TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AD	Arbre décisionnel
ANN	Artificial neural network
BP	Backpropagation
CART	Classification and regression tree
CSP	Contrôle statistique de procédé
DKL	Divergence Kullback-Leibler
DT	Decision tree
ES	Expert system
FA	Forêt aléatoire
FÉO	Fabrication d'équipement d'origine
IA	Intelligence artificielle
knn	k nearest neighbor
LMC	Large margin classifier
LVQ	Linear vector quantization
MLP	Multilayer perceptron
MMC	Machine de mesure de coordonnées
MMCO	Machine de mesure de coordonnées optique
MVS	Machine à vecteurs de support

OEM	Original equipment manufacturer
OVA	One-versus-all
OVO	One-versus-one
QVL	Quantification vectorielle linéaire
RMCC	Reconnaissance de motifs de cartes de contrôle
RNA	Réseau de neurones artificiels
RPG	Rétropropagation du gradient
SE	Système expert
SPC	Statistical process control
SVM	Support vector machine

INTRODUCTION

L'assurance qualité est un moyen, pour une entreprise, d'augmenter ses profits en réduisant à la fois ses pertes pendant la fabrication et ses coûts reliés aux garanties. Pour atteindre cet objectif, les entreprises manufacturières utilisent les outils de contrôle statistique de procédé (CSP, en anglais : *statistical process control, SPC*).

L'outil principal du CSP est la carte de contrôle – parfois aussi appelée carte de Shewhart – qui permet de visualiser l'évolution d'un indicateur de qualité. Elle a été inventée en 1924 par le statisticien Walter A. Shewhart, de Bell Labs, qui la publia en 1931 dans le livre « *Economic Control of Quality of Manufactured Product* » (Shewhart, 2015).

Au cours des décennies suivantes, l'usage des cartes de contrôle devient courant, car elles sont simples à construire et faciles à interpréter. De plus, il existe plusieurs références bibliographiques dédiées aux professionnels dont la plus populaire est le « *Statistical Quality Control Handbook* » publié par la Western Electric Co. en 1958 (Western Electric Co., 1985). Cet ouvrage présente 15 différents modes de comportement non naturels observables sur les cartes de contrôle et offre des pistes de solution concrètes pour déterminer les causes assignables et y remédier.

Par exemple, il informe que les motifs cycliques (figure 0.1) peuvent être causés par les mouvements irréguliers de va-et-vient de la machinerie, les vibrations, les changements entre les différents quarts de travail et/ou par des facteurs saisonniers comme la température.

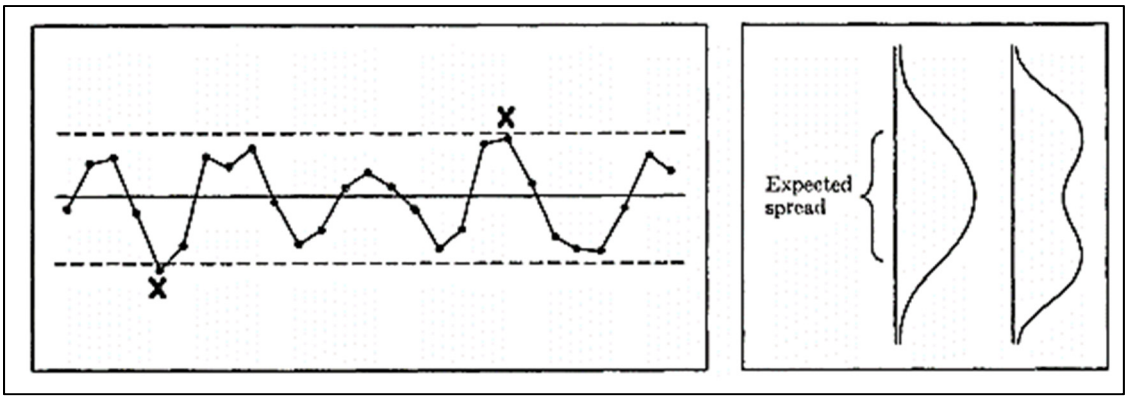

Figure 0.1 Exemple de carte de contrôle, tirée de Western Electric Co. (1985)

L'interprétation des cartes de contrôle est principalement basée sur des règles approximatives (c.-à-d. heuristiques) qui dépendent grandement de l'expérience et du jugement de l'opérateur. Il est donc important de s'assurer qu'ils sont bien formés. Quelques années plus tard, l'industrie rationalise ses procédures et centralise les connaissances de ses opérateurs dans des systèmes experts (figure 0.2).

La période de popularité des systèmes experts est relativement courte, car dès la fin des années 80, les réseaux de neurones artificiels (RNA, en anglais : *artificial neural networks*, *ANN*) commencent à être utilisés pour automatiser la lecture et l'interprétation des cartes de contrôle (Pugh, 1989). Depuis ce temps, la reconnaissance de formes, de manière générale, est dominée par l'intelligence artificielle (IA). Les RNA sont plus flexibles que les opérateurs humains et tolèrent généralement mieux le bruit statistique (Masood et Hassan, 2010). De plus, les RNA sont aussi capables d'identifier des motifs non linéaires qui sont pratiquement impossibles à détecter à l'œil, même avec l'assistance d'un système expert.

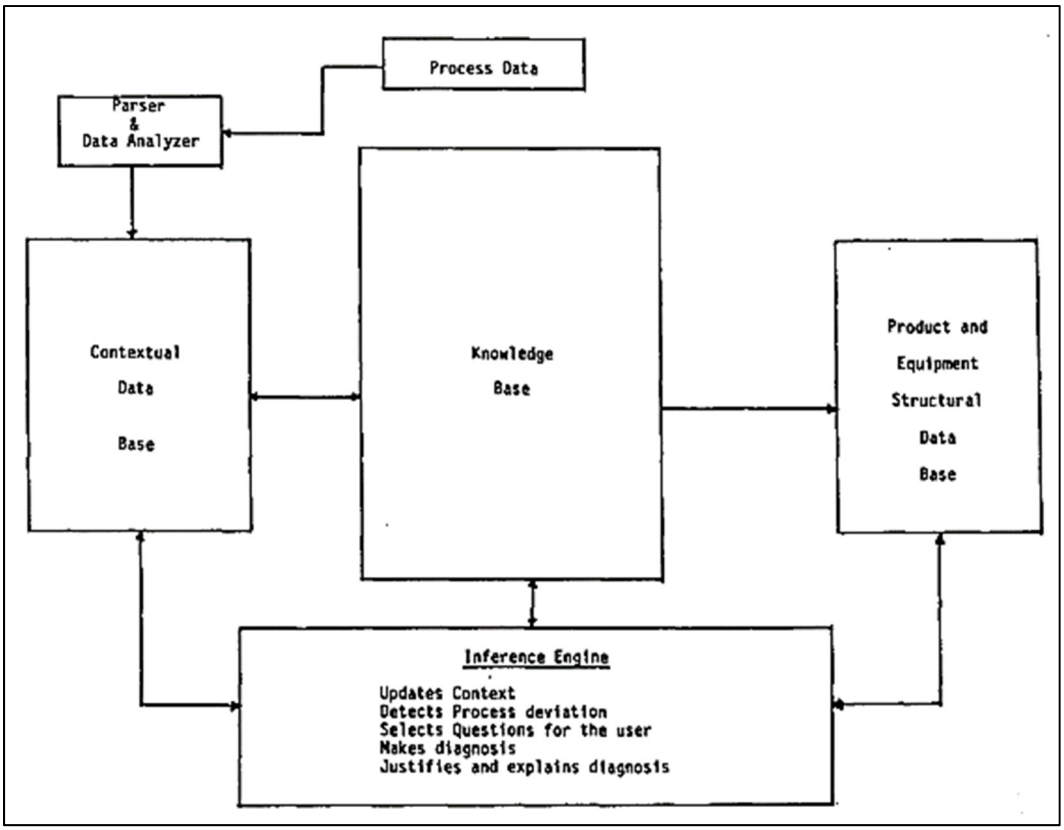

Figure 0.2 Exemple de système expert, tiré de Alexander (1986)

Au tournant du 21e siècle, le monde est secoué par l'explosion de la quantité de données numériques produites : c'est le début de l'ère des mégadonnées (en anglais : *big data*). En parallèle, la puissance des ordinateurs continue d'augmenter exponentiellement suivant ce qu'on appelle la loi de Moore. Le chevauchement de ces deux phénomènes culmine en 2011 avec le projet IBM Watson, composé d'algorithmes d'IA, qui remporte au jeu télévisé Jeopardy quatorze ans après la victoire de Deep Blue contre le champion mondial d'échecs Gary Kasparov. Ceci marque le début de la médiatisation de masse des projets d'IA accompagnée d'une hausse marquée de l'intérêt général pour l'apprentissage machine (en anglais : *machine learning*).

L'engouement pour l'analyse de donnée s'étend ensuite jusqu'aux cartes de contrôle comme il est possible de constater par l'augmentation de publications annuelles d'articles et d'actes de conférences (figure 0.3).

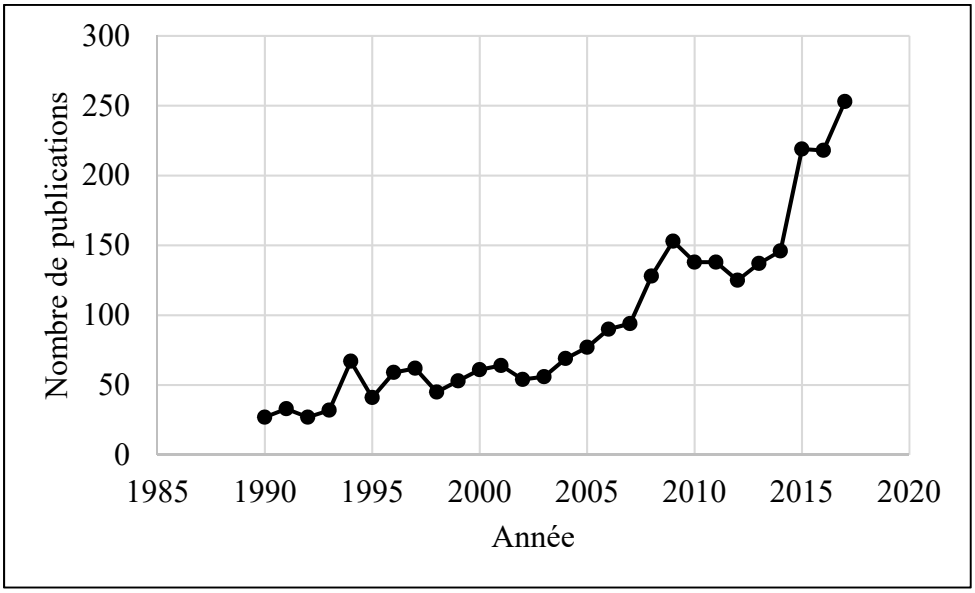

Figure 0.3 Publications annuelles sur les cartes de contrôle, tirée de Web of Science

0.1 Problématique

L'industrie automobile doit se soumettre à la norme IATF 16949 qui contient des requis relatifs au management de la qualité. Par conséquent, les sous-traitants du secteur manufacturier qui font affaire avec les géants de l'automobile doivent s'y soumettre aussi. La section 9.1.1 de la norme IATF 16949 exige que les entreprises utilisent des programmes de prise de mesures, d'analyse et d'évaluation de la qualité. Les manufacturiers sont donc tenus par des clauses contractuelles d'utiliser les outils de CSP.

Le Groupe SOGEFI est un des leaders mondiaux dans le domaine de la fabrication d'équipements d'origine (FÉO, en anglais : *original equipment manufacturer, OEM*) pour l'industrie automobile. Sa filiale montréalaise possède un système de gestion de la qualité à la fine pointe de la technologie et souhaite l'améliorer davantage pour faire de l'analyse prédictive avec ses cartes de contrôle. La première phase de ce projet consiste à automatiser la lecture et l'interprétation des cartes de contrôle et constitue le sujet de cette recherche. Ces cartes sont faites à partir de mesures géométriques prises à l'aide de machines de mesure de coordonnées (MMC). Une fois prises par un technicien spécialisé, ces mesures sont téléchargées dans une base de données SQL. Ces données sont ensuite traitées par un module de post-traitement statistique pour en faire l'analyse.

Il existe plusieurs méthodes pour automatiser l'analyse des cartes de contrôle, mais la littérature est disparate et incomplète. Il n'existe pas de référence qui permet de déterminer quel algorithme est le meilleur pour une application donnée et les informations nécessaires pour reproduire les résultats sont souvent manquantes. L'objectif de cette recherche est d'effectuer une comparaison détaillée et impartiale des différents algorithmes d'IA en CSP pour ce cas de figure.

0.2 Contribution

En prenant comme point d'appui les besoins du département de la qualité de la filiale montréalaise du Groupe SOGEFI et l'absence d'une étude comparative complète dans la littérature, les travaux présentés dans cette recherche visent à déterminer quel algorithme a le meilleur potentiel pour ce besoin spécifique.

Pour conclure, cette section a mis l'emphase sur les besoins et les événements qui ont mené à l'automatisation de la lecture de cartes de contrôle avec l'utilisation de l'IA. Les besoins de l'entreprise SOGEFI ont été. La section suivante contient la revue de littérature sur les algorithmes d'apprentissage machine utilisés dans le domaine.

CHAPITRE 1

REVUE DE LITTÉRATURE

Une étude systématique de la littérature révèle qu'il existe cinq méthodes principales utilisées pour la RMCC :

1. Les méthodes basées sur des règles empiriques ;
2. Les RNA ;
3. Les machines à vecteurs de support (en anglais : *support vector machines*, *SVM*) ;
4. Les arbres décisionnels (en anglais : *decision trees*, *DT*) ;
5. Les forêts aléatoires (en anglais : *random forests*).

La première partie de cette revue de littérature sert d'introduction aux systèmes experts (SE, en anglais : *expert systems, ES*). Ces systèmes utilisent parfois des algorithmes d'apprentissage machine, mais l'objectif de cette première discussion est de présenter leur structure d'un point de vue général. La deuxième partie discute des avancées en reconnaissance de formes avec les RNA. Les algorithmes neuronaux sont les plus utilisés dans le domain. La troisième partie présente les résultats expérimentaux de trois articles qui utilisent les techniques AD, MVS et des propositions hybrides composées de ces deux techniques. Les trois dernières parties établissent les bases théoriques des RNA, des AD et des MVS dans cet ordre.

Dans ce texte, le terme « système expert » possède deux significations distinctes. Ce problème existe aussi dans la littérature, mais la différence entre les deux est assez évidente dans le contexte. Dans la section 1.1, un système expert est un système général qui est composé d'une base de données de connaissances et d'un moteur d'inférence. Dans le reste du texte, un système expert ou spécialisé est un algorithme d'apprentissage machine entraîné sur un sous-ensemble de l'espace d'attributs.

1.1 Les systèmes experts

Les systèmes experts sont des logiciels qui sont reliés à au moins deux sources de données : une base de données qui contient un ensemble de règles et un flux de données qui vient du processus à contrôler. Les règles sont basées sur les connaissances d'experts dans le domaine et sont encodées sous la forme de conditions logiques. Le tout est relié à un moteur d'inférence qui applique les règles. Ce dernier produit un résultat qui est ensuite communiqué aux utilisateurs par le biais d'une interface graphique (figure 1.1).

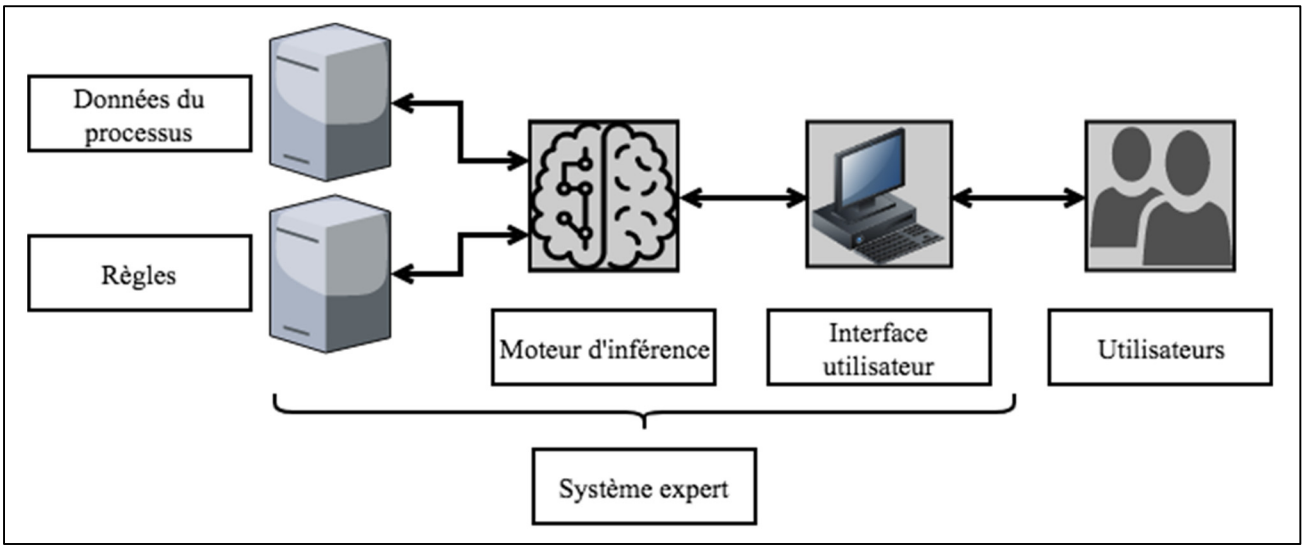

Figure 1.1 Schéma simplifié d'un système expert

Shewhart (1996) propose un système expert qui utilise un réseau de neurones comme moteur d'inférence. Les cartes de contrôles sont composées de 40 mesures $\boldsymbol{v} = \langle v_1, v_2, \dots, v_{40} \rangle$ qui sont présentées à l'algorithme sous forme de vecteurs à 20 dimensions $\boldsymbol{x} = \langle x_1, x_2, \dots, x_{20} \rangle$. Les 10 premières dimensions sont des nombres réels dont les valeurs numériques correspondent à des statistiques qui caractérisent l'ensemble de points. Guh et coll. (1999) utilisent aussi des statistiques comme entrées à leur SE. Les 10 dernières dimensions sont des booléens qui sont déterminés à partir des règles de la base de données.

Par exemple, la variable x_1 est calculée par la méthode de correspondance de modèle (en anglais : *template matching*) en comparant la carte de contrôle à une carte type idéale (c.-à-d. un modèle idéalisé d'un motif quelconque) représentée par le vecteur $\boldsymbol{w} = \langle w_1, w_2, \dots, w_{40} \rangle$.

La valeur numérique de x_1 corresponds à l'écart quadratique moyen :

$$x_1 = \frac{1}{40} \sum_{i=1}^{40} (w_i - v_i)^2 \tag{1.1}$$

Un autre exemple est le booléen x_{14} qui est posé comme étant égal à 1 si $x_1 > 1{,}8$. L'auteur justifie cette règle par le fait qu'une grande proportion des motifs de hausse subite ont historiquement un écart quadratique moyen supérieur à 1,8 pour le procédé considéré. Les données utilisées pour entraîner le modèle ont été générées à partir de 70 000 vraies cartes de contrôle auxquelles a été ajouté du bruit aléatoire. Chacune des cartes possède une étiquette qui est représentée mathématiquement par le vecteur de sortie $\boldsymbol{y} = \langle y_1, y_2, \dots, y_9 \rangle$. Le vecteur de sortie comprend 9 booléens qui servent à identifier une des neuf classes considérées. Le critère de performance utilisé est la précision telle que définie dans le contexte de classification binaire par Metz (1978) et aussi selon la norme ISO 5725. La précision (P) correspond à la proportion de bonnes classifications donnée par l'équation suivante :

$$P = \frac{VP + VN}{VP + VN + FP + FN} \tag{1.2}$$

Dans cette équation, (VP) représente le nombre de vrais positifs (VN), le nombre de vrais négatifs, (FP) le nombre de faux positifs et (FN) le nombre de faux négatifs. La précision du système expert est supérieure ou égale à 97,3% pour tous les motifs à l'exception du motif cyclique.

Les travaux de Shewhart (1996) utilisent comme vecteurs d'entrée les valeurs des cartes de contrôle sous forme de séries temporelles. Pour une fenêtre de temps de 60 points, la dimension d'un vecteur d'entrée est de 60×1. Cette approche a le désavantage d'exiger beaucoup de ressources calculatoires et de mémoire, car le nombre de paramètres du réseau de neurones est proportionnel à la dimension du vecteur d'entrée. En réponse à cette problématique, Bag et coll. (2012) proposent un SE basé sur des AD entraînés avec l'algorithme CART. Les cartes de contrôle utilisées contiennent 32 mesures et les variables utilisées sont des facteurs de formes (c.-à-d. des valeurs numériques qui caractérisent directement la géométrie des motifs) plutôt que des valeurs statistiques. Cette manière de représenter les données à l'avantage de réduire les dimensions des vecteurs d'entrée de 32×1 à 7×1. Les auteurs s'appuient sur les travaux de Pham et Wani (1997) et ceux de Gauri et Chakraborty (2006, 2007) pour justifier ce choix.

Les auteurs n'y font pas référence, mais ce phénomène fut documenté par le statisticien Francis Anscombe (Anscombe, 1973) qui créa quatre jeux de données très différents, mais qui possèdent tous la même droite des moindres carrés montrée en bleu sur la figure 1.2.

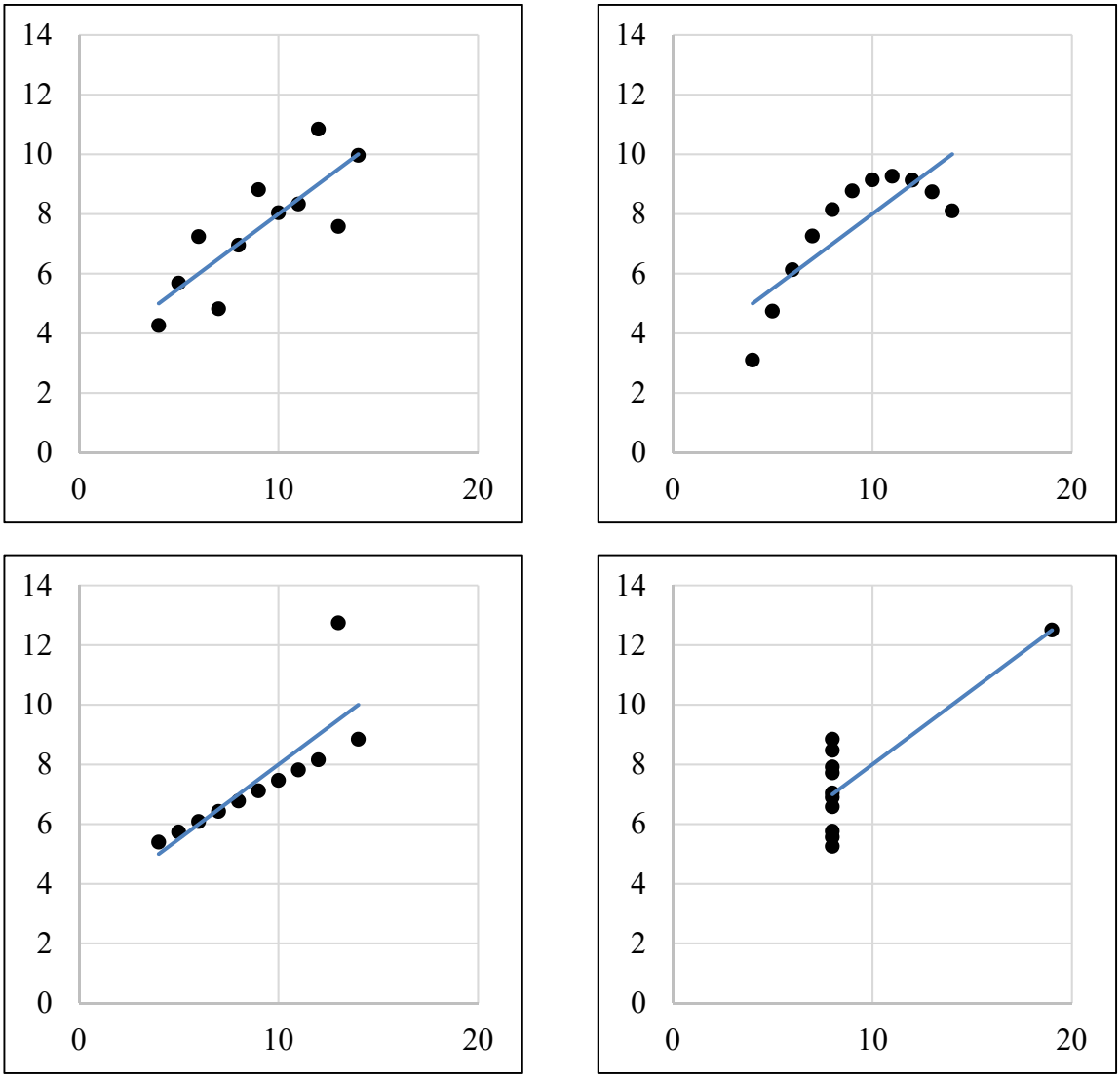

Figure 1.2 Quartet d'Anscombe, tirée de Anscombe (1973)

Un exemple de facteur de forme est la pente (β_1) de la droite des moindres carrés ($\hat{y}_i = \beta_0 + \beta_1 x_i$) de la carte de contrôle donnée par l'équation suivante :

$$\beta_1 = \left(\sum_{i=1}^{N} y_i(e_i - \bar{e})\right) \Big/ \left(\sum_{i=1}^{N} (e_i - \bar{e})^2\right) \tag{1.3}$$

Le terme e désigne l'erreur qui correspond à la différence entre la vraie valeur y_i et la prédiction $\hat{y}_i$, soit $e_i = \hat{y}_i - y_i$. Par exemple, le motif décroissant de la figure 1.3 est généré synthétiquement avec l'équation $y = -0{,}4t + 20 + r(t)$ où $r(t)$ est un nombre aléatoire situé entre -10 et 10 qui est tiré d'une distribution normale. Les pentes de la droite de régression ($\beta_1 = -0{,}4889$) et de l'équation génératrice ($\beta_1 = -0{,}4$) sont reliées par un lien de causalité trivial.

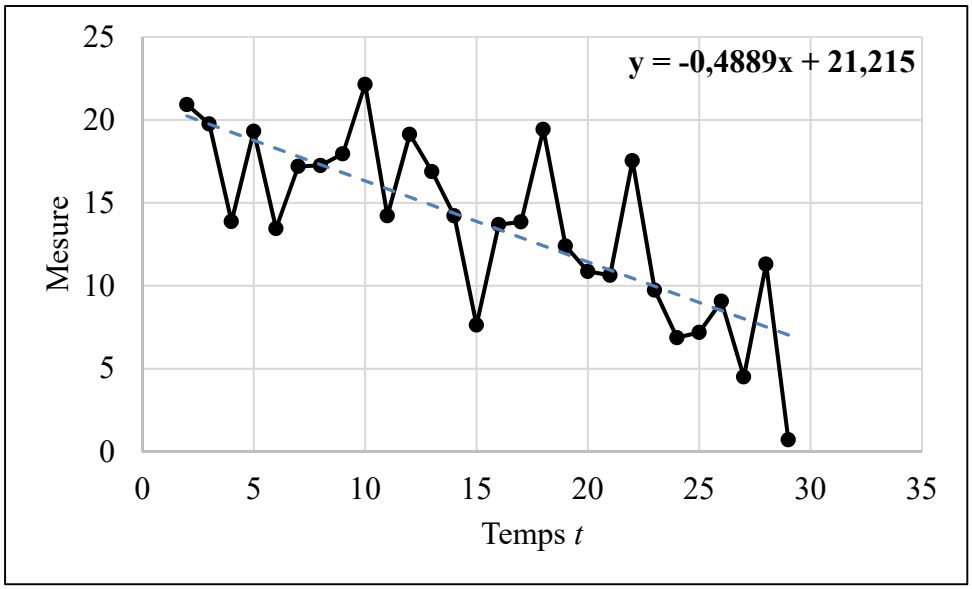

Figure 1.3 Carte de contrôle synthétique avec un motif de tendance décroissante

Le SE développé par les auteurs est muni d'une interface graphique (figure 1.4) qui permet à l'utilisateur d'entrer les données et les spécifications désirées comme les limites de contrôle.

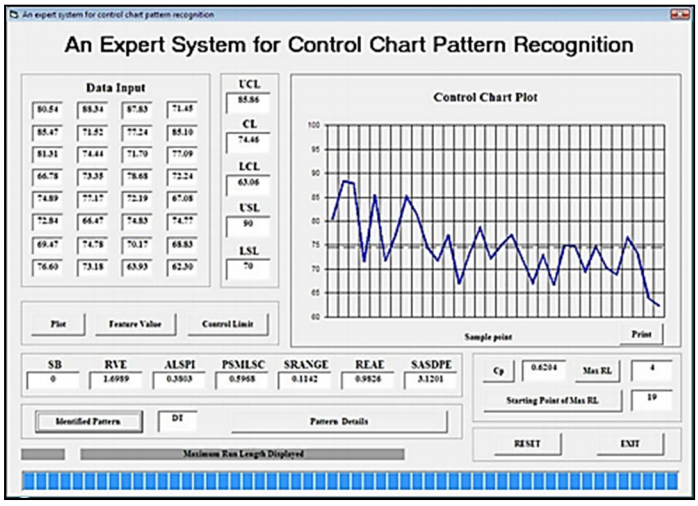

Figure 1.4 Interface utilisateur d'un système expert, tirée de Bag et coll. (2012)

Le moteur d'inférence identifie un ou plusieurs motifs et propose ensuite une liste de causes possibles et d'actions correctives à prendre (figure 1.5).

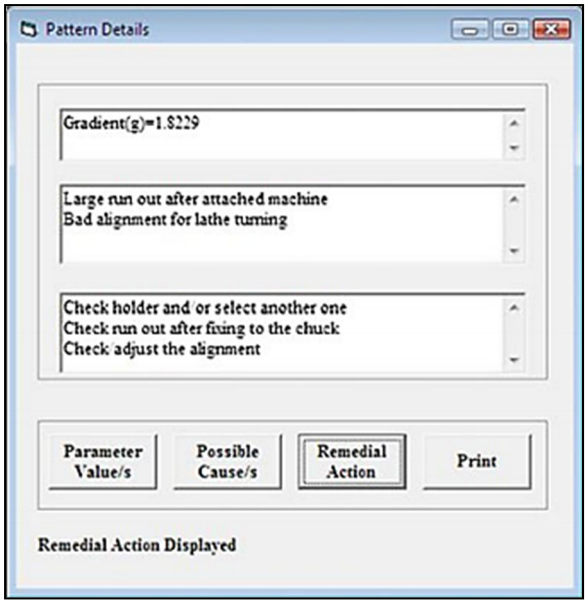

Figure 1.5 Résultats d'analyse d'un système expert, tirée de Bag et coll. (2012)

Les SE ont l'avantage d'être plus faciles à concevoir qu'un programme traditionnel, car ils utilisent principalement l'algèbre de Boole (c.-à-d. des conditions logiques simples) pour effectuer leur travail. Malgré cet avantage, l'intérêt pour les SE a grandement diminué avec le temps, tel qu'il a été souligné par Hahicha et Ghorbel (2012) qui remarquent que la proportion des articles publiés sur ce sujet passe de 28,57% pour la période allant de 1991 à 2000 à 12,32% pour la période allant de 2001 à 2010. La base de données Web of Science appuie ce constat en montrant que pendant la période allant de 1990 à 1993, plus de mille articles sont publiés annuellement en moyenne sur les systèmes experts tandis que pour la quasi-totalité de la période allant de 2000 à 2017, ce nombre est inférieur à 300. Il s'agit ici d'une baisse relative de 70%. Le problème principal des SE est celui de l'acquisition des connaissances. La base de données qui contient les règles du système est spécifique à une seule application et même s'il est facile à programmer, les experts d'un domaine donné sont souvent difficiles à rejoindre et dispendieux. Finalement, les systèmes experts ne sont pas véritablement « intelligents », car ils n'apprennent pas d'eux-mêmes, du contexte ou de leurs erreurs. Toute amélioration à ces systèmes nécessite l'intervention d'un opérateur.

Il a été montré qu'il existe plusieurs architectures possibles pour les SE. Certains emploient une méthode basée sur des règles (c.-à-d. heuristique) et d'autres utilisent des méthodes hybrides qui combinent des algorithmes d'apprentissage machine comme les réseaux de neurones artificiels et les arbres décisionnels et les règles empiriques. L'avantage du SE est qu'il peut identifier les motifs sur les cartes de contrôle et proposer des solutions basées sur les connaissances d'experts dans le domaine. Au fil du temps, cette approche fut délaissée et les chercheurs se concentrent maintenant sur les mécanismes d'inférence.

1.2 Les arbres décisionnels et les machines à vecteurs de support

Wang et coll. (2008) proposent un système hybride (figure 1.6) qui utilise un AD pour classifier les anomalies détectées par un filtrage basé sur des règles empiriques.

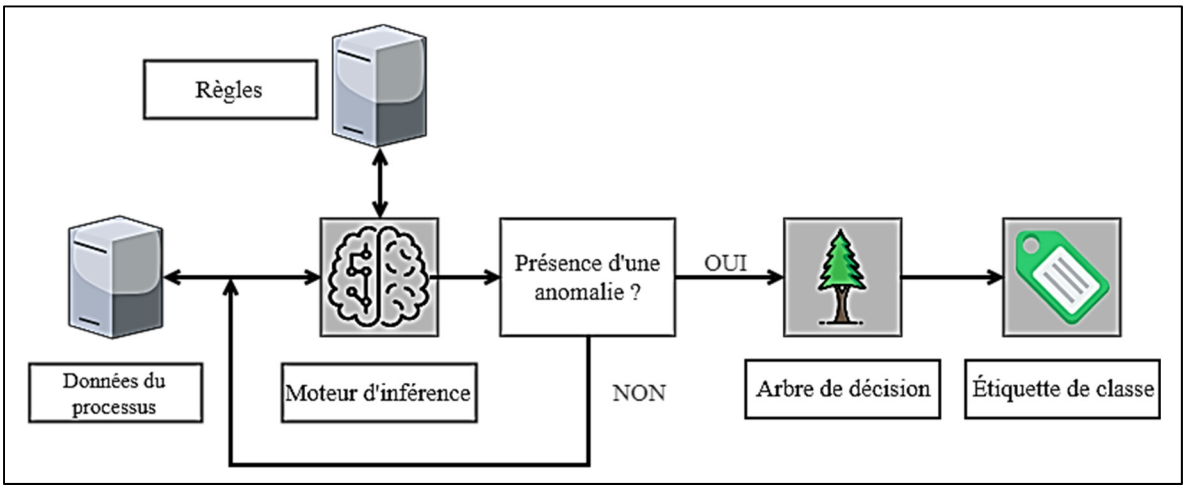

Figure 1.6 Schéma du système hybride utilisé par Wang et coll. (2008)

La classification est basée sur des facteurs de formes comme la pente de la droite des moindres carrés comme discuté dans la section sur les systèmes experts. L'algorithme utilise la technique d'émondage anticipatif (en anglais : *pre-pruning*) en minimisant l'entropie de Shannon comme critère de décision. Pour la classification d'une variable Y, l'entropie de Shannon est donnée par:

$$H(Y) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{1.4}$$

Où p_i correspond la probabilité d'appartenance à la i-ème classe. Le jeu de données utilisé est constitué de 1 200 échantillons synthétisés mathématiquement (c.-à-d. 200 pour chacune des classes). Les données ont été séparées en deux parties égales pour l'entraînement et la

validation. Les auteurs concluent d'abord que les AD sont aussi performants que les RNA pour la tâche considérée, mais n'offrent aucune comparaison directe entre les deux.

Shao (2012) présente une méthode modulaire récursive basée sur un ensemble de MVS. Il considère les six motifs suivants : normal (A), cyclique (B), tendances positives (C) et négatives (D), hausse subite (E) et baisse subite (F). Il les réorganise ensuite en quatre sous-ensembles en combinant le motif de tendance positive avec le motif de hausse subite et le motif de tendance négative avec le motif de baisse subite. Il crée ensuite une structure similaire à un AD où il remplace les nœuds par des machines à vecteurs de supports. Chaque MVS effectue une classification binaire ce qui donne finalement un arbre à cinq nœuds et dix branches (figure 1.7).

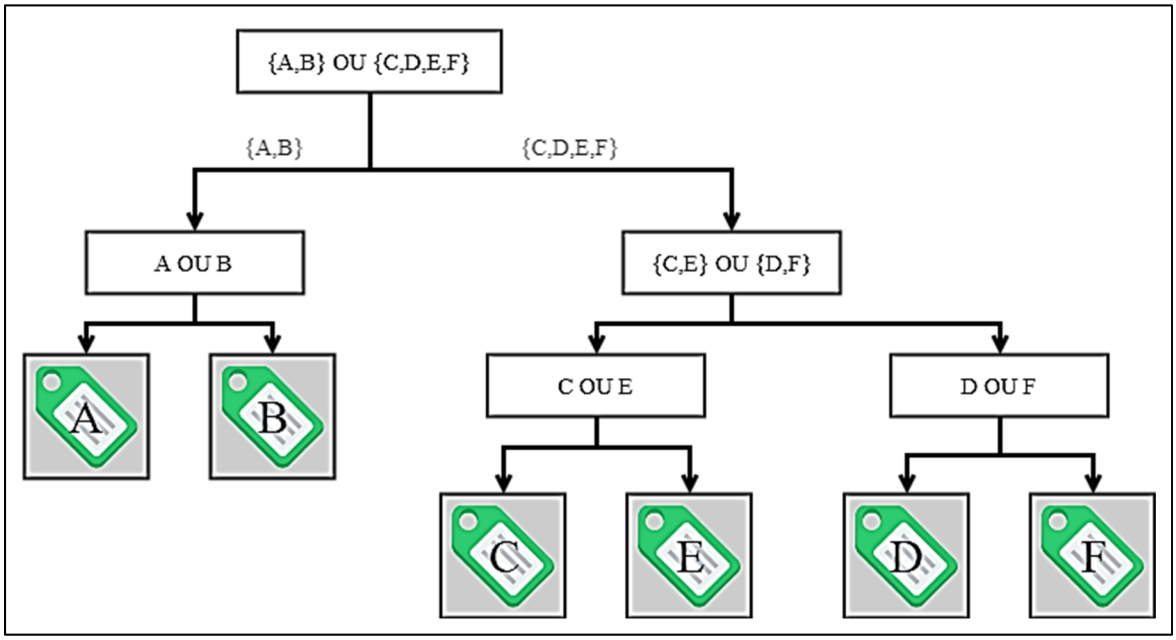

Figure 1.7 Schéma du système hybride utilisé par Shao (2012)

Le jeu de données utilisé est composé de 2 000 échantillons par classe et est généré à l'aide de la méthode de simulation de Monte-Carlo. Les données sont séparées en deux parties égales pour l'entraînement et la validation. L'étude conclut que les MVS singuliers sont plus

performants que les réseaux de neurones et que la structure hybride en forme d'arbre améliore la précision de 2% par rapport aux systèmes singuliers.

Le terme « hybride » signifie un modèle qui est composé de plus d'une instance d'un algorithme (p. ex. trois MVS ou une MVS et une FA). Les modèles composés d'une seule instance d'un algorithme sont appelés des modèles ou des systèmes singuliers.

Othman et Eshames (2012) arrivent à des conclusions différentes en comparant les RNA multicouches, les AD et les MVS. Selon eux, les AD sont les plus précis, suivis des RNA et des MVS. Ces résultats sont obtenus pour un jeu de données synthétique qui contient 100 exemples de chacune des six classes de motifs. Les auteurs se limitent à une description sommaire de chacun des algorithmes sans donner les détails des différentes architectures des algorithmes.

1.3 Les réseaux de neurones artificiels

Les RNA sont les modèles les plus utilisés en classification automatisée de motifs de cartes de contrôle. Hahicha et Ghorbel (2012) rapportent que pour la période allant de 1991 à 2000, 56,10% des publications révisées utilisent les RNA et que pour la période allant de 2001 à 2010, ce nombre grimpe à 65,75%. Ce constat est appuyé par le nombre d'articles publiés sur les RNA qui montre une hausse annuelle moyenne de 6,3% pour cette période (figure 1.8). Cette tendance s'accélère ensuite pour la période allant de 2011 à 2017 avec une hausse annuelle moyenne de 13,1%.

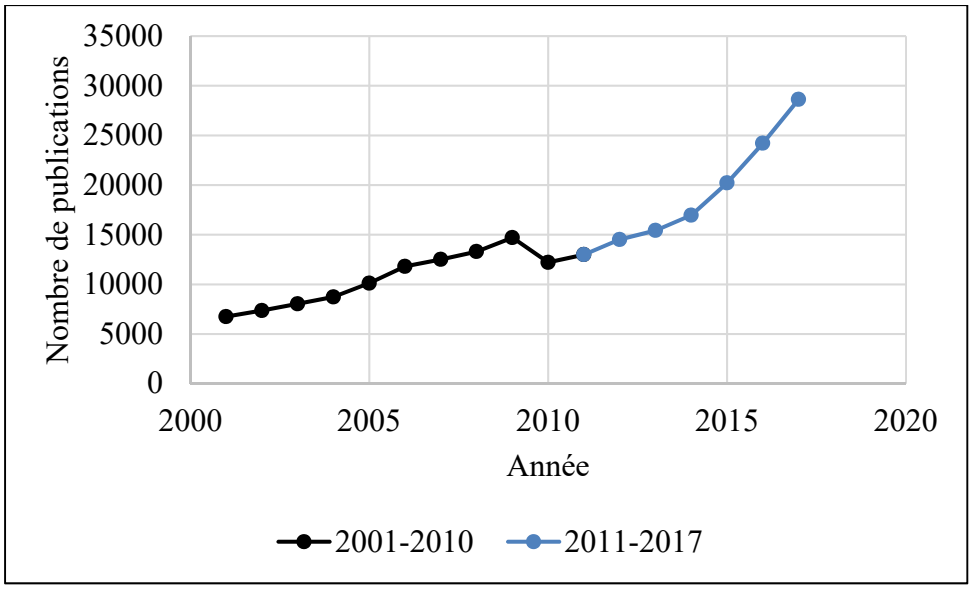

Figure 1.8 Publications annuelles sur les RNA, tirée de Web of Science

Pugh (1989) est le premier à expérimenter avec les RNA et les cartes de contrôle. Il utilise un réseau singulier avec une topologie 5-4-1, soit cinq neurones dans la couche d'entrée pleinement connectée avec les quatre neurones de la couche cachée et un neurone dans la couche de sortie. La couche cachée applique la fonction d'activation sigmoïde qui est la fonction d'activation la plus utilisée dans la grande majorité des domaines et qui est donnée par l'équation 1.5.

$$\varphi(x) = \frac{1}{1 + e^{-x}} \tag{1.5}$$

Le jeu de données utilisé est constitué de 300 cartes de contrôle synthétiques et les poids sont déterminés en utilisant l'algorithme de rétropropagation du gradient (RPG, en anglais : *backpropagation*, *BP*). La taille de la fenêtre (c.-à-d. le nombre de points qui définissent une carte) est de cinq points qui sont représentés un pour un par les nœuds de la couche d'entrée. Il conclut que les RNA sont aussi efficaces que les cartes de contrôle traditionnelles pour détecter les changements de valeurs moyennes à la suite d'une hausse subite. Cette étude constitue la preuve de concept des RNA en RMCC.

Pham et Oztemel (1993) sont les premiers à utiliser une structure hybride. Cette dernière (figure 1.9) est composée d'un module basé sur des règles et deux modules RNA spécialisés de type perceptron multicouche (en anglais: *multilayer perceptron*, *MLP*).

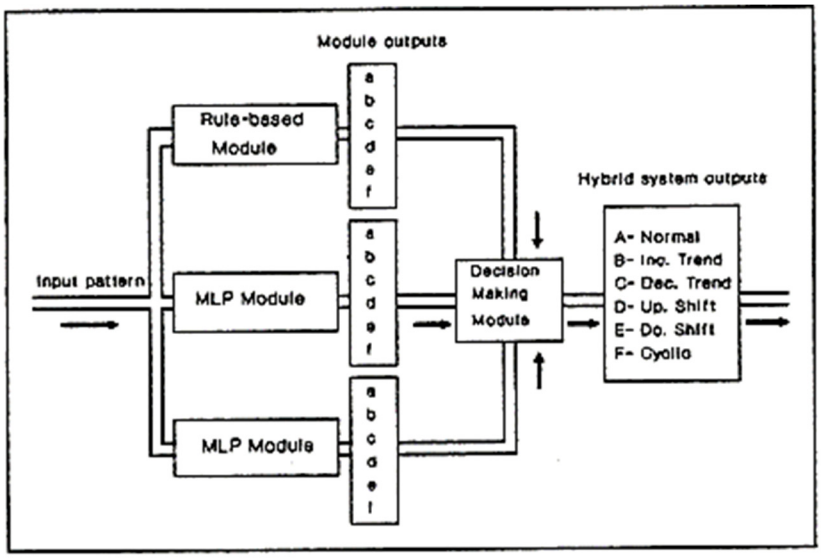

Figure 1.9 Schéma d'un système hybride, tirée de Pham et Oztemel (1993)

Les deux RNA ont une topologie 60-35-6 et des fenêtres de 60 points comme vecteurs d'entrée. Le nombre de neurones dans la couche cachée est déterminé par essai et erreur en utilisant la précision comme critère de sélection. L'entraînement est effectué avec l'algorithme de rétropropagation du gradient avec 83 cartes pour chacun des six motifs utilisés, donc 498 cartes au total.

Dans leur première expérience, les RNA utilisent le même jeu de données et dans la deuxième les jeux de données sont différents. Ils concluent que les systèmes hybrides sont plus précis que les systèmes singuliers et que les systèmes hybrides sont encore plus précis lorsque les modules sont entraînés en utilisant des jeux de données différents. Ils introduisent le terme « réseaux spécialisés » pour désigner les RNA d'un système hybride. Cheng (1997) arrive lui aussi à la conclusion que les réseaux hybrides sont plus performants que les réseaux singuliers (figure 1.10).

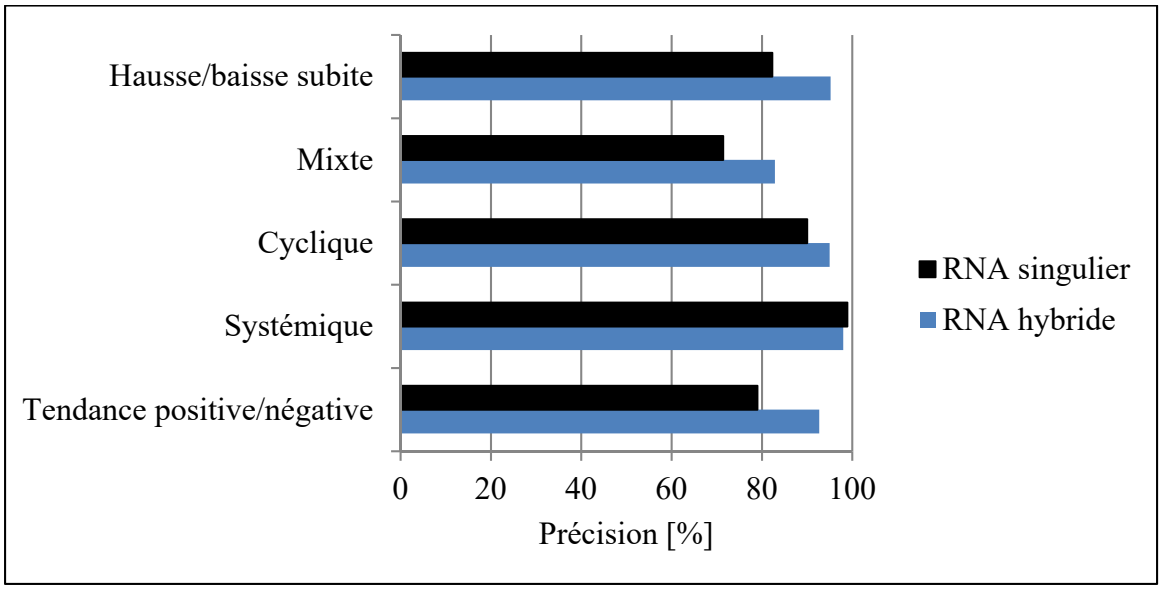

Figure 1.10 Comparaison entre les RNA singuliers et hybrides, adaptée de Cheng (1997)

Les RNA hybrides sont particulièrement adaptés à détecter les changements subits et les motifs mixtes qui se caractérisent par la présence simultanée de deux ou plusieurs motifs. La figure 1.11 montre un motif mixte composé d'un motif de tendance négative et d'un motif cyclique.

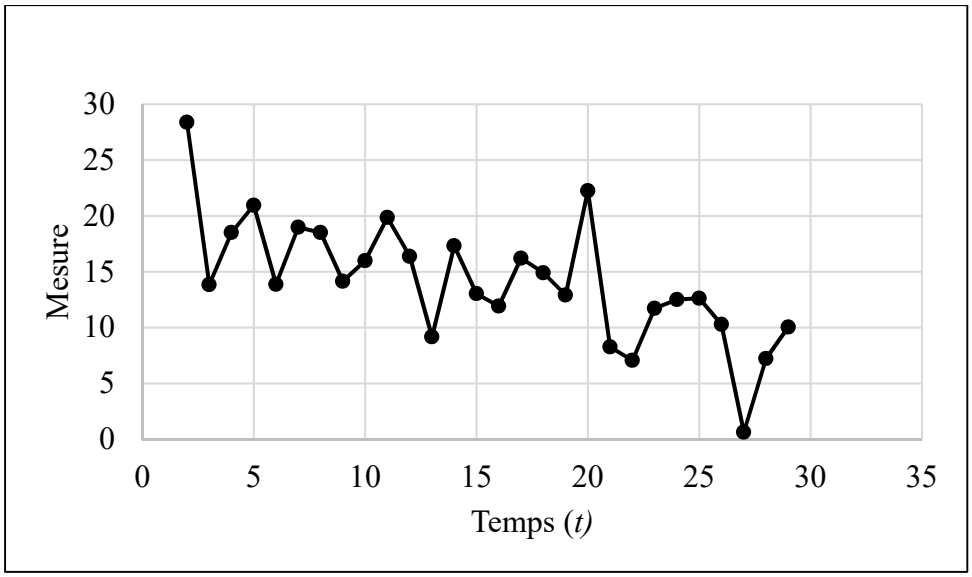

Figure 1.11 Exemple de motif mixte

L'étude de Cheng (1997) se distingue par son évaluation de la performance des RNA en simulant un environnement « en temps réel ». Pour faire cela, une partie des cartes est simulée comme étant en contrôle (c.-à-d. en l'absence de motif non naturel) et un signal non naturel est introduit à partir d'un temps donné (figure 1.12).

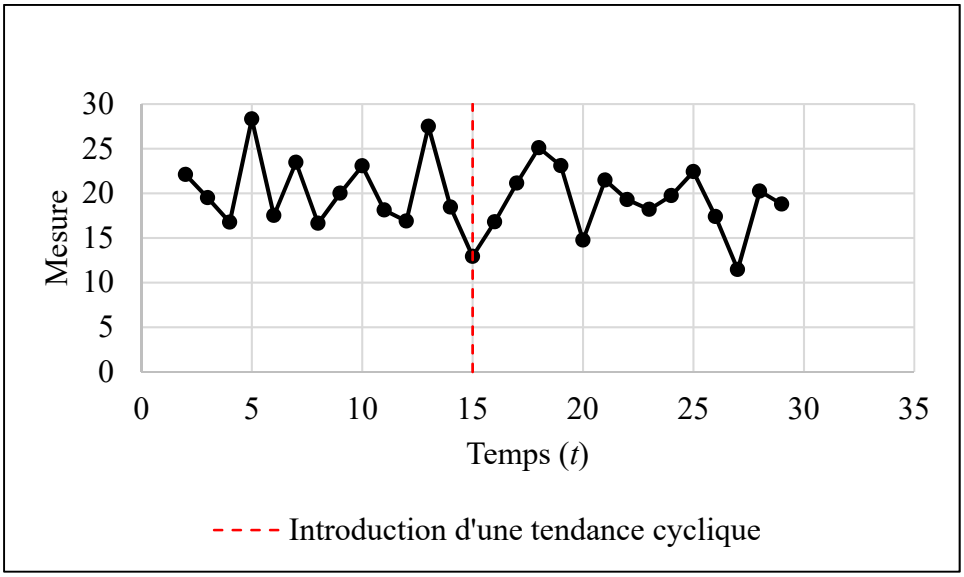

Figure 1.12 Exemple de motif mixte retardé

Jang et coll. (2003), présentent un RNA entraîné avec l'algorithme de quantification vectorielle linéaire (QVL, en anglais : *Linear Vector Quantization*, *LVQ*). La topologie 16-8-5 fut déterminée par essai et erreur pour la couche d'entrée et la couche cachée. Contrairement aux autres études mentionnées, la fonction d'activation est la fonction tangente hyperbolique donnée par :

$$\varphi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{1.6}$$

Les auteurs expliquent que l'image de la fonction bornée par $[-1,1]$ permet d'améliorer la détection des changements de direction des hausses subites et des tendances positives et négatives. L'étude se distingue de deux autres manières. Premièrement, elle évalue l'impact d'une augmentation du bruit statistique et conclut qu'une augmentation de la variance du bruit améliore la précision du classificateur. Pour les motifs cycliques de petites amplitudes, la précision passe de 100% à 74% lorsque l'écart-type du bruit passe de 0,3 à 0,1. Deuxièmement, elle présente une étude de cas dans le secteur automobile. Elle rapporte que le RNA développé fut utilisé avec des données fournies par la société General Motors générées à partir d'une machine à mesure de coordonnées optiques (MMCO) et que le réseau neural a permis de détecter des composantes usées à remplacer à partir de motifs des cartes de contrôles.

Ripley (1996) suggère d'effectuer des comparaisons en double aveugle afin d'éviter le traitement préférentiel. Il existe malgré tout quelques articles comme celui de Laosiritaworn et Bunjongjit (2012) et de Zhao et coll. (2017) qui comparent les RNA à la méthode des valeurs k plus proches voisins (en anglais : *k nearest neighbor*, *knn*) et aux machines à vecteurs de support respectivement. Il n'est cependant pas possible de déclarer de vainqueur, car il n'y a qu'une seule étude, celle de Othman et Eshames (2012), qui compare plus de deux types d'algorithmes à la fois, mais les auteurs ne fournissent pas les détails nécessaires pour reproduire leurs résultats.

1.4 Les cartes de contrôle synthétiques

Les algorithmes d'apprentissage machine étudiés sont évalués en utilisant des cartes de contrôles synthétiques générées à partir de modèles substituts. Cette approche est proposée en premier en RMCC par Alcock et Manopoulos (1997) et est ensuite est devenue courante dans le domaine. Hahicha et Ghorbel (2012) rapportent que sur 73 articles publiés entre 2001 et 2010, 68 (93,15%) utilisent ce type de données. Dans cette section, les équations mathématiques utilisées pour générer les motifs non naturels des cartes de contrôles sont présentées. Afin de les expérimenter, ces équations ont été codées sous la forme de fonctions à l'aide du langage de programmation Python version 3.6.

1.4.1 Le motif normal

Le motif normal ou naturel est celui qui est observé quand un procédé est sous contrôle (c.-à-d. en l'absence de cause assignable). Le motif normal est généré à partir de l'équation suivante :

$$y(t) = \mu + r(t)s \tag{1.7}$$

Dans l'équation 1.7, μ corresponds à la valeur moyenne, $r(t)$ est la composante aléatoire distribuée selon $\mathcal{N}(0,1)$ multipliée par le scalaire s. Un exemple est présenté à la figure 1.13.

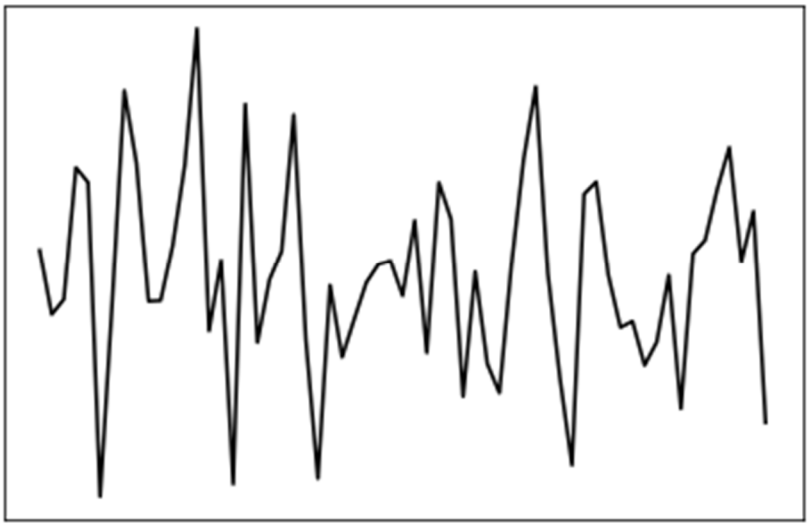

Figure 1.13 Motif normal synthétique

1.4.2 Le motif cyclique

Le motif cyclique est caractérisé par la régularité avec lequel un sous-motif est répété. Montgomery (2005) attribue les causes assignables suivantes à la présence de motifs cycliques : changements environnementaux systématiques (p. ex. le changement de température entre le jour et la nuit), la fatigue des opérateurs, les mouvements de va-et-vient des machines et la fluctuation du voltage de la source de tension. Le motif cyclique est généré à partir de l'équation suivante :

$$y(t) = \mu + r(t)s + A\sin\left(\frac{2\pi t}{T}\right) \tag{1.8}$$

L'équation 1.8 n'est rien de plus que le motif normal de l'équation 1.7 superposé à un signal sinusoïdal d'amplitude A avec une période T. Un exemple est donné à la figure 1.14.

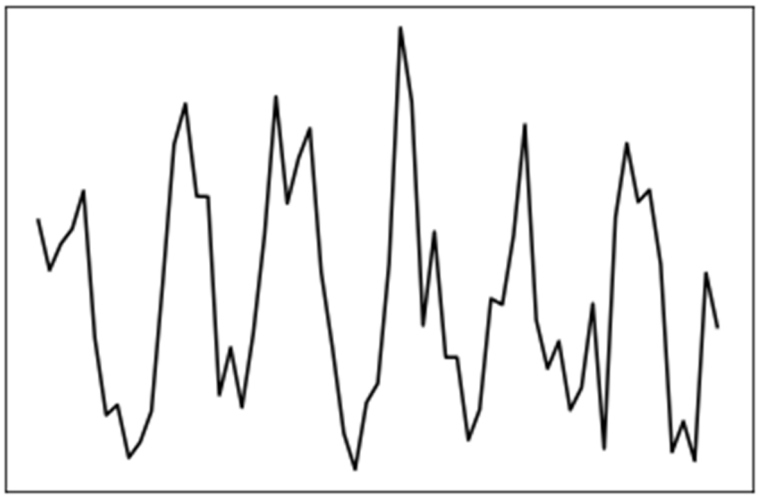

Figure 1.14 Motif cyclique synthétique

1.4.3 Les motifs de tendances positives et négatives

Les motifs de tendances positives et négatives sont caractérisés par des mouvements continus dans une direction. Montgomery (2005) attribue ces motifs à l'usure graduelle des outils, la fatigue des opérateurs et parfois aux variations saisonnières comme la température. Les motifs de tendances positives et négatives sont générés à partir de l'équation suivante où g est la pente :

$$y(t) = \mu + r(t)s \pm gt \tag{1.9}$$

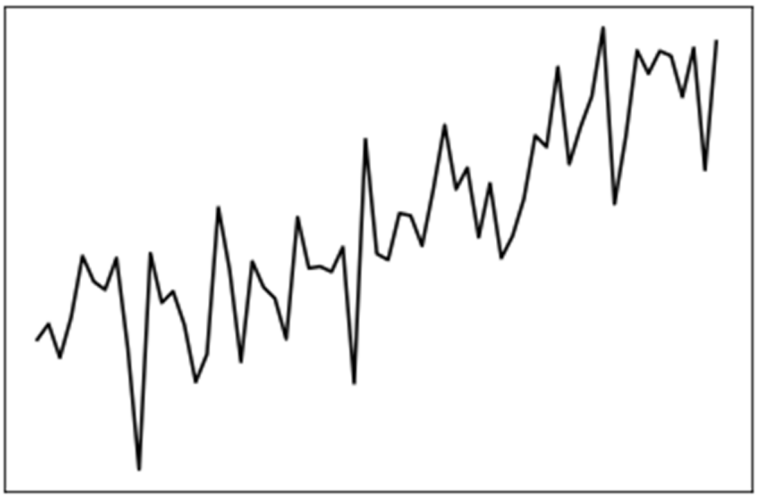

Figure 1.15 Motif de tendance positive synthétique

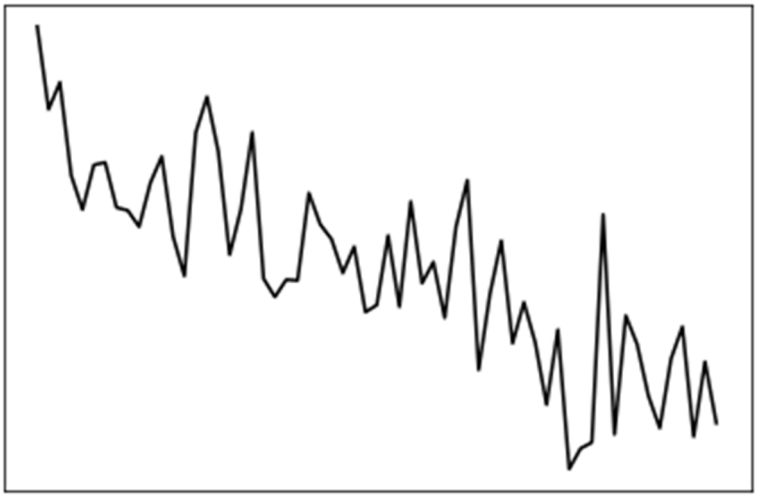

Figure 1.16 Motif de tendance négative synthétique

1.4.4 Les motifs de hausses et de baisses subites

Les motifs de hausses et baisses subites sont caractérisés par un changement rapide de la valeur moyenne d'un procédé. Montgomery (2005) attribue ces motifs à l'introduction de nouvelles méthodes de fabrication, de nouveaux matériaux ou de nouveaux opérateurs. Les motifs de hausses et de baisses subites sont générés par l'équation suivante où k représente le changement de la moyenne et $H(t-c)$ est la fonction de Heaviside avec c comme paramètre de translation :

$$y(t) = \mu + r(t)s \pm kH(t-c) \tag{1.10}$$

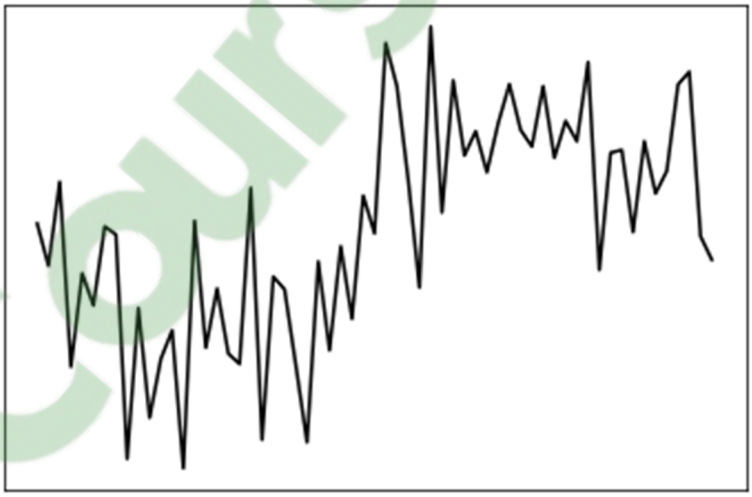

Figure 1.17 Motif de hausse subite synthétique

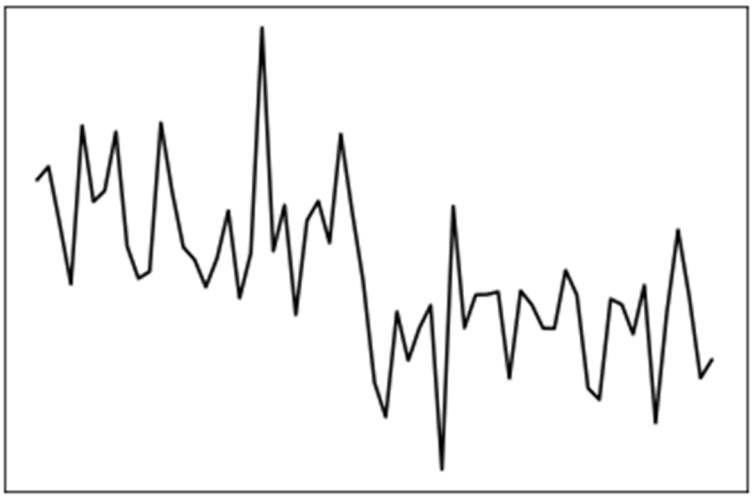

Figure 1.18 Motif de baisse subite synthétique

1.4.5 La classification avec les réseaux de neurones artificiels

La popularité des RNA en RMCC n'a cessé d'augmenter depuis leurs premières applications dans le domaine vers la fin des années 80 (voir Pugh, 1989). Ils représentent à eux seuls 65,75% de toutes les publications pour la période allant de 2001 à 2010 (Hahicha et Ghorbel, 2012).

La quasi-totalité des études publiées sur la RMCC avec les RNA utilise les réseaux multicouches entraînés avec l'algorithme de RPG. Les réseaux multicouches se distinguent du perceptron (c.-à-d. un neurone individuel) par la propriété d'être capables d'approximer n'importe quelle fonction continue à n'importe quel degré de précision souhaité (Hornik, 1991). Les bases mathématiques de cette propriété qui est connue sous le nom du théorème d'approximation universel avaient été jetées pas Cybenko (1989). Le problème est que dans les années 60, il n'existait pas encore de moyen pratique pour calculer les poids synaptiques des réseaux multicouches. Le milieu académique s'intéressait donc davantage au perceptron pour des raisons pratiques.

L'intérêt pour le perceptron et les RNA, en général, a grandement diminué quand il a été découvert qu'ils ne pouvaient pas résoudre la fonction OU exclusive, souvent appelée XOR, qui est une fonction élémentaire en algèbre relationnelle.

ENTRÉE		SORTIE
A	**B**	**A XOR B**
0	0	0
0	1	1
1	0	1
1	1	0

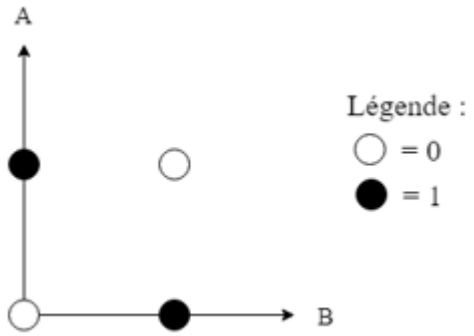

Figure 1.19 Table de vérité et représentation schématique de la fonction OU exclusive

La figure 1.19 présente l'exemple du OU exclusif que le perceptron ne peut pas résoudre. Autrement dit, le perceptron est limité aux cas qui sont linéairement séparables. Cette limite a été surmontée quand un algorithme efficace, qui existait et datait des années 70, a été utilisé pour calculer efficacement les poids synaptiques des réseaux multicouches. C'est l'algorithme de RPG que l'on utilise encore aujourd'hui. La première application spécifique aux RNA a été initialement publiée en premier par Werbos (1982), suivie par Parker (1985) et LeCun (1985). Il est important de souligner que les ordinateurs étaient aussi beaucoup plus puissants à ce moment-là que pendant les années 60.

Dans le restant de ce chapitre, la mécanique du perceptron est présentée suivie de l'algorithme de rétropropagation du gradient pour les réseaux multicouches.

1.5 Le perceptron

Dans la littérature, le terme perceptron fait référence à la fois à un algorithme de la famille de la descente du gradient (Rosenblatt, 1958) et au modèle mathématique du neurone biologique. Dans cette recherche, le terme perceptron est utilisé pour désigner un neurone artificiel. À la base, le perceptron est composé de branches et d'un noyau (figure 1.21). La i-ème valeur du j-ème vecteur d'entrée notée $x_{i,j}$ est assigné à la i-ème branche. Le j-ème vecteur d'entrée correspond à $\boldsymbol{x_j} = \langle x_{1,j}, x_{2,j}, \dots x_{n,j} \rangle^T$. La i-ème branche multiplie ensuite $x_{i,j}$ par le poids synaptique w_i et l'envoi au noyau qui n'est rien de plus qu'un sommateur qui retourne $s_j = \sum_{i=1}^{n} x_{ij} w_i$. La branche de sortie écrase ensuite s_j avec une fonction d'activation comme la fonction sigmoïde pour obtenir $\hat{y}_j(t) = \varphi(s_j)$. Ces étapes constituent la phase de propagation en avant du perceptron. L'indice t fait référence au temps qui se calcule en époques. L'erreur du j-ème exemple au temps t notée $\delta_j(t)$ est égale à la différence entre la valeur attendue y_i et la valeur calculée par le perceptron, soit $\delta_j(t) = y_j - \hat{y}_j(t)$. Pour simplifier l'algorithme d'apprentissage et la notation, le biais est modélisé par w_0, car $x_{0,j} = 1 \; \forall \; j$.

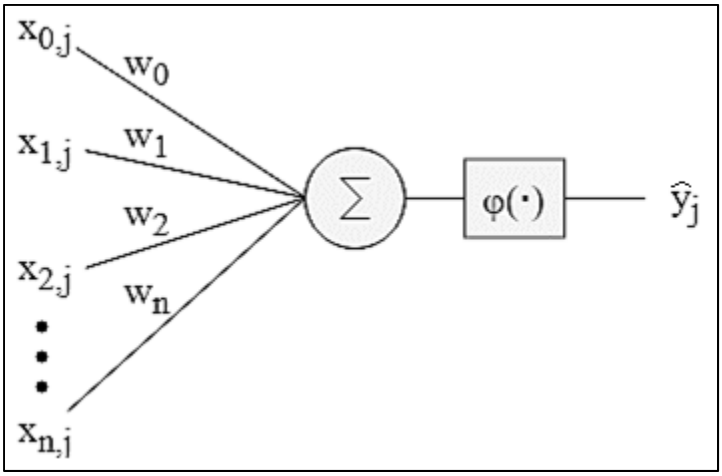

Figure 1.20 Schéma du perceptron

La tâche d'apprentissage consiste à trouver les valeurs w_i qui minimisent la somme des erreurs $\sum_{j=1}^{m} \delta_j$ pour les m exemples. Pour faire cela, l'algorithme de mise à jour des poids entre les époques t et $t+1$ fait des ajustements proportionnels à l'erreur commise multipliée par la i-ème entrée pour le i-ème poids. Il est supposé que plus l'entrée est grande, plus sa contribution à l'erreur est grande aussi. Le i-ème poids est alors ajusté par une plus grande valeur pour compenser. Pour éviter de faire de grands sauts de valeurs de poids synaptiques et accélérer la convergence, le tout est modéré avec la constante α qui est appelée le taux d'apprentissage. La procédure entière ne se résume pas les étapes suivantes :

1. Initialiser les poids avec de petites valeurs aléatoires ;
2. Faire la propagation en avant : $\hat{y}_j(t) = \sum_{i=1}^{n} w_i x_i$;
3. Calculer l'erreur : $\delta_j(t) = y_j - \hat{y}_j(t)$;
4. Mettre à jour les poids : $w_i(t+1) = w_i(t) + \alpha \delta_j(t) x_{i,j}$;
5. Répéter les étapes 2,3 et 4 pour $j \in [1, m]$ où m corresponds au nombre d'exemples.

On note les choses suivantes par rapport à la notation :

1. $\hat{y}_j$ correspond à la prédiction de la classe du j-ème exemple ;
2. y_j correspond à la véritable classe du j-ème exemple ;
3. Pour la classification binaire : $y_j \in \{0,1\}$.

Cet algorithme est connu formellement sous le nom de la règle du delta de Widrow-Hoff.

1.5.1 L'algorithme de la rétropropagation du gradient

La rétropropagation du gradient est l'algorithme utilisé par Pham et Oztemel (1993), Cheng (1997), Sukthomya et Tannock (2005), Wu et Yu (2010), Shao et coll. (2017) et la quasi-totalité des autres articles publiés sur le sujet. Il permet de faire efficacement la mise à jour des poids synaptiques des RNA multicouches et de classer des données avec des frontières de décision arbitrairement complexes. Dans cette section, les règles de mise à jour des poids synaptiques pour la couche de sortie d'un RNA sont développées en suivant une méthode qui s'applique aussi aux couches cachées. Une dérivation plus rigoureuse est offerte par Beale et Jackson (1998) et Haykin (1999).

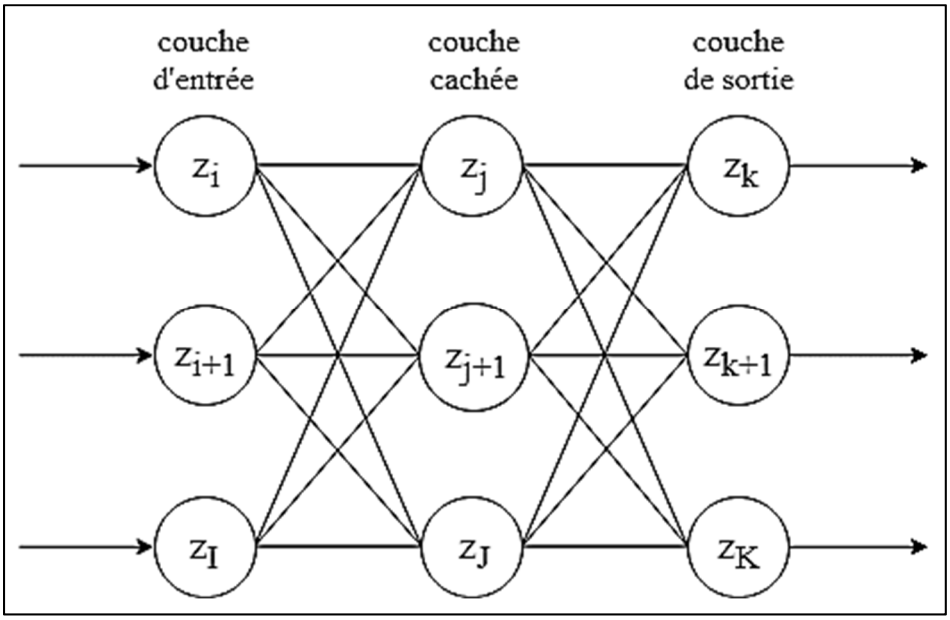

Figure 1.21 Schéma d'un réseau de neurones multicouches

Le processus débute par choisir l'erreur quadratique (équation 1.11) totale du réseau comme fonction objective à minimiser pendant l'entraînement. Le terme quadratique est utilisé par commodité pour faciliter le calcul de la dérivée et pour éviter que les erreurs négatives et positives s'annulent.

$$L = \frac{1}{2}\sum_{i=1}^{k}(z_k - y_k)^2 \tag{1.11}$$

L'algorithme est développé en se concentrant d'abord sur le k-ème nœud de la dernière couche (figure 1.22). Le terme $\partial L/\partial w_{jk}$ représente la sensibilité de l'erreur totale (L) à un changement infinitésimal du poids synaptique qui relie le nœud j de l'avant-dernière couche au nœud k de la dernière.

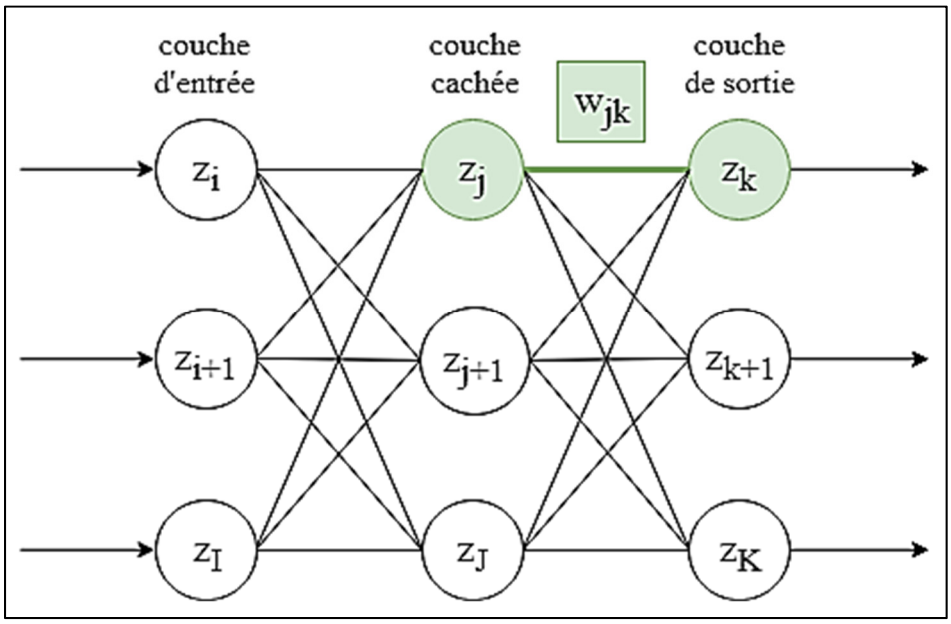

Figure 1.22 RNA partiel considéré (en vert) pour dériver l'algorithme de RPG

La règle de dérivée en chaîne est appliquée à l'équation 1.11 est appliquée :

$$\frac{\partial L}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}\left(\frac{1}{2}\sum_{k=1}^{K}(z_k - t_k)^2\right) = (z_k - t_k)\frac{\partial z_k}{\partial w_{jk}} \tag{1.12}$$

Puisque z_k corresponds à la fonction d'activation de la somme des entrées du nœud k de la couche de sortie. La résultante en équation mathématique est :

$$z_k = \varphi\left(\sum_{i=j}^{J} z_i w_{ik}\right) \tag{1.13}$$

L'équation 1.13 est développée afin d'évaluer le terme $\partial z_k / \partial w_{jk}$ de l'équation 1.12 :

$$z_k = \varphi\big(z_j w_{jk} + z_{j+1} w_{j+1,k} + z_J w_{Jk}\big) \tag{1.14}$$

Ce qui résulte en l'expression suivante :

$$\frac{\partial z_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}\Big(\varphi\big(z_j w_{jk} + z_{j+1} w_{j+1,k} + z_J w_{Jk}\big)\Big) \tag{1.15}$$

La dérivée de la sigmoïde $\varphi(x)$ est donnée par $\varphi'(x) = \varphi(x)\big(1 - \varphi(x)\big)$. La substitution $u = z_j w_{jk} + z_{j+1} w_{j+1,k} + z_J w_{Jk} = z_k$ de l'équation 1.15 est utilisée :

$$\frac{\partial z_k}{\partial w_{jk}} = \varphi(z_k)\big(1 - \varphi(z_k)\big)\frac{\partial z}{\partial w_{jk}}\big(z_j w_{jk} + z_{j+1} w_{j+1,k} + z_J w_{Jk}\big)$$

Donc :

$$\frac{\partial z_k}{\partial w_{jk}} = \varphi(z_k)\big(1 - \varphi(z_k)\big) z_j \tag{1.16}$$

L'équation 1.12 devient :

$$\frac{\partial L}{\partial w_{jk}} = (z_k - t_k)\varphi(z_k)\big(1 - \varphi(z_k)\big)z_j \tag{1.17}$$

Pour alléger la notation, la substitution suivante est effectuée :

$$\delta_k = (z_k - t_k)z_k(1 - z_k) \tag{1.18}$$

Ce n'est pas une substitution arbitraire, car δ_k provient du développement du terme $\partial z_k / \partial w_{jk}$, qui est appelé le gradient local. L'expression finale est donnée par :

$$\frac{\partial L}{\partial w_{jk}} = z_j \delta_k \tag{1.19}$$

Un exercice similaire pour les nœuds cachés démontre que la sensibilité de l'erreur totale au nœud synaptique, qui relie le i-ème nœud d'une couche quelconque au j-ème nœud de la couche cachée suivante, est donnée par :

$$\frac{\partial L}{\partial w_{ij}} = z_i \delta_j \tag{1.20}$$

Le gradient local est donné par :

$$\delta_j = z_j\big(1 - z_j\big)\sum_{k=1}^{K} \delta_k w_{jk} \tag{1.21}$$

L'équation 1.21 peut être interprétée de la même manière que l'équation 1.18 à la différence que la sensibilité de l'erreur ne se limite pas à un seul nœud, mais dépend de la somme des erreurs de la couche suivante.

L'entraînement avec la RPG se résume par les étapes suivantes :

1. Initialiser les poids avec de petites valeurs aléatoires ;
2. Faire la propagation en avant ;
3. Calculer le gradient local pour les K nœuds de sortie : $\delta_k = z_k(1 - z_k)(z_k - t_k)$;
4. Calculer le gradient local pour chacun des J nœuds cachés : $\delta_j = z_j\big(1 - z_j\big)$;
5. Mettre à jour les poids synaptiques de la couche L : $w(t+1) = w(t) + \alpha z_{L-1}\delta_L$.

L'entraînement est arrêté lorsque l'erreur quadratique moyenne $(1/2K)\sum_{i=1}^{k}(y_k - t_k)^2$ atteint une valeur cible prédéterminée, que le gradient de l'erreur devienne inférieur à une valeur critique ou qu'un certain nombre d'époques limite soient atteints.

1.5.2 La séparation des classes

En général, la dernière couche d'un RNA multicouche utilise la fonction softmax comme fonction d'activation pour transformer les valeurs des sorties en probabilités. La classe prédite par le RNA est l'étiquette avec la plus grande probabilité.

1.5.3 Conclusion

Les RNA multicouches sont des RNA qui ont une ou plusieurs couches cachées entre la couche d'entrée et la couche de sortie. Chaque couche est constituée de neurones qui sont en fait des perceptrons. Les neurones prennent la somme des signaux entrants et appliquent une fonction d'activation linéaire ou non linéaire et retournent un signal transformé aux neurones de la couche suivante. L'apprentissage consiste à trouver les valeurs numériques des poids associés aux branches qui relient les neurones de manière à minimiser une fonction objective comme l'erreur quadratique moyenne. L'algorithme le plus utilisé pour calculer les poids synaptiques est l'algorithme de RPG.

La prochaine section présente les fondements théoriques de la classification statistique avec les arbres décisionnels.

1.6 La classification avec les arbres décisionnels

Dans la dernière section, la théorie concernant la classification statistique avec les RNA a été présentée. Cette section contient les mêmes informations appliquées aux AD. Othman et Eshames (2012) se basent sur les travaux de Zhang et Su (2006) qui proposent une version plus rapide de l'algorithme C4.5. Wang et coll. (2008) utilisent la version originale de l'algorithme C4.5. Dans cette section, le fonctionnement général d'un AD est présenté suivi par les algorithmes C4.5 et ID3.

1.6.1 Le fonctionnement des arbres décisionnels

Un AD est un graphe qui représente les résultats possibles d'une série de choix interconnectés. La structure d'un arbre décisionnel (figure 1.23) comprend trois types d'éléments :

1. Le nœud de décisions, symbolisé par un carré, représente un choix à faire ;
2. Le nœud de hasard, symbolisé par un cercle, représente un résultat incertain ;
3. Le nœud terminal, symbolisé par un triangle, représente un choix final.

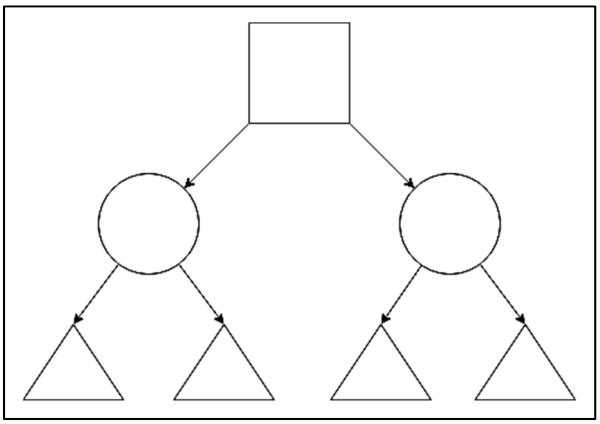

Figure 1.23 Schéma d'un arbre décisionnel simple

Les AD de ce type sont parfois utilisés, dans le domaine de la finance, pour calculer la valeur espérée du profit qui découle d'un investissement. Ils sont aussi utilisés par les banques pour déterminer l'éligibilité d'un client à recevoir un prêt hypothécaire. Ce type d'arbre est aussi appelé arbre décisionnel probabiliste, car chaque branche multiplie la valeur du nœud parent par une probabilité.

En apprentissage supervisé, les arbres décisionnels sont utilisés pour la classification et pour la régression. Le premier nœud de ce type d'arbre est appelé nœud racine (□) et les nœuds secondaires sont appelés nœuds internes (○). Ces deux types de nœuds contiennent des données et les branches appliquent conditions logiques qui permettent de les séparer. Les nœuds à l'autre extrémité de l'arbre sont appelés nœuds feuilles (△) et contiennent soit une classe unique ou mélange sous forme de distribution fréquentielle de deux ou plusieurs classes selon le cas.

Par exemple, les classes $\{A, B, C, D\}$ définis par les attributs $\{x_1, x_2\}$ de la figure 1.24 sont parfaitement séparées (c.-à-d. classifiées) par l'arbre décisionnel montré à la figure 1.25.

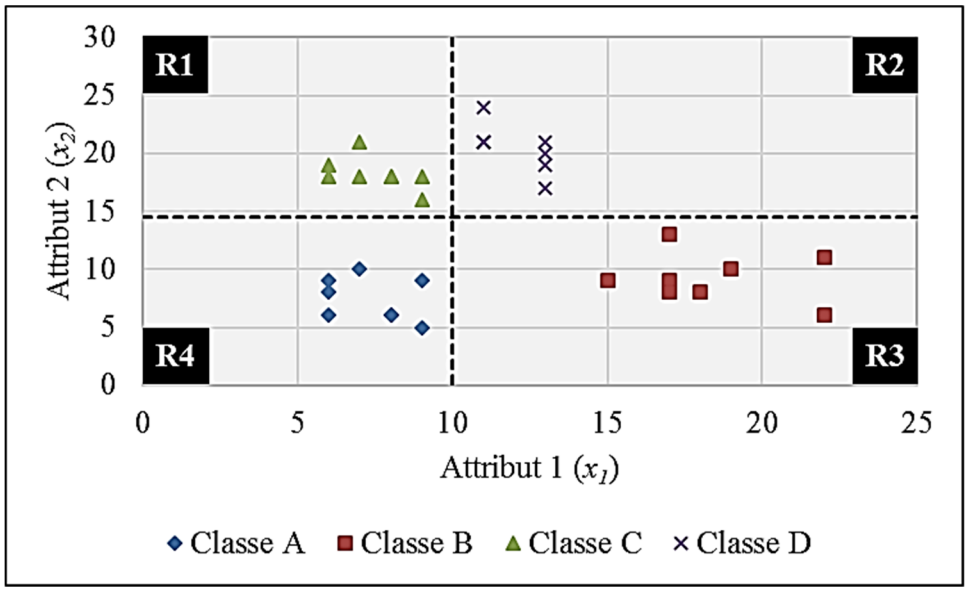

Figure 1.24 Segmentation de l'espace d'attributs

Formellement, chaque observation est définie par le doublet $(\boldsymbol{x_i}, y_i)$ où $\boldsymbol{x} = \langle x_{i,1}, x_{i,2}, \ldots, x_{i,n} \rangle$ est constitué des attributs et y_i correspond à la classe. Dans ce cas de figure, chaque observation est notée $(\boldsymbol{x_i}, y_i) = \langle x_{i,1}, x_{i,2} \rangle$.

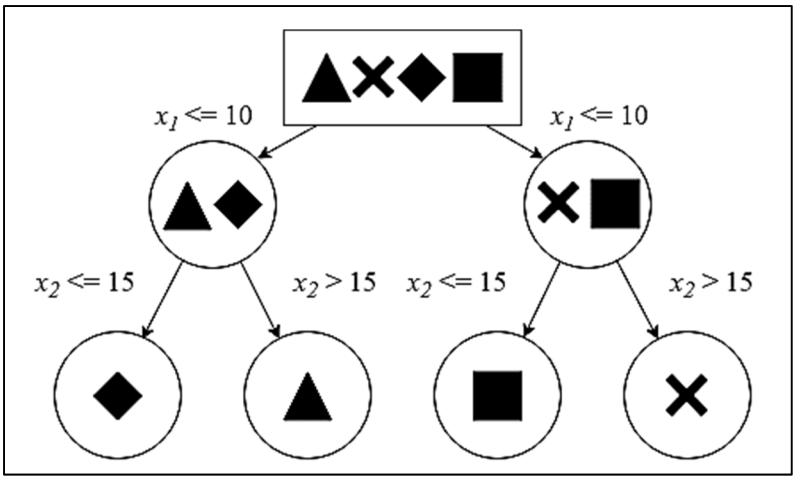

Figure 1.25 Exemple d'arbre décisionnel pour la classification

Les régions créées par les frontières pointillées de la figure 1.24 sont définies mathématiquement par les inégalités des branches de l'arbre à la figure 1.25. Dans cet exemple, les classes sont bien ordonnées et faciles à séparer (c.-à-d. que les nœuds feuilles contiennent une seule classe), mais ceux de la figure 1.26 ne le sont pas.

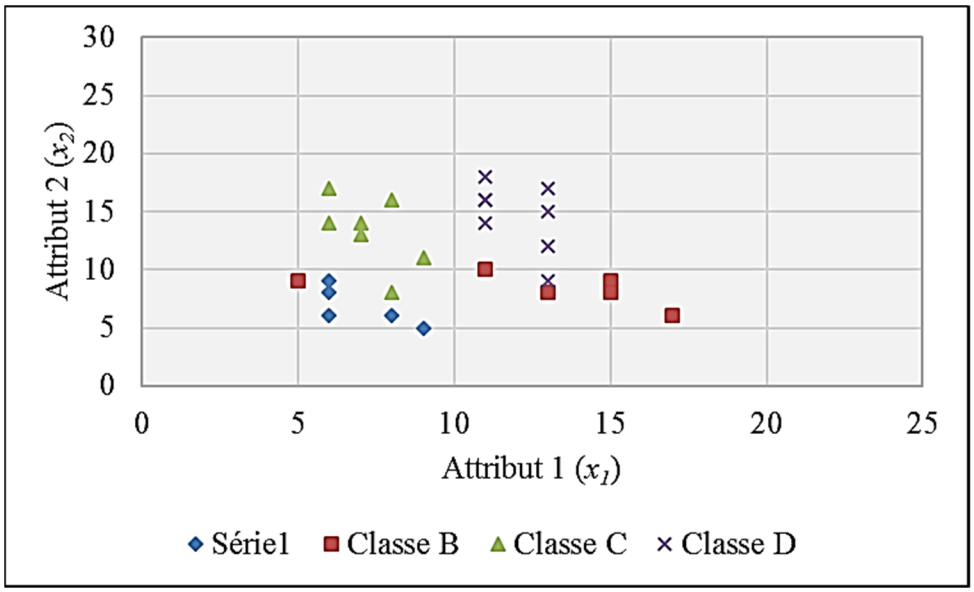

Figure 1.26 Jeu de données synthétiques linéairement inséparables

Les données de la figure 1.26 sont montrées au tableau 1.1 ci-dessous.

Tableau 1.1 Table de données linéairement inséparables

Classe A		**Classe B**		**Classe C**		**Classe D**	
x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
6	9	11	10	7	13	11	16
9	5	17	6	9	11	13	15
6	8	5	9	6	17	11	16
6	6	15	9	6	14	11	18
8	6	15	8	8	8	13	17
-	-	13	8	8	16	13	12
-	-	-	-	7	14	11	14
-	-	-	-	-	-	13	9

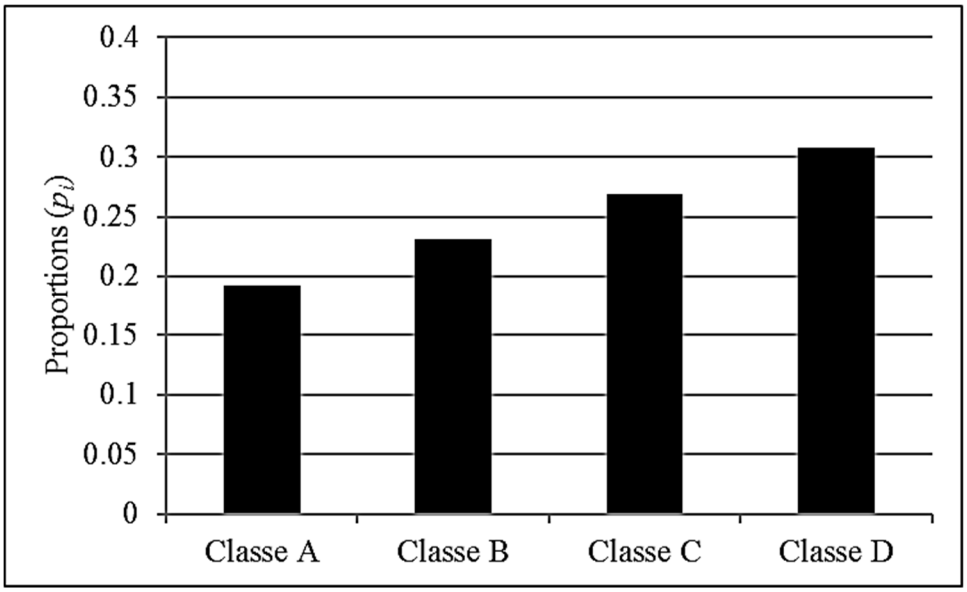

Figure 1.27 Distribution fréquentielle des classes

Cette section présente comment un AD sépare un espace d'attribut de manière générale. Dans la section suivante, l'algorithme ID3 basé sur la minimisation de l'entropie est expliqué.

1.6.2 L'algorithme ID3

L'algorithme ID3, proposé par Quinlan (1983), construit un AD en commençant par le nœud racine. Il choisit le premier attribut discriminant en se basant sur la divergence de Kullback-Leibler (DKL) qui utilise l'entropie de Shannon dont la valeur est donnée par :

$$H(Y) = \sum_{i=1}^{n} -p_i \log_2 p_i \tag{1.22}$$

Concrètement, l'entropie permet de déterminer laquelle des classes ou des attributs séparent le mieux l'ensemble. La variable p_i représente les proportions des différentes valeurs des classes ou des attributs. La DKL est donnée par l'équation suivante :

$$D_{KL}(Y, X = x_i) = H(Y) - H(Y|X = x_i) \tag{1.23}$$

C'est une mesure de la réduction d'entropie dans un ensemble de choix (c.-à-d. des classes) possibles (Y) occasionnée par la connaissance de la valeur d'un attribut ($X = x_i$). Par exemple, le fait de savoir que l'attribut x_1 d'un échantillon vaut 6, aide à faire une meilleure prédiction de sa classe, en moyenne, que de savoir que l'attribut x_2 vaut 10. Les attributs dont la DKL est plus élevée sont préférés. L'entropie du jeu de données, dont la distribution fréquentielle des classes, est présentée à la figure 1.27 est donc :

$$\begin{aligned} H(Y) &= H(Y = A) + H(Y = B) + H(Y = C) + H(Y = D) \\ &= -\left(\frac{5}{26} \cdot \log_2 \frac{5}{26}\right) - \left(\frac{6}{26} \cdot \log_2 \frac{6}{26}\right) - \left(\frac{7}{26} \cdot \log_2 \frac{7}{26}\right) - \left(\frac{8}{26} \cdot \log_2 \frac{8}{26}\right) \approx \boxed{1.9785} \end{aligned}$$

Pour choisir lequel des deux attributs utilisé pour séparer l'ensemble, l'algorithme calcule l'entropie de chacun à partir de leurs distributions fréquentielles données à la figure 1.28 et à la figure 1.29.

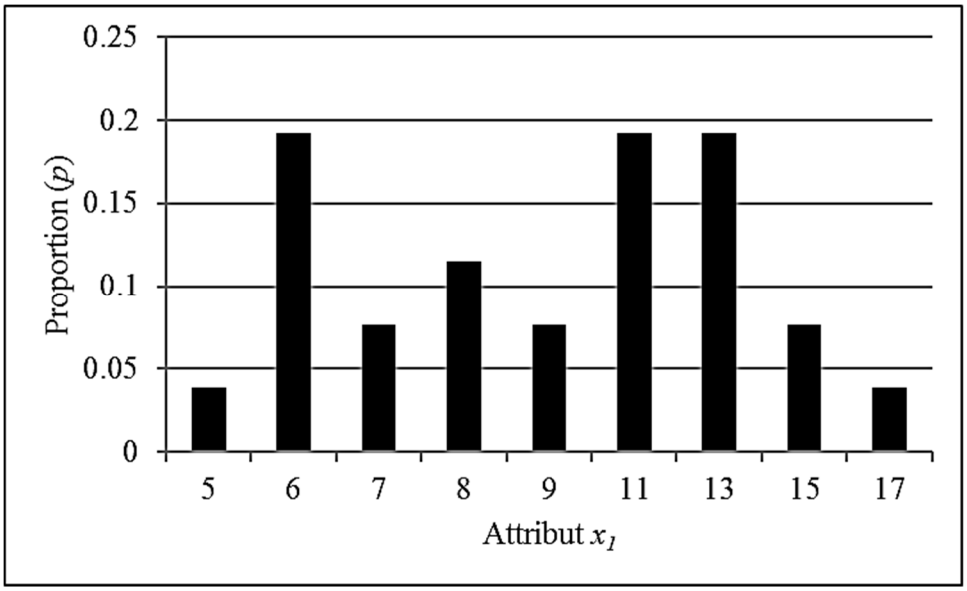

Figure 1.28 Distribution fréquentielle de l'attribut x_1

$$H(x_1) = -\left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right)-\left(\frac{5}{26}\cdot\log_2\frac{5}{26}\right)-\left(\frac{2}{26}\cdot\log_2\frac{2}{26}\right)-\left(\frac{3}{26}\cdot\log_2\frac{3}{26}\right)-\left(\frac{2}{26}\cdot\log_2\frac{2}{26}\right)$$
$$-\left(\frac{5}{26}\cdot\log_2\frac{5}{26}\right)-\left(\frac{5}{26}\cdot\log_2\frac{5}{26}\right)-\left(\frac{2}{26}\cdot\log_2\frac{2}{26}\right)-\left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right)\approx \boxed{2.9472}$$

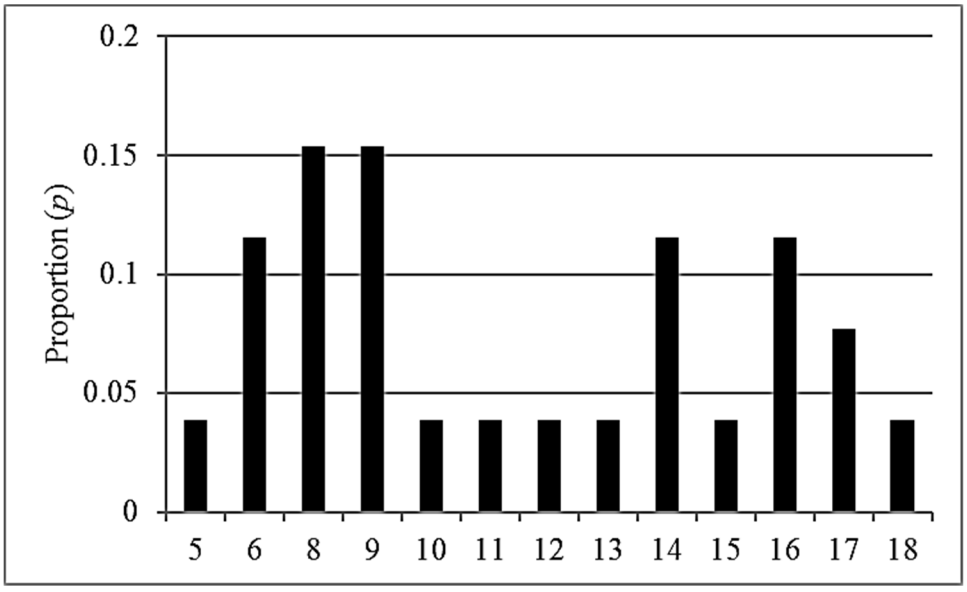

Figure 1.29 Distribution fréquentielle de l'attribut x_2

$$H(x_2) = -\left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right) - \left(\frac{3}{26}\cdot\log_2\frac{3}{26}\right) - \left(\frac{4}{26}\cdot\log_2\frac{4}{26}\right) - \left(\frac{4}{26}\cdot\log_2\frac{4}{26}\right)$$
$$- \left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right) - \left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right) - \left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right) - \left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right)$$
$$- \left(\frac{3}{26}\cdot\log_2\frac{3}{26}\right) - \left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right) - \left(\frac{3}{26}\cdot\log_2\frac{3}{26}\right) - \left(\frac{2}{26}\cdot\log_2\frac{2}{26}\right)$$
$$- \left(\frac{1}{26}\cdot\log_2\frac{1}{26}\right) \approx \boxed{3.4595}$$

Puisque l'entropie du premier attribut est inférieure à celui du deuxième, c'est celui qui est utilisé pour segmenter l'espace d'attributs au nœud racine. Pour déterminer à quelle valeur de x_1 la frontière doit être tracée, l'algorithme calcule la DKL pour différentes valeurs et choisit celle qui la maximise. L'algorithme calcule ensuite la DKL pour les différentes valeurs de l'attribut et trouve que $x_1 = 10$ est la valeur optimale. La frontière est montrée à la figure 1.30.

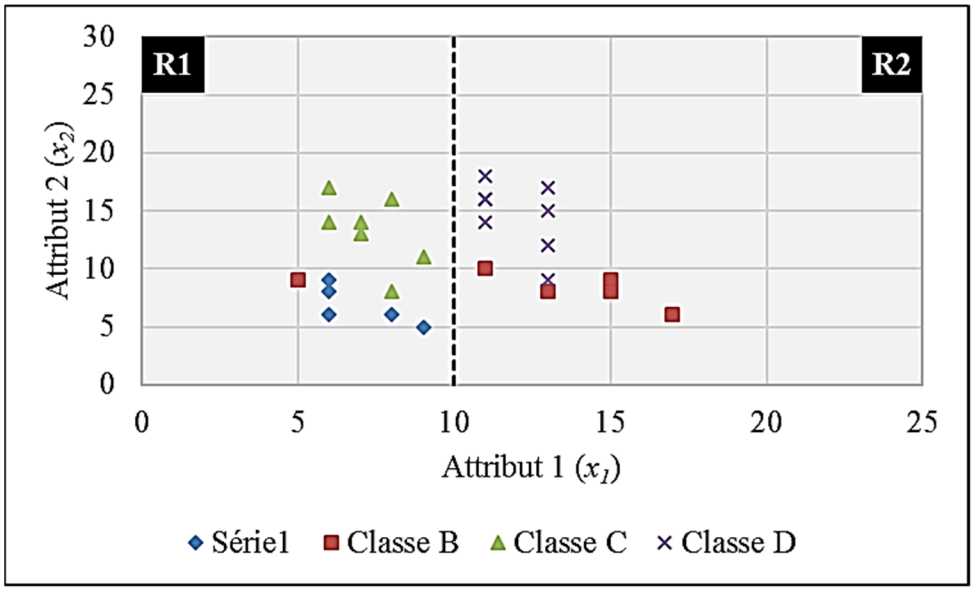

Figure 1.30 Espace d'attribut séparé à $x_1 = 10$

Cette séparation donne $\mathcal{R}_1 = \{\diamond, \diamond, \diamond, \diamond, \diamond, \diamond, \diamond, \triangle, \triangle, \triangle, \triangle, \triangle, \triangle, \triangle, \square\}$ et $\mathcal{R}_2 = \{\square, \square, \square, \square, \square, X, X, X, X, X, X, X, X\}$ qui sont tous les deux des sous-espaces plus homogènes (c.-à-d. de plus faible entropie) que l'espace d'attributs initial. L'arbre résultant est montré à la figure 1.31.

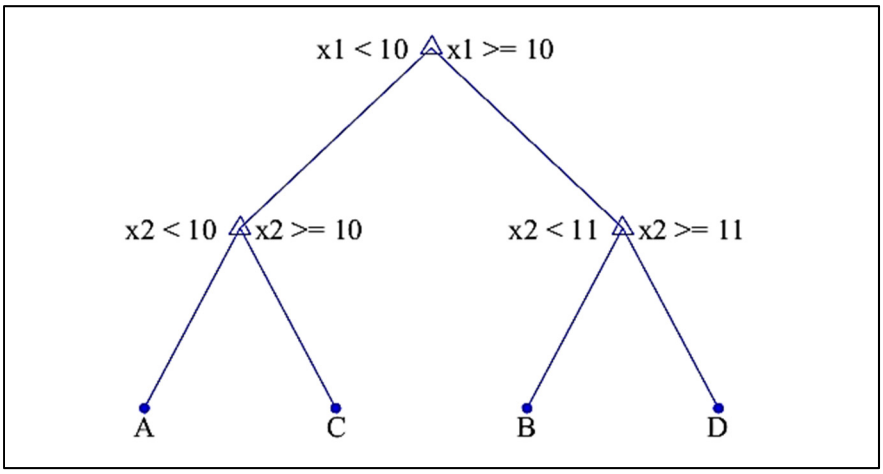

Figure 1.31 Arbre décisionnel entraîné avec l'algorithme ID3

Les frontières des sous-ensembles créées par l'AD sont montrées à la figure 1.32.

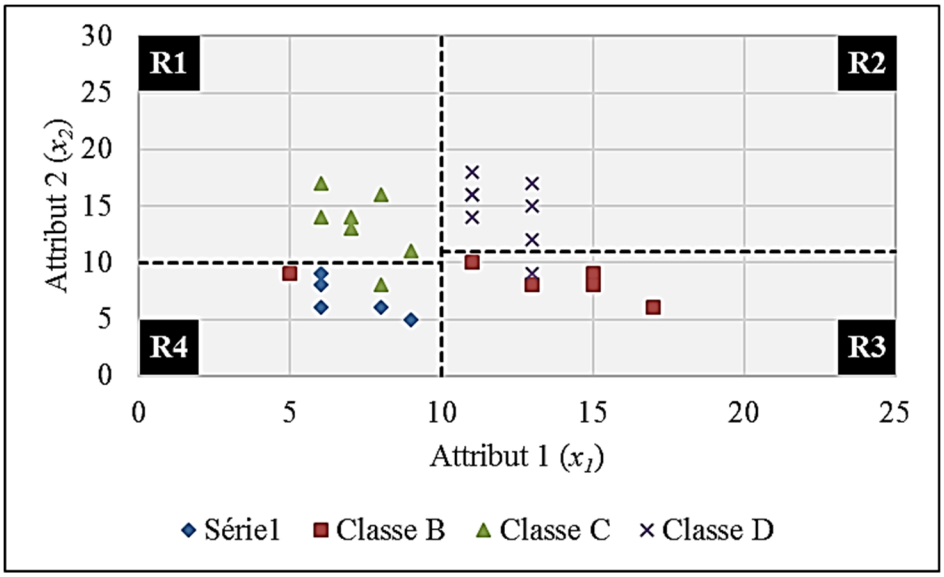

Figure 1.32 Frontières déterminées par l'algorithme ID3

En résumé, l'algorithme ID3 se décrit par la séquence suivante :

1. Calculer la DKL de chaque attribut x_i de l'ensemble X;
2. Séparer l'ensemble X en utilisant l'attribut x_i ayant la plus grande DKL;
3. Créer un nœud pour cet attribut ;
4. Répéter sur les sous-ensembles en utilisant les attributs restant pour chaque région $\mathcal{R}_j$.

La segmentation donne les sous-ensembles $\mathcal{R}_1 = \{\triangle, \triangle, \triangle, \triangle, \triangle, \triangle\}$, $\mathcal{R}_2 = \{\mathrm{X}, \mathrm{X}, \mathrm{X}, \mathrm{X}, \mathrm{X}, \mathrm{X}\}$, $\mathcal{R}_3 = \{\square, \square, \square, \square, \square, \square, \mathrm{X}\}$ et $\mathcal{R}_4 = \{\diamondsuit, \diamondsuit, \diamondsuit, \diamondsuit, \diamondsuit, \diamondsuit, \triangle\}$. L'entropie moyenne pondérée de ces sous-ensembles est de 0,4686, ce qui correspond à une réduction de 76,3% par rapport à l'entropie de l'ensemble initial.

La section suivante présente un moyen d'améliorer l'algorithme ID3 en réduisant sa propension au surapprentissage en remplaçant l'entropie par le gain normalisé et par l'introduction de l'élagage anticipatif.

1.6.2 Une version améliorée de l'algorithme ID3 : C4.5

Un désavantage de l'algorithme ID3 est qu'il est vulnérable au surapprentissage. Il a tendance à favoriser les attributs qui ont beaucoup d'observations, car ils sont plus faciles à séparer en petits sous-ensembles (c.-à-d. plusieurs branches) de faible entropie. Cela mène quasi inévitablement à un modèle sous-optimal, car l'attribut le plus fourni n'est pas nécessairement celui qui sépare le mieux l'ensemble. L'algorithme C4.5 règle ce problème en maximisant le gain normalisé (en anglais : *gain ratio*) :

$$\widehat{D}_{KL} = \frac{H(Y) - H(Y|X = x_i)}{H(x_i)} \tag{1.24}$$

Le dénominateur de l'équation 1.24 a pour effet de pénaliser x_i comme discriminant potentiel en réduisant le gain normalisé lorsqu'il y a beaucoup d'exemples.

L'algorithme C4.5 introduit aussi la notion d'élagage qui consiste à couper des branches de l'arbre afin d'augmenter son aptitude à généraliser. La généralité d'un algorithme d'apprentissage machine fait référence à sa capacité à bien classer des exemples qui ne font pas partie du jeu de données avec lequel il a été entraîné. Les sous-ensembles dont les branches ont été coupées sont combinés et remontés au nœud parent qui devient un nœud feuille.

La procédure d'élagage consiste à soumettre les données du jeu de validation au modèle et à calculer le taux d'erreur p_i de l'AD sans élagage. L'intervalle de confiance de la proportion d'erreurs de la population (P_i) d'un nœud interne est ensuite calculé en prenant la moyenne pondérée de celles de ses m nœuds feuilles pour un niveau de confiance statistique α :

$$P_1 = \sum_{i=1}^{m} \left\{ \frac{n_i}{N} \left(p_i \pm z_\alpha \sqrt{\frac{p_i(1-p_i)}{n_i}} \right) \right\} \tag{1.25}$$

Dans l'équation 1.25, n_i représente le nombre d'exemples classifiés par la i-ème feuille du nœud parent, N représente la somme de tous les exemples classifiés par les nœuds feuilles et p_i représente le pourcentage d'erreur du i-ème nœud feuille. L'intervalle de confiance du nouveau nœud résultant de l'élagage se calcule avec l'équation suivante :

$$P_2 = p \pm z_\alpha \sqrt{\frac{p(1-p)}{n}} \tag{1.26}$$

La cote Z (z_α) est calculée avec la fonction `invNorm(alpha)`, disponible dans la librairie Scipy ou `norminv(alpha)` sur MATLAB. Cette fonction est exprimée en forme close avec la fonction d'erreur inverse elle aussi disponible dans Scipy :

$$z(\alpha) = \sqrt{2}\,\mathrm{erf}^{-1}(2\alpha - 1) \tag{1.27}$$

Finalement, l'opération d'élagage est maintenue si $P_2 < P_1$.

1.6.3 Deux algorithmes supplémentaires : C5.0, CART

Il existe plusieurs autres algorithmes pour créer des AD, tels que le C5.0, commercialisé par Ross Quinlan et le très populaire algorithme CART (en anglais : *Classification And Regression Tree*). La mécanique de C5.0 est très similaire à celle de C4.5, mais il est plus efficace en consommation de mémoire et en temps d'exécution. En théorie, C5.0 génère les mêmes classificateurs que C4.5. CART ressemble lui aussi à C4.5 et C5.0 à la différence qu'il minimise l'indice d'impureté de Gini au lieu de l'entropie :

$$IndiceGini(\mathrm{Y}) = 1 - \sum_{i=1}^{n} p_i^2 \tag{1.28}$$

Dans cette équation, p_i correspond à la probabilité (c.-à-d. l'estimé de la distribution fréquentielle) que l'observation appartienne à la classe y_i. CART donne des résultats similaires à C4.5, mais il est plus rapide, car il ne demande pas de calculer de logarithmes.

La prochaine section introduit les forêts aléatoires qui sont une technique d'apprentissage ensembliste – un ensemble d'AD pour être plus précis – basée sur la minimisation de la variance par le rééchantillonnage.

1.6.4 Les forêts aléatoires

Un des désavantages des AD est qu'ils ont tendance à faire du surapprentissage. Un bon exemple est l'algorithme ID3 qui sépare les données de manière récursive jusqu'à ce qu'elles soient parfaitement séparées. Un moyen simple de palier à ce problème est de faire de l'élagage comme les algorithmes C4.5, C5.0 et CART. Le problème avec cette méthode est qu'elle coupe des branches de l'AD en suivant un ou plusieurs critères qui sont arbitraires. Par exemple, si la création d'un nœud de hasard ne réduit pas l'entropie moyenne des deux sous-groupes résultants par rapport au groupe parent plus qu'une certaine valeur de tolérance arbitraire, il est possible d'introduire une règle qui crée un nœud feuille à la place. Il est aussi possible d'imposer un nombre minimal d'exemples dans un nœud pour qu'il puisse devenir un nœud de hasard, mais ce nombre est encore une fois un paramètre arbitraire. Une alternative proposée par Ho (1995) est d'utiliser un ensemble d'AD qu'on appelle une forêt aléatoire (FA) qui utilise un sous-ensemble des attributs pour l'entraînement. Plus spécifiquement, chacun des AD de la FA choisit quelques attributs qu'il sépare de manière aléatoire et exclut les autres. De plus, pour chacun des attributs, il effectue un rééchantillonnage aléatoire avec remplacement. Autrement dit, pour un attribut donné, une observation peut se retrouver plus d'une fois dans le jeu de données d'un AD particulier et pas du tout dans un autre. L'intuition derrière cette approche est basée sur le phénomène de régression vers la moyenne. En supposant que l'échantillon initial est représentatif de la population, en ré échantillonnant, cela réduit les chances que les éléments « extrêmes » soient surreprésentés, car étant moins nombreux, ils sont moins susceptibles de se retrouver dans les nouveaux échantillons.

1.6.5 Conclusion

Les AD séparent l'espace d'attributs en créant des frontières qui maximisent l'homogénéité des sous-espaces créés par celles-ci. Ils ont l'avantage d'être faciles à implémenter et simples à interpréter. Ils ont toutefois le désavantage d'être susceptibles au surapprentissage. Les FA

sont une variante des AD basés sur le rééchantillonnage aléatoire avec remplacement et sont généralement plus précis et plus stables que les AD singuliers. Les AD et les FA attribuent une étiquette probabiliste à un échantillon en fonction du sous-espace auquel il appartient qui dépend de ses attributs.

La prochaine section introduit les fondements théoriques derrière les MVS et montre comment transformer un problème complexe en un problème d'optimisation qui se prête plus facilement à la résolution.

1.7 La classification avec les machines à vecteurs de support

Les MVS, aussi appelés séparateurs à vastes marges (SVM, en anglais : *large margin classifiers*, *LMC*), sont une famille d'algorithmes puissants utilisés en apprentissage machine pour la classification et la régression. Othman et Eshames (2012) concluent toutefois qu'ils sont moins performants que les RNA et les AD pour la RMCC. Ils ne fournissent malheureusement pas les détails de l'algorithme comme le type de noyau (en anglais : *kernel*) ou les valeurs des hyperparamètres pour que l'on puisse mettre leurs résultats en contexte ou les reproduire. Cheng et coll. (2011) utilisent une fonction à base radiale (c.-à-d. la courbe gaussienne) et obtiennent des précisions mesurées allant de 69,1% à 94,8%. Cette étude porte uniquement sur les motifs de hausses/baisses subites de la variance d'un processus.

1.7.1 Les classificateurs linéaires généralisés

Les MVS **sont** des classificateurs linéaires généralisés. Un classificateur linéaire est un algorithme qui classe les échantillons d'un jeu de données en séparant leur espace avec une combinaison linéaire de ses paramètres. L'équation d'un classificateur linéaire bidimensionnel peut être construite simplement à partir de l'équation d'une droite :

$$y = ax + b \tag{1.29}$$

Les variables x et y sont ensuite remplacées par les variables générales x_1 et x_2 respectivement et isolées :

$$ax_1 - x_2 + b = 0 \tag{1.30}$$

Cette expression est simplifiée en utilisant le produit scalaire des vecteurs $\boldsymbol{w} = \langle a, -1 \rangle$ et $\boldsymbol{x} = \langle x_1, x_2 \rangle$:

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0 \tag{1.31}$$

L'équation 1.31 est aussi valide en n dimensions. C'est l'équation d'un plan lorsque $n = 3$ et d'un hyperplan lorsque $n \geq 4$. Cette équation est utilisée comme fonction d'hypothèse $y_i(\boldsymbol{x_i}, b)$ et elle possède les propriétés suivantes :

1. $y_i(\boldsymbol{x_i}, b) = 0$ lorsque $\boldsymbol{x_i}$ est sur le plan ;
2. $y_i(\boldsymbol{x_i}, b) > 0$ d'un côté du plan ;
3. $y_i(\boldsymbol{x_i}, b) < 0$ de l'autre côté du plan.

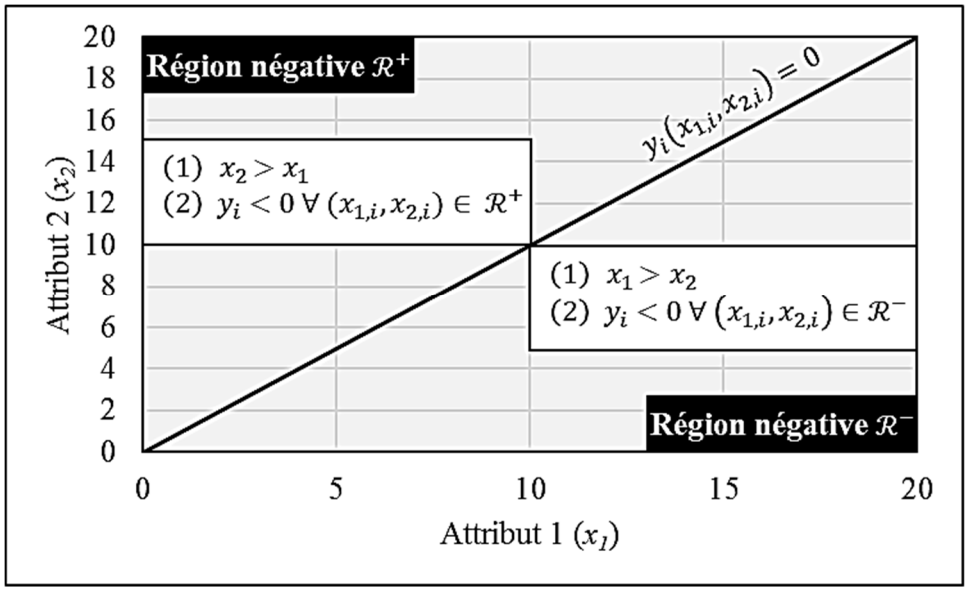

Figure 1.33 Frontière de décision d'un classificateur linéaire

Le jeu de données linéairement séparable $\mathcal{D}$ composé de m doublets de dimensions $n = 2$ est défini formellement par l'expression suivante :

$$\mathcal{D} = \left\{(\boldsymbol{x_i}, y_i) \mid \boldsymbol{x_i} \in \mathbb{R}^2, y_i \in \{-1,1\}\right\}_{i=1}^{m}$$

Les paramètres $\boldsymbol{w}$ d'un hyperplan de dimension $n-1$ peuvent être calculés en utilisant la règle de Hebb qui se décrit comme suit :

1. Initialiser le vecteur $\boldsymbol{w}$ avec de petites valeurs aléatoires;
2. Pour chaque exemple $\boldsymbol{x_i}$ dans le jeu d'entraînement :
    a. Calculer la sortie $\hat{y}_i = \boldsymbol{w} \cdot \boldsymbol{x_i}$;
    b. Mettre à jour le vecteur poids avec $\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \alpha(y_i - \hat{y}_i)\boldsymbol{x_i}$.
3. Répéter l'étape 2 jusqu'à ce que l'espace $\mathcal{D}$ soit parfaitement séparé.

Dans la règle de Hebb, y_i correspond à l'étiquette véritable du i-ème exemple et α est le taux d'apprentissage. Il contrôle la taille de l'ajustement des poids pour améliorer la convergence. Les droites de couleur noire de la figure 1.34 ont été calculées avec l'algorithme du perceptron et la droite en bleu est celle donnée par une MVS.

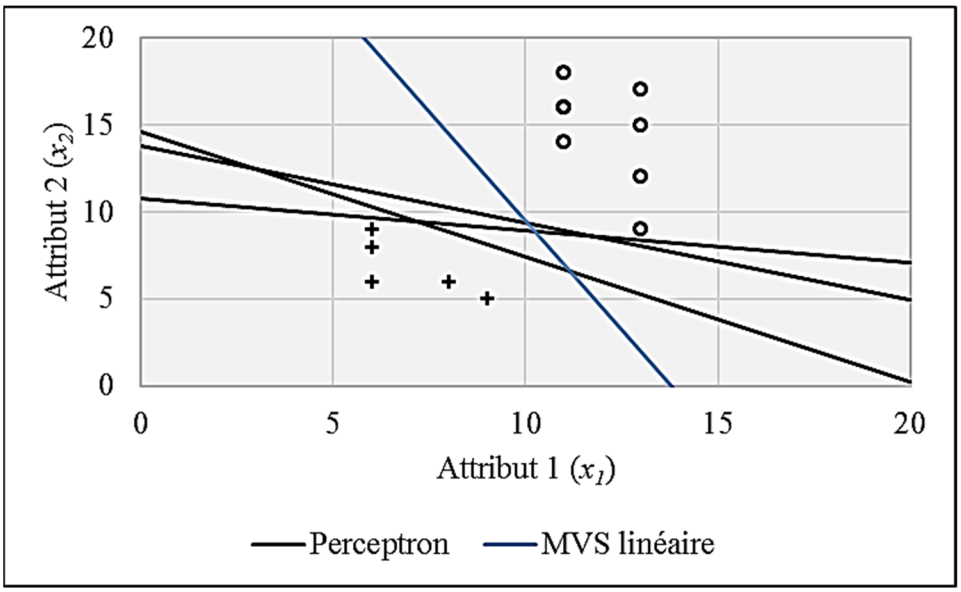

Figure 1.34 Classification avec l'algorithme du perceptron et une MVS linéaire

L'algorithme du perceptron donne un résultat différent pour différentes valeurs initiales de $\boldsymbol{w}$. Cette propriété est indésirable pour deux raisons principales :

1. Si les valeurs *réelles* des exemples sont légèrement différentes de celles mesurées, le modèle fera de mauvaises prédictions. Cela correspond à un surapprentissage ;
2. En pratique, la surface de décision « réelle » est fort probablement non linéaire et donc même si les mesures sont exactes, le modèle résultant est sous-optimal.

Autrement dit, il y a moins de confiance dans les prédictions qui sont plus près de la surface de décision. L'exemple de la figure 1.35 montre l'impact d'une translation de 2 unités vers le haut résultant d'une erreur systématique (c.-à-d. les symboles rouges) par exemple.

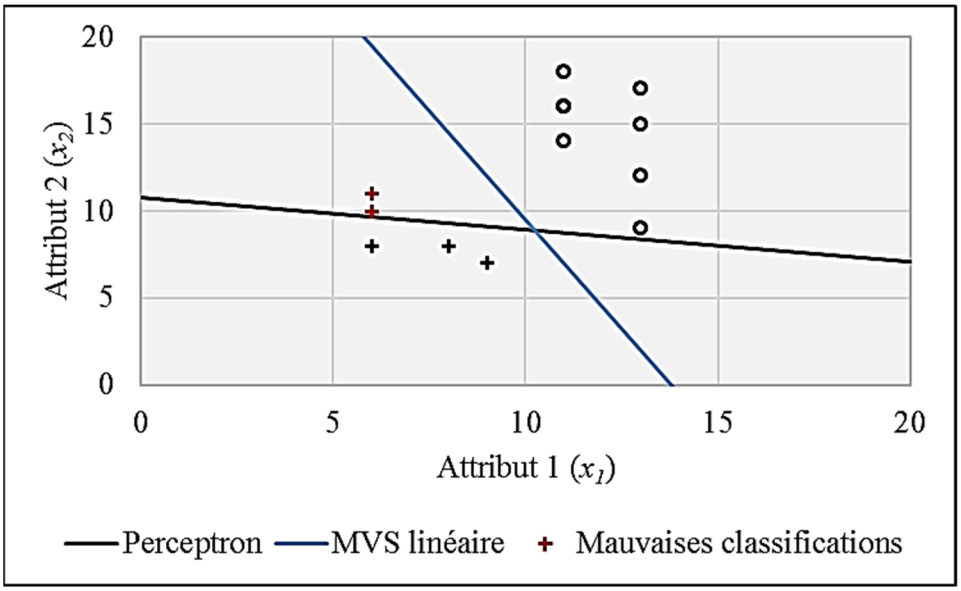

Figure 1.35 Comparaison entre le perceptron et une MVS linéaire

Pour surmonter ces problèmes, Vapnik et coll. (1992) proposent de maximiser l'espace entre la surface de décision et les exemples (figure 1.36). C'est l'idée de base des MVS.

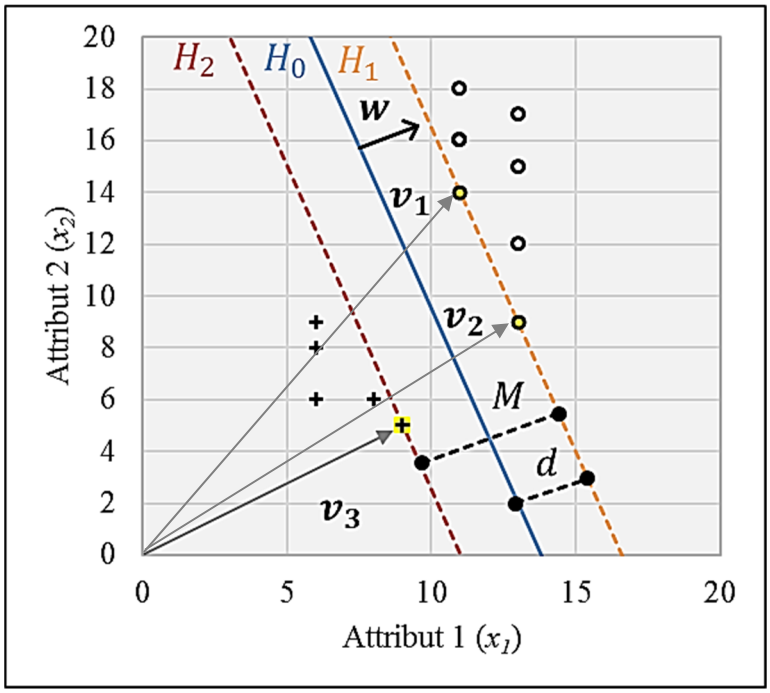

Figure 1.36 Représentation schématique des MVS

À la figure 1.36, le vecteur $\boldsymbol{w}$ est le vecteur directeur de la surface H_0 et les points jaunes en surbrillance représentent les vecteurs de support ($\boldsymbol{v}_i$). Les vecteurs de support correspondent aux points qui touchent les frontières. Il est dit qu'ils « supportent » les frontières, car s'ils sont déplacés, les frontières sont déplacées elles aussi. La distance entre les frontières H_1 et H_2 s'appelle la marge et est notée M. Elle mesure deux fois la distance entre la surface médiane H_0 et les frontières, soit $M = 2d$. Pour séparer les données en deux groupes, il est souhaité de trouver $\boldsymbol{w}$ tel que $y_i \leq -1$ d'un côté et $y_i \geq +1$ de l'autre. La mesure de d correspond à la distance géométrique entre H_0 et H_1 (ou H_2). Cela revient à calculer la distance perpendiculaire entre les plans $\boldsymbol{w} \cdot \boldsymbol{x} + b_0 = 0$ et $\boldsymbol{w} \cdot \boldsymbol{x} + b_1 = 1$ donnée par :

$$d = \frac{|b_1 - b_0|}{\|\boldsymbol{w}\|} \tag{1.32}$$

En soustrayant les équations des deux plans, le résultat est $b_1 - b_0 = 1$. L'équation 1.32 devient :

$$M = \frac{2}{\|\boldsymbol{w}\|} \tag{1.33}$$

Pour maximiser l'espace entre la surface de décision H_0 et les données, on cherche à maximiser l'équation 1.33. Cela revient à résoudre le problème d'optimisation suivant :

maximiser	$\frac{2}{\|\boldsymbol{w}\|}$
s. c.	$y_i(\boldsymbol{w} \cdot \boldsymbol{x} + b) \geq 1 \; \forall i$

Il est toutefois préférable de s'attaquer au problème équivalent suivant :

$$\begin{array}{ll} \textbf{minimiser} & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \textbf{s.c.} & y_i(\boldsymbol{w}\cdot\boldsymbol{x}+b)\geq 1\ \forall i \end{array}$$

En effet, maximiser $2/\|\boldsymbol{w}\|$ est équivalent à minimiser $\frac{1}{2}\|\boldsymbol{w}\|^2$. Cela découle du fait que la procédure d'optimisation utilise la dérivée et que la dérivée est simplement égale à $\|\boldsymbol{w}\|$. Il s'agit d'un problème d'optimisation quadratique qui peut être résolu avec différents algorithmes comme celui du simplexe proposé par Dantzig (1948). La prochaine section montre comment relaxer les contraintes de ce problème d'optimisation.

1.7.2 Les machines à vecteurs de support à marge souple

Dans la partie précédente, ce qu'on appelle une MVS à marge dure a été formulé. Ce nom vient du fait qu'il n'est pas permis que des points se trouvent entre les marges H_1 et H_2. Cette propriété devient problématique dans le cas où le jeu de données n'est pas linéairement séparable. Cette faiblesse est contournée par Vapnik et Cortes (1995) qui proposent d'utiliser des variables lâches (en anglais : *slack variables*) notées ζ_i pour « assouplir » la marge. Ils introduisent aussi la constante C qui sert à contrôler la permissivité du modèle aux erreurs de classification en pénalisant les grandes valeurs de ζ_i et leur somme.

$$\begin{array}{ll} \textbf{minimiser} & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\zeta_i \\ \textbf{s.c.} & y_i(\boldsymbol{w}\cdot\boldsymbol{x}+b)\geq 1-\zeta_i, i=1\ldots m \\ & \zeta_i\geq 1, i=1\ldots m \end{array}$$

1.7.3 Les machines à vecteurs de support à marge non linéaire

En pratique, il est rare que les relations sous-jacentes soient linéaires. L'algorithme publié par Vapnik et coll. (1992) utilise la méthode des multiplicateurs de Lagrange (λ) pour résoudre le problème d'optimisation qui est survenu dans les sections précédentes. Le problème d'optimisation résultant est le suivant :

$$\begin{array}{ll} \textbf{maximiser} & \sum_{i=1}^{m} \lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j (\boldsymbol{x_i} \cdot \boldsymbol{x_j}) \\ \textbf{s. c.} & \alpha_i - \zeta_i, i = 1 \ldots m \\ & \sum_{i=1}^{m} \lambda_i y_i = 0 \end{array}$$

L'indice *j* fait référence au j-ème vecteur de support. Cette formulation a permis de découvrir l'astuce du noyau (en anglais : *kernel trick*) qui consiste à établir une relation fonctionnelle entre un noyau non linéaire *K* et le produit scalaire de deux vecteurs.

Un exemple populaire est le noyau gaussien :

$$K(\boldsymbol{x_1}, \boldsymbol{x_2}) = \exp\left(-\frac{\|\boldsymbol{x_1} - \boldsymbol{x_2}\|^2}{2\sigma^2}\right) \tag{1.34}$$

La variable σ est un hyperparamètre fixé par l'utilisateur. Le problème d'optimisation résultant est le suivant :

$$\begin{array}{ll} \textbf{maximiser} & \sum_{i=1}^{m} \lambda_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j \exp\left(-\frac{\|\boldsymbol{x_1} - \boldsymbol{x_2}\|^2}{2\sigma^2}\right) \\ \textbf{s. c.} & \alpha_i - \zeta_i, i = 1 \ldots m \\ & \sum_{i=1}^{m} \lambda_i y_i = 0 \end{array}$$

1.7.4 La séparation en classes multiples

Les sections précédentes ont présenté comment séparer deux classes avec une MVS, mais pas comment en séparer plusieurs. Il existe deux méthodes : la classification un contre un (en anglais : *one-versus-one*, *OVO*) et la classification un contre tous (en anglais : *one-versus-all*, *OVA*). Pour les définitions qui suivent, le cas où il y a n classes à séparer est considéré.

En classification un contre un, l'algorithme construit C_2^n MVS spécialisées où chacune est entraînée sur une des paires d'attributs possibles. Chaque modèle retourne une étiquette et la classe prédite par la MVS correspond à l'étiquette qui est prédite le plus souvent. Cette méthode est appelée la décision par vote majoritaire.

En classification un contre tous, l'algorithme construit les mêmes MVS spécialisées qu'en classification un contre un, mais il utilise ensuite la régression logistique pour attribuer une probabilité à chacune des étiquettes. La classe prédite correspond à l'étiquette avec la plus grande probabilité.

1.7.5 Conclusion

Il a été présenté que les MVS ont été développées à partir des classificateurs linéaires. Le premier algorithme utilisé pour calculer les paramètres d'un classificateur linéaire est celui du perceptron. Ce dernier a pour désavantage d'être un algorithme glouton dont la réponse dépend de la graine aléatoire (en anglais : *random seed*) utilisée pour initialiser $\boldsymbol{w}$. Il a aussi été démontré comment généraliser les hyperplans pour obtenir la première forme des MVS linéaires à marge dure. Finalement, il a été présenté comment passer de la marge dure à la marge souple avec l'introduction de variables lâches et comment l'astuce du noyau permet de modéliser les relations non linéaires.

1.8 Conclusion

La littérature révèle quatre algorithmes principaux utilisés pour classifier les motifs sur les cartes de contrôle : les réseaux de neurones artificiels, les arbres de décision, les forêts aléatoires et les machines à vecteurs de support. En théorie, chacun d'eux est capable d'effectuer la tâche avec autant de précision que souhaité. La littérature à ce sujet utilisant les réseaux de neurones est exhaustive et rigoureuse dans son ensemble. Les articles présentés dans ce chapitre aident à mieux comprendre les détails nécessaires pour reproduire les résultats présentés. La littérature concernant les autres méthodes, comme les arbres de décision et les machines à vecteurs de support, est plus limitée. Bien qu'il existe un biais évident en faveur des réseaux de neurones, les articles cités qui comparent les différentes méthodes ont des lacunes qui les rendent difficiles à utiliser. L'objectif de cette recherche est de fournir une base commune pour la comparaison de différents algorithmes de classification en les plaçant sur un pied d'égalité.

Les notions mathématiques présentées dans cette section servent à établir le contexte d'interprétation des résultats. Dans le prochain chapitre, la méthodologie est présentée en commençant par le cadre de Basili pour la planification du projet suivi d'une section sur les facteurs de formes, une section sur l'entraînement des algorithmes et finalement une section sur les intervalles de confiance.

CHAPITRE 2

MÉTHODOLOGIE

L'objectif de ce chapitre est de faire un survol des notions spécifiques utilisées dans cette recherche incluant la planification : la classification de séries temporelles basée sur leurs attributs structurels l'entraînement des algorithmes et les intervalles de confiance.

2.1 Planification du projet

Cette section contient une vue d'ensemble de l'approche utilisée pour répondre à la question de recherche. Le cadre de Basili (Tableau 2.1) proposé par Basili et coll. (1986) et repris par Abran et coll. (1999) a été utilisé pour organiser les travaux. Il sert à planifier les travaux de recherche exploratoire en génie logiciel en les divisant dans les phases suivantes :

Phase I – Définition :

La première phase, de manière générale, répond aux questions « QUOI ? POURQUOI ? QUI ? » Elle est définie par les activités suivantes :

1. Identifier un besoin dans l'industrie et poser la question de recherche ;
2. Diviser la question de recherche en sous-objectifs ;
3. Cerner le public cible de la recherche.

La motivation principale de ce travail consiste à aider à réduire les défauts de fabrication dans l'industrie manufacturière à l'aide de l'IA. L'objectif, c'est de trouver le meilleur algorithme d'apprentissage machine pour la détection de motifs non naturels sur les cartes de contrôle. Le but est de permettre aux étudiants, aux chercheurs dans le domaine de l'IA et aux professionnels du domaine du contrôle de la qualité de faire un meilleur choix d'algorithme dans la conception d'un système de CSP automatisé.

Phase II – Planification :

La deuxième phase sert à faire l'étude de l'existant. Elle consiste à faire une revue de littérature qui couvre les systèmes experts, les arbres de décision, les machines à vecteurs de support et les réseaux de neurones artificiels. Le résultat de cette phase est un document qui contient une description technique des algorithmes d'IA et un compte-rendu des travaux effectués par d'autres chercheurs. Finalement, c'est durant cette phase que les modèles d'IA qui seront étudiés plus en détail sont choisis.

Phase III – Opération :

La troisième phase est définie par la construction des modèles par la programmation des algorithmes d'apprentissage machine et de la collecte des données qui découlent de leur évaluation.

Phase IV – Interprétation :

La quatrième et dernière phase sert à faire un retour sur la question de recherche, résumer les résultats avec leur portée et identifier les travaux futurs. Plus spécifiquement, dans le contexte de la production manufacturière, le meilleur algorithme est celui qui est le plus précis (P) et qui fait le moins d'erreurs de type II (c.-à-d. l'algorithme dit que le motif est normal quand il ne l'est pas). L'importance relative de l'une et de l'autre de ces mesures dépend du contexte. Par exemple, pour un test médical d'une maladie, le coût d'une erreur de type II, c.-à-d. un diagnostic négatif quand le patient est réellement atteint de la maladie, est de loin supérieur au coût d'une erreur de type I. Dans le contexte manufacturier, il est préférable de faire des erreurs de type II, mesuré par la puissance statistique (β), mais pas aux dépens d'une baisse incontrôlée de la précision. Le meilleur compromis dépend du contexte et c'est pourquoi les deux valeurs sont rapportées.

La portée des résultats de cette recherche concerne leur généralité vis-à-vis la population étudiée (c.-à-d. les données utilisées). Les résultats de cette recherche se généralisent naturellement aux populations étudiées par d'autres chercheurs et à des données qui viennent de vrais procédés manufacturiers pour deux raisons principales :

1. Les données synthétiques sont générées par la même méthode ;
2. Il a été démontré expérimentalement que les données synthétiques générées de cette manière sont similaires aux données qui viennent de vrais procédés manufacturiers (Jang et coll., 2003).

Tableau 2.1 Cadre de Basili

Phase I - Définition			
Motivation	**Objectifs**	**But**	**Utilisateurs**
• Réduire les défauts de fabrication dans l'industrie manufacturière à l'aide de l'IA.	• Concevoir un modèle substitut du processus de fabrication pour générer des cartes de contrôles synthétiques ; • Déterminer quel algorithme est le plus apte à prédire les motifs sur les cartes de contrôle.	• Comparer la performance prédictive des méthodes suivantes : RNA, les machines à vecteurs de support (MVS) et les arbres décisionnels (AD) en RMCC.	• Étudiants; • Chercheurs dans le domaine de l'IA; • Professionnels dans le domaine du contrôle de la qualité.

Tableau 2.1 Cadre de Basili (suite)

Phase II - Planification		
Étapes du projet	**Entrants**	**Livrables**
• Revue de littérature; • Comparaison des algorithmes d'IA.	• Revue de littérature sur les algorithmes d'IA suivants en RMCC : SE, RNA, AD et MVS; • Jeu de données synthétiques.	• Description technique de : RMCC, SE, RNA, AD et MVS; • Synthèse des résultats obtenus par d'autres chercheurs incluant les points faibles; • Comparaison de la performance de chacun des algorithmes ; • Chapitre 1 du mémoire.

Phase III - Opération		
Préparation	**Exécution**	**Analyse**
• Revue des protocoles expérimentaux d'autres chercheurs.	• Conception des modèles de RNA, AD et MVS sur Python; • Analyse préliminaire exploratoire et prétraitement du jeu de données ; • Expérimentation des algorithmes avec le jeu de données issues du modèle substitut.	• Récupération et mise en commun des résultats; • Analyse préliminaire des résultats ; • Chapitres 2 et 3 du mémoire.

Tableau 0.1 Cadre de Basili (suite)

Phase IV - Interprétation		
Contexte d'interprétation	**Extrapolation des résultats**	**Travaux futurs**
• Comparaison des algorithmes en fonction des intervalles de confiance de la précision obtenue pour chacun des algorithmes avec les techniques de bootstrap ; • Comparaison des algorithmes en fonction de la puissance statistique ; • Description de l'impact de l'ajustement des paramètres algorithmiques sur la précision.	• Comparaison entre les résultats de cette recherche et de ceux de la revue de littérature ; • Comparaison entre la méthode d'évaluation des algorithmes de cette recherche et de celles de la revue de littérature;	• Étude de l'impact du bruit statistique non gaussien sur la précision des algorithmes ; • Étude approfondie des algorithmes d'IA en apprentissage ensembliste ; • Étude de l'impact des techniques de prétraitement des données sur la précision des algorithmes ; • Chapitre 4 du mémoire.

2.2 Les facteurs de forme

Les cartes de contrôles sont des suites ordonnées de valeurs numériques qui représentent l'évolution d'une variable aléatoire en fonction du temps. C'est la même définition que celle des séries temporelles. La littérature sur la classification des séries temporelles est très riche (p. ex. en reconnaissance vocale et en analyse des séries temporelles financières) et Fulcher (2017) en fait un excellent résumé. Un des problèmes qu'il faut résoudre est de choisir comment représenter une fenêtre d'une série temporelle de manière consistante pour que les algorithmes d'IA puissent la classer.

La méthode la plus couramment utilisée est celle développée par Gauri et Chakraborty (2007) qui proposent une série de 30 mesures qui caractérisent les cartes de contrôle. Ces mesures sont appelées des facteurs de forme. Cette étude est reprise par Gauri et coll. (2010) qui proposent de n'utiliser que 7 des 30 facteurs pour minimiser la corrélation croisée et ainsi obtenir de meilleurs résultats. Cette proposition est motivée par Montgomery (2005). Les auteurs ont utilisé un corrélogramme, qui est un outil statistique utilisé en réduction de la dimensionnalité, pour isoler ces 7 facteurs qui sont repris et qui sont présentés dans cette section. La figure 3.12 montre le corrélogramme de l'espace d'attributs. Dans les sections qui suivent, les facteurs de formes utilisés sont expliqués en détail.

2.2.1 Facteur de forme no. 1 : FF1

Le premier facteur de forme correspond au signe + ou − de la pente de la droite des moindres carrés de la carte de contrôle. Il permet de distinguer les motifs de tendances positives et négatives ainsi que les hausses et les baisses subites.

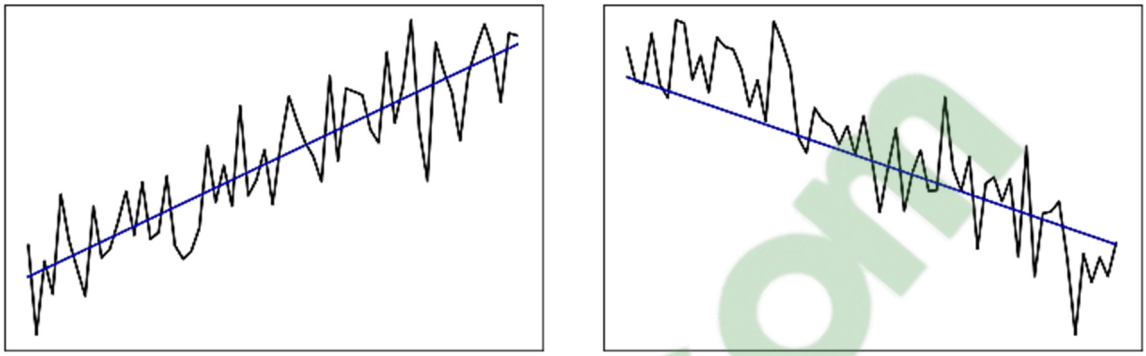

Figure 2.1 Représentation schématique du facteur de forme FF1

2.2.2 Facteur de forme no. 2 : FF2

Le deuxième facteur de forme correspond au ratio entre la variance de la série temporelle et la somme des erreurs quadratiques de la droite des moindres carrés. Pour les motifs normaux et cycliques, FF2 est environ égal à 1 et est supérieur à 1 pour les autres motifs.

2.2.3 Facteur de forme no. 3 : FF3

Le troisième facteur de forme se calcule en divisant l'aire entre la série temporelle et la droite des moindres carrés par la variance de la série temporelle. Il a été démontré expérimentalement que ce facteur est un bon discriminant pour tous les motifs considérés dans cette recherche Gauri et Chakraborty (2007).

2.2.4 Facteur de forme no. 4 : FF4

Le quatrième facteur de forme se calcule en divisant le nombre de fois que la série temporelle croise la droite des moindres carrés par le nombre de fois qu'elle croise la droite moyenne. Ce facteur de forme permet de mieux distinguer les motifs de tendances des motifs de changements subits.

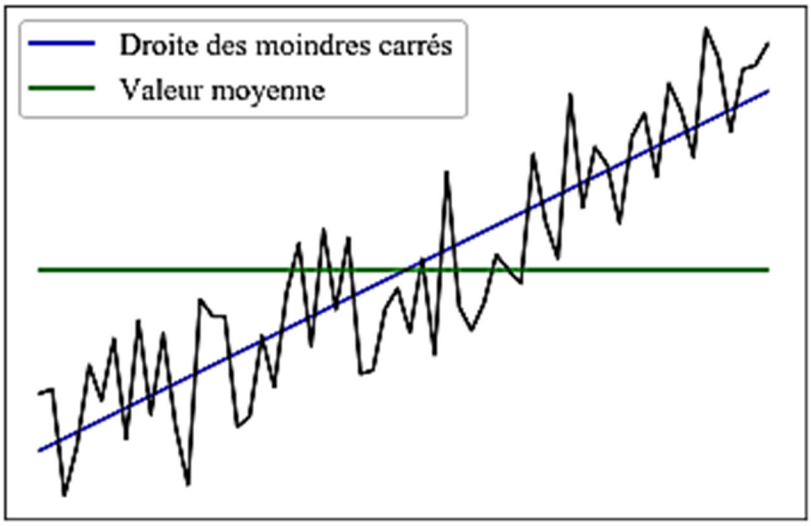

Figure 2.2 FF4 avec le motif de tendance positive

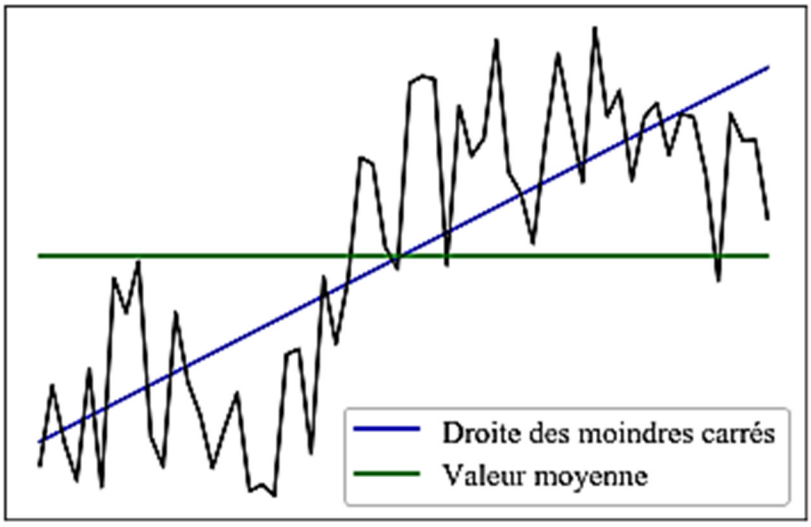

Figure 2.3 FF4 avec le motif de hausse subite

2.2.5 Facteur de forme no. 5 : FF5

Pour calculer le cinquième facteur de forme, on commence par diviser la carte de contrôle en 6 segments égaux et on calcule les 6 pentes $\boldsymbol{m} = \langle m_1, m_2, m_3, m_4, m_5, m_6 \rangle$ de la droite des moindres carrés correspondante. L'étendue des pentes correspond au cinquième facteur de forme, soit $FF5 = \max(\boldsymbol{m}) - \min(\boldsymbol{m})$. Il a été démontré expérimentalement que ce facteur est un bon discriminant pour tous les motifs utilisés dans le cadre de cette recherche Gauri et Chakraborty (2007).

2.2.6 Facteur de forme no. 6 : FF6

Le sixième facteur de forme correspond au ratio de l'erreur quadratique moyenne de la droite de régression de la série temporelle en entier et la moyenne des erreurs quadratiques moyennes des 6 droites de régression calculées pour le cinquième facteur de forme. Ce facteur de forme sert à distinguer le motif normal du motif cyclique.

2.2.7 Facteur de forme no. 7 : FF7

Le septième facteur de forme sert à distinguer les motifs de hausses et de baisses subites. Il consiste à séparer la série temporelle en n segments à condition que chacun comporte au moins 8 points. Les segments sont ensuite combinés en paires, soit C_2^n combinaisons en tout, puis la différence des erreurs quadratiques moyennes des droites des moindres carrés est calculés. La valeur de FF7 correspond à la plus petite valeur calculée.

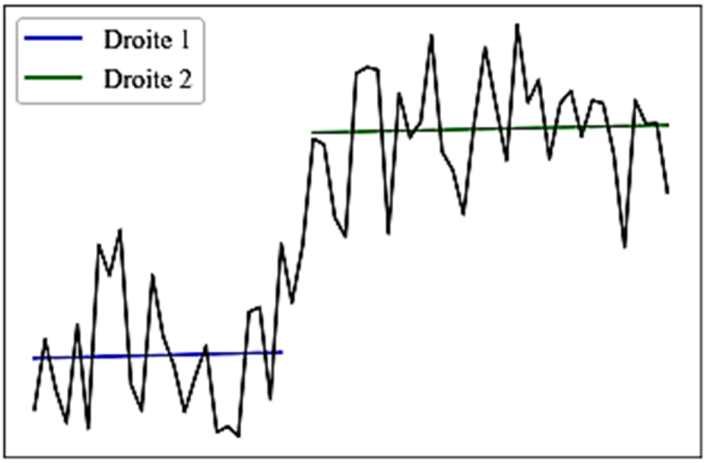

Figure 2.4 FF7 avec le motif de hausse subite

2.3 Évaluation des algorithmes

L'évaluation des algorithmes commence par l'entraînement avec 80% des données et la validation avec le 20% qui reste. Les données de chacun de ces deux groupes sont choisies au hasard et sont les mêmes pour chacun des algorithmes. Pendant la phase d'entraînement, les hyperparamètres sont ajustés de manière à minimiser une fonction objective. Pendant la validation, la fonction objective (c.-à-d. l'erreur) est calculée à la toute fin, car aucune mise à jour des paramètres n'est faite.

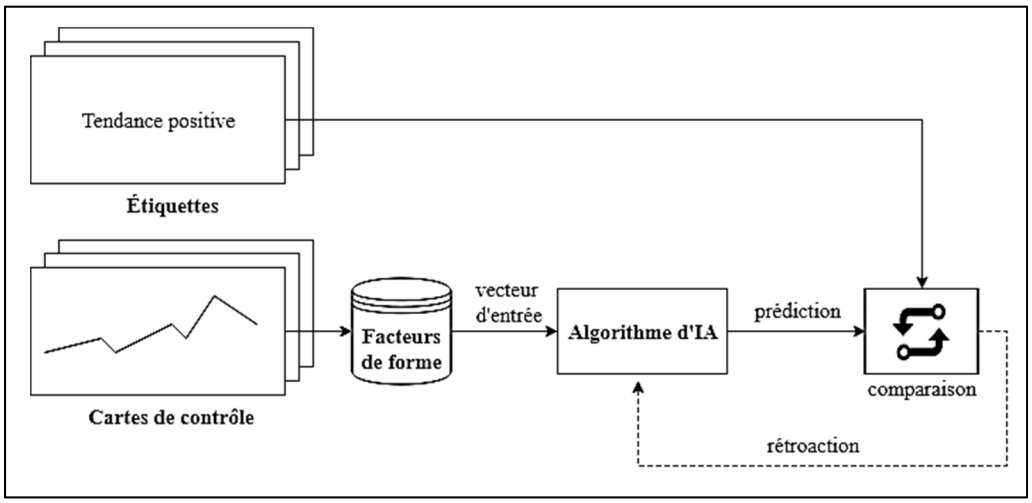

Figure 2.5 Évaluation de la performance des algorithmes

La précision moyenne est notée et le tout est répété 1 000 fois. Le résultat est l'échantillon de précision $P = (p_1, p_2, \ldots, p_{999}, p_{1000})$. L'intervalle de confiance non paramétrique de P est calculé en utilisant 10 000 échantillons de bootstrap. La procédure est la suivante pour un niveau de confiance $\alpha = 0.95$:

1 **pour** $b \leftarrow 1$ à 10 000 **faire**
2 $P_{bs}[b] \leftarrow$ rééchantilloner P aléatoirement avec remise
3 $\bar{P}_{bs}[b] \leftarrow 1/1\,000 \sum P_{bs}[b]$
4 **fin pour**
5 $\bar{P}_{bs}[b] \leftarrow$ trier $\bar{P}_{bs}[b]$ en ordre croissant
6 $P_{moyenne} \leftarrow 1/10\,000 \sum \bar{P}_{bs}[b]$
7 $P_{inférieure} \leftarrow \bar{P}_{bs}[251]$
8 $P_{supérieure} \leftarrow \bar{P}_{bs}[9\,750]$

La procédure est inspirée de Bradley et Tibshirani (1994). Le niveau de confiance $\alpha = 0.95$ fait que $(100 - \alpha)/2\,\%$ des données doivent être éliminées à gauche et à droite de la distribution fréquentielle obtenue à l'étape 5 du pseudocode. Cela correspond à éliminer 2,5% des données de chaque côté. Les données qui restent (c.-à-d. 95%) sont approximativement centrées autour de la moyenne et les extrémités sont des estimateurs non paramétriques de l'intervalle de confiance pour la précision (figure 2.30). Une méthode non paramétrique a été utilisée, car la distribution des précisions résultantes n'est pas gaussienne.

Finalement, la puissance statistique des algorithmes est donnée par $1 - \beta$ où β représente le risque d'erreur de type II. Sa valeur est calculée à partir d'une matrice de confusion normalisée. La matrice de confusion est un outil qui permet de mesurer la tendance d'un modèle à faire des erreurs de type I et type II. Dans cette recherche, l'hypothèse nulle h_0 est qu'il y a un motif non naturel présent et l'hypothèse alternative h_a est qu'il n'y en a pas (c.-à-d. que le motif est classé normal).

Une erreur de type II est commise quand un motif est classé normal alors qu'il ne l'est pas en réalité. Par exemple, la deuxième ligne de la matrice de confusion de la figure 3.1 montre que pour le motif cyclique, $\beta = 0.08$. Le paramètre β global est calculé en prenant la moyenne des sommes des éléments de la partie triangulaire inférieure pour l'ensemble des lignes de la matrice. La puissance statistique est donc égale à $1 - (0.08 + 0 + 0 + 0 + 0)/5 = 0.984$.

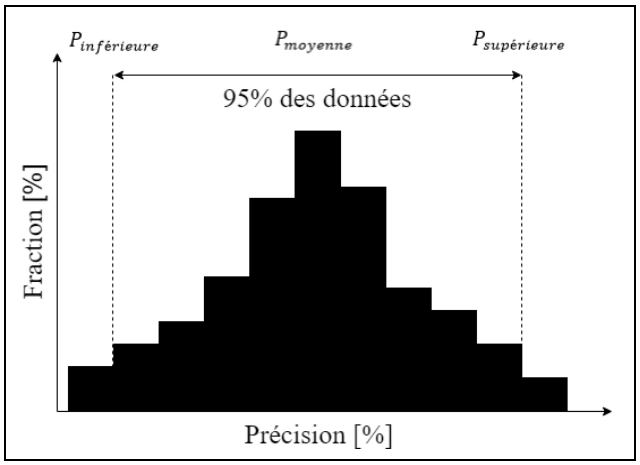

Figure 2.6 : Intervalle de confiance avec l'échantillon de bootstrap

Cette méthode a été choisie au lieu de la validation croisée pour deux raisons. La première est que le bootstrap donne une mesure généralement plus robuste de la stabilité grâce à sa faible variance (Hastie et coll., 2017). La deuxième est que même s'il est reconnu que le bootstrap a un plus grand biais que la validation croisée, ce n'est pas toujours le cas (Kohavi, 1995). Le graphe de la figure 3.31 qui montre les intervalles de confiance superposées verticalement est donc, selon l'auteur de cette recherche, un outil plus pratique pour comparer les modèles entre eux.

Les notions mathématiques présentées dans cette section servent à établir le contexte d'interprétation des résultats. Dans le prochain chapitre, les résultats obtenus sont présentés et analysés pour chacun des algorithmes.

CHAPITRE 3

PRÉSENTATION ET ANALYSE DES RÉSULTATS

3.1 Les arbres décisionnels

L'arbre décisionnel exhaustif (en anglais : *fully grown tree*) est l'arbre décisionnel obtenu quand toutes les feuilles de l'arbre sont pures (c.-à-d. que le coefficient de Gini ou l'entropie est égal à zéro et qu'il n'y a qu'une seule étiquette par feuille). En l'absence d'un critère d'arrêt restrictif, la majorité des algorithmes d'AD aboutissent à une version quelconque de cet arbre. Le problème est qu'il risque fortement de faire du surapprentissage. Ceci est dû au fait qu'il ne discrimine pas les valeurs aberrantes et que la nature des algorithmes gloutons rend le modèle si flexible qu'il modélisera partiellement le bruit statistique. Dans cette section, les résultats obtenus avec l'arbre décisionnel exhaustif à titre de référence sont présentés.

L'AD exhaustif a une précision moyenne de 92,72% avec une borne inférieure de 92,59% et une borne supérieure de 92,86%. La puissance statistique est de 0.983. Par définition, l'arbre exhaustif a une précision de 100% avec le jeu d'entraînement, ce qui signifie une perte de précision moyenne de 7,28% entre le jeu d'entraînement et le jeu de validation. Une baisse de performance de cette grandeur supporte l'hypothèse qu'il est question de surapprentissage (Hastie et coll., 2017).

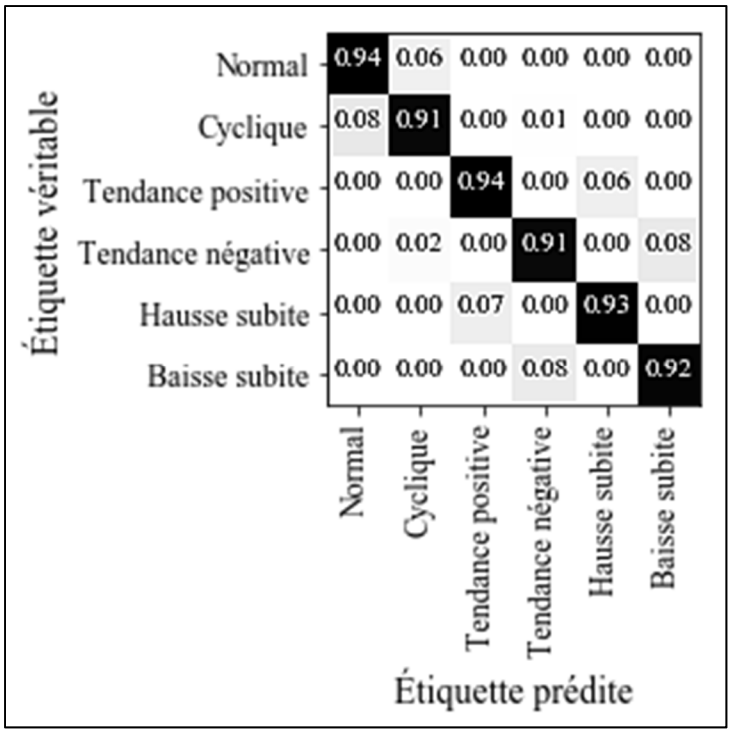

Figure 3.1 Matrice de confusion des AD exhaustifs

Un exemple d'instance incorrectement classée de motif de hausse subite est montré à la figure 3.2 côte à côte avec un exemple correctement classé.

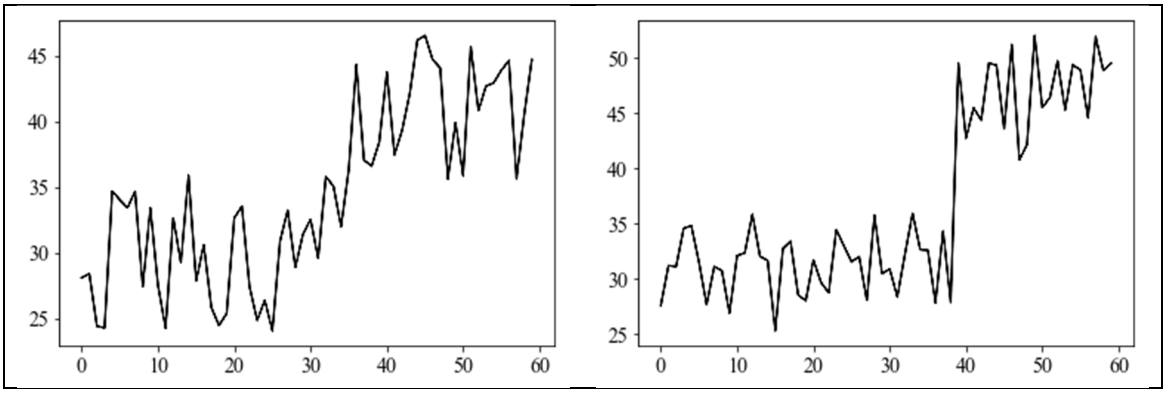

Figure 3.2 Comparaison entre un motif mal classé (gauche) et un motif correctement classé (droite)

Il est possible de constater que le motif de gauche peut facilement être confondu pour un motif de tendance positive même par un observateur humain. La hausse subite de l'image de gauche est beaucoup moins prononcée que celui de l'image de droite. Le même genre d'analyse a été effectué pour chacun des motifs mal classés et la même conclusion a été obtenue à chaque fois. La distinction entre les différents motifs n'est pas toujours évidente.

Puisque la précision du modèle varie en fonction des jeux d'entraînement et de validation qui sont choisis au hasard, le résultat obtenu est montré sous forme de distribution à la figure 3.3. C'est à partir de cette distribution que l'intervalle de confiance est calculé en prenant des échantillons de bootstrap. La précision varie entre 85,00% et 99,17% avec une moyenne de 92,72 %. Le modèle qui a obtenu une précision de 99,2% n'est pas nécessairement celui qui sera le plus performant en pratique, car il se peut que les données du jeu de validation soient composées des motifs les plus clairs (p. ex. la figure 3.2) seulement par le fruit du hasard. Sur les 1 000 modèles générés, seulement 4, soit 0,4%, ont une précision supérieure ou égale à 98%. Finalement, il est plus réaliste de rapporter la précision moyenne de 92,72% puisque la distribution est assez symétrique.

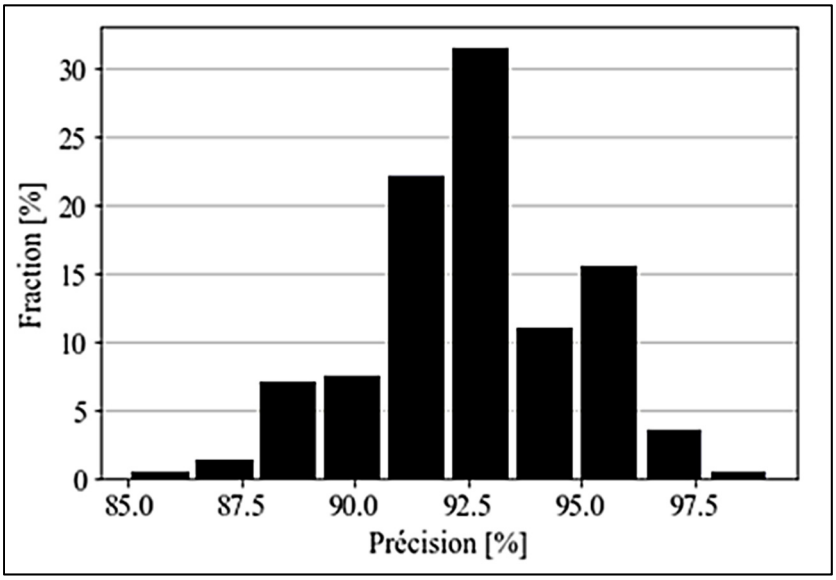

Figure 3.3 Distribution de la précision du modèle d'AD exhaustif

3.2 Les systèmes spécialisés basés sur les AD

Un modèle plus complexe (figure 3.4) a été construit pour tenter de répliquer le constat général relevé de la littérature qui prétend que les systèmes spécialisés sont plus performants que les systèmes singuliers. Un premier AD dit généraliste a été entraîné avec un jeu de données composé des 6 classes. Les prédictions de cet AD passent ensuite à travers une batterie de conditions logiques qui les dirige vers un de trois AD spécialisés entraînés sur une des paires de motifs qui sont le plus souvent confondus. L'arbre spécialisé fait le dernier tri et classe l'exemple. Dans ce cas de figure, la différence de performance est nulle et la distribution des précisions obtenues est identique à celle de la figure 3.3. La précision du système composé d'algorithmes spécialisés est donc égale à 92,72 % pour les mêmes raisons que l'AD seul. Ce constat est en désaccord avec la littérature dans un sens général, car une amélioration de la performance était attendue (Shao, 2012), mais aucune littérature spécifique sur les systèmes spécialisés composés uniquement d'AD n'a été trouvée. Il n'y a donc pas de contradiction.

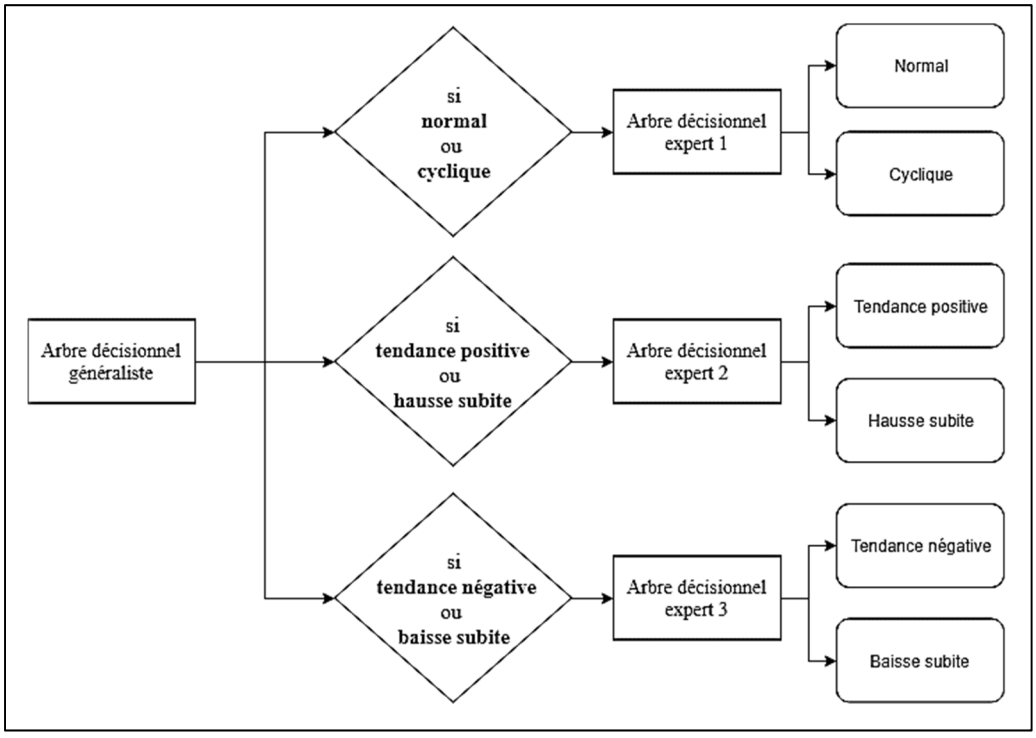

Figure 3.4 Schéma du modèle composé d'arbres décisionnels spécialisés

3.3 Les forêts aléatoires

Une série de 1 000 forêts aléatoires composées de 100 AD a été construite et utilisée pour classer les données avec une précision moyenne de 93,42%, une limite inférieure de 93,38%, une limite supérieure de 93,46% et une puissance statistique de 1. La matrice de confusion de la figure 3.5 révèle que les forêts aléatoires performent mieux avec les motifs normaux et cycliques, mais moins bien avec les autres en comparaison aux AD singuliers.

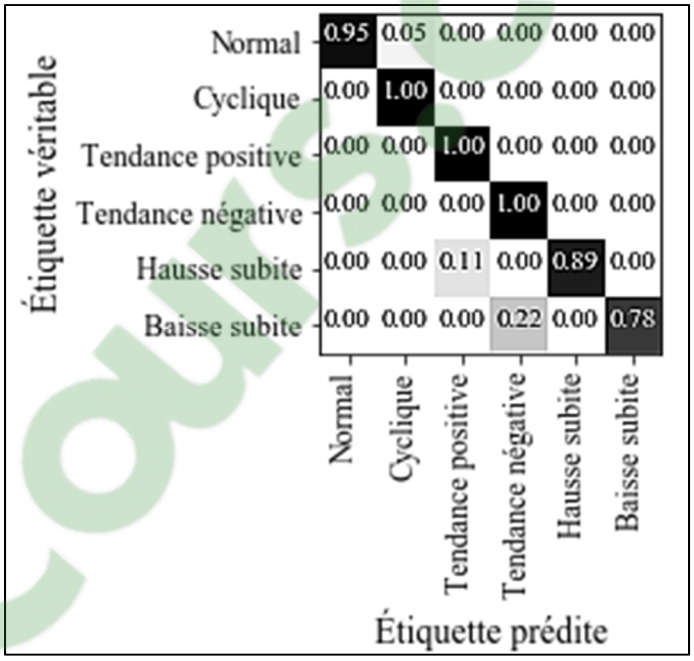

Figure 3.5 Matrice de confusion des forêts aléatoires

Les FA sont beaucoup plus stables, car la variance des précisions (figure 3.6) obtenues est beaucoup plus petite, soit $4{,}0 \times 10^{-5}$ contre $49{,}3 \times 10^{-5}$ ou une réduction de 91,9%. Ce constat est en accord avec la littérature (Hastie et coll., 2017).

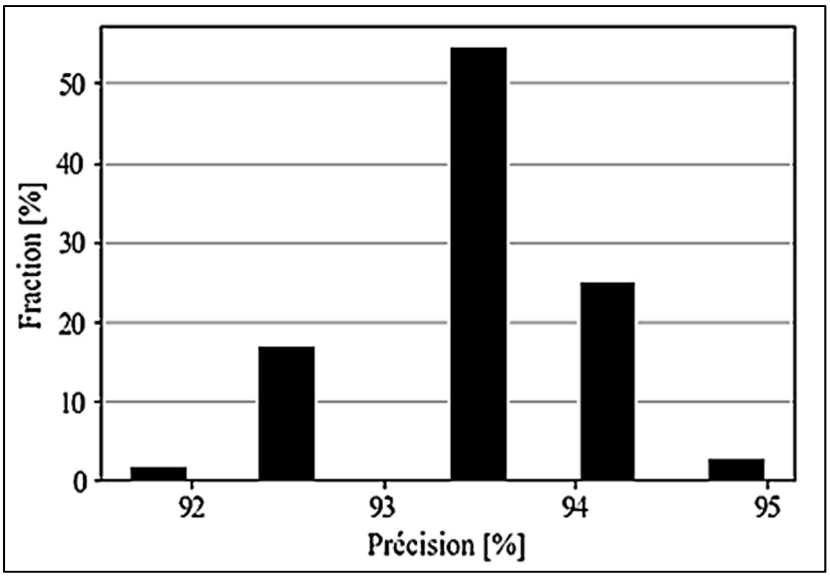

Figure 3.6 Distribution de la précision du modèle de forêt aléatoire

3.4 Les systèmes hybrides basés sur les arbres décisionnels et les forêts aléatoires

L'AD du système spécialisé de la figure 3.7 est d'abord entraîné à reconnaître les 6 classes et il renvoie ensuite les instances prédites comme des motifs normaux ou cycliques à une FA spécialisée qui fait la classification finale. La précision moyenne est de 94,80% avec une limite inférieure de 94,72% et une limite supérieure de 94,87%. La puissance est de 0.995.

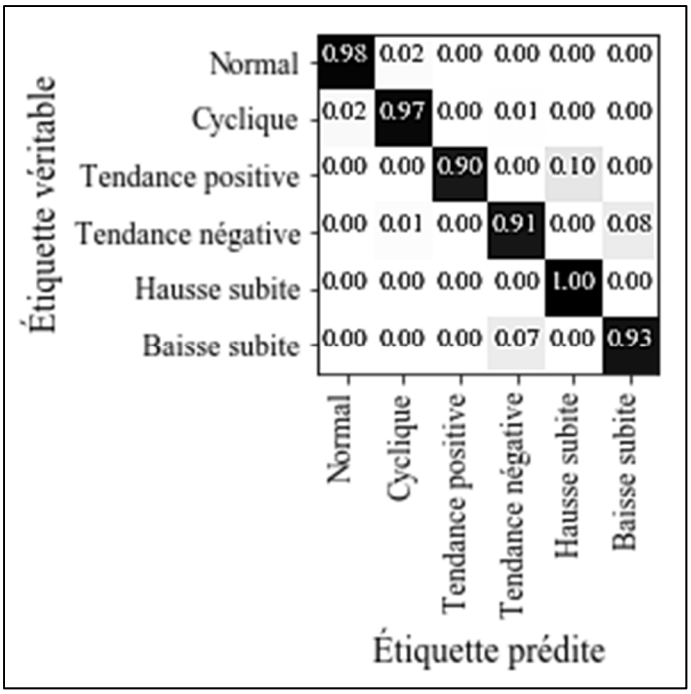

Figure 3.7 Matrice de confusion du système hybride fait d'un AD et d'une FA

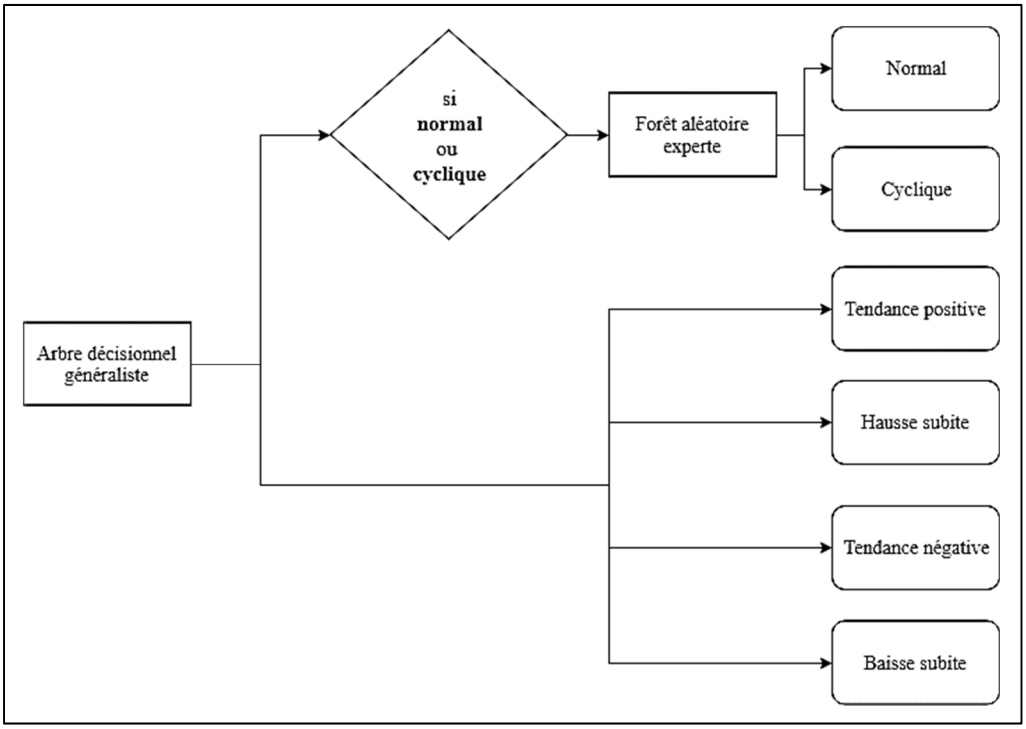

Figure 3.8 Schéma du modèle composé d'un AD singulier et d'une FA spécialisée

Le système hybride de la figure 3.8 améliore la précision de 1,38%, ce qui est en accord avec Shao (2012). Cette augmentation de la précision moyenne est causée par la réduction de la variance causée par le rééchantillonnage. Le rééchantillonnage permet donc aux AD de mieux séparer l'espace d'attributs. Chaque AD de la FA est entraîné sur un jeu de données créé par rééchantillonnage par tirage avec remise. C'est ce qui s'appelle une technique de bootstrap. En plus du rééchantillonnage, les forêts aléatoires redéfinissent l'espace d'attributs de chaque AD de manière aléatoire. Chaque arbre est entraîné et validé avec un sous-ensemble de l'espace d'attributs initial. Autrement dit, les AD n'utilisent pas tous les attributs et la figure 3.8 montre que la précision varie avec le nombre d'attributs pour 1 000 AD.

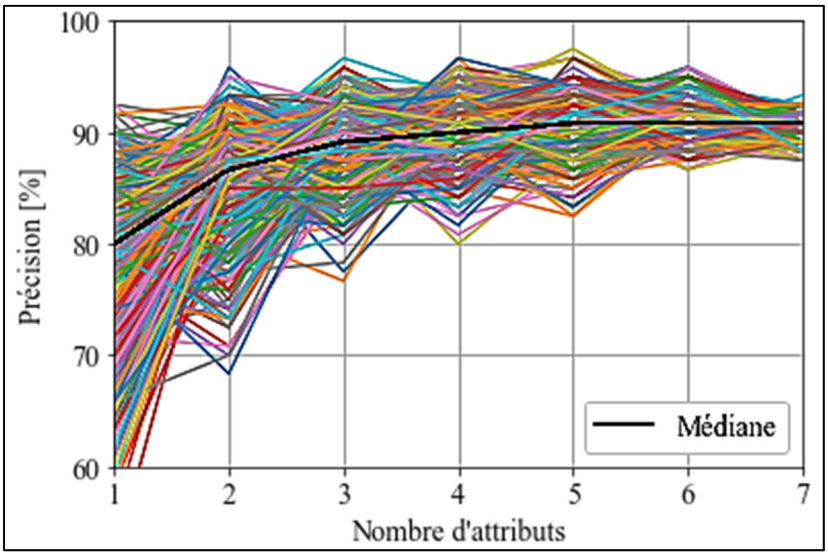

Figure 3.9 Influence du nombre d'attributs sur la précision d'un AD

La figure 3.9 laisse croire qu'il pourrait exister un nombre d'attributs qui, en moyenne, résulte en un modèle qui est plus précis. Par souci de clarté, la courbe de précision médiane est présentée pour montrer que la précision médiane de 1 000 arbres est une fonction croissante monotonique du nombre d'attributs. En d'autres mots, plus il y a d'attributs, meilleure est la précision. En contrarie, trop d'attributs pourraient réduire la précision à cause de la colinéarité qui rend l'estimation des paramètres plus difficile et moins stable.

3.5 Influence des hyperparamètres sur la précision des AD et des FA

La figure 3.10 montre que la profondeur maximale de l'AD n'a pratiquement plus d'influence sur la précision en validation au-delà d'une valeur de 3 ou 4. La figure 3.11 montre qu'il n'existe pas de lien clair entre le nombre minimal de valeurs qu'un nœud feuille doit contenir.

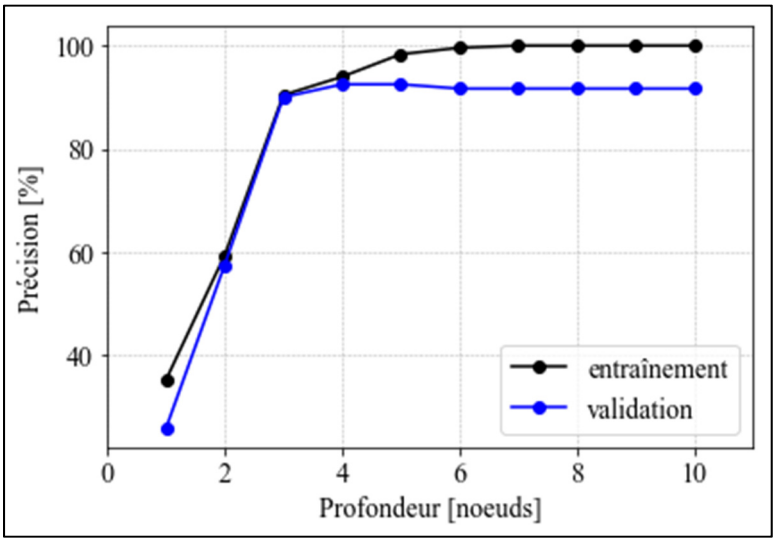

Figure 3.10 Influence de la profondeur sur la précision d'un AD

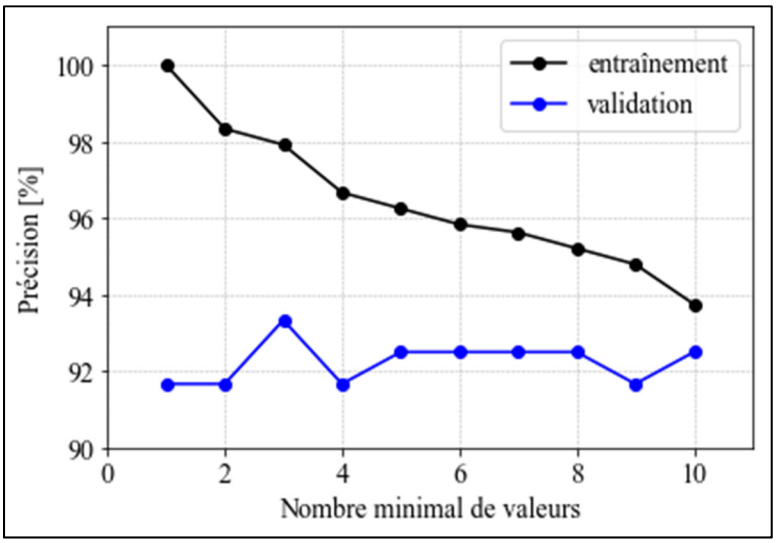

Figure 3.11 Influence du nombre minimal de valeurs par feuille sur la précision d'un AD

3.6 Résumé des résultats pour les AD et les FA

Les AD singuliers ont une précision qui varie de 85,00% à 99,17% avec une moyenne de 92,72%. Les systèmes composés d'AD spécialisés qui ont été entraînés seulement sur les paires de motifs qui sont le plus souvent confondus (c.-à-d. normal-cyclique, hausse subite-tendance positive, baisse subite-tendance négative) n'améliorent pas la performance. Un premier gain en précision moyenne de 0,7% a été obtenu en évaluant 1 000 forêts aléatoires composées de 100 AD chacune. La précision moyenne résultante est de 93,42% avec une limite inférieure de 93,38% et une limite supérieure de 93,46%. La puissance statistique est de 1. Un deuxième et dernier gain en précision de 1,38% a été obtenu, donc 2,08% en tout, avec un système hybride composé d'un AD généraliste qui renvoie les instances classées soit comme motif normal ou cyclique à une FA spécialisée entraînée exclusivement avec ces motifs. Le modèle final a une précision moyenne de 94,80% avec une limite inférieure de 94,74% et une limite supérieure de 94,87%. La puissance statistique est de 0.994. Une recherche par quadrillage sur les hyperparamètres des AD singuliers montre que tous les attributs sont corrélés à la classe des échantillons (figure 3.9). La profondeur des AD ne montre pratiquement aucune influence sur la précision au-delà de 3 ou 4 nœuds (figure 3.10). Le nombre minimal de données par feuille ne semble avoir aucune influence mesurable sur la précision (figure 3.11). Finalement, le système hybride est le modèle le plus prometteur, car même si un AD seul peut atteindre une précision de 99,17%, il a été démontré par l'intervalle de confiance qu'il est improbable qu'il performe aussi bien sur de nouvelles données.

3.7 Analyse des résultats pour les MVS linéaires

Les machines à vecteurs de support de base ont obtenu des résultats comparables aux AD et aux FA. La MVS avec un noyau linéaire a une précision moyenne de 93,00% avec une limite inférieure de 92.86% et une limite supérieure de 93.14%. La puissance statistique est de 0.968. La matrice de confusion de la figure 3.12 montre que 15% des motifs classés en motifs cycliques sont des motifs normaux en réalité.

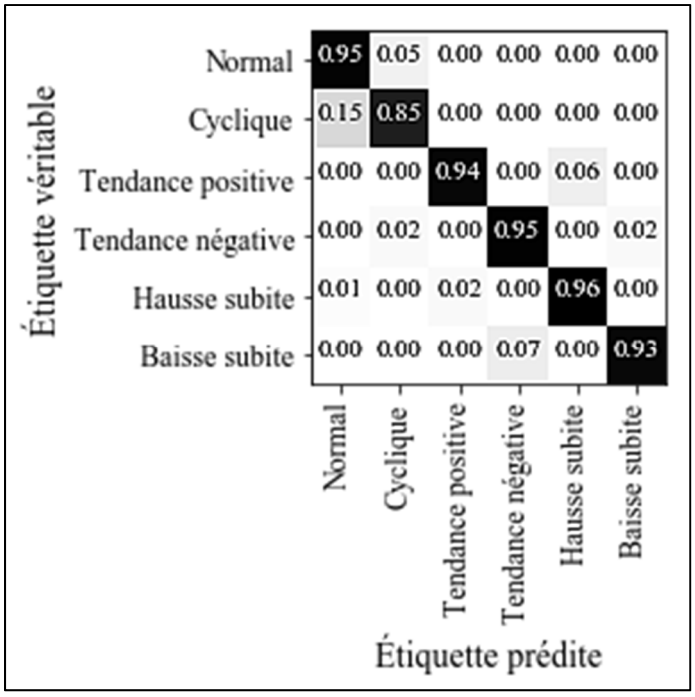

Figure 3.12 Matrice de confusion de la MVS linéaire

Afin de mieux comprendre la raison de cette baisse de la puissance statistique, le corrélogramme de la figure 3.13 est utilisé. Il permet d'étudier les dispersions relatives des motifs normaux (en bleu) et cycliques (en rouge) en fonction des $C_2^7 = 21$ combinaisons de facteurs de formes possibles.

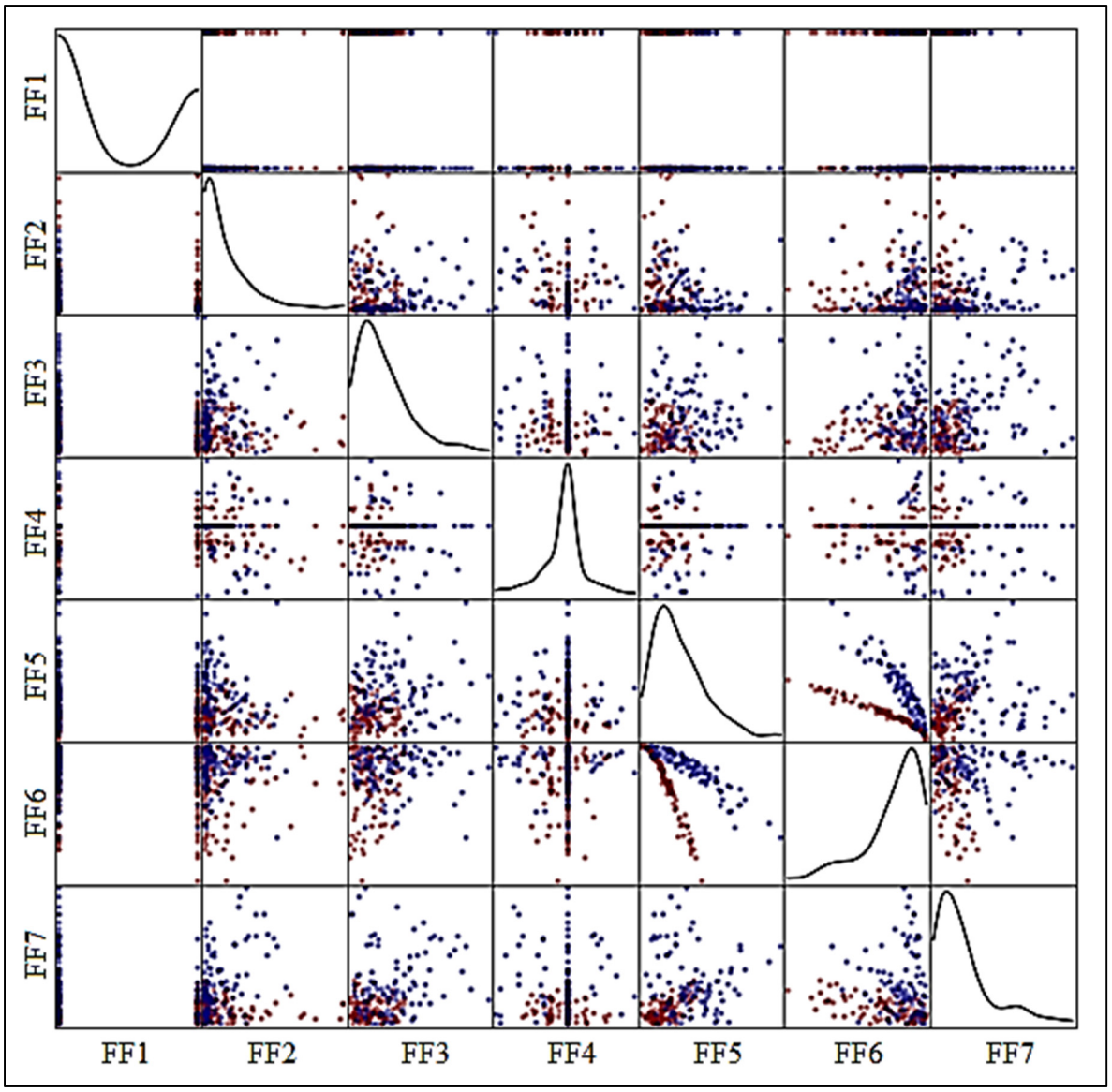

Figure 3.13 Corrélogramme pour les motifs normal (bleu) et cyclique (rouge)

Il est possible de constater que les facteurs FF5 et FF6 sont des bons candidats pour améliorer la précision, car ils ne se chevauchent presque pas et la frontière de décision de la figure 3.14 peut clairement être améliorée.

Il est possible d'améliorer la précision du modèle en changeant le paramètre de régularisation C qui caractérise la porosité de la marge.

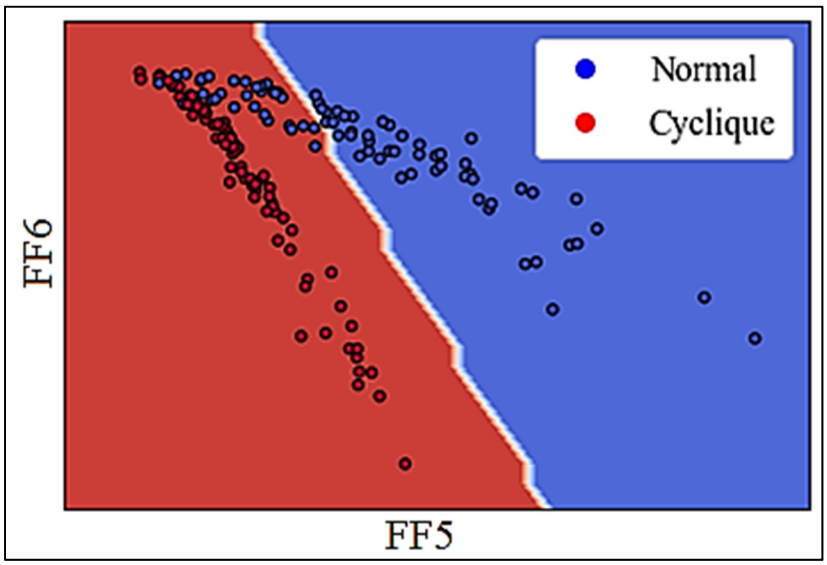

Figure 3.14 Surface de décision d'une MVS linéaire ($C = 1$)

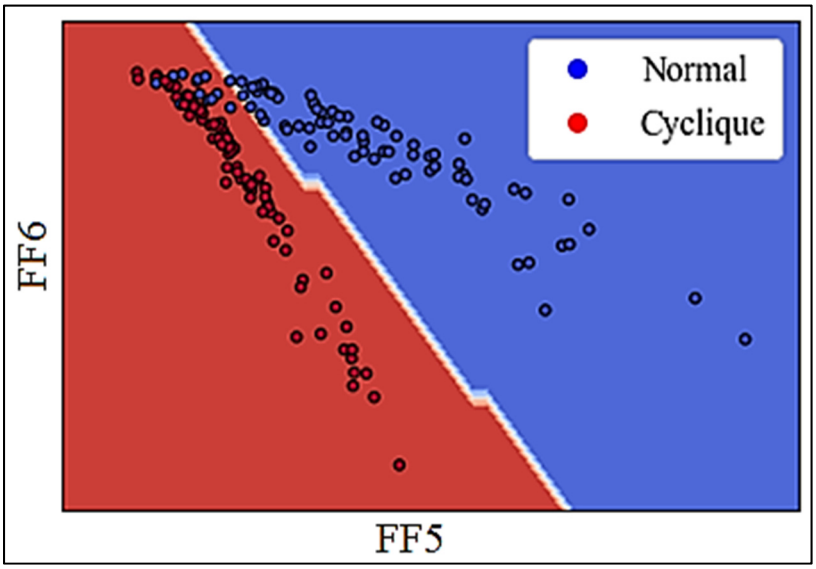

Figure 3.15 Surface de décision d'une MVS linéaire ($C = 8{,}5$)

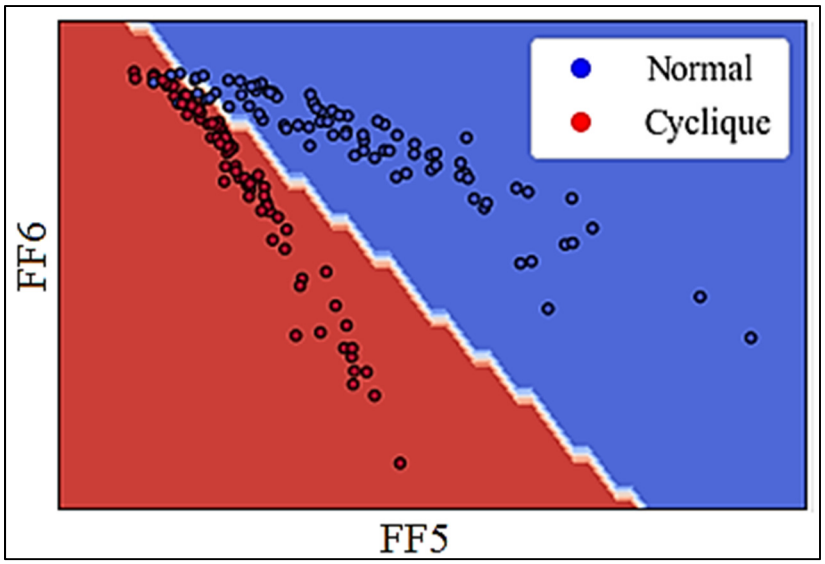

Figure 3.16 Surface de décision d'une MVS linéaire ($C = 100$)

Cent valeurs de C entre 0,001 et 100 ont été testés et un bon compromis est $C = 8{,}5$ (figure 3.15), car la précision du modèle est pratiquement identique à quand $C = 100$ (figure 3.16). Une petite valeur de C comme $C = 0{,}001$ par exemple, rend le modèle très flexible, mais elle le rend aussi très instable. Cela résulte en une précision moyenne de 23,3%. C'est une des conséquences du dilemme biais-variance. Une valeur de $C = 8{,}5$ donne suffisamment de flexibilité au modèle et résulte en une précision moyenne de 95,41% avec une limite inférieure de 95.30% et une limite supérieure de 95.52%. La puissance statistique est de 0.984.

La matrice de confusion de la figure 3.17 montre une amélioration moyenne de 2,41% de la précision pour les motifs normaux et cycliques par rapport au modèle original.

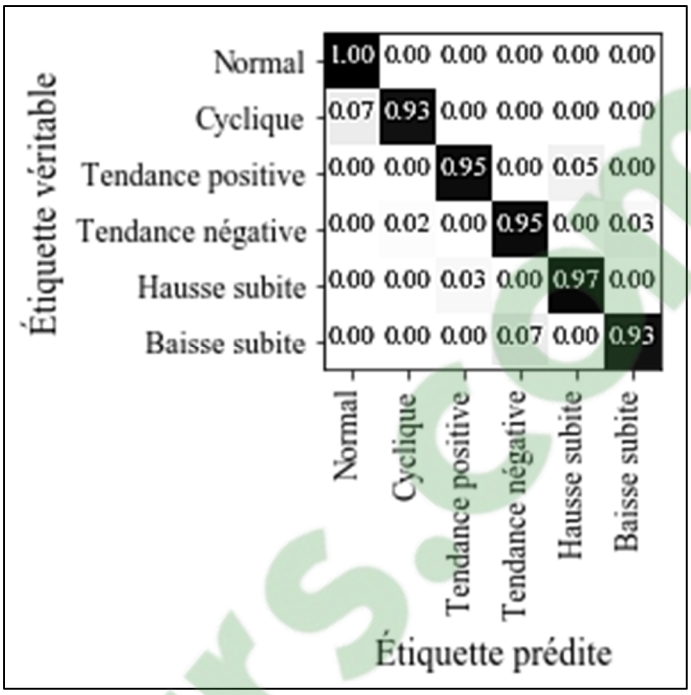

Figure 3.17 Matrice de confusion d'une MVS linéaire avec $C = 8{,}5$

Comme pour les AD, des performances supérieures à 99% ont été observées, mais seulement dans des cas très restreints. Il ne sera pas possible de s'attendre à d'aussi bons résultats quand l'algorithme est soumis à de nouvelles données. C'est pour cette raison que les intervalles de confiance sont donnés.

3.8 Les modèles hybrides basés sur les MVS linéaires spécialisées et les FA

Un modèle composé de trois MVS linéaires spécialisées identique à celui des AD de la figure 3.4 a été construit et la performance n'a pas changé du tout comme pour les AD. Autrement dit, dans ce cas de figure, une MVS linéaire entraînée avec les 6 classes est aussi précise qu'une combinaison de trois MVS entraînées sur une paire d'attributs chacune.

Le modèle hybride composé d'une FA généraliste entraînée avec les 6 classes combinées à une MVS linéaire spécialisée de la figure 3.18 donne une précision moyenne de 96,36% avec une limite inférieure de 96,30% et une limite supérieure 96,42%, soit une amélioration de 3,30% par rapport au modèle de référence. La puissance statistique est de 0.986.

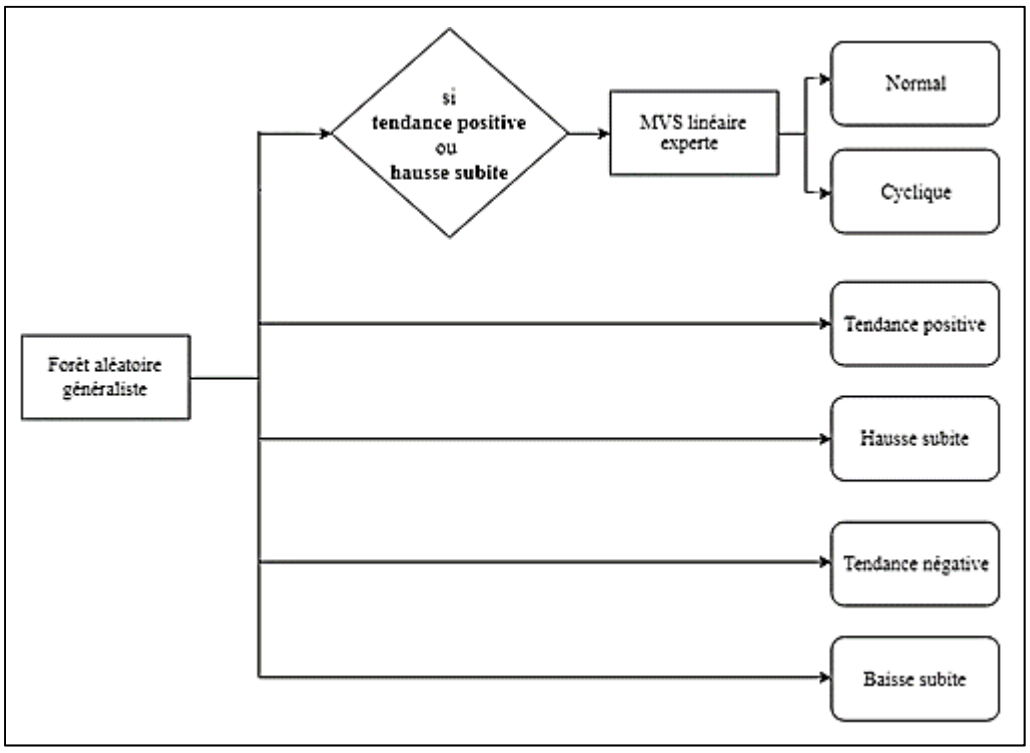

Figure 3.18 Modèle composé d'une MVS linéaire spécialisée et d'une FA généraliste

3.9 Les MVS gaussiennes

Les MVS avec un noyau gaussien ($C = 1$, $\gamma = 1$) ont une précision moyenne de 87,75% avec une limite inférieure de 87,57% et une limite supérieure de 87,97%. La puissante statistique est de 0.938. Ces résultats sont inférieurs ceux de la MVS linéaire, mais les hyperparamètres ont été choisis au hasard. De plus, il n'y a rien dans la morphologie des dispersions du corrélogramme de la figure 3.13 qui suggère qu'une frontière de décision sphéroïdale serait nécessairement meilleure qu'une frontière linéaire pour les motifs normaux et cycliques par exemple. La pertinence de cet exemple réside dans le fait que la MVS gaussienne fait la majorité de ses erreurs de classification avec ces deux motifs (figure 3.19).

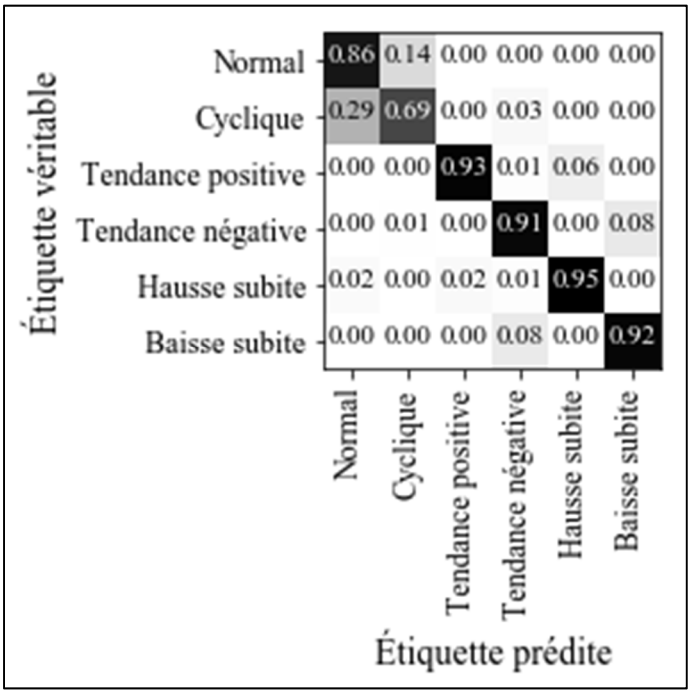

Figure 3.19 Matrice de confusion d'une MVS avec un noyau gaussien

La surface de décision de cette MVS (figure 3.20) est très similaire à celle qui a le noyau linéaire. La similarité est locale, car un changement d'échelle montre que l'espace d'attributs est bel et bien séparé par un sphéroïde (figure 3.21).

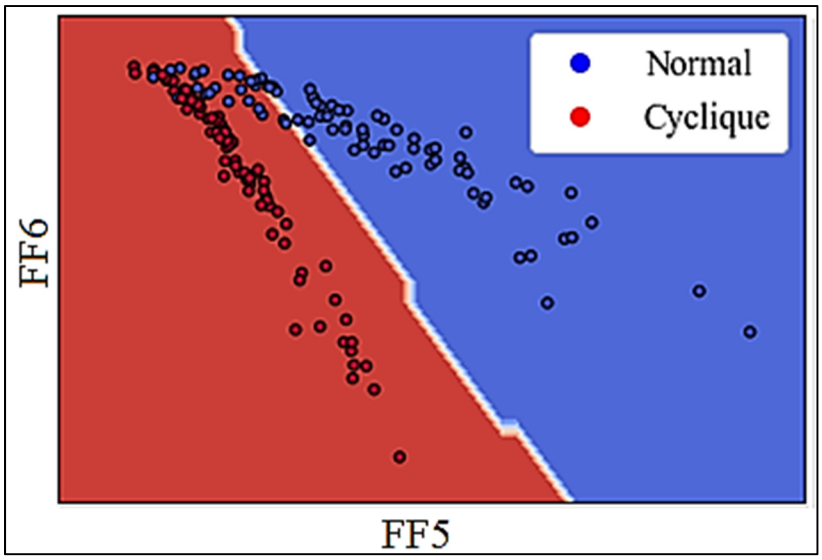

Figure 3.20 Surface de décision locale d'une MVS gaussienne ($C = 1,\ \gamma = 1$)

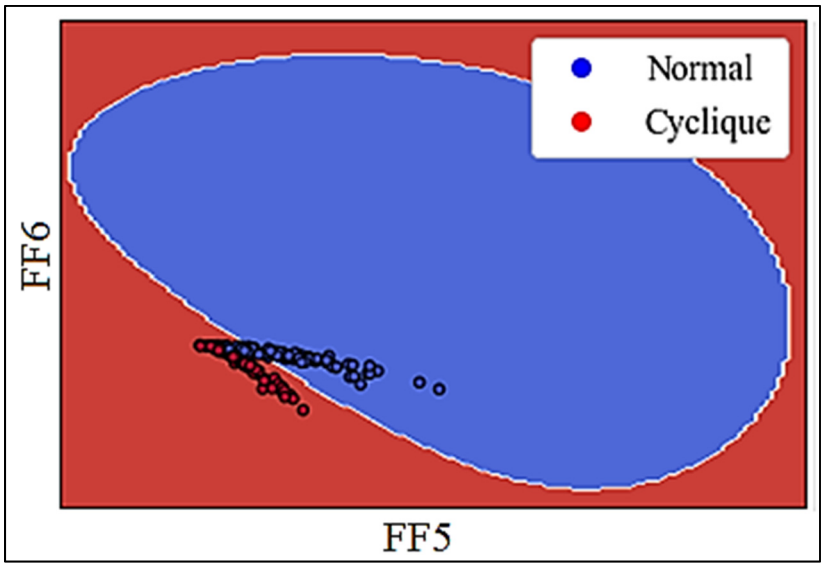

Figure 3.21 Surface de décision globale d'une MVS gaussienne ($C = 1,\ \gamma = 1$)

3.10 Influence des hyperparamètres sur la précision des MVS gaussiennes

La figure 3.22 montre que le paramètre C a peu d'influence au-delà d'une valeur de $C = 5{,}6$. Cette valeur est du même ordre de grandeur que celle utilisée pour la MVS linéaire. Cette configuration fait grimper la précision moyenne du modèle à 91,43% avec une limite inférieure de 91,27% et une limite supérieure de 91,59%. La puissance statistique grimpe à 0.962.

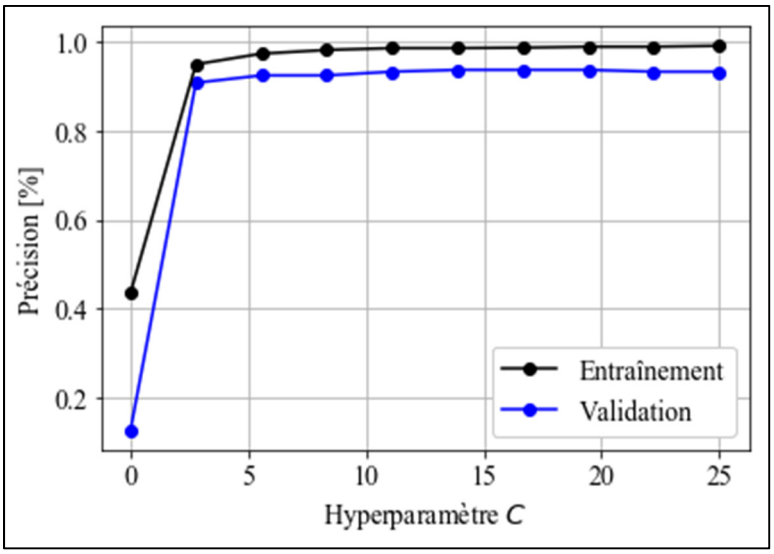

Figure 3.22 Influence de C sur la précision d'une MVS gaussienne ($\gamma = 1$)

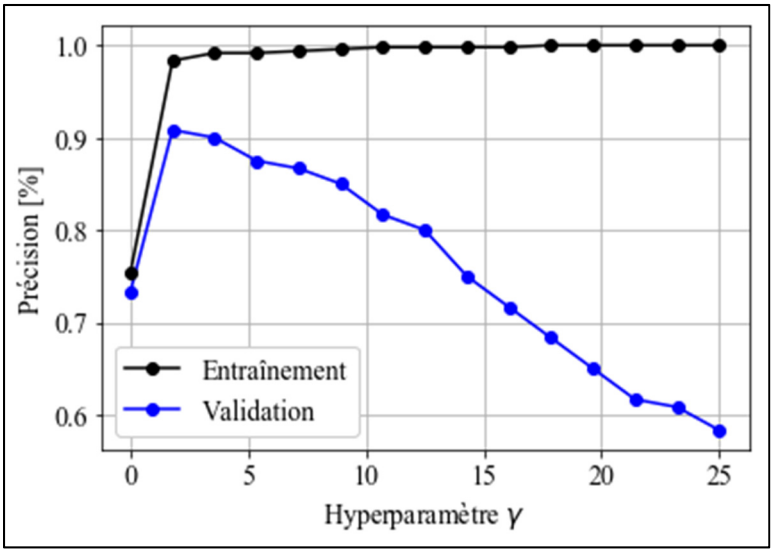

Figure 3.23 Influence de γ sur la précision d'une MVS gaussienne ($C = 5{,}6$)

L'influence de l'hyperparamètre γ sur la précision est montrée graphiquement à la figure 3.23. Le rôle de ce paramètre peut être mieux compris en regardant l'équation 2.28 :

$$K(\boldsymbol{x_1}, \boldsymbol{x_2}) = \exp\left(-\frac{\|\boldsymbol{x_1} - \boldsymbol{x_2}\|^2}{2\sigma^2}\right)$$

Dans cette équation, le paramètre γ est égal à $1/2\sigma^2$. Plus il est grand (c.-à-d. plus σ est petit), plus le terme $\exp\left(-\frac{\|\boldsymbol{x_1}-\boldsymbol{x_2}\|^2}{2\sigma^2}\right)$ est petit. Aussi, plus le terme $\|\boldsymbol{x_1} - \boldsymbol{x_2}\|^2$ est grand, plus $\exp\left(-\frac{\|\boldsymbol{x_1}-\boldsymbol{x_2}\|^2}{2\sigma^2}\right)$ est petit. En termes de valeurs absolues, une grande valeur de γ réduit davantage l'influence des points distants sur la surface de décision que l'influence des points locaux. Autrement dit, une augmentation de γ rend la surface de décision invariante aux points plus éloignés, donc le modèle devient moins flexible dans un sens global, mais il devient plus flexible localement. Le point d'équilibre se trouve aux alentours de $\gamma = 1{,}8$ avec une précision de 90,80%. En augmentant la résolution dans les environs de $\gamma \in [0{,}001\ , 2]$, la précision initiale de 91,43% dans les voisinage de $\gamma = 1$. La valeur quasi optimale avait été trouvée par hasard du premier coup.

La précision moyenne finale avec les paramètres $\gamma = 1$ et $C = 5{,}6$ est de 91,43%. La matrice de confusion est donnée à la figure 3.24.

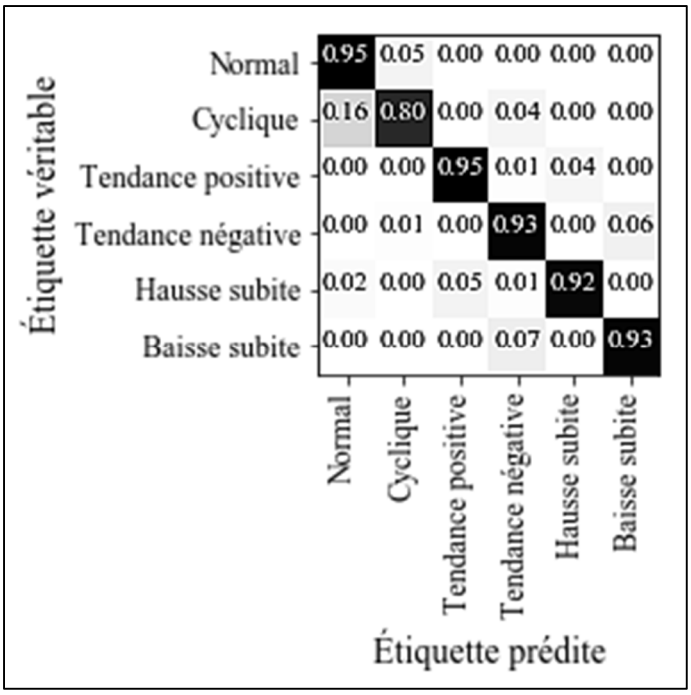

Figure 3.24 Matrice de confusion d'une MVS gaussienne ($C = 5{,}6,\ \gamma = 1$)

3.11 Les modèles hybrides spécialisés basés sur les MVS gaussiennes

Le modèle spécialisé de la figure 3.25 a été construit et ces performances sont identiques à la MVS gaussienne généraliste beaucoup plus simple de la sous-section précédente. Puisque la MVS gaussienne n'est pas meilleure à classer quelque combinaison de motifs que ce soit, aucun système hybride n'a été construit, car la performance serait nécessairement inférieure à celui de la figure 3.24.

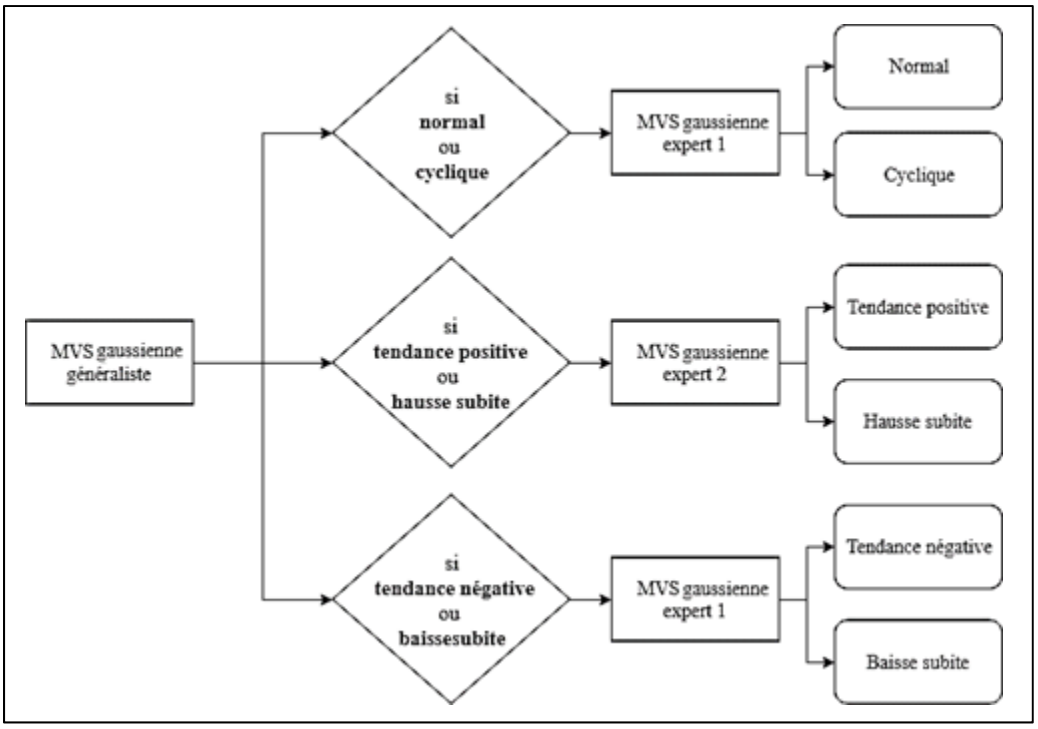

Figure 3.25 Schéma du modèle composé de MVS gaussiennes spécialisées

3.12 Résumé des résultats pour les MVS

La MVS linéaire de base avec $C = 1$ a une précision moyenne de 93,00% avec une limite inférieure de 92,86% et une limite supérieure de 93,14%. La puissance statistique est de 0.968. La majorité de ses erreurs de classification viennent du fait que 15% des motifs qu'elle classe comme cycliques sont en fait des motifs normaux. Suite à une recherche exhaustive sur l'hyperparamètre C, une valeur de $C = 8{,}5$ fait grimper la précision moyenne à 95,41% avec une limite inférieure de 95,30% et une limite supérieure de 95,51%. La puissance statistique est de 0.984. C'est une amélioration moyenne globale de 2,41% et de 6,50% pour les motifs normaux et cycliques. Un gain de précision supplémentaire de 0,95% a été obtenu en combinant une MVS linéaire spécialisée à une forêt aléatoire généraliste qui se concentre sur les motifs de tendances positives et de hausses subites.

La précision moyenne est de 96,36% avec une limite inférieure de 96,30% et une limite supérieure de 96,42%. Cela représente une amélioration de 4,9% par rapport au modèle de référence. La puissance statistique passe de 0.984 à 0.986.

La MVS gaussienne de base ($C = 1, \gamma = 1$) a donné de moins bons résultats : 87,75% en moyenne avec une limite inférieure de 87,57% et une limite supérieure de 87,94%. La puissance statistique est de 0.938. Comme pour la MVS linéaire, la MVS gaussienne ne montre pas d'augmentation appréciable de la précision au-delà d'une certaine valeur critique du paramètre C. Pour réduire la chance de faire du surapprentissage, la valeur la plus petite de $C = 5{,}6$ qui donne une précision moyenne quasi maximale de 91,43% a été choisie. Par hasard, la valeur initiale du paramètre $\gamma = 1$ était déjà optimale. La précision d'un modèle composé de MVS à noyaux gaussiens spécialisées est identique à une MVS gaussienne singulière généraliste. Il n'y a donc aucun intérêt à quadrupler le temps de calcul en utilisant un système plus complexe de ce genre. Finalement, le meilleur système est celui qui combine une MVS linéaire généraliste qui utilise une forêt aléatoire pour classer les motifs de tendances positives et de hausses subites. Les moins bons résultats ont été obtenus avec les MVS à noyaux gaussiens.

3.13 Les réseaux de neurones multicouches

La première configuration de RNA évaluée comporte trois couches : une couche d'entrée avec 7 neurones, une couche cachée avec 10 neurones et une couche de sortie avec 6 neurones (c.-à-d. un RNA de configuration 7-10-6). La couche cachée applique la fonction d'activation sigmoïde. La précision moyenne obtenue est de 95,31% avec une limite inférieure de 95,19% et une limite supérieure de 95,44%. La puissance statistique est de 0.990. La distribution des précisions pour 1 000 brassages aléatoires est montrée à la figure 3.26 et la matrice de confusion à la figure 3.27.

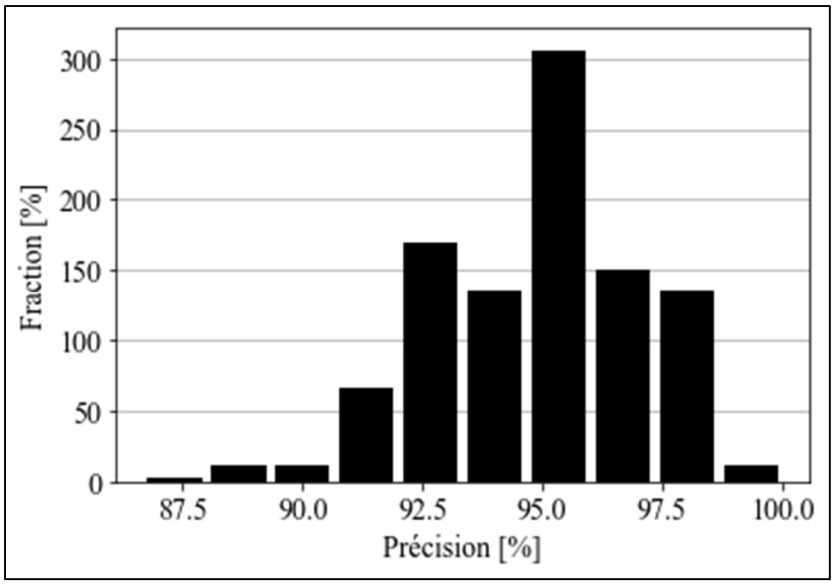

Figure 3.26 Distribution de la précision du modèle de RNA de base

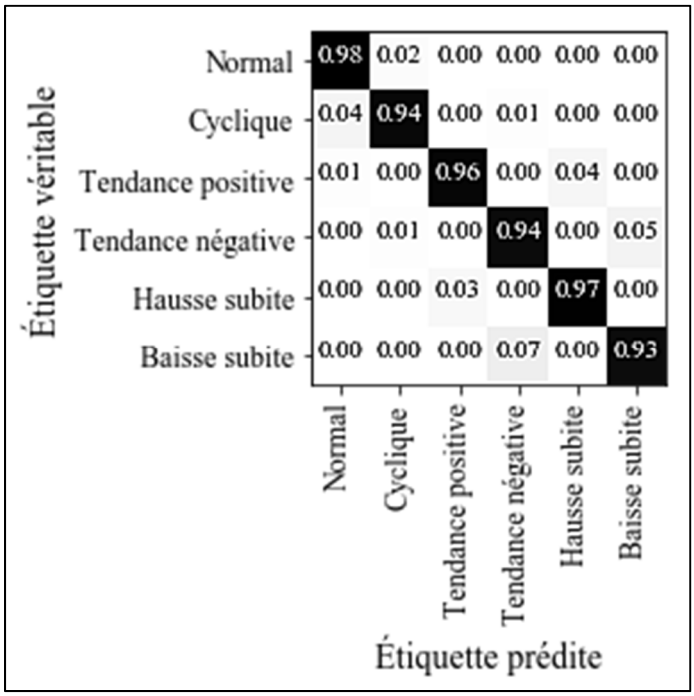

Figure 3.27 Matrice de confusion d'un RNA 7-10-6

La figure 3.28 montre que la surface de décision du RNA arrive à mieux séparer les motifs normaux et cycliques que la MVS de la figure 3.20.

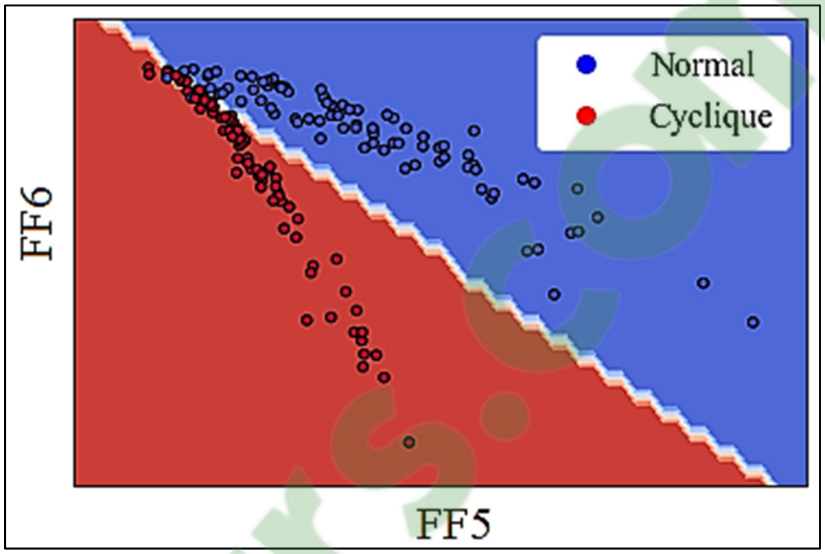

Figure 3.28 Surface de décision d'un RNA 7-10-6

3.14 Les systèmes spécialisés basés sur les RNA multicouches

Le RNA spécialisé de la figure 3.29 a été construit et la précision moyenne est passée de 95,3% à 95,40%. Jusqu'à présent, seuls les RNA deviennent meilleurs lorsqu'ils sont assemblés en systèmes spécialisés. Toutefois, cette augmentation n'est pas nécessairement statistiquement représentative étant donné qu'elle n'est que de 0,1%.

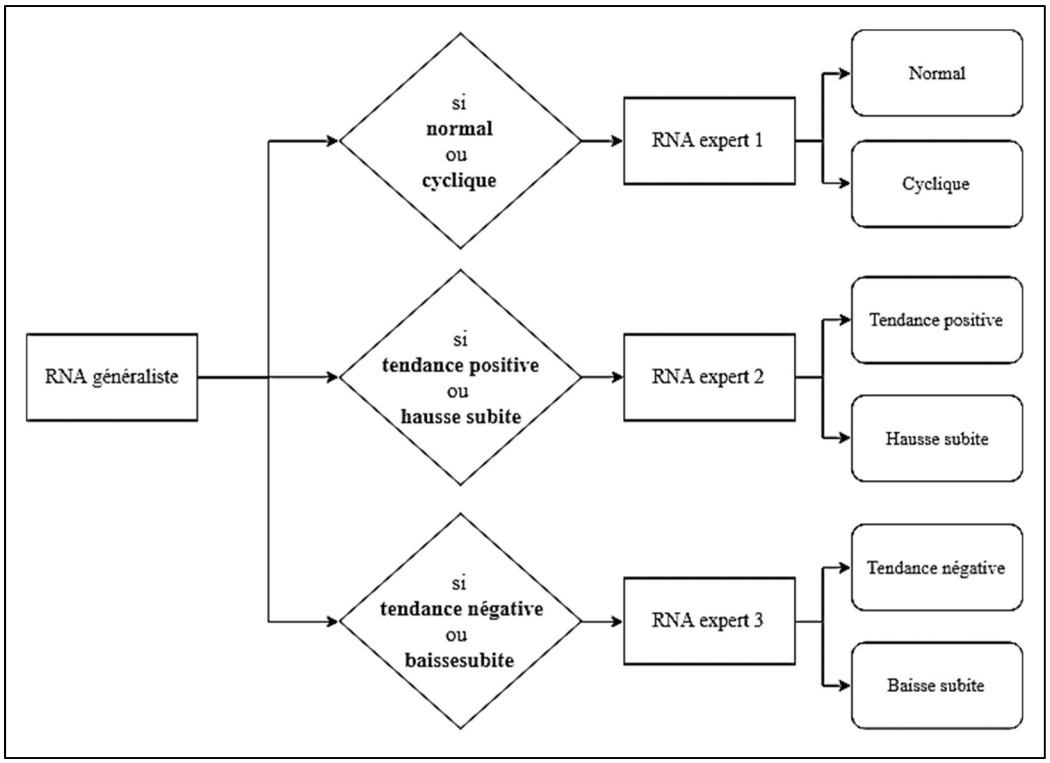

Figure 3.29 Schéma du modèle composé de RNA spécialisés

3.15 Influence du nombre de neurones de la couche cachée sur la précision d'un RNA

La figure 3.30 montre que la précision n'est pas influencée par le nombre de neurones dans la couche cachée au-delà de 5 neurones. La valeur de 5 est donc optimale considérant que le nombre de paramètres du réseau de neurones est proportionnel au nombre de neurones total dans celui-ci. Un plus petit nombre de paramètres résulte en de plus petits temps de calcul. Cela diminue aussi le risque de surapprentissage.

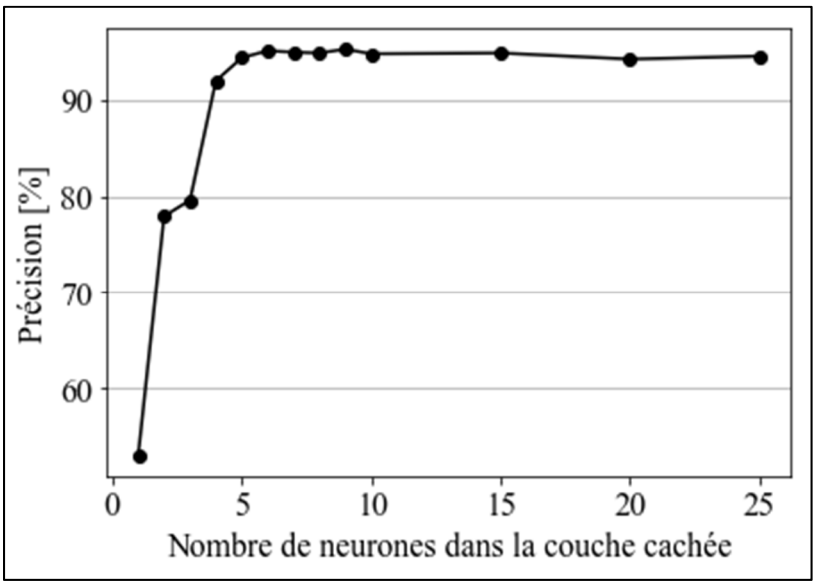

Figure 3.30 Influence du nombre de nœuds de la couche cachée sur la précision

3.16 Influence de la topologie sur la précision d'un RNA

Pour comprendre l'influence de la topologie, la précision de 25 différentes configurations est rapportée au tableau 3.1. Les résultats montrent qu'aucune des configurations n'est aussi précise que le RNA 7-10-6 de la section 3.13. Il semble que l'important est d'avoir assez de neurones pour que la mémoire (c.-à-d. les connaissances stockées par le RNA dans ses poids synaptiques) soit suffisante pour bien « comprendre » le jeu de données.

Par exemple, quand qu'il y a au moins 10 neurones dans chacune des couches, la précision est strictement supérieure à 90% avec une moyenne de 93,2%. Cette valeur chute à 88,1% quand une des deux couches possède 5 neurones. Finalement, il est observé que l'important, c'est d'avoir plus de neurones dans la dernière couche, car même avec une première couche de 5 neurones, la précision moyenne est de 92,3%. À l'inverse, quand la dernière couche cachée n'a que 5 neurones, la précision moyenne est de 84,1%.

Tableau 3.1 Influence de la structure sur la précision d'un RNA

		Nombre de neurones dans la 2e couche cachée				
		5	**10**	**15**	**20**	**25**
Nombre de neurones dans la 1re couche cachée	**5**	88,7	91,9	93,1	94,5	92,7
	10	91,5	90,2	94,4	94,4	93,6
	15	86,7	93,8	93,2	93,5	94,0
	20	77,9	92,4	91,8	93,6	94,0
	25	75,8	93,2	92,9	92,8	92,9

3.17 Récapitulatif de l'ensemble des résultats

La meilleure performance est obtenue par les MVS linéaires, suivis des RNA, FA, AD et finalement les MVS gaussiennes. La meilleure performance fut toutefois obtenue avec un système hybride avec une FA spécialisée une MVS linéaire généraliste. La précision moyenne de ce système est de 96,36% avec une limite inférieure de 96,30% et une limite supérieure de 96,42%. La puissance statistique est de 0.986.

Tableau 3.2 Paramètres expérimentaux et résultats

Algorithme	**Détails**		**Précision [%]**	**Puissance**
Arbre décisionnel	Algorithme : Critère :	CART Gini	$P_{inférieure} = 92{,}59$ $P_{moyenne} = 92{,}72$ $P_{supérieure} = 92{,}86$	0,983
Forêt aléatoire	Nombre d'arbres : Algorithme : Critère :	100 CART Gini	$P_{inférieure} = 93{,}38$ $P_{moyenne} = 93{,}42$ $P_{supérieure} = 93{,}46$	1,000
Machine à vecteur de support	Noyau : Marge :	Linéaire $C = 8{,}5$	$P_{inférieure} = 95{,}30$ $P_{moyenne} = 95{,}41$ $P_{supérieure} = 95{,}51$	0,968
Machine à vecteur de support	Noyau : Marge : Influence :	Gaussien $C = 1$ $\gamma = 5{,}56$	$P_{inférieure} = 91{,}27$ $P_{moyenne} = 91{,}45$ $P_{supérieure} = 91.59$	0,962
Réseau de neurones artificiel	Topologie : Activation : Optimisation : Taux d'apprentissage : Momentum :	7-10-6 Sigmoïde RPG $\alpha = 0{,}01$ $\eta = 0{,}9$	$P_{inférieure} = 95{,}19$ $P_{moyenne} = 95{,}31$ $P_{supérieure} = 95{,}44$	0,990

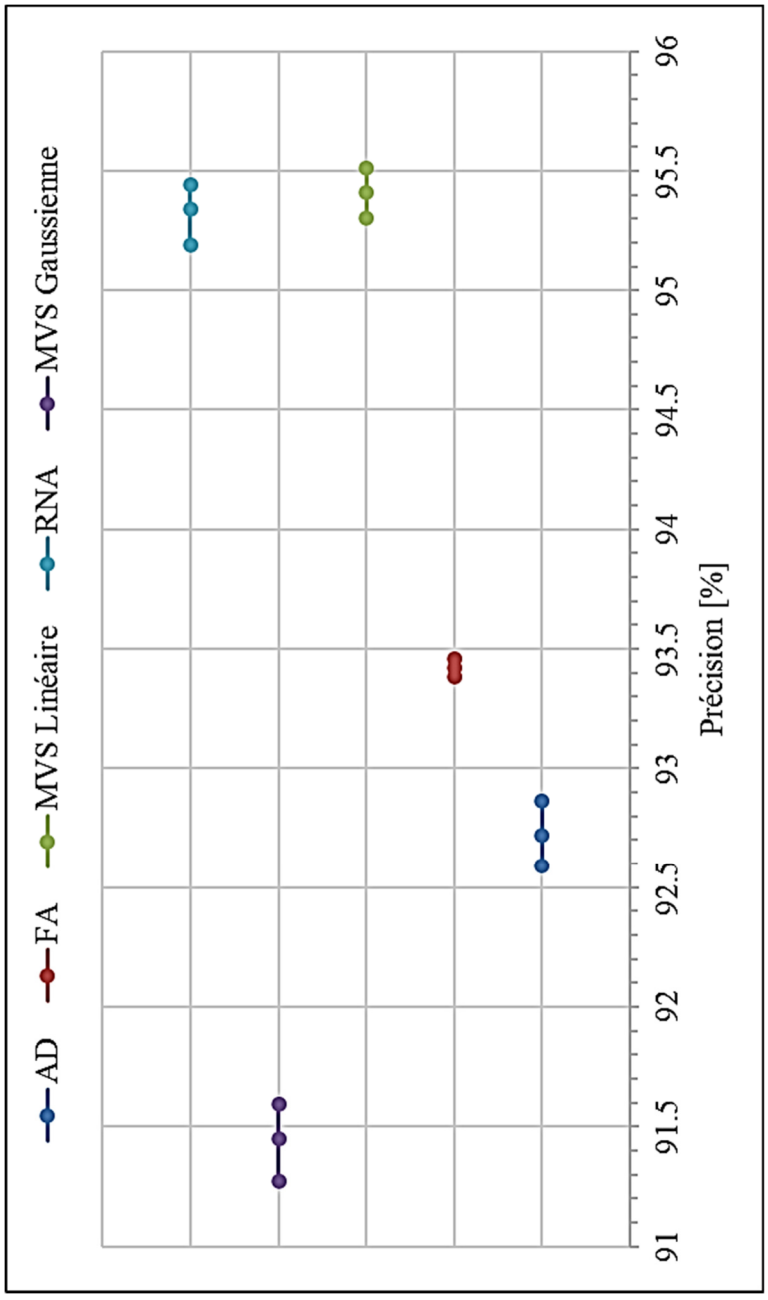

Figure 3.31 Intervalles de confiance de la précision des algorithmes

CHAPITRE 4

DISCUSSION

Il est important de souligner que les différences en valeurs absolues entre les résultats obtenus et ceux de la littérature sont petites et s'expliquent par les différentes approches utilisées. L'objectif principal de cette recherche était de comparer les algorithmes d'IA sur un pied d'égalité et de trouver le meilleur pour la RMCC.

Pour commencer, Wang et coll. (2008) rapportent une précision de 96,5% pour les AD contre une précision moyenne de 92,72% dans cette recherche. Il a aussi été observé des précisions similaires à cette valeur (figure 3.3), mais l'intervalle de confiance obtenu avec le bootstrap montre qu'il est improbable d'observer cette valeur dans la réalité. L'étude précédente ne mentionne pas avoir fait de validation croisée et ne donne pas d'intervalle de confiance, ce qui laisse croire que leurs résultats sont optimistes. Le même algorithme CART été utilisé, mais il faut supposer que nos jeux de données sont différents pour expliquer les écarts.

Aucun ouvrage spécifique à la RMCC n'a été trouvé dans la littérature qui étudiait l'influence des hyperparamètres des arbres décisionnels sur la précision. Cette partie de la recherche constitue une nouvelle contribution à la littérature. Il a été démontré par cette recherche que le nombre d'attributs a peu d'influence au-delà d'une valeur critique (figure 3.8), mais que la précision augmente toujours tant avec le jeu d'entraînement que le jeu de validation avec l'augmentation du nombre de paramètres. Cela est attribué au fait que les attributs utilisés sont faiblement corrélés entre eux, car autrement il serait attendu de constater une baisse de la performance avec le nombre croissant d'attributs. Cette conclusion est appuyée par le corrélogramme de la Figure 3.13 et les travaux de Hastie et coll. (2017). Un constat similaire a été fait pour la profondeur de l'AD (figure 3.9). Au-delà d'une certaine valeur, la précision cesse d'augmenter en entraînement et en validation.

Finalement, le nombre minimal de valeurs utilisées pour transformer un nœud feuille en nœud de hasard n'a presque pas d'influence sur la précision en validation (figure 3.11). Cette constatation est aussi en accord avec Hastie et coll. (2017) et est causée par le surapprentissage. Les feuilles qui possèdent peu d'exemples tentent de catégoriser ce qui est en fait du bruit statistique et ne se généralisent pas au jeu de validation. C'est normal, car le bruit statistique est aléatoire par définition.

Ensuite, il a été démontré que pour l'espace d'attributs et le jeu de données choisi, les AD spécialisés (figure 3.4) n'améliorent pas la précision globale. Ce résultat va à l'encontre de Shao (2012), mais s'explique par le fait que des attributs différents sont utilisés. En l'absence de colinéarité, il n'y a aucune garantie que les systèmes spécialisés sont plus performants que les systèmes singuliers. En revanche, en présence de colinéarité, la précision des systèmes singuliers diminue, car les variances des différents groupes de données s'additionnent.

En ce qui concerne les MVS, les meilleurs résultats obtenus proviennent de la MVS linéaire avec une précision moyenne de 95,41% contre le 96,66% d'Othman et Eshames (2012). Il est proposé que ces disparités sont dues aux différences de protocole et de jeux de données, car les auteurs ne spécifient pas le noyau de la MVS ni les hyperparamètres utilisés. Une autre différence est que leur espace d'attributs consiste en une approximation agrégée par morceaux (en anglais : *piecewise aggregate approximation, PAA*) des cartes de contrôles suivie d'une réduction de la dimensionnalité avec l'analyse en composantes principales.

Les résultats obtenus pour les RNA sont en accord avec Jang et coll. (2003) et Zhao et coll. (2017). Cette fois-ci, les RNA spécialisés démontrent une précision moyenne supérieure de 0,1% au RNA généraliste, ce qui est en accord avec Shao (2012). À première vue, cette augmentation est modeste, mais il reste que 37,7% des RNA spécialisés ont une précision supérieure à 96% contre 29,7% pour les RNA généralistes. La valeur de 96% est arbitraire, mais elle montre qu'il est possible d'atteindre une précision supérieure à 96% avec les systèmes spécialisés 8% plus souvent que pour les RNA généralistes.

Enfin, même si les FA ne sont pas les plus précis, leur puissance statistique est la plus élevée de toutes et ils sont un excellent candidat pour la RMCC. Les FA ne commettent pratiquement pas d'erreur de type II et leur précision est inférieure de seulement 2,94% au meilleur résultat de 96,36% obtenu par le système hybride de la section 3.8. En pratique, il est impératif de trouver un équilibre entre la précision et la propension à faire des erreurs de type II. En effet, même si l'algorithme sonne l'alarme pour un motif de hausse subite quand il est question d'un motif de tendance naturelle en réalité, l'important est qu'il y aura une intervention qualité qui permettra sans doute de découvrir la cause racine malgré l'erreur, surtout si les opérateurs sont conscients de cette faiblesse.

Somme toute, les résultats de cette recherche reflètent la majorité des constats des ouvrages cités dans la revue de littérature et les écarts sont attribués soit à l'absence de validation croisée qui donne des résultats très élevés, mais impossibles à généraliser d'un point de vue statistique, soit à l'utilisation de données différentes. Par contre, voici une observation finale, ce n'est pas parce que les auteurs des études citées ne mentionnent pas avoir fait de validation croisée de manière explicite qu'ils n'en ont pas fait. Si tel est le cas, les conclusions devraient être révisées et il serait supposé que les écarts proviennent des différentes données avec lesquelles les modèles ont été évalués.

CONCLUSION

L'objectif principal de cette recherche était d'évaluer les différents algorithmes d'apprentissage machine en reconnaissance de motifs sur cartes de contrôle. Deux des objectifs secondaires étaient d'atteindre l'objectif principal sans traitement préférentiel à un des algorithmes et sans utiliser d'algorithmes d'optimisation spécialisés. Le tout dans le but d'évaluer l'aptitude de l'algorithme en soi plutôt que d'une technique d'optimisation particulière. Tous ces objectifs ont été atteints.

Le troisième et dernier objectif secondaire était de fournir tous les détails nécessaires pour qu'une tierce partie puisse répliquer les résultats en partant de la création des données synthétiques jusqu'aux algorithmes. Cet objectif a aussi été atteint et le nécessaire se trouve à la section 2.5 et au tableau 3.2. Au moment de publier ce mémoire, cet ouvrage est le seul dans le domaine qui présente la totalité des détails nécessaires pour reproduire les résultats.

En conclusion, le modèle le plus performant est le modèle hybride constitué d'une FA généraliste et d'une MVS linéaire experte. Ce modèle a une précision moyenne de 96,36%. Ensuite, les modèles généralistes sont classés dans l'ordre décroissant de précision qui suit : MVS linéaire, RNA, FA, AD et MVS gaussiennes.

Dans le cadre de cette recherche, plusieurs avenues de recherche intéressantes restent à explorer, mais, pour les découvrir, il était nécessaire en premier d'avoir des modèles d'IA fonctionnels pour la RMCC. Par exemple, en pratique, le bruit statistique d'un procédé est rarement gaussien et il est difficile de comprendre comment réagiraient nos modèles vis-à-vis cette réalité. Montgomery (2005) suggère que la précision diminuerait. Existe-t-il des moyens pour rendre un algorithme invariant à la nature du bruit statistique de séries temporelles dans le cadre de la RMCC ? À ce jour, il n'est pas possible de le savoir.

Une autre avenue de recherche intéressante est la combinaison de l'apprentissage ensembliste et de l'optimisation. Cette recherche montre que les systèmes hybrides – une forme d'apprentissage ensembliste – sont plus performants que les systèmes singuliers. Il a aussi été démontré que les FA, qui sont un ensemble d'AD, sont d'excellents candidats pour la RMCC. Il serait intéressant de voir la précision maximale atteignable avec différentes techniques d'optimisation comme l'algorithme AdaBoost, les algorithmes génétiques, la simulation d'essaims ou l'optimisation par colonies de fourmis par exemple. Cette recherche n'a pas utilisé d'algorithme d'optimisation simplement pour isoler la contribution des algorithmes en soi.

Finalement, aucune technique de prétraitement des données ou d'optimisation n'a été utilisée. Il serait intéressant de voir leur impact sur les modèles. Il est prévu, à l'avenir, d'évaluer les modèles présentés avec des données issues d'un vrai procédé manufacturier, ce qui est un des plus grands défis dans le domaine selon Hahicha et Ghorbel (2012).

APPENDICES

La majorité du code généré dans le cadre de cet ouvrage est programmé en Python 3.6 et 3.7. La librairie Matplotlib 3.0.2 a été utilisée pour générer les graphiques à l'exception de la figure 2.19 qui a été créée avec la boîte à outils Machine Learning de MATLAB R2017b 64-bit. Les algorithmes d'apprentissage automatique proviennent des librairies Scipy 1.2.0, Scikit-learn 0.20.2 et Keras 2.2.4.

Le code est mis à la disposition de tous dans le dépôt Git de l'adresse suivante : https://github.com/iangagn/code_maitrise. N'hésitez pas à communiquer avec l'auteur par courriel à l'adresse suivante : iannick.gagnon.1@ens.etsmtl.ca.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Alcock, J., & Manopoulos, Y. (1999, août*). Time-Series Similarity Queries Employing a Feature-Based Approach*. Communication présentée à 7th Hellenic Conference on Informatics, Ioannina, Grèce, 27-29.

Alexander, S. M. (1986). The application of expert systems to manufacturing process control. *Computers & Industrial Engineering*, 12(4), 307-314.

Anscombe, F. J. (1973). Graphs in Statistical Analysis, *The American Statistician*, 27(1), 17-21.

Bag, M., Gauri, S., & Chakraborty, S. (2012). An expert control system for control chart pattern recognition. *International Journal of Advanced Manufacturing Technology*, 62, 291-301.

Basili, V. R., Selby, R. W., & Hutchens D. H. (1986). Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*. 12(7), 933-743.

Beale, R., & Jackson, T. (1998). *Neural Computing : an introduction*. Bristol, Royaume-Uni : Institute of Physics Publishing, 264 p.

Boser, B. E., Guyon, I. M, & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers, *Proceedings of the fifth annual workshop on computational learning theory*, 1, 144-152.

Cheng, C. S. (1997). A neural network approach for the analysis of control chart patterns. *International Journal of Production Research*, 35(3), 667-697.

Cheng, C. S., & Cheng, H. P. (2011). Using Neural Networks to Detect the Bivariate Process Varaince Shifts Pattern, *Computers and Industrial Engineering*, 60(2), 269-278.

Cortes, C., & Vapnik, V. N. (1995), Support-Vector Networks, *Machine Learning*, 20(3), 273-297.

Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function, *Mathematics of Control, Signals, and Systems*, 2, 313-314.

Efron B., & Tibshirani (1994). *An Introduction to the Bootstrap* (1ère éd.). Boca Raton, Florida : CRC Press., 456 p.

Fulcher, D. B. (2017). Feature-Based Time Series Analysis. Dans Dong, G., & Huan, L. (Éds), *Feature Engineering for Machine Learning Data Analytics.* (pp. 87-109). Boca Raton, Florida : CRC Press.

Gauri, S. K., & Chakraborty, S. (2006). Feature-based recognition of control chart patterns. *Computers and Industrial Engineering*, 51(4), 726-742.

Gauri, S. K., & Chakraborty, S. (2007). A study on the various features for effective control chart pattern recognition. *International Journal of Avvanced ManufacturingTechnology*, 34(3-4), 385-398.

Gauri, S. K., & Chakraborty, S. (2009). Recognition of control chart patterns using improved selection of features. *Computers and Industrial Engineering*. 56(4), 1577-1588.

Guh, R. S., & Hsieh, Y. C. (1999). A neural network based model for abnormal pattern recognition of control charts. *Computers and Industrial Engineerint*, 36(1), 97-108.

Hastie, T., James, G., Tibshirani, R., & Witten, D. (2017). *An Introduction to Statistical Learning with Applications in R*. New York, NY : Springer.

Haykin, S. (1999). *Neural Networks : A Comprehensive Foundation* (2e éd.). Upper Saddle River, New Jersey : Prentice Hall, 842 p.

Ho, T. K. (1995), Random decision forests, *Proceedings of the Third International Conference on Document Analysis and Recognition*, vol. 1, 278 p.

Hornik. K. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5), 359-366.

Jang, K. Y., Yang, K., & Zeng, Z. (2003). Application of artificial neural network to identify non-random variation patterns on the run chart in automotive assembly process. *International Journal of Production Research*, 41(6), 1239-1254.

Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Proceedings of the Fourteenth Internation Joint conference on Artificial Intelligence*, vol. 2, 1137-1143.

Laosiritaworn, W., & Bunjongjit, T. T. (2012). Classification Techniques for Control Chart Pattern Recognition : A Case of Metal Frame for Actuator Production. *Chiang Mai Journal of Science*. 40, 701-712.

Le Cun, Y. (1985). A learning scheme for asymmetric threshold networks, *Cognitiva 85*, 1, 599-604.

Masood, I. & Hassan, A. (2010). Issues in Development of Artificial Neural Network-Based Control Chart Recognition Schemes. *European Journal of Scientific Research*, 39, 336-355.

Metz, C. E. (1978). Basic Principles of ROC Analysis, *Seminars in Nuclear Medicine*, 8(4), 283-298.

Montgomery, D. C. (2005). *Introduction to Statistical Quality Control* (5e éd.). Hoboken, NJ : John Wiley & Sons, 776 p.

Othman, Z., & Eshames, H. F. (2012). Abnormal patterns detection in control charts using classification techniques. *International Journal of Advancements in Computing Technology*, 4(10), 61-70.

Parker, D. B. (1985). *Learning-logic* (Rapport technique n° 47). Cambridge, Massachussets : Sloan School of Management, MIT.

Pham, D. T., & Oztemel, E. (1993). Control Chart Pattern Recognition Using Combinations of Multi-Layer Perceptrons and Learning-Vector-Quantization Neural Networks, *Proceedings of the Institution of Mechanical Engineers*, 207(29), 113-118.

Pham, D. T., & Wani, M. A. (1997). Feature-based control chart pattern recognition. *International Journal of Production Research*, 35(7), 1875-1890.

Pugh, G. A. (1989). Synthetic neural networks for process control. *Computers and Industrial Engineering*, 17(1-4), 24-26.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, vol. 1, 81-106.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks.* Cambridge, Royaume-Uni : Cambridge University Press, 416 p.

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386-408.

Shao, X. (2012). Recognition of Control chart Patterns Using Decision Tree of Multi-class SVM, Dans Janusz, K. (Éd.), *Advances in Intelligent and Soft Computing* (pp. 33-41). vol. 138, New York, NY : Springer.

Shewhart, W. A. (2015). *Economic Control of Quality of Manufactured Product.* New York : Van Nostrand.

Shewhart, M. (1991). Application of machine learning and expert systems to Statistical Process Control (SPC) chart interpretation. *Second CLIPS Conference Proceedings*, vol. 1, 123-138.

Su, J., & Zhang, H. (2006). A Fast Decision Tree Learning Algorithm. *Proceedings of the 21st national conference on artificial intelligence*, vol. 1, 500-505.

Sukthomya, W., & Tannock, J. (2005). The training of neural networks to model manufacturing processes, *Journal of Intelligent Manufacturing*, 16(1), 39-51.

Wang, C. H., Guo, R. S., Chiang, M. H., & Wong, J. Y. (2008). Decision tree based control chart pattern recognition. *International Journal of Production Research*, 46(17), 4889-4901.

Werbos, P. (1982). Applications of Advances in Nonlinear Sensitivity Analysis. Dans Drenick, R. F., & Kozin, F. (Éds), *System Modeling and Optimization: 10th IFIP Conference, New York City, États-Unis, Août 31 - Septembre 4, 1981, Lecture notes in Control and Information Sciences* (pp. 762-770). New York, États-Unis : Springer-Verlag.

Western Electric Company (1982). *Statistical Quality Control Handbook.* Indiana : Western Electric Co. Inc, 328 p.

Wu, B., & Yu, J. (2010). A neural network ensemble model for on-line monitoring of process mean and variance shifts in corellated processes, *Expert Systems with Applications*, 37(6), 4058-4065.

Zhao, C., Wang, C., Hua, L, Liu, X., Zhang, Y., & Hu, H. (2017). Recognition of Control Chart Pattern Using Improved Supervised Locally Linear Embedding and Support Vector Machine, *Procedia Engineering*, vol. 174, 281-288.