, Linda Sackett-Lundeen, and Francesco Portaluppi. Circadian disruption : New clinical perspective of disease pathology and basis for chronotherapeutic intervention. *Chronobiology international*, 33(8) :1101–1119, 2016.

[108] Alexandra Stefan, Vassilis Athitsos, and Gautam Das. The move-split-merge metric for time series. *IEEE transactions on Knowledge and Data Engineering*, 25(6) :1425–1438, 2012.

[109] Ralf Östermark. A fuzzy vector valued KNN-algorithm for automatic outlier detection. *Applied Soft Computing*, 9(4) :1263–1272, 2009. Publisher : Elsevier.

[110] Pei Sun, Sanjay Chawla, and Bavani Arunasalam. Mining for outliers in sequential databases. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 94–105. SIAM, 2006.

[111] Boleslaw K Szymanski and Yongqiang Zhang. Recursive data mining for masquerade detection and author identification. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 424–431. IEEE, 2004.

[112] Christian Thiel. Classification on soft labels is robust against label noise. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 65–73. Springer, 2008.

[113] Jacques Thimonier. Détermination de l'état physiologique des femelles par analyse des niveaux de progestérone. *Productions animales*, 13(3) :177–183, 2000.

[114] Robert S Thompson, Rachel Roller, Benjamin N Greenwood, and Monika Fleshner. Wheel running improves rem sleep and attenuates stress-induced flattening of diurnal rhythms in f344 rats. *Stress*, 19(3) :312–324, 2016.

[115] Ruey S Tsay, Daniel Pena, and Alan E Pankratz. Outliers in multivariate time series. *Biometrika*, 87(4) :789–804, 2000.

[116] John Joseph Valletta, Colin Torney, Michael Kings, Alex Thornton, and Joah Madden. Applications of machine learning in animal behaviour studies. *Animal Behaviour*, 124 :203–220, 2017.

[117] Isabelle Veissier, Alain Boissy, Anne Marie dePassillé, J Rushen, CG Van Reenen, S Roussel, Stéphane Andanson, and Philippe Pradel. Calves' responses to repeated social regrouping and relocation. *Journal of animal science*, 79(10) :2580–2593, 2001.

[118] Isabelle Veissier, J Capdeville, and Eric Delval. Cubicle housing systems for cattle : Comfort of dairy cows depends on cubicle adjustment. *Journal of animal science*, 82(11) :3321–3337, 2004.

[119] Isabelle Veissier, Pierre Le Neindre, and Gilbert Trillat. The use of circadian behaviour to measure adaptation of calves to changes in their environment. *Applied Animal Behaviour Science*, 22(1) :1–12, 1989.

[120] Isabelle Veissier, Marie-Madeleine Mialon, and Karen Helle Sloth. Early modification of the circadian organization of cow activity in relation to disease or estrus. *Journal of dairy science*, 100(5) :3969–3974, 2017.

[121] Clothilde Villot, Bruno Meunier, Jonathan Bodin, Cécile Martin, and Mathieu Silberberg. Relative reticulo-rumen ph indicators for subacute ruminal acidosis detection in dairy cows. *animal*, 12(3) :481–490, 2018.

[122] MAG Von Keyserlingk, D Olenick, and DM Weary. Acute behavioral effects of regrouping dairy cows. *Journal of Dairy Science*, 91(3) :1011–1016, 2008.

[123] Cyril Voyant, Gilles Notton, Soteris Kalogirou, Marie-Laure Nivet, Christophe Paoli, Fabrice Motte, and Alexis Fouilloy. Machine learning methods for solar radiation forecasting : A review. *Renewable Energy*, 105 :569–582, 2017.

[124] Nicolas Wagner, Violaine Antoine, Jonas Koko, and Romain Lardy. Fuzzy k-nn based classifiers for time series with soft labels. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 578–589. Springer, 2020.

[125] Nicolas Wagner, Violaine Antoine, Jonas Koko, Marie Madeleine Richard, Romain Lardy, and Isabelle Veissier. Use of a precision-livestock-farming technology coupled with time series methods to identify abnormal circadian pattern of activity. *Measuring Behavior 2018*, 2018.

[126] Nicolas Wagner, Violaine Antoine, Marie-Madeleine Mialon, Romain Lardy, Mathieu Silberberg, Jonas Koko, and Isabelle Veissier. Machine learning to detect behavioural anomalies in dairy cows under subacute ruminal acidosis. *Computers and Electronics in Agriculture*, 170 :105233, 2020.

[127] Nicolas Wagner, Marie-Madeleine Mialon, Karen Helle Sloth, Romain Lardy, Dorothée Ledoux, Mathieu Silberberg, Alice de Boyer des Roches, and Isabelle Veissier. Detection of changes in the circadian rhythm of cattle in relation to disease, stress, and reproductive events. *Methods*, 2020.

[128] Nicolas Wagner, Antoine Violaine, Koko Jonas, Mialon Marie-Madeleine, Lardy Romain, and Veissier Isabelle. Comparison of machine learning methods to detect anomalies in the activity of dairy cows. In *International Symposium on Methodologies for Intelligent Systems*. Springer, 2020.

[129] Thijs J Walbeek, Deborah AM Joye, Ila Mishra, and Michael R Gorman. Physiological, behavioral and environmental factors influence bifurcated circadian entrainment in mice. *Physiology & behavior*, 210 :112625, 2019.

[130] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks : A strong baseline. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 1578–1585. IEEE, 2017.

[131] Andrew W Williams, Soila M Pertet, and Priya Narasimhan. Tiresias : Black-box failure prediction in distributed systems. In *2007 IEEE international parallel and distributed processing symposium*, pages 1–8. IEEE, 2007.

[132] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1) :1–37, 2008.

[133] Yanling Xie, Qingming Tang, Guangjin Chen, Mengru Xie, Shaoling Yu, Jiajia Zhao, and Lili Chen. New insights into the circadian rhythm and its related diseases. *Frontiers in physiology*, 10 :682, 2019.

[134] Qiang Yang and Xindong Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04) :597–604, 2006. Publisher : World Scientific.

[135] Nong Ye et al. A markov chain model of temporal behavior for anomaly detection. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, volume 166, page 169. West Point, NY, 2000.

[136] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. Clustering time series using unsupervised-shapelets. In *2012 IEEE 12th International Conference on Data Mining*, pages 785–794. IEEE, 2012.

[137] Yang Zhang, Nirvana Meratnia, and Paul Havinga. Outlier detection techniques for wireless sensor networks : A survey. *IEEE communications surveys & tutorials*, 12(2) :159–170, 2010.

Annexe A

Résultats FBAT et LSTM sur les jeux de données 1, 2, 3 et 4

1 Résultats jeu de données 1

TABLE A.1 – Résultats hard avec la méthode FBAT sur le jeu de données 1.

z		rap_-	rap_+	$prec_-$	$prec_+$	$prec$	tps app	tps test	tps total
0	avg	0.8	0.28	0.27	0.81	0.41	866.4	8.6	876.3
	std	0.043	0.04	0.003	0.01	0.02	1.28	0.04	1.01
1	avg	0.76	0.32	0.27	0.8	0.43	871.6	14.8	1763.3
	std	0.012	0.011	0.003	0.005	0.006	1.24	0.05	1.59
2	avg	0.94	0.07	0.25	0.77	0.28	882.1	20.8	2666.2
	std	0.027	0.023	0.002	0.015	0.01	0.6	0.09	1.71
3	avg	0.96	0.05	0.25	0.77	0.28	892.2	26.9	3585.3
	std	0.003	0.002	0.002	0.012	0.002	0.6	0.09	2.06
4	avg	0.9	0.12	0.25	0.79	0.32	907.9	32.9	4526.1
	std	0.006	0.005	0.002	0.008	0.004	1.12	0.15	2.81
5	avg	0.89	0.13	0.25	0.79	0.32	916.6	39.1	5481.7
	std	0.011	0.008	0.002	0.009	0.005	0.72	0.13	3.3
6	avg	0.91	0.11	0.25	0.79	0.31	927.6	45.0	6454.3
	std	0.027	0.024	0.002	0.012	0.011	0.6	0.15	3.47
7	avg	0.82	0.2	0.25	0.77	0.35	940.5	51.2	7446.0
	std	0.037	0.036	0.003	0.007	0.018	0.6	0.18	4.13
8	avg	0.84	0.18	0.25	0.78	0.34	951.7	57.1	8455.1
	std	0.065	0.059	0.003	0.015	0.027	1.36	0.18	5.06
9	avg	0.78	0.24	0.26	0.78	0.37	967.0	63.2	9485.3
	std	0.262	0.259	0.015	0.013	0.13	0.8	0.18	5.65
10	avg	0.77	0.24	0.26	0.77	0.37	975.9	69.1	10530.6
	std	0.264	0.262	0.014	0.011	0.131	0.79	0.22	5.94
11	avg	0.68	0.34	0.27	0.77	0.42	988.6	75.3	11594.5
	std	0.339	0.339	0.03	0.01	0.171	1.19	0.28	6.76
12	avg	0.58	0.43	0.27	0.76	0.47	994.4	78.1	12667.7
	std	0.378	0.379	0.034	0.009	0.191	1.15	0.31	8.16

TABLE A.2 – Résultats hard avec la méthode LSTM sur le jeu de données 1.

l		rap_-	rap_+	$prec_-$	$prec_+$	$prec$	tps app	tps test	tps total
1	avg	0.0	0.99	0.05	0.75	0.75	688.0	2.0	690.0
	std	0.01	0.014	0.133	0.002	0.008	224.01	0.12	224.07
2	avg	0.01	0.99	0.28	0.75	0.75	783.4	2.1	785.5
	std	0.009	0.013	0.086	0.002	0.008	247.84	0.51	247.91
3	avg	0.02	0.97	0.23	0.75	0.74	783.2	2.0	785.2
	std	0.006	0.009	0.028	0.002	0.005	252.42	0.4	252.54
4	avg	0.03	0.96	0.22	0.75	0.73	773.5	2.0	775.5
	std	0.008	0.008	0.017	0.002	0.005	234.39	0.45	234.5
5	avg	0.07	0.91	0.22	0.75	0.7	752.1	2.3	754.5
	std	0.004	0.008	0.011	0.002	0.006	196.55	0.54	196.55
6	avg	0.07	0.91	0.22	0.75	0.71	711.5	2.2	713.7
	std	0.008	0.012	0.012	0.002	0.008	97.52	0.71	97.69
7	avg	0.1	0.88	0.22	0.75	0.69	697.9	2.2	700.0
	std	0.038	0.041	0.012	0.003	0.021	23.03	0.81	23.16
8	avg	0.11	0.87	0.22	0.75	0.68	696.5	2.3	698.8
	std	0.026	0.027	0.011	0.003	0.014	23.38	0.91	23.47
9	avg	0.14	0.84	0.22	0.75	0.67	704.0	2.4	706.4
	std	0.032	0.035	0.01	0.003	0.019	23.04	1.02	23.08
10	avg	0.17	0.81	0.22	0.75	0.65	710.0	2.9	712.9
	std	0.023	0.024	0.008	0.003	0.013	22.11	1.39	22.2
11	avg	0.17	0.8	0.23	0.75	0.65	715.7	2.3	718.0
	std	0.026	0.024	0.009	0.003	0.013	24.24	1.01	24.3
12	avg	0.17	0.8	0.23	0.75	0.65	720.6	2.3	722.9
	std	0.026	0.023	0.009	0.003	0.012	22.5	0.94	22.52

2 Résultats jeu de données 2

3 Résultats jeu de données 3

4 Résultats jeux de données 4

TABLE A.3 – Résultats hard avec la méthode FBAT sur le jeu de données 2.

z		rap_-	rap_+	$prec_-$	$prec_+$	$prec$	tps app	tps test	tps total
0	avg	0.58	0.48	0.88	0.15	0.56	1971.8	20.1	1994.0
	std	0.016	0.016	0.002	0.003	0.012	2.61	0.06	2.5
1	avg	0.69	0.37	0.88	0.15	0.65	1991.8	34.5	4021.7
	std	0.026	0.027	0.002	0.005	0.019	2.48	0.18	4.64
2	avg	0.47	0.56	0.88	0.14	0.49	2024.2	48.7	6094.9
	std	0.074	0.072	0.002	0.004	0.055	2.46	0.26	7.23
3	avg	0.41	0.62	0.88	0.14	0.44	2050.1	62.9	8208.6
	std	0.079	0.085	0.003	0.003	0.057	2.3	0.37	8.82
4	avg	0.37	0.67	0.88	0.14	0.41	2086.7	77.0	10372.9
	std	0.004	0.007	0.002	0.003	0.003	1.81	0.4	10.37
5	avg	0.4	0.63	0.88	0.14	0.43	2110.0	91.1	12574.8
	std	0.025	0.025	0.002	0.003	0.019	3.69	0.61	14.21
6	avg	0.31	0.72	0.88	0.14	0.37	2128.6	105.1	14808.8
	std	0.025	0.02	0.002	0.003	0.019	1.72	0.58	15.58
7	avg	0.31	0.71	0.88	0.14	0.37	2161.3	119.5	17089.9
	std	0.046	0.05	0.002	0.003	0.033	2.76	0.79	17.87
8	avg	0.45	0.57	0.88	0.16	0.47	2191.0	133.5	19415.1
	std	0.292	0.293	0.004	0.053	0.214	2.24	0.86	19.31
9	avg	0.51	0.51	0.88	0.17	0.51	2225.3	147.7	21789.0
	std	0.338	0.338	0.006	0.052	0.248	3.0	0.96	21.14
10	avg	0.32	0.71	0.88	0.14	0.37	2243.5	161.6	24194.4
	std	0.004	0.007	0.001	0.003	0.004	3.47	0.93	23.91
11	avg	0.32	0.7	0.88	0.14	0.37	2271.4	176.1	26641.9
	std	0.012	0.012	0.001	0.003	0.009	2.67	0.99	26.64
12	avg	0.39	0.63	0.88	0.15	0.43	2291.1	182.8	29117.2
	std	0.211	0.212	0.003	0.051	0.154	5.95	0.99	32.01

TABLE A.4 – Résultats hard avec la méthode LSTM sur le jeu de données 2.

l		rap_-	rap_+	$prec_-$	$prec_+$	$prec$	tps app	tps test	tps total
1	avg	0.98	0.03	0.87	0.21	0.86	781.6	4.3	785.9
	std	0.009	0.009	0.002	0.026	0.007	66.7	0.53	66.77
2	avg	0.97	0.04	0.87	0.19	0.85	878.7	4.4	883.2
	std	0.014	0.015	0.002	0.042	0.01	87.11	0.56	87.08
3	avg	0.95	0.06	0.87	0.17	0.83	884.8	4.5	889.3
	std	0.115	0.115	0.003	0.039	0.085	87.59	0.47	87.61
4	avg	0.98	0.03	0.87	0.21	0.86	883.5	4.6	888.2
	std	0.01	0.012	0.002	0.049	0.008	75.99	0.23	76.01
5	avg	0.98	0.03	0.87	0.21	0.86	914.9	4.8	919.8
	std	0.009	0.011	0.002	0.038	0.007	98.9	0.7	98.86
6	avg	0.97	0.04	0.87	0.18	0.85	884.5	5.0	889.5
	std	0.012	0.013	0.002	0.025	0.009	74.84	0.76	74.84
7	avg	0.96	0.05	0.87	0.17	0.84	908.7	4.9	913.6
	std	0.011	0.012	0.002	0.011	0.009	91.24	0.91	91.21
8	avg	0.8	0.21	0.87	0.14	0.72	914.1	5.4	919.5
	std	0.266	0.266	0.003	0.008	0.196	83.6	1.05	83.81
9	avg	0.47	0.54	0.87	0.13	0.48	900.9	5.2	906.1
	std	0.341	0.345	0.005	0.002	0.25	73.34	1.23	73.34
10	avg	0.18	0.83	0.88	0.13	0.27	902.2	5.8	908.0
	std	0.105	0.108	0.005	0.002	0.077	52.92	1.49	53.19
11	avg	0.18	0.83	0.88	0.13	0.27	922.7	5.9	928.6
	std	0.087	0.091	0.005	0.002	0.064	74.03	1.65	74.26
12	avg	0.18	0.83	0.88	0.13	0.27	920.4	5.9	926.3
	std	0.077	0.081	0.005	0.002	0.056	76.89	1.81	77.27

TABLE A.5 – Résultats hard avec la méthode FBAT sur le jeu de données 3.

z		rap_-	rap_+	$prec_-$	$prec_+$	$prec$	tps app	tps test	tps total
0	avg	0.73	0.38	0.94	0.09	0.71	587.4	5.9	595.1
	std	0.01	0.02	0.002	0.004	0.008	0.84	0.03	2.7
1	avg	0.73	0.37	0.94	0.09	0.71	592.5	9.9	1197.4
	std	0.036	0.036	0.002	0.003	0.032	0.81	0.08	2.66
2	avg	0.54	0.53	0.94	0.08	0.54	601.2	14.0	1812.7
	std	0.061	0.067	0.004	0.003	0.053	0.86	0.09	3.07
3	avg	0.52	0.53	0.94	0.08	0.52	608.3	18.0	2439.0
	std	0.062	0.077	0.004	0.003	0.052	0.88	0.13	3.5
4	avg	0.49	0.52	0.93	0.08	0.49	616.4	22.1	3077.5
	std	0.325	0.318	0.004	0.013	0.281	0.58	0.11	3.78
5	avg	0.6	0.41	0.93	0.08	0.58	624.4	26.2	3728.1
	std	0.311	0.301	0.003	0.013	0.269	0.74	0.13	4.02
6	avg	0.37	0.65	0.94	0.08	0.39	630.9	30.2	4389.2
	std	0.408	0.406	0.01	0.012	0.352	0.67	0.16	4.54
7	avg	0.26	0.76	0.94	0.07	0.29	640.7	34.3	5064.2
	std	0.36	0.362	0.011	0.007	0.31	0.84	0.17	5.23
8	avg	0.66	0.36	0.94	0.09	0.64	649.1	38.3	5751.6
	std	0.405	0.399	0.005	0.019	0.35	0.84	0.2	6.1
9	avg	0.85	0.18	0.93	0.09	0.8	657.3	42.4	6451.2
	std	0.259	0.249	0.003	0.016	0.225	1.29	0.22	7.34
10	avg	0.84	0.18	0.93	0.09	0.79	665.3	46.4	7162.8
	std	0.258	0.255	0.004	0.011	0.224	1.07	0.26	8.46
11	avg	0.65	0.36	0.93	0.08	0.63	673.5	50.5	7886.9
	std	0.396	0.393	0.004	0.012	0.342	0.94	0.25	9.43
12	avg	0.28	0.75	0.95	0.07	0.31	677.6	52.4	8616.9
	std	0.355	0.36	0.01	0.007	0.306	0.88	0.27	10.33

TABLE A.6 – Résultats hard avec la méthode LSTM sur le jeu de données 3.

l		rap_-	rap_+	$prec_-$	$prec_+$	$prec$	tps app	tps test	tps total
1	avg	0.58	0.42	0.94	0.06	0.57	280.0	1.5	281.5
	std	0.424	0.429	0.015	0.012	0.366	124.72	0.23	124.93
2	avg	0.49	0.51	0.94	0.06	0.49	311.5	1.5	313.0
	std	0.424	0.429	0.016	0.01	0.366	141.05	0.27	141.26
3	avg	0.53	0.47	0.93	0.06	0.53	313.4	1.5	314.9
	std	0.37	0.381	0.008	0.01	0.318	142.92	0.34	143.12
4	avg	0.51	0.5	0.93	0.06	0.51	312.0	1.5	313.5
	std	0.35	0.362	0.008	0.012	0.302	142.72	0.5	143.07
5	avg	0.49	0.52	0.93	0.07	0.49	305.8	1.6	307.4
	std	0.318	0.328	0.007	0.005	0.274	144.05	0.52	144.25
6	avg	0.57	0.43	0.93	0.06	0.56	256.3	1.6	257.9
	std	0.351	0.359	0.006	0.009	0.302	75.69	0.63	75.88
7	avg	0.51	0.5	0.93	0.07	0.5	243.4	1.4	244.8
	std	0.282	0.287	0.005	0.003	0.243	8.48	0.42	8.49
8	avg	0.52	0.48	0.93	0.07	0.52	243.5	1.5	245.0
	std	0.298	0.3	0.005	0.006	0.257	8.78	0.62	8.79
9	avg	0.51	0.49	0.93	0.07	0.51	245.6	1.7	247.3
	std	0.292	0.295	0.005	0.003	0.252	8.76	0.89	8.77
10	avg	0.53	0.47	0.93	0.07	0.53	247.4	1.4	248.8
	std	0.289	0.292	0.004	0.004	0.249	6.88	0.44	6.95
11	avg	0.52	0.48	0.93	0.07	0.52	249.5	1.5	251.0
	std	0.279	0.283	0.005	0.004	0.24	7.38	0.62	7.31
12	avg	0.52	0.48	0.93	0.07	0.52	250.9	1.4	252.3
	std	0.28	0.283	0.004	0.004	0.241	8.46	0.39	8.5

TABLE A.7 – Résultats hard avec la méthode FBAT sur le jeu de données 4

z		rap_-	rap_+	$prec_-$	$prec_+$	$prec$	tps app	tps test	tps total
0	avg	0.85	0.18	0.75	0.29	0.68	46605.9	431.0	47092.6
	std	0.013	0.013	0.0	0.003	0.007	5765.92	54.45	5821.12
1	avg	0.8	0.24	0.76	0.29	0.66	47683.8	746.0	95523.0
	std	0.021	0.021	0.0	0.004	0.011	5886.52	92.23	11797.64
2	avg	0.74	0.29	0.75	0.28	0.63	48260.3	1046.2	144830.1
	std	0.002	0.002	0.001	0.001	0.001	5842.9	131.24	17758.59
3	avg	0.75	0.28	0.75	0.28	0.63	48761.0	1350.5	194942.7
	std	0.023	0.022	0.0	0.002	0.011	5822.88	168.17	23744.6
4	avg	0.78	0.24	0.75	0.27	0.64	49353.2	1649.7	245945.7
	std	0.035	0.035	0.001	0.005	0.017	6161.93	208.87	30113.21
5	avg	0.84	0.18	0.75	0.28	0.67	50149.6	1959.8	298055.5
	std	0.075	0.074	0.001	0.009	0.037	6219.57	246.33	36576.67
6	avg	0.98	0.04	0.75	0.38	0.74	50746.8	2259.5	351062.6
	std	0.003	0.003	0.0	0.014	0.002	6389.68	278.46	43241.41
7	avg	0.98	0.04	0.75	0.38	0.74	51131.9	2568.7	404763.8
	std	0.002	0.002	0.0	0.009	0.001	6166.42	316.98	49719.49
8	avg	0.97	0.05	0.75	0.35	0.74	52077.7	2866.5	459708.7
	std	0.001	0.001	0.0	0.004	0.001	6206.85	355.19	56275.43
9	avg	0.97	0.05	0.75	0.36	0.74	52492.7	3168.9	515370.8
	std	0.004	0.004	0.0	0.016	0.002	5878.1	395.28	62525.53
10	avg	0.98	0.04	0.75	0.39	0.74	46763.2	2989.6	495505.0
	std	0.008	0.008	0.0	0.047	0.004	419.33	12.64	4321.22
11	avg	0.98	0.04	0.75	0.4	0.74	47200.0	3252.0	545957.9
	std	0.005	0.005	0.0	0.04	0.003	483.13	15.39	4153.24
12	avg	0.98	0.03	0.75	0.42	0.74	46785.1	3393.7	596136.7
	std	0.005	0.005	0.0	0.042	0.003	689.62	38.62	4805.52

Annexe B

Résultats fuzzy k-NN, F-DTW et F-BOSS sur les données de l'UCR

1 Influence du nombre de voisins

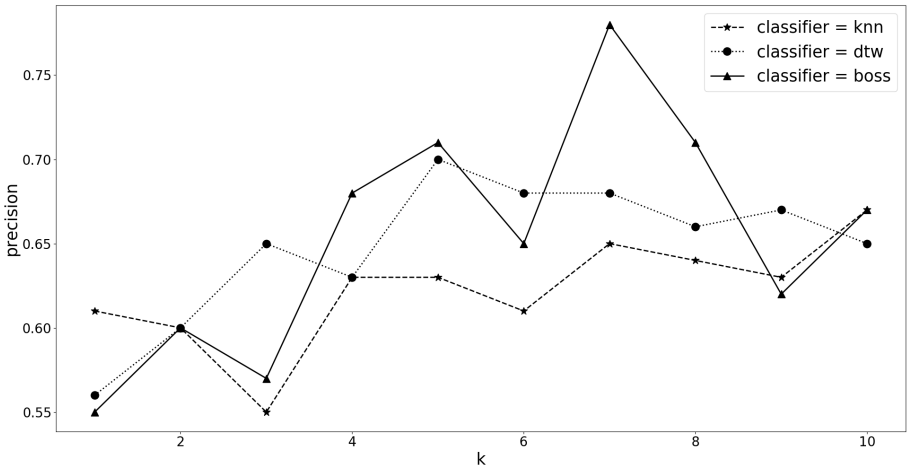


FIGURE B.1 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données Earthquakes.

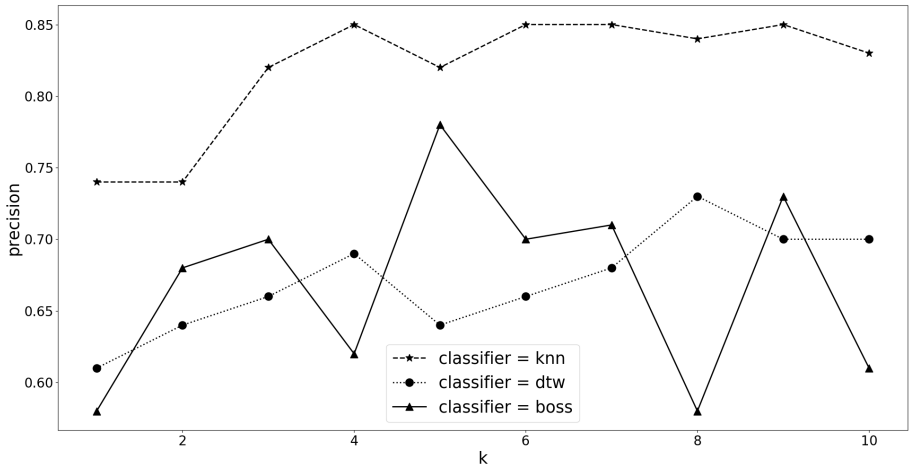


FIGURE B.2 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données ECG200.

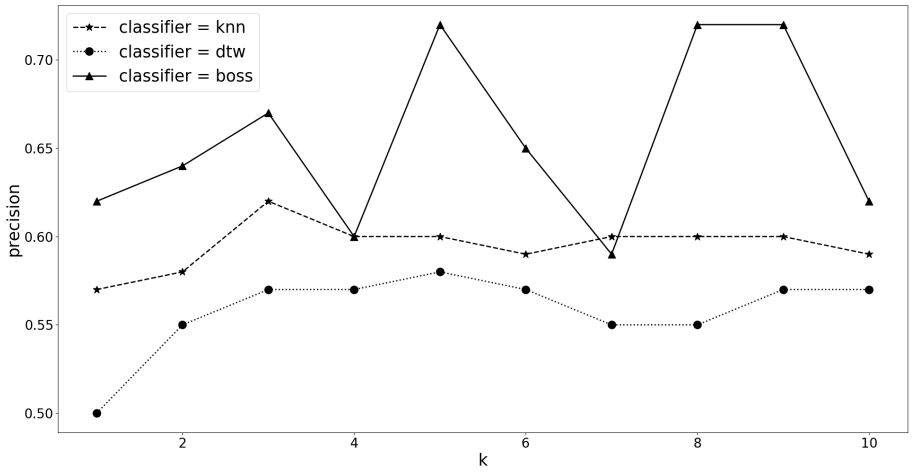


FIGURE B.3 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données ECGFiveDays.

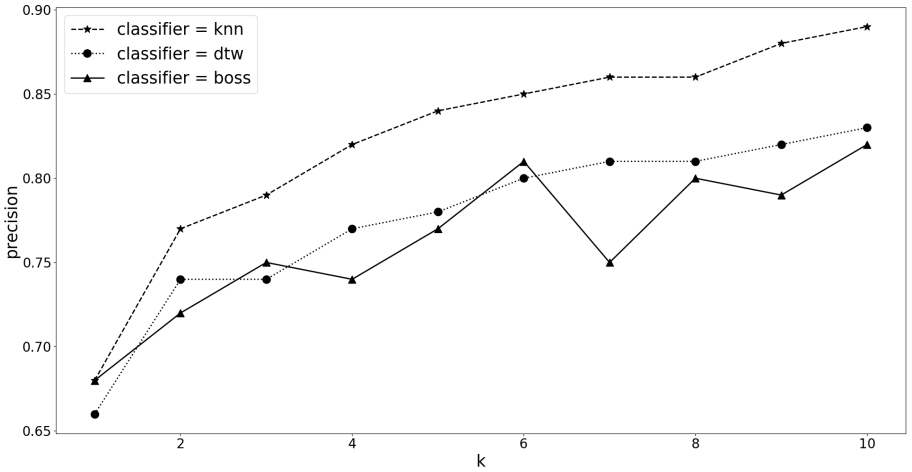


FIGURE B.4 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données ItalyPowerDemand.

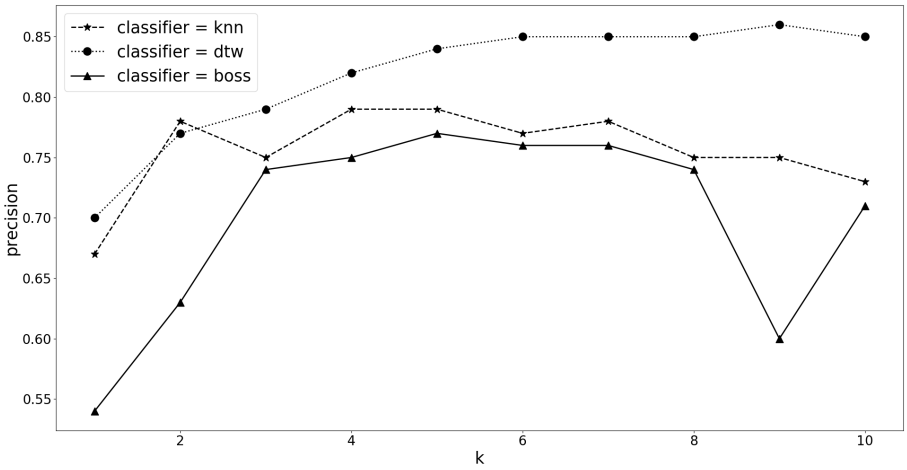


FIGURE B.5 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données MoteStrain.

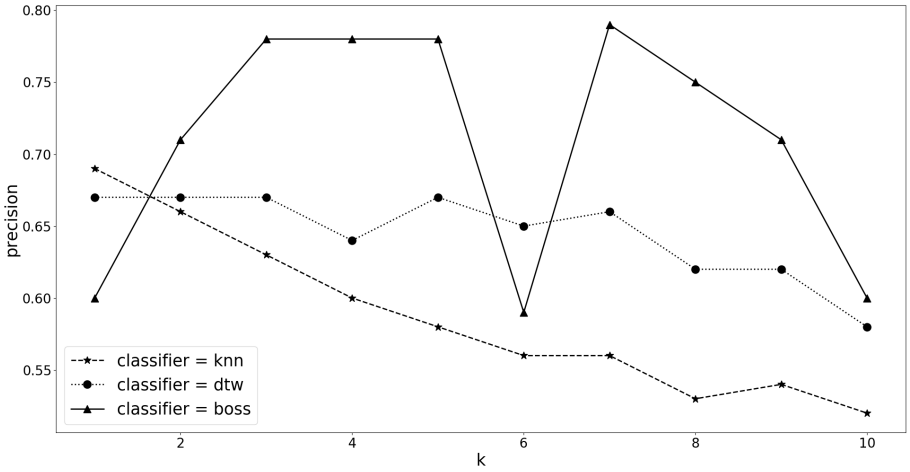


FIGURE B.6 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données SonyAIBORobotSurface1

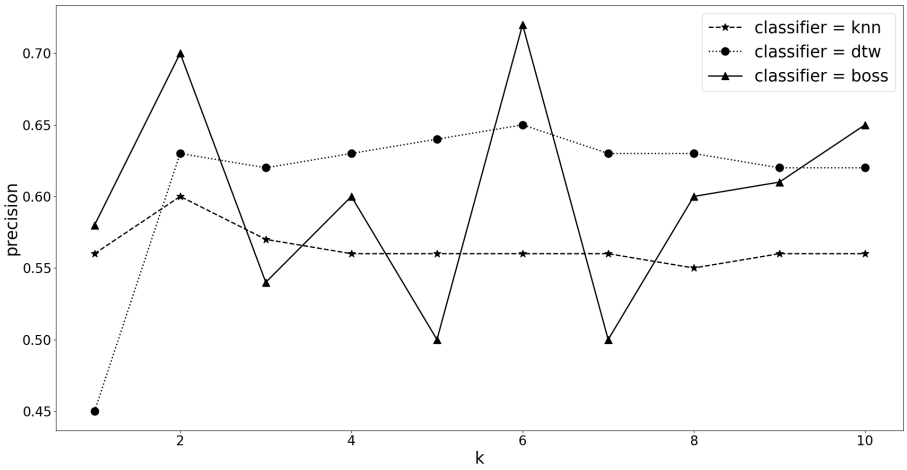


FIGURE B.7 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données TwoLeadECG

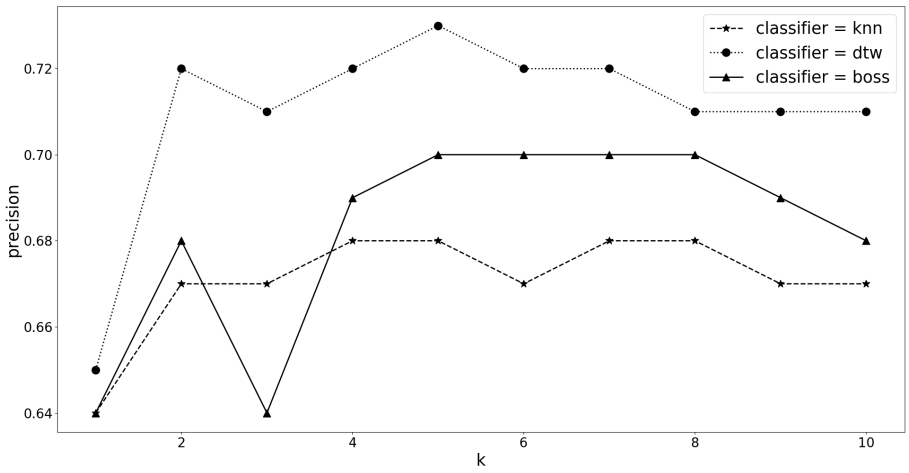


FIGURE B.8 – Précision obtenue par les classifieurs fuzzy k-NN, F-DTW et F-BOSS en fonction du nombre de voisins k avec la stratégie 3 et$\mu = 0,3$ sur le jeu de données Yoga