Remerciement

Le travail présenté dans ce rapport de thèse est le fruit de la collaboration de plusieurs personnes que je tiens à remercier vivement pour m'avoir accompagné, soutenu et guidé le long de ma vie d'étude.

Je voudrais tout d'abord exprimer tous mes respects et remerciements à mon directeur de thèse, Monsieur ETTAOUIL Mohamed, Professeur à la Faculté des Sciences et Techniques de Fès, pour m'avoir encadré depuis mon master en m'offrant le pas initial de mon cursus de recherche. Sa disponibilité, sa patience, ses précieux conseils et ses encouragements m'ont permis de préparer ma thèse dans de bonnes conditions. Enfin, j'ai été extrêmement sensible à ses précieuses qualités humaines et son comportement de père plus que professeur qui m'ont offerts la chance de travailler dans un environnement familial.

Je tiens à remercier mon co-directeur de thèse, Monsieur MASSRAR Mohammed, Professeur à la Faculté des Sciences et Techniques de Fès, pour m'avoir conseillé, encouragé et examiné le côté technique de mes travaux. Je lui exprime ma gratitude pour ses discussions, ses précieux conseils et avis, ainsi que ses remarques qui m'ont inspiré d'autres idées pour des prochains travaux.

Mes remerciements les plus respectueux s'adressent à Monsieur ELHILALI Alaoui Ahmed, Professeur à la Faculté des Sciences et Techniques de Fès, pour l'honneur qu'il m'a fait en acceptant d'être président de mon jury de thèse. Je ne raterai pas cette occasion sans le remercier pour ses encouragements, ses compliments et son accompagnement durant mes années d'études à la faculté de sciences et techniques.

J'exprime ma profonde gratitude à Madame EZZAKI Fatima, Professeur à la Faculté des Sciences et Techniques de Fès et Directrice du Laboratoire Modélisation et Calcul Scientifique, pour le temps précieux qu'il m'a accordé en acceptant d'être rapporteur de ce travail malgré ses nombreuses occupations. Je la remercie pour toute l'attention qu'il a portée à ce travail et les efforts qu'elle fournit pour assurer un environnement confortable de travail au sein du laboratoire.

Mes remerciements sincères vont à Monsieur BAHAJ, Professeur à la Faculté des Sciences et techniques de Settat, pour l'intérêt qu'il a porté à l'égard de cette recherche en s'engageant à être rapporteur. Je le remercie aussi pour le temps qu'il a consacré et pour son déplacement

afin de juger mon travail.

Je tiens à remercier également Monsieur ELAFIA Abdelatif, Professeur à l'ENSIAS de Rabat, de m'avoir fait le privilège de rapporter mon manuscrit et de se déplacer pour juger mon travail de thèse. Je le remercie également pour sa sympathie, ses précieux conseils et ses encouragements qui m'ont marqués.

J'adresse un sincère merci à Monsieur GHANOU Youssef, Professeur à l'école supérieure de technologie de Meknès, qui m'a fait l'honneur en acceptant de juger mon travail de recherche en tant qu'examinateur. Je le remercie fortement pour ses encouragements, son soutien, et ses conseils durant mon cursus de recherche.

Je tiens à remercier l'ensemble des enseignants de la Faculté des Sciences et Techniques de Fès, grâce à eux, j'ai acquis plusieurs connaissances.

Je tiens à remercier également Monsieur le Doyen, Monsieur le Vice-Doyen et tout le personnel de la Faculté des Sciences et Techniques de Fès qui ont aidé à la réalisation de cette thèse.

Un remerciement spécial à tous mes collègues docteurs et doctorants du Laboratoire Modélisation et Calcul Scientifique, pour leur sympathie, leurs gentillesses, leurs soutien et leurs discussions enrichissantes le long des années que nous avons passé ensemble.

Je n'aurais jamais pu réaliser ce travail doctoral sans le grand soutien moral et matériel de mes parents. Je leurs adresse mes grands remerciements de m'avoir offert la liberté de choix de mes études, pour la confiance qu'ils m'ont accordée, et pour les sacrifices qu'ils ont consentis pour ma formation et mon bien-être. Le mot merci ne peut pas exprimer à quel point je vous suis reconnaissante, mais j'espère que le dédicace de ce travail vous rendra fière de votre fille.

Mes remerciements chaleureux s'adressent à mes sœurs, cousins, oncles, tantes et à mes frères et sœurs que les expériences de la vie m'ont offertes pour leur grand amour, leur intérêt à ce que je fais et pour leur soutien de prêt et de loin. J'espère qu'ils trouvent dans ce travail l'expression de mon profond amour, mes vœux de succès, de réussite et de bonheur dans leur vie.

Avec une grande joie et gaieté je remercie toute personne que j'ai rencontrée dans ma vie et qui a contribué, directement ou indirectement, à la réalisation de cette thèse. Très heureuse d'avoir partagé avec vous des moments soit agréables ou désagréables qui m'ont beaucoup appris.

Avant Propos

Dans ce rapport de thèse, nous présentons le fruit de nos travaux de recherche effectués au sein du laboratoire de modélisation et calcul scientifique de la faculté de sciences et techniques de Fès, université SIDI MOHAMED BEN ABDELLAH. Ce travail a fait l'objet de publications dans des journaux internationaux et communications à des congrès nationaux et internationaux. Ces travaux ont été élaborés sous la direction de monsieur ETTAOUIL MOHAMED professeur de l'enseignement supérieur à la FST de Fès et membre du centre d'études doctorales : Calcul Scientifique et Informatique, Science de l'ingénieur.

Dans ce rapport j'ai essayé de traiter toutes les notions nécessaires à l'aboutissement aux résultats trouvés durant les années de recherche, en espérant qu'il puisse être une référence bien structurée et bien enrichie pour les étudiants qui voudront chercher dans le domaine de l'apprentissage automatique, les réseaux de neurones artificiels et la classification. Aussi, j'ai veillé présenter de nouveaux axes de recherches dans le domaine des réseaux neuronaux pour ouvrir la porte devant les nouveaux chercheurs.

Table des matières

Liste des Symboles **5**

Liste des Figures **7**

Liste des Tableaux **10**

Liste des Algorithmes **12**

Introduction Générale **13**

Chapitre - 1 Apprentissage artificiel et Classification

1.1 Introduction 20
1.2 Apprentissage Artificiel 21
1.2.1 Généralités 21
1.2.2 Modes d'apprentissage 22
1.3 Introduction à la classification 24
1.3.1 Notion de partition 25
1.3.2 Qualité de la partition et nombre de classes 25
1.3.3 Types de classification 30
1.4 Classifieurs de partitionnement 33
1.4.1 Généralités 33
1.4.2 Méthode K-moyennes 34
1.4.3 Méthode ISODATA 34

1.4.4 Nuées dynamiques 34
1.4.5 Méthodes k-modes, k-prototypes 34
1.5 Classifieurs neuronaux 35
1.5.1 Généralités sur les réseaux de neurones artificiels 35
1.5.2 Réseaux multi-couches 37
1.5.3 Cartes topologiques 40
1.6 Classifieurs probabilistes 51
1.6.1 Modèle de mélange 51
1.6.2 Algorithme EM 52
1.6.3 Cartes topologiques probabilistes 53
1.6.4 Réseaux neuronaux probabilistes 54
1.7 Discussion 57
1.8 Conclusion 59

Chapitre - 2 Sélection d'architecture neuronale du PrSOM pour la construction du code-book optimal du signal des chiffres arabes

2.1 Introduction 60
2.2 Compression de la parole 61
2.3 Quantification vectorielle 62
2.3.1 Détermination du dictionnaire optimal 63
2.3.2 Quantification vectorielle par la carte topologique 64
2.4 Cartes topologiques probabilistes 65
2.4.1 Architecture de la carte probabiliste 65
2.4.2 Fondement probabiliste 66
2.4.3 Estimation des paramètres 67
2.5 Problématique 69
2.6 Description de l'approche H-PrSOM 70
2.6.1 Phase de sélection de paramètres 71
2.6.2 Phase d'apprentissage 72
2.6.3 Phase d'optimisation 72
2.7 Résultats expérimentaux 73
2.8 Conclusion 76

Chapitre - 3 Contribution à la sélection des modèles neuronaux de type PMC

3.1 Introduction 78
3.2 Réseaux Multi-couches 79
3.2.1 Architecture du Perceptron Multi couches 79
3.2.2 Fonction de transfert 82
3.2.3 Modélisation mathématique 82
3.2.4 Apprentissage du Perceptron Multi couches 84
3.2.5 Pouvoir et faiblesse du Perceptron Multi couches 86
3.3 Problématique du choix de l'architecture 87
3.4 Description de la nouvelle modélisation mathématique 90
3.4.1 Variables du modèle proposé 91
3.4.2 Fonction objectif 91
3.4.3 Contraintes du modèle 92
3.5 Approche de résolution par les Algorithmes génétiques 94
3.5.1 Généralités sur les Algorithmes génétiques 94
3.5.2 Adaptation des Algorithmes génétiques au modèle proposé 96
3.6 Résultats expérimentaux 100
3.7 Conclusion 104

Chapitre - 4 Contribution à la classification automatique par le réseau PMC et une nouvelle version de k-moyennes

4.1 Introduction 105
4.2 Méthode k-moyennes 106
4.2.1 Fondement mathématique de k-means 106
4.2.2 Phases d'apprentissage 107
4.3 Problématique 108
4.4 État de l'art 111
4.5 Description de l'hybridation proposée 112
4.5.1 Phase de sélection de paramètres 113
4.5.2 Phase d'étiquetage de données 114
4.5.3 Phase d'apprentissage 115

4.6 Résultats expérimentaux 116
4.6.1 Évaluation de l'approche de clustering proposée 116
4.6.2 Évaluation de l'hybridation proposée 117
4.7 Conclusion 123

Chapitre - 5 Nouvelle méthode de clustering pour la Sélection de caractéristiques pertinentes

5.1 Introduction 124
5.2 Réduction de la dimension 125
5.3 Extraction de caractéristiques 126
5.3.1 Méthodes linéaires 127
5.3.2 Méthodes non linéaires 127
5.4 Sélection de variables 128
5.4.1 Méthodes à base d'apprentissage connexionniste 129
5.4.2 Méthodes à base de modèles d'optimisation 129
5.4.3 Méthodes à base de classification 130
5.5 Approche proposée pour la réduction de la dimensionnalité 130
5.5.1 Approche de clustering 131
5.5.2 Étape de réduction de la dimension 133
5.6 Validation de la méthode proposée 134
5.7 Conclusion 137

Conclusion Générale **138**

Annexes **142**

Annexe A - Description des Instances utilisées pour la validation des méthodes proposées **142**

Annexe B - Autres critères d'évaluation de classification **144**

Bibliographie **159**

Liste des symboles

AA	Apprentissage Artificiel
ACP	Analyse en Composantes Principales
Adaline	Adaptative Linear Element
AG	Algorithme Génétique
BB	Branch and Bound
BD	Davies-Bouldin
BP	Back-Propagation
CAH	Classification Ascendante Hiérarchique
CCA	(Curvilinear Component Analysis
CDH	Classification Descendante Hiérarchique
CS	Classification Supervisée
ECD	Extraction de Connaissances à partir des Données
GTM	Generative Topographic Mapping
IA	Intelligence Artificielle
ISOMAP	Isometric feature Mapping
LLE	Locally Linear Embedding
Madaline	Multiple Adaline
MDS	Multi-Dimensional Scaling
NLM	NonLinear Mapping
OA	Outer Approximation
OPT-MLP	Perceptron Multi-Couches Optimale

PMC	Perceptron Multi-Couches
PNN	Probabilistique Neural Networks
PRSOM	PRobabilistic Self Organizing Map
QV	Quantification Vectorielle
RD	Réduction de la Dimensionnalité
Rdf	Reconnaissance De Formes
RNAs	Réseaux de Neurones Artificiels
SCL	Simple Competitive Learning
SOM	Self Organizing Map

Liste des Figures

1 Schéma d'un processus d'extraction de connaissances à partir des données [98] 14

2 Organigramme de lecture possible de cette thèse 19

1.1 Différents types de classification 29

1.2 Différents types de classification 30

1.3 catégorisation des différents méthodes de classification non supervisée 31

1.4 Catégorisation des types de RNAs 36

1.5 Modélisation du neurone biologique sous forme d'un Neurone formel 37

1.6 Perceptron simple 38

1.7 Réseau Adaline 38

1.8 Connection de plusieurs adalines pour arriver à résoudre des problèmes non linéaires 39

1.9 Illustration de l'importance des couches cachées dans les réseaux à couches . 40

1.10 Architecture du réseau de Kohonen 42

1.11 Fonction de voisinage (figure tirée de [28]) 43

1.12 Autres types de voisinage 43

1.13 Illustration de l'influence du paramètre T sur l'ordre induit par la carte topologique pour deux décroissances différentes de T et une même initialisation aléatoire au centre du nuage et un même intervalle de croissance (figure tirée de [63]) 46

1.14 Schéma de l'algorithme d'apprentissage du réseau de Kohonen 48

1.15 Illustration du déploiement de la carte pour s'approcher de l'espace d'entrée : Déplacement du vecteur gagnant et ces voisins vers l'entrée présentée 49

1.16 Illustration du déroulement ($A \longrightarrow B \longrightarrow C$) de l'auto-organisation dans la carte de Kohonen 50

1.17 Architecture générale d'un réseau probabiliste 56

2.1 Schéma général illustrant le principe de quantification vectorielle 62
2.2 Modélisation de la duplication de la carte auto-organisatrice sous forme d'un modèle de mélange de densités gaussiennes 66
2.3 Exemple illustratif des types de neurones de la carte obtenus après apprentissage [74] 70
2.4 Exemple illustratif de la technique du choix des centres initiaux 71
2.5 Etapes principales de l'approche proposée 71
2.6 Visualisation du signal original du chiffre zéro et son signal décompressé après compression par notre approche 76

3.1 Représentation graphique du réseau multi-couches général. Dans ce graphisme, l'information se propage de la gauche vers la droite 80
3.2 Calcul des sorties de la première couche cachée 80
3.3 Calcul des sorties du réseau 81
3.4 Calcul des sorties de la j^{me} couche cachée 81
3.5 Description de l'activation et la sotie d'un neurone 87
3.6 Illustration des deux problèmes d'apprentissage 89
3.7 Illustration du problème de sur apprentissage 89
3.8 Différent étapes d'un algorithme génétique 95
3.9 Example illustrant la modification d'un gène pour la mutation 96
3.10 Exemple du codage adapté pour l'approche OPT-MLP d'un individu de la population 98
3.11 Exemple illustratif de la manière avec laquelle deux parents de la population sont croisés à l'aide du croisement en un seul point mais différent pour chaque chromosomes 99
3.12 Description de la technique utilisée pour la mutation d'un individu de la population 100

4.1 Diagramme de Voronoi (tiré de [139]) : définition d'une partition telle que chaque sous ensemble de la partition est associé à l'un des référents 107
4.2 : Illustration des étapes de la méthode de k-moyennes de l'étape (a) d'initialisation des centres des classes, à l'étape (c), de convergence en passant par des itérations de mises à jour et de réaffectation des données 108
4.3 Schéma de l'algorithme d'apprentissage de la méthode de k-moyennes 109
4.4 Exemple (inspiré de [122]) d'illustration du problème d'initialisation de l'algorithme de k-moyennes 110
4.5 Etapes principales de l'approche de clustering 113

4.6 Diagramme de la méthode H-kmeans 115
4.7 Schéma illustratif des étapes de la l'approche globale 115
4.8 Variation de R^2 en fonction du nombre de classes pour la base IRIS 117
4.9 Variation de R^2 en fonction du nombre de classes pour la base WINE 118
4.10 Variation de R^2 en fonction du nombre de classes pour la base SOYBEAN . 118
4.11 Variation de R^2 en fonction du nombre de classes pour la base HAYES-ROTh 119
4.12 Variation de R^2 en fonction du nombre de classes pour la base GLASS . . . 120

5.1 Processus général de la technique d'extraction de caractéristiques (reprise de [106]) 126
5.2 Principe général de la technique de sélection de variables (reprise de [106]) . 129
5.3 Technique de construction de la nouvelle base de données 131
5.4 Technique de détermination de l'interval de variation de ϵ 132
5.5 Schéma général de la méthode de clustering proposée 134
5.6 Illustration schématique de la méthode proposée 135

Liste des tableaux

2.1 Résultats de compression des dix chiffres arabes : architectures définies automatiquement par l'approche proposée et leurs PSNR et MSE associés 74

2.2 Génération arbitraire de plusieurs architectures de la carte pour le chiffre zéro 75

3.1 Caractéristiques des bases de données utilisées pour la validation de notre modèle 100

3.2 L'architecture optimale obtenue par OPT-MLP pour les cinq bases de données 101

3.3 Résultats de l'approche OPT-MLP en comparaison avec l'approche AOM pour la base de données IRIS 102

3.4 Résultats obtenus par l'approche OPT-MLP en comparaison avec six méthodes d'optimisation de l'architecture pour la base IRIS 102

3.5 Résultats de classification obtenus par l'approche OPT-MLP en comparaison avec douze classifieurs 103

4.1 Description des données utilisées pour la validation de notre approche 116

4.2 Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de IRIS 117

4.3 Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de WINE 118

4.4 Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de SOYBEAN 119

4.5 Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de HAYES-ROTh 119

4.6 Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de GLASS 120

4.7 Résultats de classification des données selon la variation du seuil de similarité 121

4.8 Comparaison de l'approche proposée avec des méthodes neuronales et des méthodes de clustering 122

5.1 Description de quelques information sur les bases de données Benchmarks utilisées 135

5.2 Résultats de clustering avant et après la réduction de la dimensionnalité . . . 136

5.3 Résultats de réduction obtenus par l'approche proposée 136

Liste des Algorithmes

1 Algorithme de Kohonen 47
2 Algorithme EM 53
3 Algorithme PRSOM avec T constant 69
4 Algorithme de rétro-propagation 88
5 Algorithme de k-moyennes 109

Introduction Générale

Contexte général

Cette thèse s'inscrit dans le cadre de l'Extraction de Connaissances à partir des Données (ECD) à l'aide de l'apprentissage automatique et la classification. Cette discipline, appelée aussi communément "fouille de données" ou "data-mining", a pour objectif la découverte d'informations implicites pour l'extraction d'un savoir à partir de données par des méthodes automatiques ou semi-automatiques. L'ECD est la discipline la plus répandue dans les entreprises soucieuses d'extraire l'information pertinente dissimulée dans leurs bases de données, en vue d'améliorer leurs processus et gérer leur relation client et leur maîtrise de risques. Le processus d'extraction d'information utile passe par plusieurs étapes (figure 2) : l'acquisition de données multiformes (textes, images, séquences vidéos, etc.), la préparation de données (pré-traitement), la fouille de données, et enfin la validation et la mise en forme des connaissances.

Cependant, le processus d'ECD ne se passe pas très facile. Le passage des données aux connaissances est porteur de nombreux défis scientifiques et technologiques qui requièrent une démarche interdisciplinaire. Au cours des deux dernières décennies, l'humanité a créé plus d'informations que pendant toute son histoire. Cette réalité désigne notre âge sous le nom de l'âge d'information en croyant que l'information est le secret du succès et de la puissance. Grâce à l'évolution de l'informatique (apparition du Web 2.0, l'utilisateur est le leader de l'Internet) et des technologies sophistiquées, telles que les ordinateurs, les satellites, etc., nous avions rassemblé des quantités ainsi que des ressources de l'information énormes. Grâce aux performances accrues des ordinateurs, les moyens de la mémoire numérique de masse, la création de bases de données et des systèmes de gestion structurés de base de données (les SGBD), il est possible d'analyser des corpus volumineux de données particulièrement pour la recherche pertinente et efficace d'information particulière dans une grande collection. Néanmoins, aujourd'hui, les informations collectées engendrent des masses de données de plus en plus volumineuses, hétérogènes, imprécises, incertaines, incomplètes, bruitées, éparpillées, etc. De ce fait, la prise de décision critique dans un temps limité et l'examen de tous les documents contenant l'information souhaitée dans un temps raisonnable provoquent

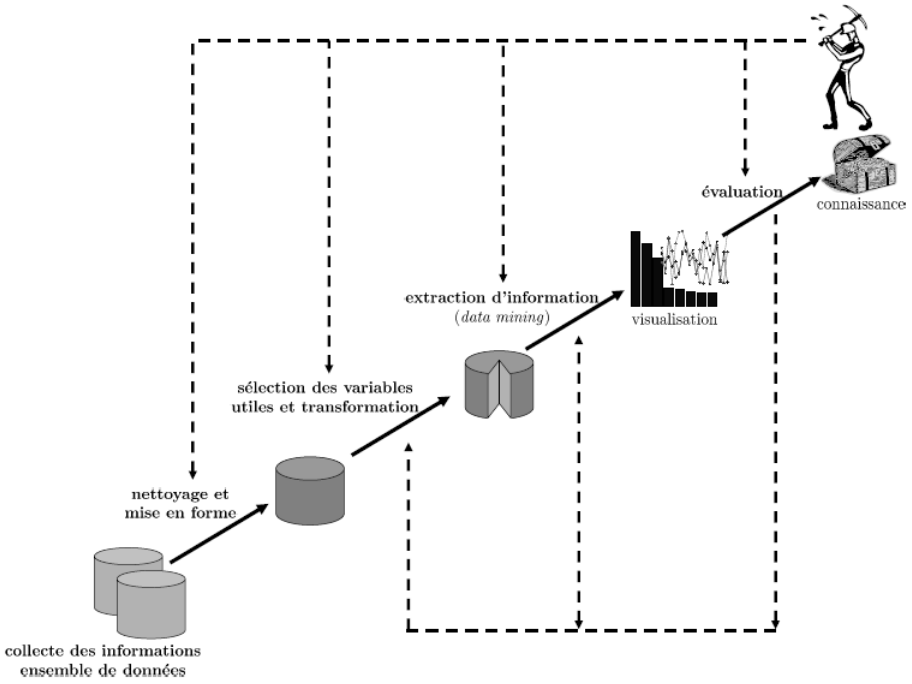

Figure 1 – Schéma d'un processus d'extraction de connaissances à partir des données [98]

un scénario difficile. Il a ainsi fallu développer des outils permettant une prédiction et appréhension rapide de l'information contenue dans de "grands tableaux de données". C'est l'objet de l'analyse, apprentissage artificiel et classification de données.

Motivation et Problématique

La classification non-supervisée est une tâche primordiale en apprentissage automatique et data mining. Elle consiste à représenter la meilleure structure induite de la distribution d'un ensemble de données non étiquetées. Dans ce cadre, son principe revient à regrouper les données en clusters (collection d'enregistrements similaires l'un à l'autre, et différents de ceux existants dans les autres clusters) en respectant deux propriétés intéressantes : la cohésion et la séparation. En effet, le clustering vise à maximiser l'homogénéité (similarité) à l'intérieur de chaque groupe (cohésion) et la minimiser entre les différents groupes (séparation).

Quant à la classification supervisée, elle présente à son tour la tâche la plus commune de la fouille de données qui semble être une tâche humaine primordiale. Elle consiste à apprendre à partir d'un jeu d'apprentissage pour construire un modèle et puis employer celui-ci pour classifier de nouveaux objets qui n'ont pas participé à l'apprentissage.

Les algorithmes de classification supervisée et non supervisée sont particulièrement nombreux ; Les réseaux de neurones se montrent très efficaces particulièrement pour des

problèmes de classification et de prédiction. Leur principale caractéristique est l'aspect non linéaire de leur réponse. Ainsi, ils sont une alternative prometteuse aux techniques plus traditionnelles pour des tâches de prédiction temporelle non linéaire. Ils sont en relation plus ou moins étroite avec la modélisation de processus cognitifs (capable de connaître ou faire connaître) comme une alternative à l'intelligence artificielle.

Cependant, l'estimation et l'identification des modèles neuronaux utilisent des techniques sophistiquées et il n'est pas facile de déterminer leur architecture adéquate. En effet, ces modèles sont par définition sur-paramétrés et les fonctions d'erreur à minimiser ont de nombreux minima locaux. Aussi, la mise en œuvre s'avère souvent délicate. Par exemple, le nombre de neurones par couche cachée est particulièrement important car il détermine la capacité de calcul du réseau. Un nombre insuffisant de neurones cachés peut compromettre la capacité du réseau à résoudre le problème (sous apprentissage). Inversement, un nombre élevé de neurones force le réseau à apprendre par cœur (sur-apprentissage).

L'algorithme de k-means présente une autre méthode dite de partitionnement qui utilise l'optimisation alternée pour chercher la partition optimale de l'ensemble des données. C'est une méthode très connue et très utilisée pour la classification non supervisée. Il peut être montré que l'algorithme de cette méthode converge en un nombre fini d'opérations. Cependant la convergence est locale, ce qui pose le grand problème de l'initialisation des paramètres de la méthode. La difficulté réside non seulement dans le choix de ces paramètres pour avoir la bonne partition, mais également dans le nombre de classes qui doit rester relativement faible. En fouille de données, l'utilisateur n'a en général, aucune connaissance a priori des données et ne peut pas connaître le nombre idéal de classes.

Toutes ces problématiques nous ont poussés à chercher une issue pour les problèmes d'estimation de paramètres des architectures neuronaux et à déterminer le nombre de classes.

Objectifs et contributions

En s'intéressant particulièrement à la classification non supervisée et aux réseaux connexionistes, les travaux présentés dans cette thèse s'articulent autour de trois objectifs principaux :

- Proposer de nouvelles méthodes de classification non supervisée capables d'estimer les paramètres.
- Déterminer le nombre de classes d'un ensemble de données.
- Proposer des modèles mathématiques et heuristiques pour la sélection des architectures neuronales.

De ce fait, Nous développons, dans cette thèse, quatre approches :

En s'intéressant à la sélection de modèles neuronaux probabilistes, notre première contribution propose une nouvelle méthode pour la sélection de l'architecture neuronale de la carte topologique probabiliste (PrSOM). Nous développons une nouvelle méthode

hiérarchique qui opère sur la structure de données pour estimer un nombre de neurones adéquat ainsi qu'une bonne initialisation des poids de la carte probabiliste. Dès lors, Cette nouvelle méthode accroit la performance et génère considérablement un nombre de neurones adéquat pour éviter le grand problème de sur-apprentissage. Dans un autre côté, les cartes topologiques, particulièrement dont la version est probabiliste, sont connues par leur efficacité pour la quantification vectorielle. Nous proposons d'utiliser notre nouvelle méthode, baptisée H-PRSOM, afin de déterminer le dictionnaire optimal pour la compression de la parole. L'approche H-PrSOM, fondée sur l'heuristique divisive et l'algorithme PrSOM, a montré sa reproductibilité lors des expérimentations sur la base du signal des chiffres arabes.

Dans la deuxième approche, nous présentons une nouvelle modélisation mathématique du problème de choix du nombre de couches et neurones cachés du perceptron multi-couches (PMC) ainsi que la stabilisation des poids. La modélisation proposée consiste à associer à chaque couche et chaque neurone une variable binaire qui décide sa présence ou absence dans le réseau. A ce dernier, une nouvelle fonction coût est associée pour mesurer son erreur empirique. En se basant sur la résolution de la modélisation proposée par l'optimisation évolutionnaire des algorithmes génétiques, nous cherchons le réseau optimal parmi plusieurs réseaux générés dans la population initiale. La comparaison de la nouvelle méthode à celles existantes dans la littérature a permis de découvrir de constater sa capacité à sélectionner le réseau optimal et performant (donnons une bonne généralisation). Ceci améliore, par conséquent, le compromis optimalité/efficacité. Ces avantages sont prouvés lors des tests expérimentaux sur des bases de données benchmarks.

Restant dans le cadre de sélection de paramètres, la troisième approche prend en charge d'étiqueter les données d'apprentissages du perceptron multi-couches (PMC) sans avoir aucune information a priori sur les données. Etant un bon classifieur supervisé, nous visons à adapter le PMC à la classification automatique. Pour se faire, nous développons une nouvelle méthode de clustering, baptisée H-kmeans, en amont du perceptron multi-couches pour sélectionner les donnes d'apprentissage. L'hybridation des deux méthodes a pour objectif d'allier les points forts des deux méthodes afin d'avoir une meilleure classification. En effet, H-kmeans s'inspire du processus hiérarchique de notre première contribution et génère plusieurs partitions différentes en utilisant l'algorithme de k-means comme méthode de recherche locale de la partition optimale. A chaque itération, nous cherchons de nouveaux paramètres en étudiant la structure des données. La sélection des paramètres par H-kmeans surmonte le problème d'initialisation qui affecte gravement la qualité des résultats. Dès lors, cette nouvelle approche enrichie la gamme des méthodes de clustering. En plus, elle augmente la capacité de généralisation du PMC. Ces avantages se sont vérifiés lors d'une validation expérimentale sur des données synthétiques.

Dans la dernière contribution, Nous continuons à s'intéresser au processus d'extraction de connaissances à partir des données. En effet, nous attaquons l'étape de sélection de variables pertinentes pour la réduction de la dimensionnalité des données. Pour se faire, Nous présentons une nouvelle méthode itérative de sélection de paramètres initiaux de l'algorithme de k-means baptisée Hb-kmeans. Cette dernière consiste, à chaque itération, à générer un ensemble de boules englobant les données les plus proches selon un rayon de similarité bien

déterminé. A l'aide du nombre et des centres de ces boules, nous déterminons les paramètres initiaux pour une application locale de l'algorithme de k-means. Ainsi, nous enrichissons, une nouvelle fois, les méthodes de clustering par la méthode Hb-kmeans reposant sur le principe itératif de génération de plusieurs partitions pour la sélection de la meilleure. En outre, nous visons, dans cette contribution, à classifier les caractéristiques des données et sélectionner les plus pertinentes grâce à la méthode Hb-kmeans donnant la meilleure partition. Vue que l'information est contenue dans les classes, nous sélectionnons un représentant des caractéristiques de chaque classe de la partition optimale. Nous éliminons par suite les autres caractéristiques pour avoir des données non bruitées et non redondantes. La validation de cette nouvelle technique sur des bases de données benchmarks s'est avérée prometteuse et encourageante en effectuant les tests expérimentaux.

Organisation de la thèse

Ce manuscrit est structuré en deux parties : la première concerne l'état de l'art et la seconde décrit nos contributions.

La première partie, résumée dans un seul chapitre, introduit les différentes notions nécessaires pour entamer la deuxième partie des contributions. Nous ébauchons le chapitre 1 par le cadre général des méthodes abordées dans cette thèse : l'apprentissage artificiel. Dans la même section, nous présentons trois catégories de tâches qu'on peut accomplir avec des algorithmes d'apprentissage artificiel. La notion de classification qui représente le cœur de nos travaux de recherches est introduite dans la troisième section. Dans cette dernière, nous présentons la notion de partition, la problématique de sélection de la meilleure partition et du nombre de classes ainsi que les types de classification. Le long de trois sections, nous mettons l'accent sur trois catégories d'approches dédiées à la classification : Approches de partitionnement, neuronales et probabilistes. Nous veillons à définir le principe, objectif et fondement mathématique de chaque méthode des trois catégories. Nous laissons la description des méthodes utilisées dans nos contributions aux chapitres suivants. Avant de clôturer ce chapitre, nous présentons une discussion sur les méthodes hiérarchiques et les avantages et inconvénients des méthodes neuronales dans l'objectif d'introduire et de positionner l'apport de nos contributions au regard de l'existant.

La deuxième partie est scindée en quatre chapitres qui exposent nos contributions.

Dans le chapitre 2 de cette thèse, nous nous sommes intéressés au problème de compression de la parole. Il s'agit de déterminer le dictionnaire optimal par la nouvelle méthode H-PrSOM de la carte topologique probabiliste (PrSOM). Pour donner une vision claire sur la thématique traitée, nous présentons les notions de bases de la compression de la parole dans la deuxième section. La quantification vectorielle, étant la technique la plus utilisée pour la compression de la parole, sera détaillée dans la troisième section en décrivant son principe, les techniques de détermination du cood-book optimal et plus précisément par la carte topologique. A ce titre, nous définissons la carte topologique probabiliste, son architecture, modélisation et estimation de paramètres. Nous détaillons la nouvelle méthode de choix du

dictionnaire optimale dans la section 5 après la description de la problématique de sélection d'architecture neuronal de la carte probabiliste PrSOM. Nous finirons ce chapitre par la validation de l'approche proposée sur la base de données du signal des chiffres arabes.

Le chapitre 3, à son tour, s'intéresse à la sélection d'architectures neuronales particulièrement pour les réseaux de types PMC. Pour cela, en premier lieu, nous présentons avec soin les différentes entités qui composent un réseau multi-couches. En traitant les avantages et les inconvénients de ce type de réseau, nous expliquons la problématique que nous tentons de résoudre dans ce chapitre et nous donnons un bref état de l'art sur les catégories de méthodes proposées pour le même objectif. Nous réservons deux sections pour la description de la modélisation mathématique proposée pour la sélection d'architecture. La première est dédiée à la définition des différentes composantes du nouveau modèle mathématique alors que la deuxième est réservée à la description de la résolution du modèle via les algorithmes génétiques. Enfin, nous terminons ce chapitre par un ensemble de tests numériques pour montrer la performance de notre approche, en utilisant un ensemble de jeux de données disponible sur internet.

Le chapitre 4 aborde un nouvel axe d'estimation des paramètres des algorithmes de k-means et des données d'apprentissage du PMC via une hybridation entre la nouvelle méthode H-kmeans et le PMC. Nous commençons par exposer une description détaillée de la méthode de k-means suivie de la problématique liée au choix de paramètres de l'algorithme de k-means ainsi que les données d'apprentissage du perceptron multi-couches. Nous évoquons brièvement, dans la quatrième section, quelques versions de k-means ainsi que des travaux réalisés concernant l'hybridation du PMC avec les k-means pour la résolution de plusieurs problèmes. Les différentes étapes de la nouvelle méthode proposée sont présentées dans la cinquième section. Enfin, pour tester la performance de la méthode présentée, nous avons validé cette dernière sur des jeux de données disponibles sur internet et ainsi nous avons comparé les résultats obtenus avec ceux d'autres approches de classification.

Le dernier chapitre présente une nouvelle méthode dédiée à l'extraction de caractéristiques pertinente dans le but de réduction de la dimensionnalité. La méthode Hb-kmeans repose sur une nouvelle technique itérative pour la sélection de paramètres initiaux de k-means et la convergence vers une meilleure partition. Une introduction sur la nécessité de la réduction de dimension des variables est présentée dans la deuxième section de ce chapitre. Dans la troisième et la quatrième section, nous donnons un aperçu général sur les méthodes d'extraction de caractéristiques et de sélection de variables. Dans la cinquième section, nous décrivons les différentes étapes de la nouvelle approche proposée. Avant de conclure, nous testons la performance de la méthode à l'aide de testes expérimentaux.

La figure 2 propose une démarche de lecture de cette thèse et indique les différentes connexions entre les chapitres.

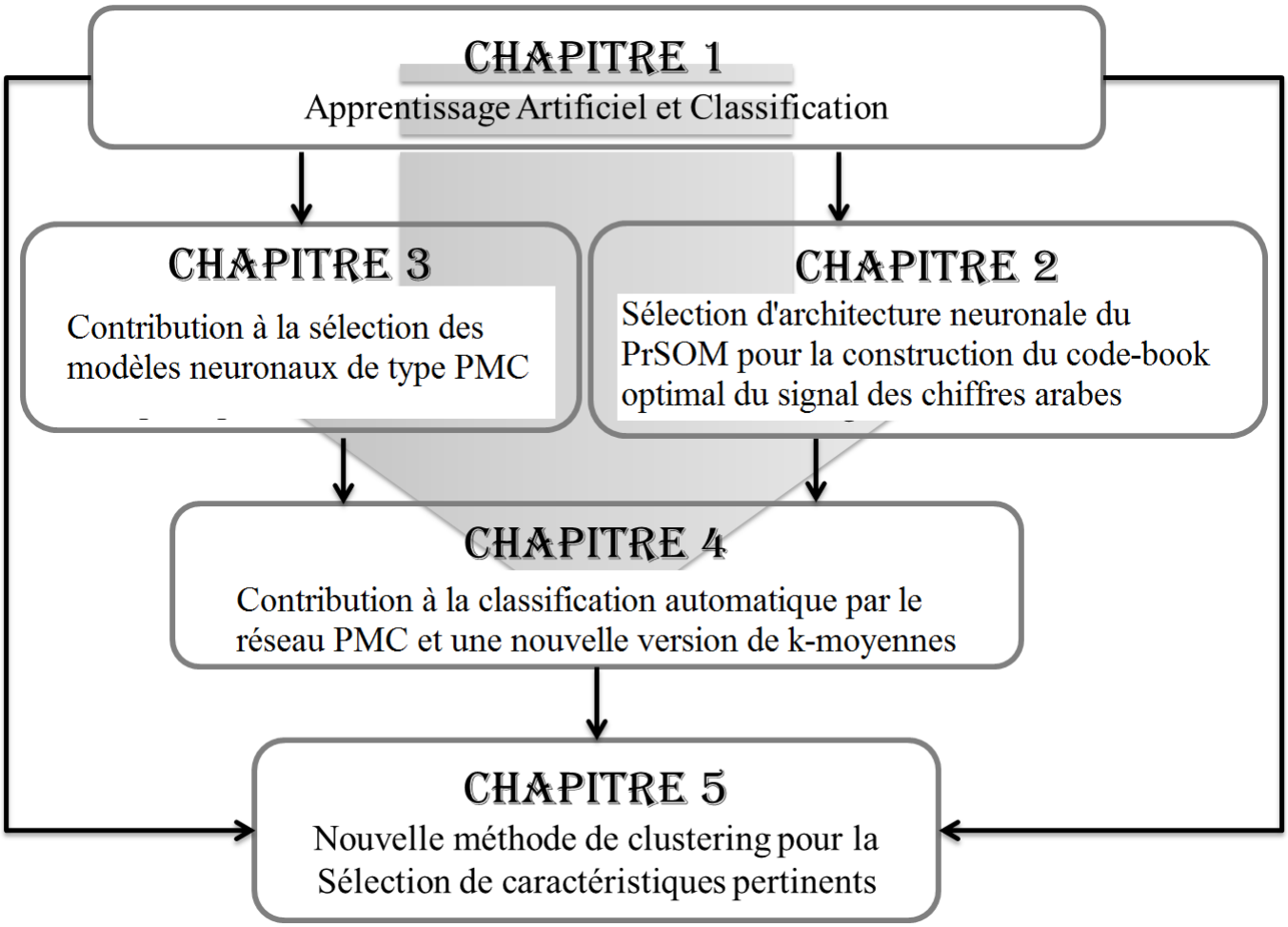

Figure 2 – Organigramme de lecture possible de cette thèse

Chapitre 1

Apprentissage artificiel et Classification

1.1 Introduction

Passionnés par le fonctionnement des créatures vivantes et leur manière d'agir, les chercheurs ont tenté d'imiter l'architecture ou le fonctionnement de plusieurs créatures. Cela a créé un grand défi pour les scientifiques qui réside dans le fait de générer des machines intelligentes exploitant une des plus grandes qualités humaines, l'apprentissage. D'un point de vue artificiel, l'apprentissage représente la capacité de généraliser les résultats d'une expérience effectuée sur une population réduite à une autre plus grande. Cette technique permet alors d'accumuler des expériences et tirer des règles qui vont être appliquées (généralisation) sur des situations non encore rencontrées. De ce fait, la qualité de généralisation est le grand dilemme de l'apprentissage. Actuellement, l'apprentissage est le produit d'une histoire de cinquante ans de recherche et de réalisations. Vue son importance, un grand nombre de tâches d'intelligence artificielle, de reconnaissance des formes (RdF) et de fouille de données sont fondées sur des modules d'apprentissage. Nos travaux de recherche s'articulent autour de ces thématiques et plus précisément les méthodes d'apprentissages pour la classification automatique et discrimination. Pour cela, le premier chapitre va permettre, dans un premier lieu, d'exposer le fondement et les particularités de l'apprentissage artificiel, tout en réalisant le lien entre les modes d'apprentissages les plus utilisés et la classification supervisée et non supervisée. Dans un second lieu, nous consacrons les autres sections de ce chapitre à une description détaillée du principe des réseaux neuronaux, probabilistes et déterministes, vue leur importance et performance comme méthodes d'apprentissage, ainsi que quelques méthodes de partitionnement pour la classification nécessaires à la résolution de nos problèmes de recherche. Puisque nous nous sommes fortement intéressés aux méthodes neuronales, ce chapitre représente aussi une occasion pour présenter les problèmes rencontrés dans le domaine de l'apprentissage avec les méthodes neuronales et les méthodes de partitionnement. Avant de conclure, nous discutons les méthodes hiérarchiques ainsi que les avantages et limites des RNAs.

1.2 Apprentissage Artificiel

1.2.1 Généralités

L'intelligence artificiel (IA) est devenue (il y a une vingtaine d'années) un champs d'investigation attirant dans le monde de la recherche pour concevoir des machines intelligentes dans presque tous les domaines. Elle permet d'utiliser plus intelligemment un ordinateur en facilitant la communication entre l'homme et la machine et en exploitant au mieux la puissance de traitement [104]. L'apprentissage artificiel (AA) se trouve à l'intersection de l'intelligence artificielle et des statistiques, et il est devenu à son tour une branche très importante des mathématiques appliquées. Elle est défini dans [48] comme une notion englobant toute méthode permettant de construire un modèle réel à partir d'un ensemble de données soit en améliorant un modèle partiel (ou moins général), soit en créant complètement le modèle.

Son objectif est de réaliser des modèles de conception, d'analyse, de développement et d'implémentation de méthodes qui apprennent par l'exemple (Acquisition de connaissances) et permettent à une machine d'évoluer par un processus systématique. Il permet ainsi de remplir des tâches difficiles ou impossibles à remplir par des moyens algorithmiques plus classiques. L'utilité de l'apprentissage artificiel augmente lorsque l'on cherche la modélisation de processus complexes qui sont caractérisés par des connaissances théoriques trop imprécises pour permettre des prédictions précises (généralisation). Vu son importance, plusieurs acteurs d'activités sollicitent l'application de l'intelligence artificielle : analyse de données, bio-informatique, télécommunications, génie des procédés, diagnostic médical, interface cerveau-machines, et bien d'autres.

La conception d'un modèle par apprentissage nécessite de définir une fonction de cout calculée à partir des exemples de la base d'apprentissage et à l'aide de variables susceptibles d'avoir une influence sur la grandeur à modéliser. Cette fonction est capable de reproduire les données existantes le mieux possible et de trouver une généralisation (prédiction du comportement de la grandeur à modéliser) pour les données qui ne font pas partie des données d'apprentissage.

L'apprentissage vise alors à ajuster les paramètres dont dépend la fonction de cout de telle manière à obtenir un modèle présentant les qualités requises d'apprentissage et de généralisation. Par exemple, pour les Réseaux de neurones artificiels (RNAs), le principe fondamental de l'apprentissage consiste à ajuster de façon incrémentale les poids des connexions entre neurones à partir de présentation d'exemples en entrée du réseau. Cet apprentissage fait appel, selon le type de réseau, à deux types de lois : une loi d'inspiration biologique et une loi mathématique. La première revient à augmenter la force de la connexion entre deux neurones qui sont simultanément excités (loi de Hebb), alors que la deuxième est fondée sur une méthode d'optimisation d'une fonction de coût calculée à partir de l'erreur commise par le réseau (par exemple, la méthode de rétro-propagation). Bien que les principaux réseaux se distinguent par l'architecture, le niveau de complexité (le nombre de neurones) et par le type des neurones (leurs fonctions de transition), le mode

d'apprentissage est le point de distinction majeure entre les RNAs.

Il existe deux tendances principales en apprentissage : apprentissage symbolique et apprentissage numérique. Le premier est issu de l'intelligence artificielle alors que le deuxième est issu des statistiques. Dans notre thèse, On s'intéresse aux méthodes appartenant au deuxième type et nous allons par la suite décrire les modes d'apprentissage existants.

1.2.2 Modes d'apprentissage

Il existe plusieurs modes d'apprentissage que nous pouvons catégoriser sous trois types principaux : apprentissage supervisé, non supervisé et par renforcement. L'auteur de [75] définit premièrement ces modes, d'une façon simple, à travers un exemple qui décrit les manières qu'un professeur peut suivre pour présenter l'information aux élèves.

1.2.2.1 Apprentissage supervisé

On entend par apprentissage supervisé une technique très importante en apprentissage automatique où l'on cherche à produire automatiquement des modèles à partir d'un ensemble d'entrées $E = \{X_i/i = 1, \ldots, p\}$ et leurs sorties associés $S = \{o_i/i = 1, \ldots, p\}$ avec $X_i = (x_1, \ldots, x_n)$. Ces données représentent des cas déjà traités et validés et sont appelées des données étiquetées et l'ensemble $A = \{(X_i, o_i)/i = 1, \ldots, p\}$ (d'entrée,sortie) est appelé ensemble d'apprentissage. A l'aide de cet ensemble, ce type d'apprentissage consiste à forcer le réseau à converger vers un état final précis. Il s'agit dans ce cas de déterminer une fonction g qui réalise la meilleure approximation d'une fonction f liant un ensemble d'entrée E à un ensemble de sortie S tel que $o_i = f(X_i)$ c'est-à-dire que la fonction g doit minimiser l'erreur de généralisation définie par [201] :

$$E(g) = \int_X (g(X) - f(X))^2 p(X)\,\mathrm{d}x \tag{1.1}$$

Où p(X) est la distribution de probabilité de la variable X qui gère les données de l'ensemble. Cette quantité est mesurée sur un ensemble de test formé des données qui n'ont pas participé à la phase d'apprentissage pour tester l'efficacité du modèle (la fonction g) à atteindre une meilleure généralisation. La fonction g (fonction de prédiction) est choisie parmi un ensemble de modèles pouvant réaliser une approximation de la fonction f et ayant de bonnes garanties en généralisation. Le but d'une méthode ou algorithme d'apprentissage supervisé est donc de généraliser pour des entrées inconnues ce qu'il a pu « apprendre » grâce aux données déjà étiquetées.

Les méthodes d'apprentissage supervisées produisent des solutions pour trois types de problèmes :

- Problème de régression : Lorsque la sortie que l'on cherche à estimer est une valeur dans un ensemble continu de réels. La fonction g est appelée dans ce cas un régresseur.
- Problème de discrimination (classification supervisée (CS)) : Ce problème revient à attribuer une étiquette à chaque entrée lorsque l'ensemble des valeurs de sortie est fini. La fonction de prédiction est alors appelée un classifieur.

— Problème de prédiction structurée : Ce problème revient à attribuer une sortie complexe à chaque entrée lorsque l'ensemble de sortie est un ensemble de données structurées.

1.2.2.2 Apprentissage non supervisé

Cette méthode se distingue de l'apprentissage supervisé par le fait que les données d'apprentissages sont non étiquetées tel qu'aucune sortie n'est connu a priori. Ce type d'apprentissage utilise alors, en entrée, un ensemble de données collectées pour que l'algorithme traite ces données comme des variables aléatoires et construit un modèle de « densités jointes » pour cet ensemble de données. Par conséquent, l'objectif de ce type d'apprentissage est la détermination d'une fonction h (parmi un ensemble de fonctions) qui optimise un critère spécifique définit selon le problème traité.

Les problèmes solubles par les méthodes d'apprentissage non supervisé sont :

— Classification non supervisée (Clustering en anglais) : c'est l'exemple typique de l'apprentissage non supervisé. Dans ce cas, le critère à optimiser est une fonction qui mesure la séparation entre les classes et la dispersion des données à l'intérieur de celles-ci.

— L'estimation de la fonction de densité : il y a deux grandes catégories de méthodes qui estime la densité : Méthodes paramétriques (incluant densité-mélange) et Méthodes non-paramétriques. Pour la première catégorie, les densités de probabilité ($p(x)$) sont posées à l'avance en visant la recherche de la paramétrisation de ces densités. Les méthodes non-paramétriques visent à estimer la densité de probabilité directement à partir des données sans avoir aucune hypothèse a priori sur la distribution des données.

— Représentation dans une faible dimension : Ce problème revient à chercher des surfaces planes (ou non linéaires) dans lesquelles la densité des données est la plus grande. Cela peut servir pour visualiser les données et réduire leur dimension.

1.2.2.3 Apprentissage par renforcement

Ce genre d'apprentissage peut être situé entre les deux autres précédents. En effet, comme l'apprentissage non supervisé, les sorties désirées ne sont pas disponibles, mais elles peuvent être analysées et évaluées. Par suite, le système favorise la réapparition de celles jugées bonnes. En revanche, la réapparition de mauvaises sorties n'est pas souhaitée par le système.
Plus précisément, le principe de ce type d'apprentissage est inspiré des travaux en psychologie empirique de Thorndike qui se résument comme suit [199] : lorsqu'une décision prise par le réseau engendre un indice de satisfaction positif, alors la tendance du réseau à prendre cette décision doit être renforcée. De ce fait, le réseau n'est pas dirigé pour prendre une décision, mais il doit découvrir tout seul quelles actions sont plus bénéfiques en les essayant. Par conséquent, les caractéristiques les plus importantes de l'apprentissage par renforcement sont la recherche par essai-erreur et récompense à long terme. Une autre clé de l'apprentissage par renforcement est de considérer le problème dans sa globalité,

l'information obtenue ainsi que l'interaction avec l'environnement permet au réseau de générer ses propres comportements et résoudre le problème en vision globale.

Il existe autres types qui sont peu connu comme : l'apprentissage semi-supervisé, l'apprentissage par transfert et l'apprentissage partiellement supervisé.

Les méthodes issues de l'apprentissage artificiel se regroupent habituellement en deux champs industriels : la reconnaissance de formes (Rdf) [122][31] et la fouille de données [98]. Les processus de ces deux champs montrent peu de différence, seul l'objectif diffère. En effet, la fouille de données ou Extraction de connaissances à partir des données (ECD) est apparue, afin de permettre l'analyse de données de n'importe quelle nature, et d'offrir une visualisation interactive des données (stockage dans une base de données, sélection des données à étudier, si nécessaire : nettoyage des données, puis utilisation des apprentissages numériques et symboliques afin de proposer des modèles à l'utilisateur, enfin validation des modèles proposés). Tandis qu'un outil développé suivant le processus de reconnaissance de formes permettrait quant à lui de faire une décision.

La classification est une capacité fondamentale de l'intelligence et un problème central en reconnaissance de formes et fouille de données. Ces dernières années, les domaines scientifiques nécessitent de plus en plus de catégoriser leurs données dans un but descriptif ou décisionnel. Cela a augmenté significativement les besoins de classification.

1.3 Introduction à la classification

L'objectif principal de la classification est l'extraction des informations exploitables en détectant certaines caractéristiques de l'ensemble de données traitées. Pratiquement, elle consiste à partitionner un jeu de données en sous-groupes le plus homogènes possible [87]. Pour une bonne introduction à ce domaine, nous recommandons de consulter par exemple les travaux suivants : [58], [92] ou [129].

La classification a plusieurs avantages qui facilitent plusieurs tâches dans des problèmes différents. Parmi ces avantages on trouve :

- Réduction de temps de recherche de l'information : le temps consommé pour accéder à l'information désiré est réduit car la méthode d'accès est facilitée par la classification des objets.
- Réduction de dimension : la classification donne plusieurs solutions pour réduire la dimension des données.
- Représentation et visualisation des données : les données appartenant à la même classe peuvent être représentées par le centre de leur classe. Visualiser les données de grande dimension devient possible avec la classification.
- Prévision et prise de décision : la classification peut tirer des connaissances sur le comportement de certains objets. A travers ces connaissances, on peut prédire et décider le comportement des autres.

- Référence des objets : chaque objet appartient à une référence précise qui est sa classe.

1.3.1 Notion de partition

Une notion très importante et fondamental dans la pratique de la classification est la notion de partition. La problématique de classification consiste à déterminer une ou plusieurs partitions sur un ensemble bien déterminé. Définissons d'abord cette notion.

Définition 1. *On appelle partition d'un ensemble E, l'ensemble P, $P = \{C_i, i \in I\}$, tel que :*
I : ensemble d'indices
C_i : une classe de E (un groupe d'individus présentant des similitudes communes) possédant les propriétés suivantes :

1. $\forall i \in I, C_i \neq \emptyset$
2. $\forall i \in I, \forall j \in I, i \neq j \longrightarrow C_i \cap C_j \neq \emptyset$
3. $\cup_{i \in I} C_i = E$

Une autre notion très importante est la notion de ressemblance entre objets. Si on définit l'objectif de la classification de la manière suivante : la classification vise à constituer des groupes d'objets tels que :
— Les objets soient les plus similaires possibles au sein d'un groupe
— Les groupes soient aussi dissemblables que possible
On aura alors besoin de définir la notion de similarité ou proximité entre objets.

Définition 2. *On appelle similarité ou dis-similarité, toute application à valeurs numériques qui permet de mesurer le lien entre les individus d'un même ensemble ou entre les variables.*

Définition 3. *Un indice de similarité sur un ensemble E est une application $Sim : E \times E \longrightarrow \mathbb{R}^+$ ayant les propriétés : $\forall x \in E, \forall y \in E$ avec $x \neq y$, on a :*

$$P(1) : Sim(x, y) = Sim(y, x)$$
$$P(2) : Sim(x, x) = Sim(y, y) = Sim_{max} > Sim(x, y)$$

Définition 4. *Un indice de dis-similarité sur un ensemble E est une application $Diss$: $E \times E \longrightarrow \mathbb{R}^+$ ayant les propriétés : $\forall x \in E, \forall y \in E$ avec $x \neq y$, on a :*

$$P(3) : Diss(x, y) = Diss(y, x)$$
$$P(4) : Diss(x, x) = 0$$

1.3.2 Qualité de la partition et nombre de classes

L'objectif des approches de classification automatique (classification non supervisée) est de produire une partition réalisant un compromis entre une cohésion maximale (similarité intra-classe) et une séparation maximale (dis-similarité inter-classe) entre les classes de la

partition obtenue. Par conséquent le problème majeur du choix de la meilleure partition se pose.

Pour calculer la complexité du problème de partitionnement, on peut envisager d'énumérer toutes les partitions possibles d'un ensemble d'objets. On note par $S(N, K)$ le nombre de partitionnements de N objets en K groupes. Ce nombre peut être calculé par récurrence sur K selon l'équation suivante [120] :

$$S(N,K) = \frac{1}{K!}\sum_{i=1}^{K}(-1)^{K-i}\binom{K}{i}(i)^N \tag{1.2}$$

L'énumération de toutes les partitions devient en effet impraticable même pour des problèmes de petite taille. Cela pose le grand problème de choix de la meilleure partition et la Nécessité d'algorithmes pour trouver une bonne partition.

Cela donne naissance à deux problèmes : le choix des classes adéquates et le nombre de classes.

Pour ces différentes techniques, l'un des challenges et des problèmes les plus difficiles est la détermination du nombre de classes dans un ensemble de données, qui est un paramètre d'entrée de base pour la plupart des algorithmes de clustering. Pour d'autres méthodes, le nombre de classes de la partition est défini par un niveau de coupure d'un dendrogramme contenant plusieurs partitions, qui n'est pas toujours facile à déterminer. Cependant, ce nombre est généralement un paramètre inconnu qui doit être soit spécifié par l'utilisateur en fonction des connaissances antérieures ou estimé d'une certaine manière. Une sur-estimation ou sous-estimation du nombre de clusters affectera considérablement la qualité des résultats de clustering. Souvent plusieurs niveaux de coupure sont retenus et les partitions qui s'en déduisent sont comparées afin de retenir la meilleure en termes de compacité et de séparabilité des classes. Pour les autres techniques de classification, généralement, on fixe plusieurs valeurs pour le nombre de classes, et les partitions correspondantes sont ensuite comparées.

Ainsi, la détermination du nombre de classes par les approches de clustering nécessite l'évaluation de la qualité des partitions de données obtenues. Or cette évaluation n'est pas un processus immédiat.

Il existe plusieurs critères associés à la qualité du clustering. On distingue deux catégories : critères internes et critères externes. Les critères internes sont basés sur les données et les ressemblances entre elles (similarites/dis-similarites). Les critères externes quant a eux sont basés sur des informations extérieures comme le label de classes ou l'avis d'un expert. Nous ne nous intéressons qu'aux critères internes. Lorsqu'on utilise ces derniers, le problème de classification automatique est alors considéré comme un problème d'optimisation dont les performances peuvent être déterminées.

Pour la présentation de ces indices, nous proposons d'expliciter quelques notations préliminaires. Pour une partition P en k classes $\{C_1, C_2, C_3, ..., C_k\}$ de l'ensemble d'individus $\{X_1, X_2, X_3, ..., X_n\}$, la dispersion intra-classe $s_a(C_i)$ et la séparation inter-classe $d_a(C_i, C_j)$ sont données par les formules suivantes :

$$\forall C_i \in P : s_a(C_i) = \frac{1}{|C_i|(|C_i|-1)} \sum_{u=1}^{|C_i|} \sum_{q=1}^{|C_i|} d(X_u, X_q) \tag{1.3}$$

$$\forall C_i, C_j \in P : d_a(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{u=1}^{|C_i|} \sum_{q=1}^{|C_j|} d(X_u, X_q) \tag{1.4}$$

où d représente la mesure de dis-similarité définie sur l'ensemble d'individus X et $|C_i|$ le cardinal (nombre d'individus de la classe C_i).

1.3.2.1 Indice de Davies-Bouldin

L'indice de Davies-Bouldin (BD) est basé sur la minimisation du rapport des dispersions intra-classe et de la séparation inter-classe. Il est calculé comme suit :

$$DB(P) = \frac{1}{k} \sum_{i=1}^{k} \max_{\substack{i \leq j \leq k \\ j \neq i}} \left\{ \frac{s_a(C_i) + s_a(C_j)}{d_a(C_i, C_j)} \right\} \tag{1.5}$$

On constate ainsi que le ratio sera d'autant plus faible que les classes seront compactes et éloignées les unes des autres. Par conséquence, la partition de meilleure qualité sera celle qui minimisera l'indice de DB.

1.3.2.2 Indice de Silhouette

La silhouette [183] d'une classification est une grandeur montrant comment chaque observation appartient plus ou moins à sa classe. Supposons que n observations aient été réparties en k classes par un algorithme quelconque. La silhouette $s(x_i)$ de l'observation i est calculée à l'aide de $a(x_i)$ et $b(x_i)$ tels que :

— $a(x_i)$ est la dis-similarité moyenne de x_i avec toutes les autres observations au sein d'une même classe. Plus $a(x_i)$ est petit, meilleure est l'assignation de i à sa classe C_i :

$$\forall x_i \in X; a(x_i) = \frac{1}{|C_i - 1|} \sum_{\substack{x_j \in C_i \\ x_j \neq x_i}} d(x_j, x_i) \tag{1.6}$$

— $b(i)$ est la plus faible moyenne des dis-similarités (ou distances) de l'observation i à chaque autre classe dont i ne fait pas partie. La classe avec cette plus faible dissimilarité moyenne est appelée classe voisine de i car c'est la meilleure classe suivante

pour l'observation i :

$$\forall x_i \in X; b(x_i) = \min_{\substack{C \in P \\ C \neq C_i}} d(x_i, C) \tag{1.7}$$

$$o \quad d(x_i, C) = \frac{1}{|C|} \sum_{x_j \in C} d(x_i, x_j) \tag{1.8}$$

La silhouette de la ième observation est alors donnée par :

$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}} \tag{1.9}$$

Cet indice prend des valeurs entre -1 et 1. Plus la valeur de $s(x_i)$ est proche de 1, plus la classification de x_i dans C_i est meilleure. Quand $s(x_i)$ est proche de 0, x_i est située entre deux classes. Finalement, quand $s(x_i)$ s'approche de -1, x_i est mal classée dans C_i. Dans ce cas, l'observation x_i doit être rattachée à un autre cluster plus proche.

Chaque classe de la partition obtenue est aussi représentée par un indice silhouette pour évaluer l'affectation des objets. En effet, cet indice montre quels objets sont correctement classés à l'intérieur de cette classe et lesquels n'ont simplement qu'une position intermédiaire.

On définit l'indice de silhouette de la classe C_i par la moyenne des indices de silhouette des individus qui lui appartiennent :

$$\forall C_i \in P; s(C_i) = \frac{\sum_{x_j \in C_i} s(x_j)}{|C_i|} \tag{1.10}$$

De même, on définit l'indice de silhouette global de la partition P par la moyenne globale des largeurs de silhouettes dans les différentes classes C_i qui composent la partition :

$$s(P) = \frac{\sum_{C_i \in P} s(C_i)}{k} \tag{1.11}$$

La meilleure partition retenue est alors celle qui permet d'obtenir une silhouette globale maximale.

1.3.2.3 Indice de R^2

Cet indice est une proportion de la variance expliquée par les classes. Son expression est donnée par :

$$R^2 = \frac{I_B}{I} \tag{1.12}$$

Avec I_B et I sont respectivement l'inertie inter-classe et l'inertie totale de l'ensemble de données. Si l'ensemble X des points, de centre g, est regroupé en K classes chacune de centre g_k et de poids p_k, ces deux inerties sont définies par :

$$I_B = \sum_{k=1}^{K} p_k \|g_k - g\|^2 \tag{1.13}$$

$$I = \sum_{x_i \in X} p_i \|x_i - g\|^2 \tag{1.14}$$

L'indice R^2 doit être le plus proche possible de 1 sans avoir un grand nombre de classes. Dans le graphe de représentation de cet indice, il faut s'arrêter après le dernier saut important. Un exemple est représenté dans la figure (1.1) pour détecter le nombre de classes à l'aide de l'indice R^2.

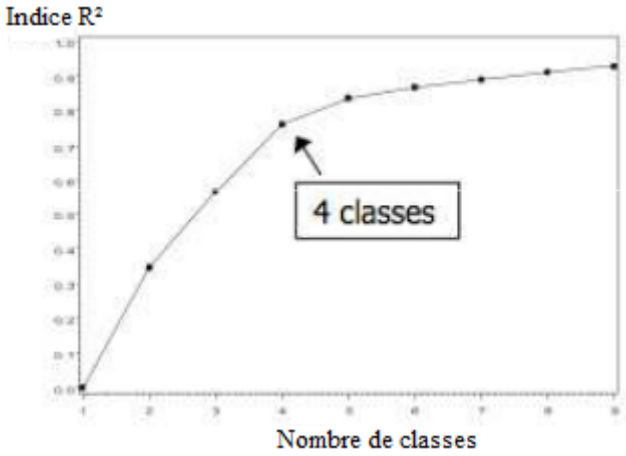

Figure 1.1 – Différents types de classification

1.3.2.4 Indice de pseudo F

L'indice pseudo F [41] décrit le rapport entre la variance des clusters et la variance à l'intérieur des clusters. Il mesure la séparation entre toutes les classes de la partition.

$$F = \frac{\frac{R^2}{k-1}}{\frac{1-R^2}{n-k}} \tag{1.15}$$

Où n est le nombre d'observations et k le nombre de classes.

Les grandes valeurs de Pseudo F indiquent des clusters compacts et séparés. En particulier, les pics de cet indice sont des indicateurs d'une séparation de clusters plus importante. Typiquement, ils sont repérés dans le graphique de l'indice par rapport au nombre de clusters.

Dans le contexte de mélanges supposés gaussiens, c'est-à-dire si l'hypothèse d'une situation gaussienne multidimensionnelle, le choix du nombre de classes s'apparente à une sélection de modèle par des critères AIC, BIC, spécifiques (Annexe (5.7)).

Les indices que nous venons de présenter permettent d'évaluer la qualité d'une partition des données, ce qui est nécessaire dans le cas des approches itératives que nous allons proposer dans les chapitres suivants.

1.3.3 Types de classification

Les méthodes de classification sont abondantes. La figure (1.2) présente une première catégorisation des variantes que l'on peut trouver parmi les méthodes de classification [120].

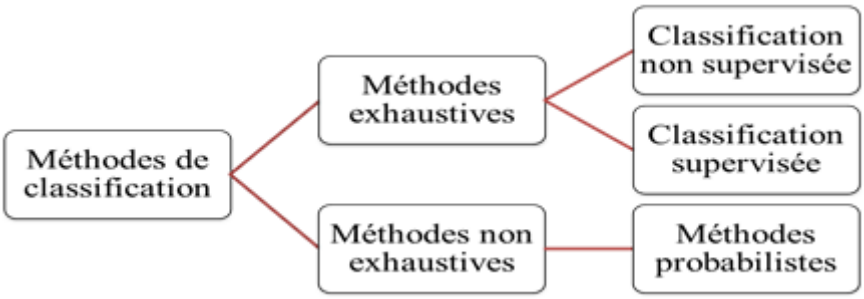

Figure 1.2 – Différents types de classification

On distingue premièrement deux grandes classes de classification : Classification exclusive et non exclusive.

— Classification exclusive (hard classification en anglais) : Cette classe de méthodes consiste à partitionner l'ensemble d'objets en imposant qu'un objet n'appartient qu'à une et une seule classe.
— Classification non exclusive (fuzzy classification en anglais) : Au contraire des méthodes de la classification exclusive, une méthode de classification non exclusive autorise qu'un objet appartienne à plusieurs classes simultanément ; dans ce cas, les classes peuvent se recouvrir.

En s'intéressant à la première classe, nous présentons une description générale de la classification supervisée et non supervisée.

1.3.3.1 Classification non supervisée

Une méthode de classification non supervisée n'utilise que la matrice de dis-similarité et ne possède aucune information sur la classe d'un objet. En d'autres termes on dit que les objets ne sont pas étiquetés et alors on cherche à découvrir leur groupe d'objets. Comme nous avons déjà signalé, ce type de classification fait partie d'une des réalisations de l'apprentissage non supervisé.

Appelée aussi classification automatique, elle représente un axe très important qui est omniprésent dans plusieurs domaines de différentes natures. La littérature est très riche en méthodes de clustering, plusieurs livres dédiés au clustering ont été publiés [120] [5] [102] [193] [69] [81] [19], en plus d'une somme d'articles de revus [122] [121]. La fourniture d'une catégorisation nette des différentes méthodes de clustering s'avère un peu difficile parce qu'on peut trouver un chevauchement entre ces catégories, de sorte qu'une méthode peut avoir des caractéristiques de plusieurs catégories. Toutefois, la présentation des différentes méthodes sous leurs catégories d'une manière hiérarchique et

organisée est très utile. Dans cette thèse nous allons mettre l'accent sur des méthodes précises que nous avons développées par la suite dans nos travaux. Pour une étude approfondie sur les différentes méthodes des différentes catégories nous référons le lecteur à [98]

La figure (1.3) présente une image organisée des différentes catégories des méthodes de clustering et les classes en cinq catégories : méthodes hiérarchiques, méthodes de partitionnement, méthodes basées sur la densité, méthodes basées sur grid, méthodes basées sur un modèle.

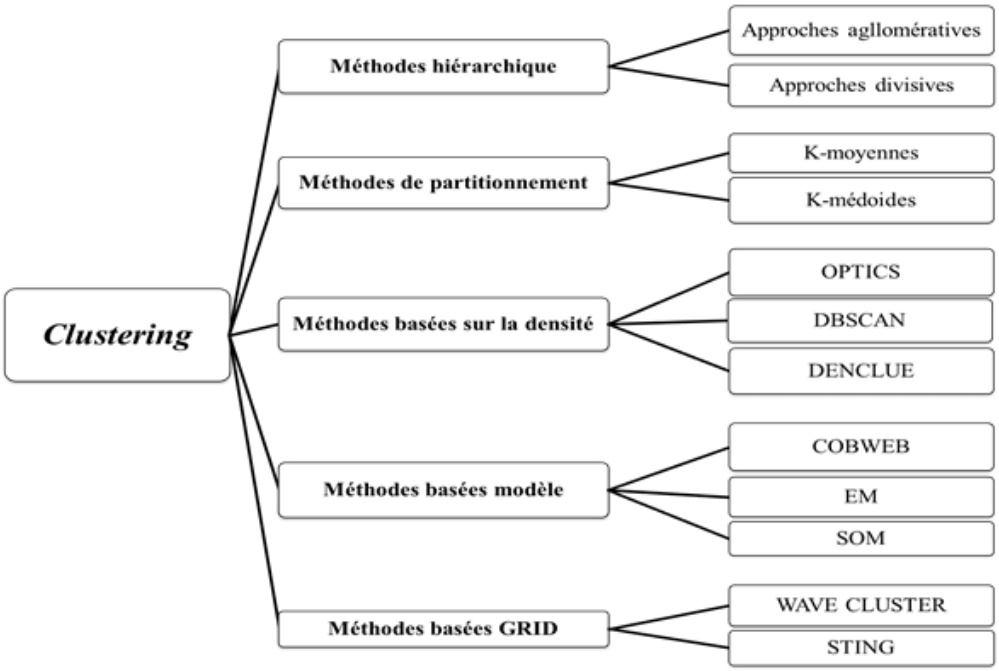

Figure 1.3 – catégorisation des différents méthodes de classification non supervisée

1.3.3.2 Classification supervisée

Au contraire de la classification non supervisée, les objets sont étiquetés tout en connaissant leurs dis-similarités. Elle consiste à utiliser les groupes connus pour découvrir ce qui les différencie ou afin de pouvoir classer de nouveaux objets dont la classe n'est pas connue. Comme vu précédemment, ce type de classification fait partie des réalisations de l'apprentissage supervisé.
Son but est de construire un classifieur qui soit capable d'attribuer une classe C_i (parmi un nombre fini de classes possibles) à un objet x_i. L'approche statistique est fondée sur deux hypothèses principales :

— Un objet est décrit comme une réalisation d'une variable aléatoire x correspondant à la description de l'objet et le numéro i de la classe qui lui est associé est également représenté par une variable aléatoire d qui est dans ce cas discrète ; il en découle que le problème de la discrimination peut se formuler comme un problème de détermination

des probabilités conditionnelles $p(d = i/x)$ permettant de représenter, dans le cadre probabiliste, la dépendance entre les variables aléatoires x et d.
— On dispose d'un échantillon (de taille finie) issu de la variable jointe parente (x,d) et de la question de la construction d'un classifieur à partir des données qui se traduit par une estimation de probabilités conditionnelles à partir de l'échantillon. On construit la règle de décision de façon que la probabilité d'erreur de classification soit minimale.

Nous distinguerons trois approches pour la construction des règles de classification à partir des données que nous présentons très brièvement. Deux premières approches reposant sur l'application du théorème de Bayes qui exprime la probabilité $p(d = i/x)$ en fonction de la densité conditionnelle de la classe i, $p(x/d = i)$, de la probabilité $p(d = i)$ et de la densité $p(x)$.

La première approche dite paramétrique se base sur l'hypothèse que la forme de la distribution de densité conditionnelle est connue a priori et que nous pouvons l'exprimer analytiquement. La construction du classifieur consiste alors à estimer les paramètres de la fonction de densité à partir des échantillons d'apprentissage. Le plus souvent, essentiellement pour des raison de simplicité, les densités de probabilités sont supposées suivre des lois normales ou multi-normales. Le désavantage de cette approche vient du fait que l'hypothèse sur la forme des densités de probabilités peut ne pas correspondre à la réalité.

La deuxième approche dite non-paramétrique permet de s'affranchir des hypothèses de la première approche. Les fonctions de densités sont estimés à partir des données sans fixer a priori la forme des densités de probabilités. On peut citer par exemple l'utilisation de fonction noyau, d'histogramme, ou de la règle des k-plus proches voisins. L'inconvénient majeur de cette technique survient en les appliquant en grande dimension : elles sont difficiles à mettre en œuvre car le nombre de paramètres augmente d'une façon exponentielle avec la dimension de l'espace dans lequel se fait l'estimation.

Une troisième approche dite semi-paramétrique ou encore flexible consiste à rechercher le meilleur classifieur à l'intérieur d'une classe de fonctions très générales dont la complexité pourra être ajustée en général par le contrôle du nombre de paramètres ou de termes intervenant dans le modèle. Le nombre de paramètres adaptatifs dans cette approche est indépendant de la taille de l'échantillon et peut être varié librement selon la flexibilité désirée. Les classes de fonctions choisies sont en général assez riches pour présenter des "capacités d'approximation universel". C'est le cas des techniques de poursuite en projection des modèles linéaires généralisées, des réseaux de neurones. La recherche d'une solution optimale au problème de discrimination demande d'une part de trouver la bonne sous classe de fonction pour cette classe de méthodes et d'autre part, de déterminer les paramètres optimaux pour cette sous classe. Il s'agit du problème de choix du modèle.

On peut se reporter à [31][65][181] pour une représentation détaillée de ces différentes approches.

La plupart de nos travaux présentés dans cette thèse s'intéressent aux méthodes neu-

ronales pour résoudre le problème de la classification supervisée et non supervisée. Nous abordons par la suite les méthodes de classification. Nous mettons l'accent sur les approches neuronales déterministes et probabilistes ainsi que les approches de partitionnement.

1.4 Classifieurs de partitionnement

1.4.1 Généralités

Au contraire des méthodes de classification hiérarchique qui consistent à créer une décomposition hiérarchique de l'ensemble de données, un algorithme de partitionnement cherche une seule partition de l'ensemble de données. L'avantage majeur des méthodes de partitionnement est leur efficacité envers les applications utilisant des données de grandes dimensions dont la construction d'un dendrogramme est très couteuse en termes de calcul. Cependant, l'utilisation de ce genre de méthodes pose le problème du choix du nombre de classes désirés à la sortie. La production des clusters via les méthodes de partitionnement est faite par l'optimisation d'un critère définit soit localement (sur un sous ensemble de données) ou globalement (définit sur tous les données).

La fonction la plus intuitive et fréquente qui est utilisée comme fonction objectif dans les méthodes de partitionnement est le critère d'erreur quadratique. Cette erreur donne de bons résultats pour un ensemble de classes compactes et isolées. Elle est définie sous la forme :

$$I\left(W,\aleph\right)=\sum_{c=1}^{p}\sum_{i=1}^{n_j}\left\|z_i^c-W_c\right\|^2 \tag{1.16}$$

La minimisation de cette fonction peut se modéliser en termes d'un problème de programmation non linéaire :

$$\begin{cases}\min_{x_{ij}}\sum_{j=1}^{m}\sum_{i=1}^{n}x_{ij}\|x_i-W_c\|^2\\ \sum_{j=1}^{m}x_{ij}=1\\ \sum_{i=0}^{n}x_{ij}\geq 1\\ x_{ij}\in\{0,1\}\end{cases} \tag{1.17}$$

avec :

$$x_{ij}=\begin{cases}1 & \text{si l'objet i est affecté à la classe j}\\ 0 & \text{sinon}\end{cases} \tag{1.18}$$

La première contrainte du modèle garantie l'appartenance d'une donnée uniquement à une seule classe (hard classification). La deuxième contrainte assure que toute classe contient au moins une seule donnée. Le modèle est un problème d'optimisation en nombre entier avec une fonction objectif non linéaire et non convexe et à contraintes discrètes. Ce problème est NP-complet [133] et peut avoir plusieurs minimums locaux. C'est pour cette raison que la plus part des méthodes de résolution de ce problème sont des heuristiques qui se contentent de chercher une solution raisonnable en identifiant un bon minimum local [75].

1.4.2 Méthode K-moyennes

L'heuristique la plus connue et la plus utilisée [122] [119] pour la résolution de la problématique de partitionnement est la méthode de k-means (ou k-moyennes). Cet algorithme est l'un des plus simples [206] et plus célèbres algorithmes de la classification non supervisé. Cet algorithme détermine les centres de clusters et les éléments qui leur appartiennent en minimisant la fonction objectif, déjà introduite, basée sur l'erreur quadratique. Le but principal de l'algorithme est de localiser les centres des clusters autant que possible éloignés les uns des autres et associer chaque point des données au plus proche centre. Malgré l'efficacité de cette méthode, le point de départ pose le problème de la convergence vers l'optimum global. En effet, elle soufre du problème d'initialisation des centres des classes et le choix du nombre k (nombre de centres ou de classes).

Nous nous sommes intéressés à la résolution de ce problème en contribuant par deux travaux [76][101]. Vu que nous présenterons notre contribution dans le chapitre 3, nous réservons plus de détail sur les différentes étapes de cette algorithme pour ce chapitre ainsi que la description de la problématique du choix de paramètres.

1.4.3 Méthode ISODATA

Cette méthode très connue [37], utilise la technique de « Merging and Splitting clusters ». Elle vise à déterminer automatiquement le nombre de classes en commençant par un nombre assez grand. En itérant l'algorithme, on fusionne deux classes si la distance entre eux est inférieure à un seuil donné et on subdivise une classe si son inertie est très grande. Cet algorithme vise à surmonter le problème du choix du nombre de classes mais exige la donnée de trois paramètres.

1.4.4 Nuées dynamiques

La variante proposée par [57] et parallèlement par [103] consiste à sélectionner une fonction cout tout à fait différente et ainsi elle décrit une approche dynamique de clustering obtenue par la nouvelle formulation du problème de clustering dans le cadre de l'estimation du maximum de vraisemblance. Au lieu des barycentres, chaque classe est représentée par un noyau constitué d'éléments représentatifs de cette classe. Certes, cela permet de remédier au problème de l'influence négative d'éventuelles valeurs extrêmes sur le calcul du barycentre. Pour trouver ces noyaux, Diday a également proposé la recherche de formes fortes (homogènes) communes à plusieurs partitions issues d'initialisations différentes. Bien que cette méthode propose une solution au problème des points extrêmes (outiliers) mais l'algorithme demande toutefois d'indiquer le nombre de classes.

1.4.5 Méthodes k-modes, k-prototypes

Cette méthode [115] représente une extension de k-means pour les variables catégoriels. Pour cela la méthode définit premièrement une nouvelle mesure de dis-similarité entre va-

riables catégoriels comme suit :

$$D(x,y) = \sum_{i=1}^{n} \delta(x_i, y_i) \quad \text{où} \quad \delta(x_i, y_i) = \begin{cases} 1 & si \quad x_i = y_i \\ 0 & \text{sinon} \end{cases} \tag{1.19}$$

Deuxièmement, Au lieu d'utiliser les moyennes des classes construites, la méthode K-modes utilise des modes de classes. En plus, Huang propose une nouvelle méthode pour la mise à jour des modes afin de minimiser le cout de la fonction objectif de classification. Pour traiter des variables mixtes (numériques et catégoriels), Huang [115] propose un autre algorithme nommé « K-prototypes » qui intègre les deux processus de k-means et k-modes. Pour calculer la dis-similarité entre variables mixtes, il définit une mesure qui additionne la mesure D utilisée par la méthode k-modes à la distance euclidienne multiplié par un paramètre qui veille à ne favoriser aucun type des attributs.

1.5 Classifieurs neuronaux

1.5.1 Généralités sur les réseaux de neurones artificiels

Le développement des recherches sur le fonctionnement du cerveau a fait d'énormes progrès qui ont contribué à la compréhension de cet organe humain très mystérieux et complexe. Considéré comme une machine, le cerveau a trois grandes caractéristiques : Apprentissage, Robustesse et Parallélisme. Ces derniers permettent une adaptation, une résistance à l'impression des entrées et à la détérioration, une interaction locale et des propriétés globales ainsi qu'une simultanéité du traitement. Cette boite noire inspire la construction des réseaux de neurones artificiels en simulant, au plus près des connaissances actuelles, ces unités de "calcul" que nous avons par milliards dans chacun de nous. En effet, les RNAs représentent une modélisation mathématique et algorithmique du cerveau et du système nerveux comme un ensemble d'unités de calculs (les neurones) connectées entre elles. On apprend les paramètres du réseau (les poids des connexions) pour reproduire le même comportement. Les réseaux de neurones artificiels sont parmi les approches directes qui déterminent les discriminants sans utiliser d'hypothèses sur les densités des données.

Les réseaux de neurones artificiels sont des réseaux de processeurs élémentaires fortement connectés et fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.

Un réseau de neurones formels émule une fonction mathématique dépendant d'un jeu de paramètres W qui associe à tout point x de l'espace d'entrée χ un point u de l'espace de sortie U tel que :

$$\psi : W \times \chi \to U \tag{1.20}$$

$$\psi : (w, x) \mapsto \psi(w, x) \tag{1.21}$$

Cette fonction peut être vue aussi comme :

$$\psi_w : \chi \to U \tag{1.22}$$

$$x \mapsto \psi_x$$

Depuis leur apparition et avec leur développement, les réseaux de neurones sont capables d'établir plusieurs tâches comme : approximation de fonctions et de distributions, analyse exploratoire de données (fouille de données, visualisation), régression non linéaire, identification et prévision de séries temporelles. Ils ont été utilisés dans toutes sortes de problèmes, de la reconnaissance de formes, de visages, de caractères [143] au traitement du langage [215] en passant par la chimie.

Au fil des ans, des architectures diverses et variées de réseaux de neurones apparurent. On peut citer en particulier les réseaux à convolution [143], les réseaux récurrents, les réseaux à propagation avant, les machines de Boltzmann et les machines de Boltzmann restreintes [108]. Nous présentons à travers la figure (1.4) une catégorisation des différents types de réseaux de neurones artificiels. Il existe deux classes principales : réseaux bouclés et autres non bouclés.

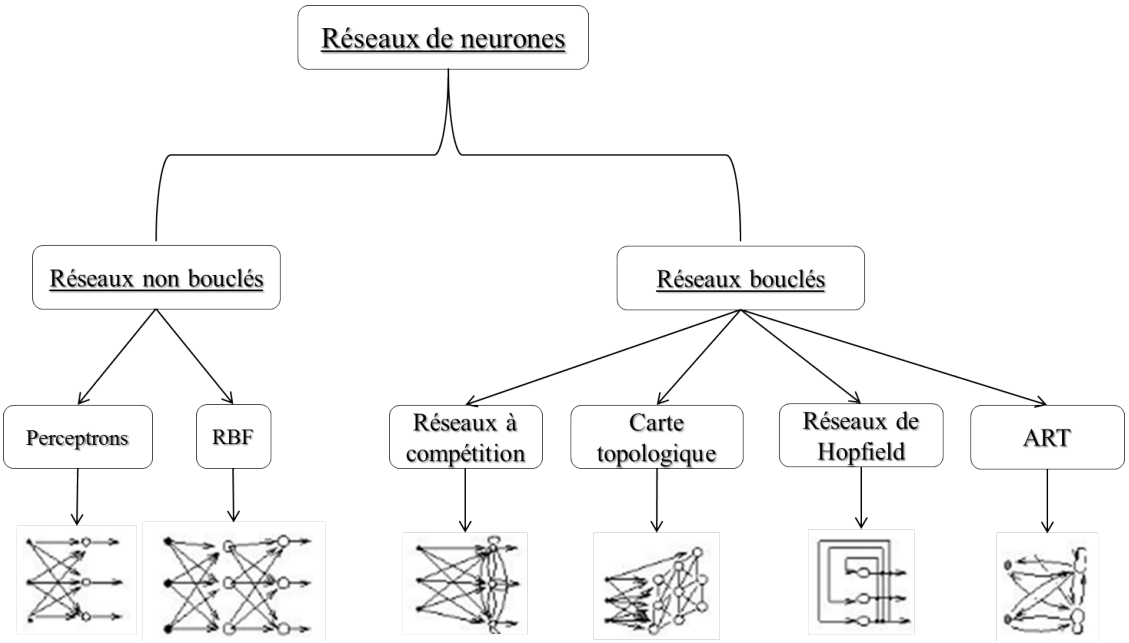

Figure 1.4 – Catégorisation des types de RNAs

Les sections suivantes sont consacrées à la présentation de deux types de réseaux connexionnistes : réseaux multi-couches et cartes topologiques. Ces réseaux constituent le background de nos travaux de recherche que nous présentons dans les autres chapitres.

1.5.2 Réseaux multi-couches

Les perceptrons multi-couches (PMCs) sont des modèles de régression non-linéaire utilisables pour la classification supervisée comme pour la régression, ils représentent des approximateurs universels qui sont robustes aux données bruitées. Ils permettent aussi de créer des modèles arbitrairement complexes. Depuis leur apparition, ces réseaux ont connus un grand succès et ont été utilisés dans différentes applications [198] [166]. En outre, ces réseaux sont de plus en plus utilisés en marketing, scoring, traitement du signal, automatique, prédiction, diagnastic... avec des succès divers et plus précisément pour la reconnaissance de formes [167].

L'invention des PMCs est due à l'incapacité des premiers perceptrons à résoudre des problèmes non linéaires.

L'apparition des réseaux multi-couches a été précédée par le développement de plusieurs réseaux de neurones [202]. J. Mc Culloch et W. Pitts [162] ont introduit en 1943 le premier neurone formel (figure (1.5)) en modélisant le neurone biologique (Modèle de Neurone biologique capable de transmettre via l'axone des informations issues par les dendrites à d'autres neurones au travers de ses différentes connexions (synapses)). Il était montré que ces premiers réseaux formels simples sont capables de réaliser des fonctions logiques, arithmétiques et symboliques complexes.

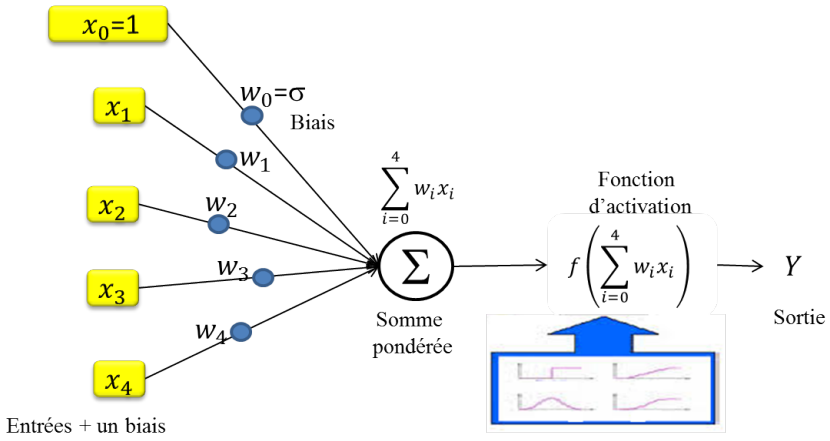

Figure 1.5 – Modélisation du neurone biologique sous forme d'un Neurone formel

En 1949 D. Hebb, un psychologiste américain, s'inspire du conditionnement chez l'animal et propose la loi de modification des propriétés des connexions entre neurones connue par la loi de Hebb. Celle ci consiste à renforcer la connexion synaptique de deux neurones qui s'activent en même temps et d'inhiber la connexion synaptique des deux neurones qui s'activent en sens opposé. Cette loi n'apprend pas de l'erreur. Le premier succès de ces types de réseaux a vu le jour avec le développement du perceptron simple (figure (1.6)) en 1957 par F. Rosenblatt [182] qui s'est inspiré des théories cognitives de Friedrich Hayek et de Donald Hebb. C'est le premier modèle pour lequel un processus d'apprentissage a pu être défini.

Basé sur ce réseau, Rosenblatt a développé le premier neuro-ordinateur et l'a appliqué au domaine de la reconnaissance de formes. Un autre modèle, très proche dans sa structure du perceptron simple, a été développé en 1962 par l'automaticien B. Widrow. Ce réseau, appelé Adaline (Adaptative Linear Element) (figure (1.7)), est caractérisé par sa loi d'apprentissage qui est à l'origine de l'algorithme de rétro-propagation de gradient qui va mettre fin par la suite à la limitation des perceptrons simples. Les réseaux de type Adaline sont toujours utilisés pour certaines applications particulières. Bien que le perceptron simple et l'adaline ont des structures similaires, mais ils ont des points de différence :

— Le perceptron utilise les étiquettes des classes pour apprendre les coefficients du modèle alors que l'adaline utilise des valeurs prédites continues pour faire l'apprentissage. Cette méthode est plus efficace puisqu'elle détermine le taux de l'erreur commise.
— Dans l'apprentissage du Perceptron, les poids sont ajustés seulement si un motifs est mal classé.
— Dans la plupart des temps l'adaline converge.

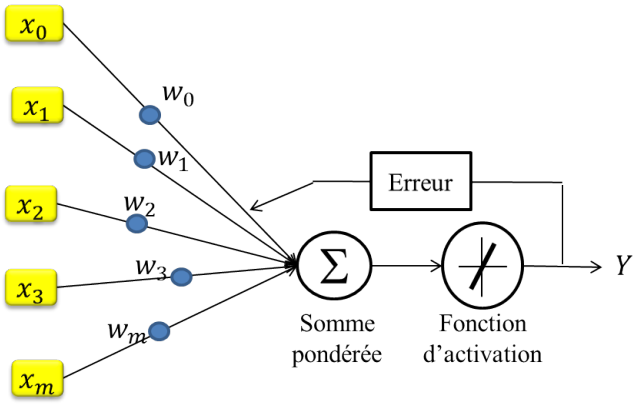

Figure 1.6 – Perceptron simple

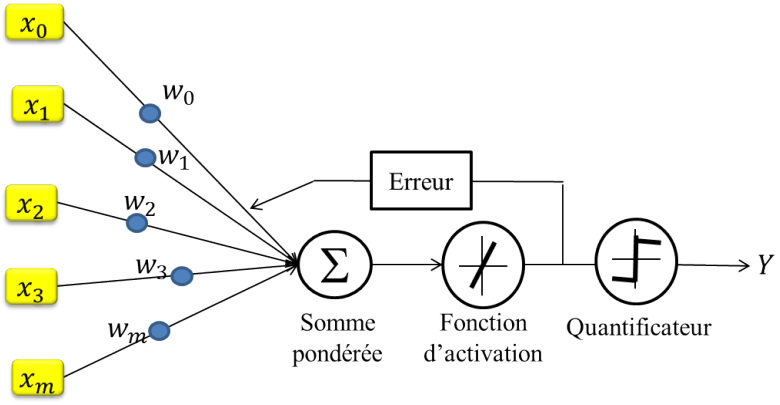

Figure 1.7 – Réseau Adaline

En 1969, M. Minsky et S. Papert [165] mettent en évidence les limitations théoriques des modèles de perceptrons concernant l'impossibilité de traiter par ces modèles des problèmes non linéaires (le problème est très bien illustré avec l'exemple du "ou exclusif" où les données

ne sont pas linéairement séparables). Pour remédier à ce problème, Paul Werbos a introduit en 1984 la nouvelle méthode d'apprentissage dédiée aux réseaux multi-couches : réseaux de neurones non bouclés à couches, dont les neurones cachés ont une fonction d'activation non-linéaire. Ces réseaux visent à décomposer la fonction non linéaire en une suite d'étapes linéairement séparables. D'où l'idée de connecter plusieurs adalines (Madaline) (figure (1.8)) ou perceptrons simples telle que chacun se spécialise dans la séparation d'une fonction linéaire.
Dès cette découverte, nous avons la possibilité de réaliser une fonction non linéaire d'entrée-sortie avec les réseaux multi-couches au travers de la rétro-propagation de gradient [213] qui restent jusqu'à nos jours le modèle le plus étudié et le plus productif au niveau des applications.

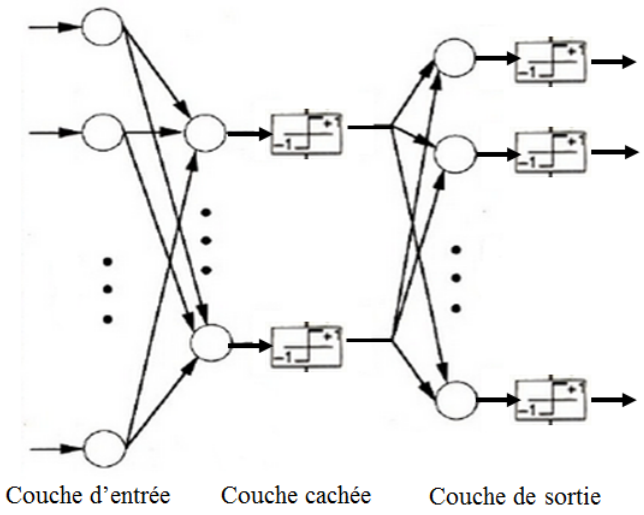

Figure 1.8 – Connection de plusieurs adalines pour arriver à résoudre des problèmes non linéaires

Pour illustrer le rôle des couches cachées dans le perceptron, nous présentons dans la figure (1.9) une comparaison de la capacité de séparation du problème XOR et des régions maillées par une, deux ou trois couches. D'après cette figure on peut remarquer qu'un perceptron à trois couches est capable de séparer d'une manière efficace les régions alors qu'un perceptron à deux couches peut apprendre seulement les régions convexes. Le perceptron simple est capable au juste de diviser le plan en deux régions à travers un hyperplan.

Les réseaux de neurones de type Perceptrons Multicouches ont montré leur efficacité en tant qu'un outil de modélisation. Cependant, le problème de choix du nombre de couches et le nombre de neurones dans chaque couche (choix de l'architecture) influence le résultat de généralisation et de convergence du PMC et cela a attiré l'attention de plusieurs chercheurs. Nous traiterons cette problématique en détail ainsi que les différents points en relation avec les perceptrons multi-couches dans le chapitre suivant.

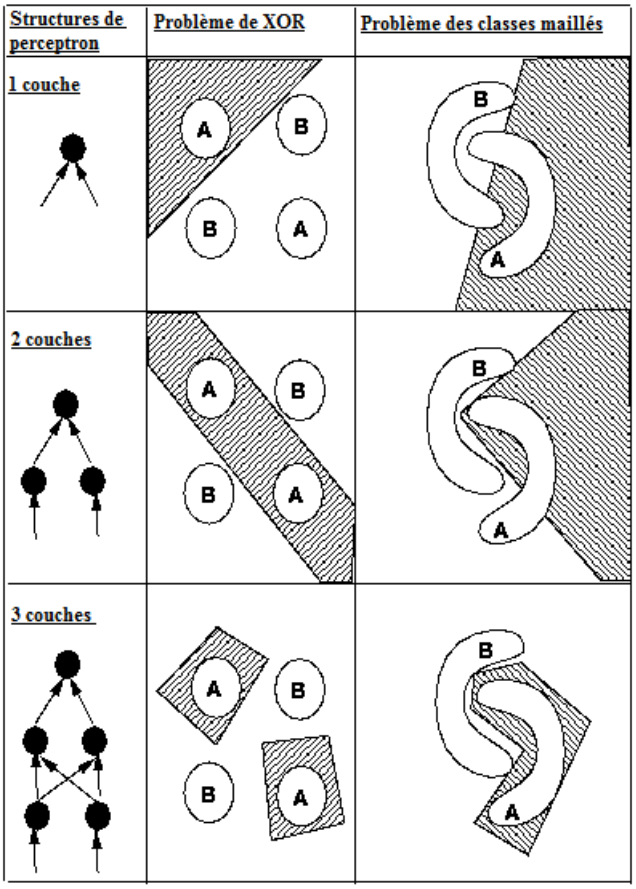

Figure 1.9 – Illustration de l'importance des couches cachées dans les réseaux à couches

1.5.3 Cartes topologiques

Dans cette section nous présentons les cartes auto-organisatrices de Kohonen et leurs fondements théoriques : origine biologique, conception artificiel, principe, formulation mathématique, algorithme d'apprentissage, utilisation dans l'analyse de données, etc. Ce modèle sera présenté sous forme détaillée car une partie de nos contributions s'articulent autour de lui.

Les cartes auto organisatrices (en anglais : Self Organizing Map (SOM)), carte auto adaptatives ou cartes topologiques forment une classe de réseaux de neurones artificiels dédiés à la discrétisation, quantification vectoriel ou classification non supervisée. Désignés aussi par cartes de Kohonen du nom de leur fondateur « TEUVO KOHONEN », leur concept [134] [135] [136] a été développé dans les années 80 à partir de motivations neuro-mimétique comme beaucoup d'autres créations de l'intelligence artificielle. Très tôt après son apparition, le réseau de Kohonen a été bénéfique pour 3343 de publications scientifiques de l'année 1981 à 1997. Le travail de [128] présente une bibliographie complète de ces travaux et le site de l'Université de Helsinki (http ://www.cis.hut.fi/research/som-research/) aborde des thèmes très divers : vision, analyse d'image, compression d'image, imagerie médicale, reconnaissance de l'écriture, reconnaissance de la parole continue, analyse du signal et de la

musique, commande de processus, robotique, recherche sur le Web, etc. La conception du réseau de Kohonen s'inspire du principe neuronal du cerveau des vertébrés. Ce dernier est composé de régions qui présentent la même topologie que les stimuli de la rétine. C'est-à-dire deux régions proches dans le cortex correspondent à deux stimuli proches dans la rétine. Aussi des stimuli de même nature excitent une région du cerveau bien particulière. Le cortex contient un grand nombre de neurones capables d'interpréter tous les types de stimuli imaginables. Tous ces caractéristiques ne sont pas génétiques dans le cerveau, mais sont dues à un apprentissage.

1.5.3.1 Conception artificiel

Rien n'est plus puissant que le cerveau ; cette réalité pousse les scientifiques d'une manière permanente à essayer de simuler des neurones artificiels dont leur fonctionnement est proche de la réalité. Les observations et caractéristiques neuro-mimétiques de Kohonen l'ont motivé à proposer un réseau de neurones artificiels [137] à apprentissage non supervisé compétitif. Ce type d'apprentissage peut être traduit par la notion d'auto-organisation et compétition dans la carte de Kohonen. On dit que l'apprentissage compétitif est local car il modifie un nombre limité de neurones du réseau et d'une manière différente [64] [27]. Contrairement aux autres types de RNAs où tous les neurones bénéficient d'un changement simultané de poids et de la même manière.

Les travaux de Kohonen se sont basés sur des techniques initialement mises au point pour l'apprentissage compétitif et parmi les premiers travaux dans ce domaine on trouve : Von Der Malsburg [205] et Didday [59]. En addition, il a introduit la notion de voisinage entre les automates (neurones) et la notion de topologie. En proposant les cartes auto-organisatrices, Kohonen cherche à satisfaire plusieurs objectifs :

- Transformer des signaux de dimension quelconque, en signaux à une ou deux dimensions (réduction de la dimension des données) tout en préservant leurs caractéristiques,
- Projeter les données initiales sur un espace discret et régulier de faible dimension (visualisation),
- Utiliser des espaces de treillis réguliers dont chacun des nœuds est occupé par un automate,
- Partitionner l'ensemble des observations en groupements similaires qui possèdent la particularité d'avoir une structure de voisinage,

1.5.3.2 Architecture du réseau de Kohonen

Différente des autres types des réseaux de neurones artificiels, l'architecture du réseau de Kohonen se compose de deux couches dont les composants sont complètement interconnectés. La première couche sert seulement à présenter les stimuli à la deuxième couche et les états de tous ses neurones sont forcés aux valeurs des données d'entrées. Cette couche contient donc exactement n neurones (avec n la dimension de l'espace des observations). Quant à la deuxième, il s'agit d'un treillis (espace discret) de faible dimension (en général 1D, 2D ou 3D). Cet espace qu'on appelle la carte et on note C, est le plus souvent de dimension deux. Cette carte est composée d'un nombre de neurones N, où chacun porte un vecteur poids qui a la même dimension que les données (figure (1.10)), inter-connectés et liés entre eux par une structure de graphe non orienté. Cette structure induit une distance

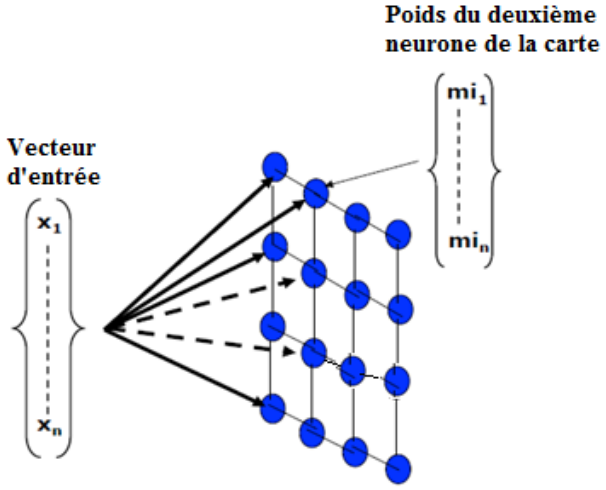

Figure 1.10 – Architecture du réseau de Kohonen

discrète δ sur C : pour toute paire de neurones (c, r) de cette carte, $\delta(c, r)$ est définie comme étant la longueur du plus court chemin entre c et r sur le graphe. Pour chaque neurone c, cette distance discrète permet de définir la notion de voisinage d'ordre d de c :

$$V_c(d) = \{r \in C, \delta(c, r) \leq d\} \tag{1.23}$$

La définition de cette nouvelle notion est une spécificité de la carte de Kohonen. Le voisinage et les liaisons entre neurones sont définis à priori avec l'architecture du réseau et selon la forme de la carte. Les figures (1.11) et (1.12) présentent l'ensemble de ces notions de distance et de voisinage pour une carte topologique constituée par un treillis à deux dimensions.

1.5.3.3 Fondement mathématique des cartes SOM

L'algorithme des cartes auto-organisatrices cherche à minimiser une fonction de coût convenablement choisie. Cette fonction doit tenir compte de deux objectifs, d'une part, de l'inertie interne de la partition dans l'espace de donnée, et d'autre part, chercher à assurer la conservation de la topologie. Pour réaliser ces deux critères, l'idée consiste à généraliser la fonction d'inertie en introduisant dans son expression des termes spécifiques qui sont définis à partir de la distance définie sur la carte et de la notion de voisinage qui lui est attachée. On s'intéresse à la notion de voisinage qui va être introduite à l'aide de fonctions noyaux positives et symétriques K avec $\lim_{|x|\to+\infty} k(x) = 0$. Les fonctions noyaux jouent le rôle de création de zone d'influence autour de chaque neurone c. Quant aux distances $\delta(c, r)$ qui lient le neurone c aux autres neurones (r) de la carte, ils permettent de faire varier l'influence relative des différents neurones : cette influence est quantifiée par $k(\delta(c, r))$. Afin de gérer la taille du voisinage, on utilise la famille de fonction K^T paramétrée par T :

$$K^T(\delta) = K(\delta/T) \tag{1.24}$$

Le paramètre T joue un rôle important pour quantifier l'influence relative entre tout couple de neurones (c, r) de la carte C. Cette influence décroit avec T et devient négligeable lorsque T tend vers 0 (figure (1.11) et (1.12)).

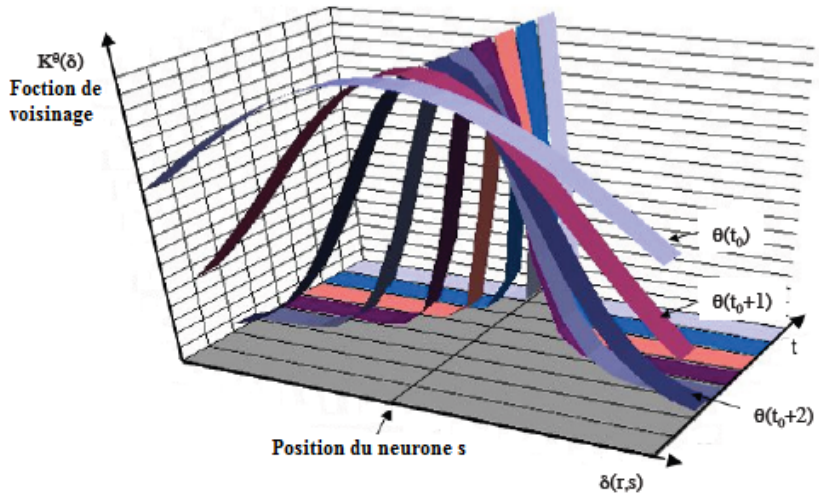

Figure 1.11 – Fonction de voisinage (figure tirée de [28])

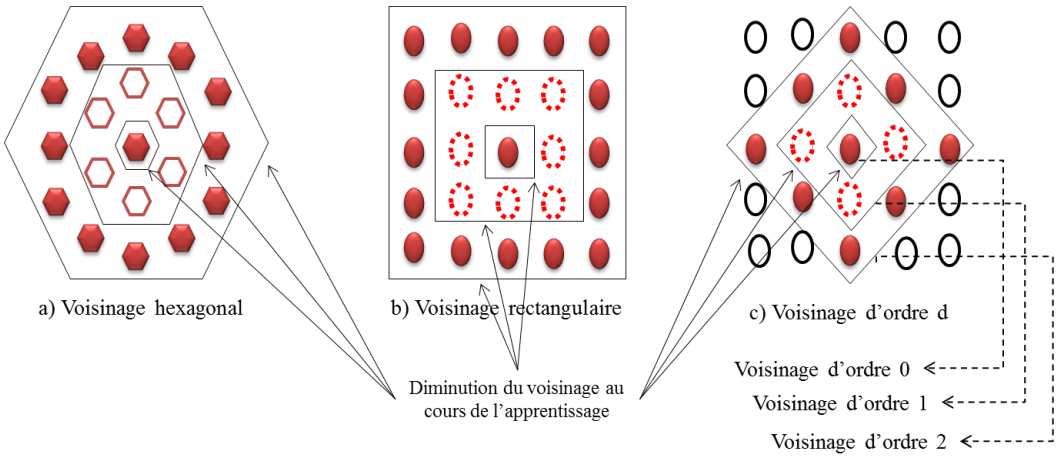

Figure 1.12 – Autres types de voisinage

Pour gérer les zones d'influences pour chaque neurone de la carte on définit :

$$V_c^T = \{r \in C / K^T(\delta(c,r)) > \alpha\} \tag{1.25}$$

Le seuil α gère l'ordre des valeurs significatives prises en compte par le calcul.
L'algorithme des cartes auto-organisatrices cherche à minimiser la fonction énergie suivante :

$$J_{som}^T(\aleph, W) = \sum_{z_i \in A} \sum_{c \in C} K^T(\delta(c, \aleph(z_i)))\|z_i - w_c\|^2 \tag{1.26}$$

où :

- W est l'ensemble des vecteurs référents qui forment la carte, $\aleph$ est la fonction d'affectation et $\aleph(z_i)$ représente le neurone particulier de la carte C qui est affecté à l'observation z_i.

- $\delta(c, \aleph(z_i))$ est la distance entre un neurone c quelconque de la carte et le neurone affecté à l'observation z_i.

Le minimum de cette fonction ne fournit pas seulement une partition formée de sous-ensembles qui sont suffisamment compacts, mais aussi la capacité de définir un ordre induit à partir de la topologie de la carte.
L'expression $J^T_{som}(\aleph, W)$ peut s'écrire sous la forme suivante avec d^T définit une nouvelle distance :

$$J^T_{som}(\aleph, W) = \sum_{z_i \in A} d^T(z_i, w_{\aleph(z_i)}) \tag{1.27}$$

Avec :

$$d^T(z_i, w_{\aleph(z_i)}) = K^T(\delta(c, \aleph(z_i)))\|z_i - w_c\|^2 \tag{1.28}$$

La distance d^T entre z_i et $w_{\aleph(z_i)}$ est une somme pondérée de la distance euclidienne de z_i à tous les vecteurs référents w_c du voisinage du neurone $\aleph(z_i)$.

Pour illustrer et mieux comprendre d'une manière plus intuitive comment se forme l'ordre topologique durant la minimisation de la fonction J^T_{som}, une décomposition de cette dernière en deux termes distincts sera étudiée en détail dans ce qui suit. La fonction de cout peut s'écrire sous la forme :

$$J^T_{som} = \sum_r \sum_{z_i \in P_r \cap A} \sum_c K^T(\delta(c, r))\|z_i - w_c\|^2 = \sum_c \sum_r \sum_{z_i \in P_r} K^T(\delta(c, r))\|z_i - w_c\|^2 \tag{1.29}$$

En décomposant cette expression, la double fonctionnalité attachée à la fonction J^T_{som} apparait :

$$J^T_{som} = \sum_c \sum_{r \neq c} \sum_{z_i \in P_r} K^T(\delta(c, r))\|z_i - w_c\|^2 + \left[K^T(\delta(c, c)) \sum_c \sum_{z_i \in P_c} \|z_i - w_c\|^2\right] \tag{1.30}$$

$$J^T_{som} = \frac{1}{2} \sum_c \sum_{r \neq c} K^T(\delta(c, r)) \left[\sum_{z_i \in P_r} \|z_i - w_c\|^2 + \sum_{z_i \in P_c} \|z_i - w_r\|^2\right] + K^T(\delta(c, c) \sum_c \sum_{z_i \in P_c} \|z_i - w_c\|^2 \tag{1.31}$$

$$J^T_{som} = \frac{1}{2} \sum_c \sum_{r \neq c} K^T(\delta(c, r)) \left[\sum_{z_i \in P_r} \|z_i - w_c\|^2 + \sum_{z_i \in P_c} \|z_i - w_r\|^2\right] + K^T(0) I(W, \aleph) \tag{1.32}$$

On voit l'apparition de deux termes dans la fonction J^T_{som} dont il faut minimiser la somme. L'interprétation de ces deux termes est comme suit :

- Le premier terme veille à assurer la contrainte de conservation de la topologie. En effet, à l'apprentissage, si on a deux neurones c et r voisins sur la carte, alors la fonction noyau K^T a une grande valeur en $\delta(c, r)$, car ce dernier est petit. Dans ce cas

la minimisation de ce premier terme permet de rapprocher les deux sous-ensembles P_c et P_r liés à ces deux neurones. Donc les proximités sur la carte entrainent des proximités dans l'espace de données D. Il faut signaler que la contribution de ce terme à la fonction de cout dépend des valeurs de T : plus T est grand, plus la contribution sera significative.

- Le second terme de l'expression veille à assurer la deuxième contrainte. On constate que ce terme correspond à la fonction de cout définit pour la méthode des k-moyennes pondéré par $K^T(\delta(c,c)) = K(0)$. Donc ce terme a tendance à obtenir une partition de l'ensemble de données D en sous-ensembles compacts, et pour laquelle les vecteurs référents deviennent les centres de gravité des différents sous-ensembles de la partition. Son importance relative dépend aussi du paramètre T : plus T est petit, plus ce terme est pris en considération durant la minimisation.

Cette décomposition de la fonction de cout montre la dépendance de cette dernière au terme T et décompose le processus de minimisation en deux étapes :

- **Étape de conservation d'ordre topologique** : Première étape correspondante aux grandes valeurs de T : à cette étape, le premier terme joue le rôle majeur et l'algorithme a tendance à assurer la conservation de l'ordre topologique.
- **Étape de minimisation de l'inertie** : Seconde étape correspondante aux petites valeurs de T : à cette étape de l'algorithme, c'est le deuxième terme qui joue un rôle prépondérant et l'algorithme minimise la partie de l'expression liée à l'inertie.

D'après cette analyse, on remarque parfaitement le rôle du paramètre de régularisation T qui permet de réaliser un compromis entre les deux termes de la fonction de cout J^T_{som}. Une mauvaise gestion de la décroissance de T peut générer des anomalies dans l'ordre induit par la carte topologique (figure (1.13)) donc il faut veiller à bien choisir des paramètres déterminants :

- L'intervalle de variation de T, la valeur initiale T_{max} et la valeur finale T_{min}
- Le nombre de fois où l'étape itérative est effectuée
- La manière dont le paramètre décroit dans l'intervalle $[T_{max}, T_{min}]$

On peut résumer le résultat de la minimisation de la fonction comme le calcul d'une solution des k-moyennes sous une contrainte d'ordre sur les référents. En effet, on commence par assurer l'ordre topologique, une fois obtenu, on cherche une partition de sous-ensembles qui assurent une haute compacité. On peut donc considérer que la première étape initialise la deuxième étape par des référents qui ont comme propriétés de respecter l'ordre topologique. La minimisation de la fonction J^T_{som}, par un algorithme d'optimisation non adaptatif à T fixée, se fait par itérations successives, chacune se décomposant en deux phases. La première phase affecte l'ensemble des observations et la seconde minimise la valeur de la fonction de coût par rapport aux référents W. Pour plus de détail sur cette version de minimisation nous référons le lecteur à [63]. Dans cette thèse nous nous sommes intéressé à l'algorithme de Kohonen, pour cela nous présentons l'algorithme de minimisation de J^T_{som} proposé par Kohonen.

1.5.3.4 Algorithme d'apprentissage proposé par Kohonen

Pour la résolution de la fonction J^T_{som}, Kohonen présente quelques particularités pour la définition de la mise à jour des poids des neurones. L'algorithme de Kohonen est réalisé

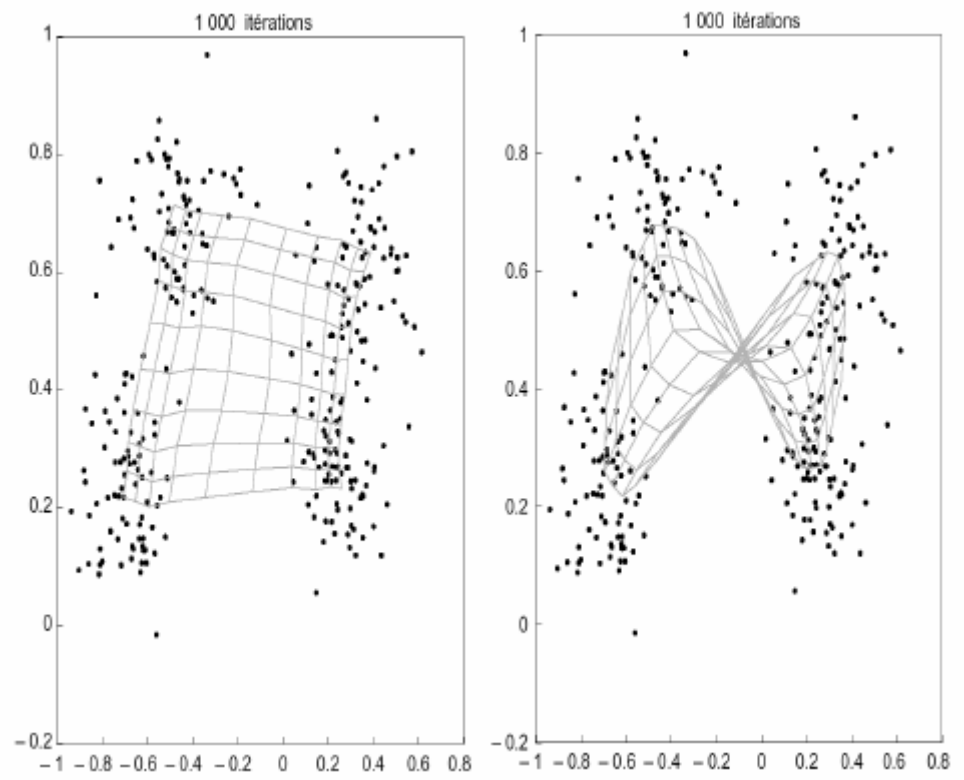

Figure 1.13 – Illustration de l'influence du paramètre T sur l'ordre induit par la carte topologique pour deux décroissances différentes de T et une même initialisation aléatoire au centre du nuage et un même intervalle de croissance (figure tirée de [63])

par itérations successives et procède d'une manière alternative pour la minimisation des paramètres de la fonction de cout. Chaque itération contient donc une phase d'affectation qui affecte l'ensemble des observations et une autre de minimisation qui minimise la fonction de cout associée à la partition. La description du mécanisme des deux phases d'une itération de l'algorithme est résumée dans les étapes suivantes :

— Phase d'affectation : Cette phase minimise la fonction J^T_{som} par rapport à la fonction d'affectation $\aleph$. On suppose que l'ensemble des référents est constant et égal à la valeur calculée précédemment. La fonction d'affectation selon laquelle on trouve le neurone gagnant (apprentissage compétitif) :

$$\aleph(z_i) = argmin_{c=1,\dots,p}\|z_i - W_c\|^2 \tag{1.33}$$

Cette phase permet de définir une fonction d'affectation et donc une partition de l'ensemble des données D. Chaque observation z est affectée au référent le plus proche au sens de la distance. En terme d'apprentissage des RNAs, on peut appeler cette phase : phase de compétition puisque chaque neurone entre en compétition avec les autres neurones de la carte pour gagner l'observation présentée à la carte.

— Phase de minimisation : Il s'agit maintenant de minimiser la quantité J^T_{som} par rapport à l'ensemble des référents W. Cette minimisation est effectuée en gardant la fonction d'affectation $\aleph$ fixée et égale à la fonction calculée durant la phase précédente. Kohonen choisit autre manière de mise à jour des référents. Il suffit de constater que durant

cette phase, il n'est pas obligatoire de trouver le minimum global de la fonction J^T_{som} pour $\aleph$ fixée, il suffit de faire décroître sa valeur. Il est donc possible de définir les poids par une méthode de gradient simple telle que :

$$w_c^t = w_c^{t-1} - \mu^t \frac{\partial J^T_{som}}{\partial w_c^{t-1}} \tag{1.34}$$

Où μ^t est le pas du gradient de l'itération t et :

$$\frac{\partial J^T_{som}}{\partial w_c^{t-1}} = 2 \sum_{zi \in A} K^T(\delta(c, \aleph(z_i)))(z_i - w_c^{t-1}) \tag{1.35}$$

Algorithm 1 Algorithme de Kohonen

Entrées: $T_{min}, T_{max}, T, t, Niter, W$

Sorties: W

Phase d'initialisation :

— Fixer les valeurs de T_{min}, T_{max}, Niter, la taille p de la carte, $t \leftarrow 0$

— Choisir la structure de la carte, les p référents (poids de neurones) initiaux d'une manière aléatoire

Tant Que Niter n'est pas atteint ou les vecteurs référents ne changent plus **faire**

Les p référents W^{t-1} étant connus à la phase précédente :

— Choisir une observation z_i parmi l'ensemble d'apprentissage

— Calculer la nouvelle valeur de T en appliquant la formule suivante :

$$T = T_{max} * \left(\frac{T_{min}}{T_{max}}\right)^{\frac{t}{Niter-1}} \tag{1.36}$$

Pour cette valeur du paramètre T faire :

— Après la compétition lancé entre les neurones, on affecte l'observation z_i au neurone gagnant $\aleph(zi)$ trouvé à partir de la fonction d'affectation (équation (1.33))

— Calcul de l'ensemble des nouveaux référents W^t selon la formule (équation (1.34)) en fonction de leur distance au neurone $\aleph(zi)$ sélectionné à l'étape précédente

$t \leftarrow t + 1$

Fin Tant Que

Cette version a été initialement présentée par Kohonen. Elle diffère de la version d'optimisation globale de l'algorithme présentée dans [63] par l'utilisation d'une seule observation par itération et aussi dans le choix de la fonction d'affectation. La première différence vient du fait qu'on suppose que l'on dispose de toutes les observations de l'ensemble d'apprentissage A, donc la contribution d'une seule observation z_i à la correction de w_c est représentée par le terme de la somme qui est : $K^T(\delta(c, \aleph(z_i)))(z_i - w_c^{t-1})$. À chaque présentation d'une observation z_i les nouveaux référents sont alors calculés pour tous les neurones de la carte C en fonction du neurone sélectionné :

$$w_c^t = w_c^{t-1} - \mu^t K^T \left(\delta(c, \aleph(z_i))\right) \left(z_i - w_c^{t-1}\right) \tag{1.37}$$

Cette phase peut comporter deux autres phases : la première de coopération et le deuxième d'adaptation. Dans la première ; les neurones voisins du gagnant dans la grille coopèrent

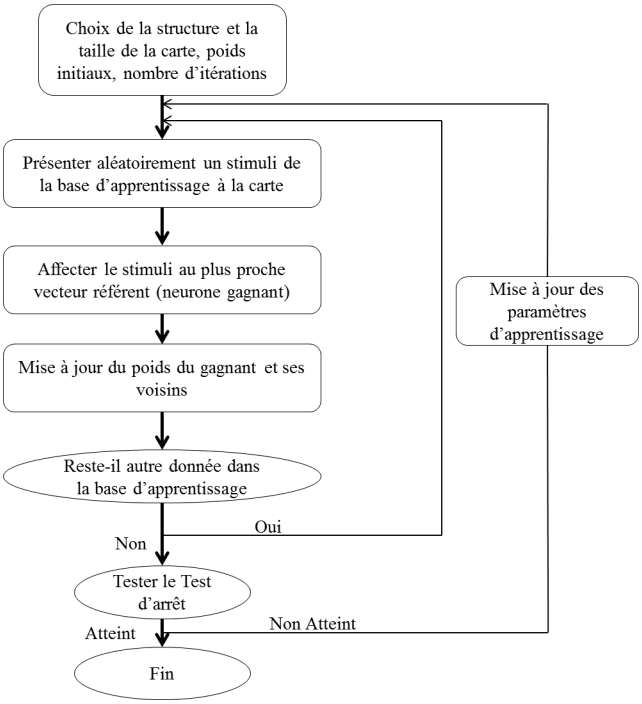

Figure 1.14 – Schéma de l'algorithme d'apprentissage du réseau de Kohonen

entre eux et auront la chance d'être adaptés et s'approchent de l'observation présentée. Quant à la phase d'adaptation, le gagnant et ses voisins augmentent la valeur de leur fonction discriminante relative aux entrées courantes.

Après cette description du déroulement de calcul, nous présentons l'algorithme principal de Kohonen (Algorithme (1)) ainsi qu'un résumé de ses étapes dans le schéma illustratif de la figure (1.14).

1.5.3.5 Propriétés typiques de la carte

La carte auto adaptative dispose de plusieurs propriétés qui la caractérisent des autres réseaux de neurones artificiels et des autres classifieurs proposés par les chercheurs. Ces propriétés intéressantes et particulières font de cet algorithme une méthode très utilisée dans différents domaines, comme : la reconnaissance de la parole arabe [4], les problèmes de finances [10], la prédiction [13], le traitement de texte [15], la classification [45], la compression d'image [79], la quantification vectorielle [14]. Dans les points qui suivent nous citons les caractéristiques de la carte de Kohonen :

- **Notion de voisinage** : cette notion est la particularité caractéristique de la carte auto-adaptative et la clé des autres propriétés. Elle permet de définir un voisinage autour de chaque neurone dans la carte (figure (1.12)) afin d'avoir des classes voisines à la sortie ou des éléments voisins qui appartiennent à la même classe et aussi d'accélérer l'apprentissage. Pour mieux voir l'utilité de cette notion, on suppose que l'espace n'est pas constitué de zones isolées, mais de sous-ensembles compacts. Donc en déplaçant

un vecteur référent vers une zone, on peut se dire qu'il y a probablement d'autres zones dans la même direction qui doivent être représentées par des vecteurs référents. Cela justifie le fait de déplacer les neurones proches du vainqueur dans la grille dans cette même direction (figure (1.15)), avec une amplitude de déplacement moins importante.

- **Préservation des relations topologiques** : l'apprentissage dans la carte veille à ce que les données similaires dans l'espace d'entrée auront des projections proches (neurones proches au sens de la distance) sur la carte. Aussi des observations voisines dans l'espace des données appartiennent après classement à la même classe ou à des classes voisines. Suite à cette propriété, la carte auto adaptative va se déployer durant l'apprentissage jusqu'à ce que sa topologie reflète celle de l'espace d'entrée.
- **Notion de compétition** : Cette notion consiste à spécifier une zone (un sous ensemble de neurones) pour bénéficier d'un changement et s'approcher de l'entrée présentée à la carte dans un instant donnée. Cette zone est constituée du neurone gagnant qui est le plus proche (au sens de la distance) au stimulus présenté et ses voisins. L'activation et l'adaptation de cette zone uniquement a pour objectif de se spécialiser sur des sous-ensembles d'entrées similaires pour devenir des détecteurs de caractéristiques.
- **Notion d'auto-organisation** : Ce terme se manifeste dans la phase d'adaptation où les poids du neurone gagnant et ces voisins sont gratifiés d'une translation vers l'entrée mise en jeu sans intervention d'aucun superviseur. Dépendant seulement des entrées du réseau, l'apprentissage permet de passer d'un état désorganisé, suite à une initialisation aléatoire des poids de neurones (figure (1.16 A)), à un état organisé respectant la topologie des données (figure (1.16 C)).

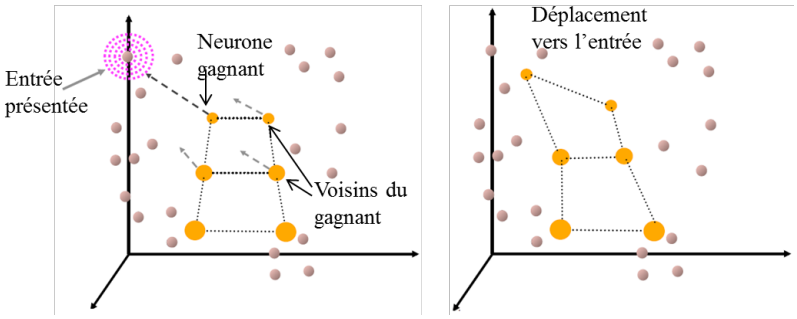

Figure 1.15 – Illustration du déploiement de la carte pour s'approcher de l'espace d'entrée : Déplacement du vecteur gagnant et ces voisins vers l'entrée présentée

1.5.3.6 Cartes topologiques pour la classification automatique

Puisque nous avons bien détaillé le principe et le fonctionnement des cartes topologiques, nous allons montrer l'importance de cette méthode comme un outil très puissant de la classification non supervisée.
Les cartes auto-organisatrices et la classification non supervisée ont des points d'intersection. Puisque les méthodes de la classification automatiques cherchent à former des groupes homogènes qui regroupent les données similaires afin de structurer un ensemble de données. Ces cartes appartiennent à la catégorie des méthodes basées modèles (figure (1.3)). La notion d'ordre topologique constitue l'apport des réseaux de neurones à apprentissage non supervisé

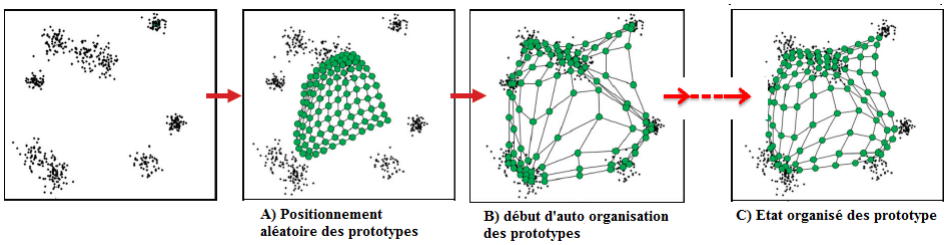

Figure 1.16 – Illustration du déroulement ($A \longrightarrow B \longrightarrow C$) de l'auto-organisation dans la carte de Kohonen

au domaine de la classification automatique, qui est un des grands thèmes abordés en analyse des données [120] [188] [66]. Pour cela la carte auto-organisatrice est un outil très fort de l'analyse de données et plus précisément de la classification non supervisée (clustering) grâce à ses caractéristiques intéressantes qui sont les suivantes :

- Légèreté de calcul, structure simple et dimension inférieur
- Représentation simple des clusters par les centres des vecteurs associés avec chaque neurone
- Possibilité de la comparaison des groupements qui ont été réalisés directement à partir des données
- Facilité du calcul de similarité en calculant seulement les distances entre les projections des données ou des représentants des classes sur la carte
- La topologie de l'ensemble d'entrées est en quelque sorte mappée dans la topologie des poids des neurones de la carte
- Apprentissage non supervisé qui est le cas le plus rencontré dans les problèmes
- Propriété d'auto organisation
- Notion de voisinage qui n'est pas prise en compte par les méthodes de classification classiques

Toutes ces propriétés ont motivé plusieurs chercheurs à utiliser les cartes de Kohonen dans différents domaines dès leur apparition [204] [53].

1.5.3.7 Avantages et limites des cartes auto adaptatives

Le réseau de Kohonen présente un type spécifique dans les réseaux de neurones artificiels par ses propriétés qui le caractérisent. Celles-ci ont attribué à ce type de réseaux plusieurs avantages :

- Léger en terme de coup de calcul : Grâce à la notion de voisinage dans la carte, Kohonen modifie le vecteur référent et ses voisins dans la même itération avec différence d'amplitude de déplacement selon la proximité au neurone gagnant. L'algorithme présente des opérations simples pour diminuer le temps de calcul.
- Portable et simple à utiliser : la légèreté en cout et en calcul et la simplicité de l'algorithme de Kohonen les rond portable dans différents domaines d'application.
- Non linéarité : L'algorithme de Kohonen représente une méthode de projection non linéaire. Cette propriété le caractérise en étant une version développé de l'algorithme (SCL) et une généralisation de la technique (ACP).
- Approximation et visualisation de l'espace d'entrée : Une carte de Kohonen réalise

l'approximation et la représentation graphique d'un ensemble de vecteurs (base d'apprentissage) par le biais d'un ensemble plus petit de vecteurs prototypes (neurones). La position spatiale d'un neurone dans la grille de sortie correspond à des caractéristiques des données en entrée.
- Sélection de caractéristiques : La carte est capable de sélectionner les meilleures caractéristiques pour réaliser une approximation des données d'entrée. Elle trouve une version discrète de la surface principale.
- Réduction de dimension

Malgré les propriétés et les avantages du réseau de Kohonen il souffre de quelques inconvénients :
- Initialisation des poids : l'algorithme impose l'initialisation des poids des neurones aléatoirement et cela influence la qualité des résultats et la vitesse de convergence.
- Nombre de neurones dans la carte : Classiquement, ce nombre est donné au hasard, parfois on a besoin d'ajouter d'autres neurones pour mieux représenter les données discontinues ou supprimer quelques uns car ils n'ajoutent aucune information et ralentissent la vitesse de convergence.
- Le nombre de classes : l'utilisation du réseau de kohonen comme classifieur pose le problème de choix du nombre de classe.
- Une mauvaise gestion de la décroissance de la taille de voisinage peut introduire des anomalies dans l'ordre.

1.6 Classifieurs probabilistes

Nous nous intéresserons dans cette section plus particulièrement aux approches probabilistes désignés pour la classification avec ses deux types. L'utilisation des modèles probabilistes de classification joue un rôle important pour :
— formaliser l'idée intuitive de la notion de classe naturelle
— évoluer de l'approche algorithmique, heuristique et géométrique vers une approche plus statistique

Les modèles probabilistes ont plusieurs avantages :
— Analyse précise et interprétation statistique de certains critères métriques dont les différentes variantes n'étaient pas toujours bien claires : $trace(S_W)$ et $|S_W|$ (S_W est la matrice de variance intra-classe
— Définition de nouvelles variantes répondant à des hypothèses précises
— définir un cadre formel pour proposer des solutions à des problèmes difficiles : nombre de classes, validation des résultats,

Les modèles de mélange apportent une réponse rigoureuse, flexible et interprétable pour les multiples besoins de la classification : classification supervisée ou non, nature des données, choix du nombre de groupes, etc.

1.6.1 Modèle de mélange

Les modèles de mélange sont de plus en plus utilisés en classification automatique vue qu'ils présentent une idée intuitive d'une population composée de plusieurs classes. En plus,

ils ont des liens forts avec des méthodes de références comme l'algorithme des k-means. Ils ont aussi la capacité de traiter de manière assez naturelle de nombreuses situations particulières.

Définition 5. *Une loi de mélange fini p sur un espace χ est une loi de probabilité s'exprimant comme une combinaison linéaire de plusieurs lois de probabilité $p_1, ..., p_g$ sur χ. Autrement dit, il existe g coefficients $\pi_1, ..., \pi_g$ ($\pi_k > 0$ et $\sum_{k=1}^{g} \pi_k = 1$) tels que, pour tout $x_1 \in \chi$:*

$$p(x_1) = \sum_{k=1}^{g} \pi_k p_k(x_1) \tag{1.38}$$

Les π_k et p_k sont respectivement appelées proportions et composantes du mélange.

Le modèle de "mélange fini de lois de probabilité" consiste à supposer que les données proviennent d'une source contenant plusieurs sous-population. Chaque sous-population est modélisée de manière séparée. La population totale est un mélange de ces sous-populations. Le modèle résultant est dit alors un modèle de mélange fini.

1.6.2 Algorithme EM

L'algorithme EM [31][138] est un algorithme itératif qui permet de trouver un maximum local de fonction de vraisemblance des observations, lorsque chaque observation contient une partie cachée. Ainsi, on suppose que chaque donnée est un couple de types (x, z) où x est sa partie observable et z sa partie cachée. Nous supposons connus d'une manière explicite la forme de la fonction densité jointe $p(x, z; \theta)$ ou θ est l'ensemble des paramètres du modèle à estimer. On suppose que l'on dispose d'une série de données indépendantes : $(x_1, z_1), (x_2, z_2), ..., (x_N, z_N)$, pour lesquelles x_i sont les parties qu'on a réellement observées et les z_i sont les parties inconnues (cachées).

Nous souhaitons par la suite maximiser le logarithme de la vraisemblance des parties des données réellement observées $A = x_1, x_2, ..., x_N$ dont le logarithme est égal à :

$$lnV(A; \theta) = lnV(x_1, x_2, ..., x_N; \theta) = \sum_{i=1}^{N} lnp(x_i; \theta) \tag{1.39}$$

où $p(x; \theta)$ est la fonction densité de la partie observée x. En pratique $p(x; \theta)$ est calculable en marginalisant la fonction densité $p(x, z; \theta)$ ($p(x; \theta) = \int p(x, z; \theta)\, \mathrm{d}z$), ce qui donne souvent une fonction log-vraisemblance $lnV(x_1, x_2, ..., x_N; \theta)$ qui n'est pas simple à optimiser.

L'algorithme EM proposé par Dempster et al [54] maximise l'expression (1.39) en utilisant le log-vraisemblance des données entières $lnV(x_1, x_2, ..., x_N, z_1, z_2, ..., z_N; \theta)$. On désigne par $Z = z_1, z_2, ..., z_N$ l'ensemble des parties correspondantes et non observées. Chaque itération de l'algorithme EM comporte deux étapes :

— L'étape d'Estimation (E)

— L'étape de Maximisation(M)

Ainsi à l'itération t ces deux étapes se présentent de la manière suivante :

— Étape E On suppose à cette étape, que la fonction densité de la partie cachée conditionnée par la partie observée (z/x) correspond à la valeur du paramètre θ^{t-1} calculée à l'itération précédente (ou égale à l'initialisation θ^0 si $t = 1$; cette fonction densité s'écrit donc $p(z/x, \theta^{t-1})$. On calcule alors l'espérance :

$$Q\left(\theta, \theta^{t-1}\right) = E\left[lnV(A, Z/\theta)/A, \theta^{t-1}\right] \tag{1.40}$$

$$= \int lnV(A, Z/\theta) p(Z/A, \theta^{t-1})\, \mathrm{d}z \tag{1.41}$$

$$= \int lnV(A, Z/\theta) \Pi_{i=1}^{N} p(z_i/x_i, \theta^{t-1})\, \mathrm{d}z_i \tag{1.42}$$

Cette expression qui est parfois appelée "la vraisemblance relative" se justifie "intuitivement". En effet, étant donné qu'on ne connait pas les valeurs des variables cachées z_i associées aux observations $x_i \in A$, on calcule l'espérance du log-vraisemblance relativement aux variables cachées.

— Étape M Ayant calculé $Q\left(\theta, \theta^{t-1}\right)$ à l'étape E, il s'agit dans cette étape de maximiser cette expression par rapport à θ. on prend alors :

$$\theta' = argmax_\theta Q\left(\theta, \theta^{t-1}\right) \tag{1.43}$$

Il est démontré alors que chaque itération (E-M) fait croitre la fonction log-vraisemblance (1.39) $(lnV(A;\theta) \geq lnV(A;\theta_{t-1}))$ [54]. Ainsi, l'algorithme EM se présente de manière suivante :

Algorithm 2 Algorithme EM

Entrées: θ^0, N_{iter}

Sorties: θ

Phase d'initialisation : Choisir les paramètres initiaux θ^0, N_{iter}

Itération de base :

Tant Que La stabilisation de θ^t n'est pas atteinte ou $t \leq N_{iter}$ **faire**

— Étape E : Estimer l'expression $Q\left(\theta, \theta^{t-1}\right)$ définie par l'expression (1.40)

— Étape M : Maximiser $Q\left(\theta, \theta^{t-1}\right)$ par rapport à θ, prendre $\theta_t = argmax_\theta Q\left(\theta, \theta^{t-1}\right)$

Fin Tant Que

1.6.3 Cartes topologiques probabilistes

La carte auto-organisatrice probabiliste, appelée en anglais Probabilistic Self Organizing Map (PrSOM) , a été proposée par [8]. Ce modèle probabiliste représente une généralisation du modèle classique des cartes topologiques. Bien que les deux modèles des cartes topologiques partagent les mêmes propriétés, le modèle probabiliste permet non seulement d'obtenir une quantification de l'espace des données, mais aussi une estimation des densités locales. Pour cela nous nous sommes intéressés à ce modèle pour la compression de la parole (chapitre 4).

Nous présentons une description des points communs et des points qui différent les deux modèles des cartes topologiques. Les points communs sont :

— La taille du voisinage décroit durant l'apprentissage sous le contrôle du paramètre T.
— La dernière fonction d'affectation de l'algorithme définie la partition associée à la carte d'après l'ordre topologique obtenu à la fin de l'apprentissage.
— L'ensemble des données est divisé en M sous-ensembles : chaque neurone c de la carte représente un sous-ensemble $P_c = \left\{z/\chi^{Niter}(z) = c\right\}$.

Les points différents sont :

— La carte et la partition obtenues par PrSOM sont déterminées en tenant compte des distributions de probabilités tandis que SOM utilise la distance euclidienne.
— L'estimation des probabilités permet d'obtenir des informations supplémentaires pouvant être utilisées avec profit dans des applications (classification).
— Il n'existe pas, pour l'algorithme PrSOM, de version stochastique : l'estimation de la variance demande de prendre en considération toute la base d'exemples avant de modifier les différentes valeurs des paramètres.
— L'algorithme PrSOM permet d'obtenir un grand nombre d'informations supplémentaires sur l'ensemble des observations étudié (recherche des données aberrantes, calcul de probabilité...).
— Le PrSOM ne peut être utilisé que si le nombre d'observations est assez grand pour permettre une estimation suffisamment précise des variances attachées aux gaussiennes.

Dans le cas des données continues, cet algorithme suppose que la distribution de probabilité $p(x/c)$ prend une forme analytique qui est représentée par une loi Gaussienne sphérique. Chaque fonction densité est définie par son vecteur moyen qui est le référent w_c ainsi que l'écart type σ_c qui varie maintenant avec chaque neurone (cellule) c. Les mélanges de fonctions considérées sont donc des mélanges de fonctions Gaussiennes. Les paramètres à estimer dans ce cas représentent, $\theta = (W, \Sigma)$, l'ensemble des référents W et l'ensemble des écarts-types $\Sigma = \{\sigma_c I, c = 1...k\}$.

Le formalisme bayésien, introduit par Luttrel [156], est utilisé dans ce modèle pour définir le mélange de densités des cartes topologiques. Aussi, les propriétés de Markov sont utilisées afin de simplifier le calcul des probabilités. Toutes ces notions, ainsi que le fondement du PrSOM seront détaillés dans le chapitre 4 décrivant notre contribution qui s'intéresse au choix de l'architecture du PrSOM pour la compression du signal parole.

1.6.4 Réseaux neuronaux probabilistes

Parmi les différents types de réseaux de neurones, les réseaux probabilistes représentent des méthodes importantes pour la résolution des problèmes de la classification. Le réseau de neurones probabiliste (PNN) a été introduit par Specht dans son article [194]. Un réseau neuronal probabiliste a une similarité avec le modèle de rétro-propagation dans la façon dont ils propagent l'information. Mais le réseau PNN est dis-similaire aux autres dans la procédure d'apprentissage. Il apporte une information supplémentaire en intégrant une notion de densité de probabilité qui est estimée en utilisant le théorème de Bayes ainsi que la théorie des fenêtres de Parzen. Pour cette raison, on dit que ce réseau est dérivé d'un réseau bayésien. les PNNs utilisent des fonctions radiales pour construire une fonction de décision locale centrée sur un sous-ensemble de l'espace d'entrée [39]. La fonction de décision globale est

calculée par la somme de toutes les fonctions locales [132]. Ce calcul résout le problème des minima locaux.

1.6.4.1 Principe des PNNs

Grâce à leurs propriétés, ces réseaux sont très utilisés dans le domaine de la classification. Il faut noter que ces modèles sont qualifiés de réseau à apprentissage supervisé. Dans ce contexte, chaque vecteur observé x de d caractéristiques est placé dans une des classes $C_i, i = 1, 2, ..., m$ prédéfini (m est le nombre de classes possibles) dans lequel x peut appartenir. Il faut signaler que l'efficacité d'un PNN en tant que classificateur est limitée par la dimension du vecteur d'entrée et le nombre de classes possibles.

Les informations connus à priori sur certaines variables sont très utiles et doivent être exploitées et introduites en tant qu'a priori sur les paramètres recherchés. La densité a priori, nous indique la valeur que doit avoir les paramètres avant de voir les données. On combine ensuite cette information a priori avec les informations issues des données, en utilisant la règle de Bayes. Cette règle associe un degré de confiance dans les différentes valeurs du vecteur des poids. Cette confiance est définie par une distribution de probabilité des poids du réseau qui sont alors considérés comme des variables aléatoires. Cette distribution est initialisée à une certaine distribution a priori. Après observation des données, celle ci peut être convertie au moyen du théorème de Bayes en une nouvelle distribution a posteriori des poids qui peut être utilisée pour évaluer les sorties du réseau pour un exemple inconnu.

Dans les réseaux PNNs, reposant sur la théorie bayésienne qui permet de relier la distribution à priori $p(C_i)$ et à posteriori $p(C_i|x)$, la probabilité $P(C_i/x)$ pour x d'appartenir à la classe C_i est donnée par :

$$p(C_i|x) = \frac{p(x|C_i)p(C_i)}{\sum_{\forall C_j} p(x|C_j)p(C_j)} \tag{1.44}$$

C'est le modèle de la règle de Bayes de probabilité conditionnelle mis en œuvre par le classificateur de Bayes où $p(x|C_i)$ est la probabilité qu'un objet de classe C_i possède les caractéristiques de x. $p(C_i|x)$ est la probabilité qu'un objet de caractéristique x soit de la classe C_i. Aussi, $p(C_j)$ est la probabilité de choisir un échantillon de la classe C_j.

Le PNN est connu pour être un classifieur optimal de Bayes, ce qui signifie que lorsque le nombre d'échantillons utilisés pour former le réseau s'approche de l'infini, la densité de probabilité estimée par le modèle s'approche de la vraie fonction de densité de probabilité.

La règle de décision d'erreur minimale (règle bayésienne) attribue à l'objet x la classe de probabilité conditionnelle sachant x maximale :

$$p(C_i|x) > p(C_j|x); \forall j = 1, ..., m; j \neq i \tag{1.45}$$

Les PNNs reposent sur une autre technique pour l'estimation des probabilités obtenues par la formule de Bayes. Ces probabilités sont estimées à partir de l'ensemble d'apprentissage (ensemble de paires (x_i, C_i)) en utilisant la technique de fenêtrage de Parzen pour l'estimation

des fonctions de densités. L'estimateur utilisé pour les réseaux PNNs est :

$$p(x|w_i) = \frac{1}{2m\pi^{p/2}\sigma^p} \sum_{i=1}^{m} exp - \left[\frac{(X - X_i^a)^2}{2\sigma^2}\right] \tag{1.46}$$

X_i^a est le ième échantillon appartenant à la classe C_a et σ est un paramètre de lissage qui sert à contrôler l'activation de la fonction exponentielle.

1.6.4.2 Architecture d'un PNN

Les PNNs ont une structure organisée en quatre couches (figure (1.17)). La première couche est la couche d'entrée qui sert seulement a présenter les données. La deuxième couche (couche d'exemples) contient un neurone pour un couple (vecteur, classe). La troisième couche (couche de sommation) est une couche d'évaluation de densité où chaque neurone activé par la fonction sigma (noyau) est évalué. La quatrième couche (couche de sortie) de probabilité est une couche de calcul des statistiques. La couche d'exemples a une similarité avec le réseau à base radiale ; Cependant, la couche de sommation est similaire au réseau à compétition. La couche d'exemples a le nombre de neurones identiques aux nombres d'échantillons d'entrées et la couche de sommation a le même nombre de neurones que la classe cible.

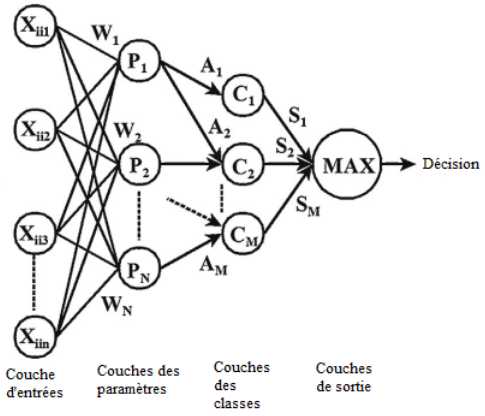

Figure 1.17 – Architecture générale d'un réseau probabiliste

1.6.4.3 Apprentissage du PNN

Le PNN est un réseau d'apprentissage supervisé qui met en œuvre l'approche de Bayes pour la classification dans son modèle d'apprentissage. Le réseau sélectionne une base d'apprentissage et estime la probabilité de l'échantillon en utilisant la fonction à base radiale. Le PNN est caractérisé par son fonctionnement en parallèle et ne nécessite aucune rétro-propagation vers l'entrée ; Ainsi, l'apprentissage du PNN est instantané et plus facile à apprendre que d'autres réseaux neuronaux.

Lorsqu'une entrée est présentée au réseau, la première couche calcule les distances entre le vecteur d'entrée et les vecteurs d'entrées d'apprentissage, puis produit un vecteur avec

des éléments qui indiquent à quel point l'entrée est proche de l'entrée d'entraînement. La couche de sommation somme ces contributions pour chaque classe d'entrées pour produire un vecteur de probabilités qui représente la sortie nette. Enfin, une fonction de transfert compétitif sur la sortie de la couche de sommation sélectionne le maximum de ces probabilités et produit un pour cette classe et 0 pour les autres classes.

L'algorithme d'apprentissage de ce type de réseaux est comme suit :

1.6.4.4 Avantages et inconvénients du PNN

Le PNN se caractérise par de nombreux avantages :

- La vitesse d'apprentissage : ce réseau apprend très vite car il est créé après un seul passage sur l'ensemble d'apprentissage [194] [22].
- La base mathématique solide : le principe de fonctionnement des PNNs est basé sur une base mathématique très solide [31].
- L'utilisation interactive : Ces réseaux peuvent être utilisés interactivement.
- La convergence est globale : ces réseaux ne souffrent pas du problème des minima locaux [55].
- La faible complexité : un seul paramètre de lissage est à calibrer.

D'autre part, ces réseaux souffrent aussi de quelques inconvénients importants :

— Le nombre de neurones cachés est fixé dans le réseau PNN et égal au nombre d'échantillons de l'apprentissage. Cela crée souvent des problèmes avec un ensemble d'apprentissage très grand qui contient dans la majorité du temps beaucoup de redondances.

— Le choix du paramètre de lissage est très important et pose un vrai problème pour ces réseaux. Ce paramètre peut affecter sérieusement la généralisation du réseau [154].

1.7 Discussion

Les méthodes hiérarchiques représentent aussi une catégorie très importante de méthodes de classification qui, contrairement aux méthodes de partitionnement, construisent les classes progressivement. Elles permettent d'obtenir des partitions non contradictoires qui sont présentées sous forme d'un arbre de classification (plusieurs partitions). Il y a deux genres de méthodes de classification hiérarchiques : La classification ascendante hiérarchique (CAH) et la classification descendante hiérarchique (CDH).

— La CAH consiste à commencer par n (nombre d'objets de la population) classes qui contiennent chacune un individu et essayer par la suite, dans chaque itération, de fusionner deux ou plusieurs classes choisies selon un critère de similarité pour former une nouvelle classe. Cette opération est itérée jusqu'à l'obtention d'une unique classe regroupant l'ensemble des individus. Ce processus permet de construire un arbre (hiérarchie) dans un ordre ascendant qui peut être coupé à différents niveaux pour obtenir un nombre de classes plus ou moins grand.

— La CDH parte de l'ensemble de tous les individus comme une seule classe puis divise successivement les classes en classes plus raffinées. Le processus de division des classes marche jusqu'à que chaque classe contienne un seul point ou bien si l'on atteint un

nombre de classes désiré. La technique de division de la CDH veille à deux points : chercher la classe qui va être divisée et choisir un mode d'affectation des objets aux sous classes résultantes de la division.

Les travaux [29][44][123][188] présentent des détails plus riches sur ce type de méthodes de classification.

Bien que ces méthodes sont applicables à tout type d'attributs et présentent une facilité de manipuler toute forme de similitude et une flexibilité concernant le niveau de granularité, elles ont une difficulté à choisir la droite arrêtant des critères et sont très couteuses en termes de calcul par rapport aux méthodes de partitionnement. Nos travaux, que nous allons présenter dans les chapitres suivants, présentent de nouvelles méthodes de clustering s'inspirant du principe des méthodes de classification hiérarchique. En plus, ces méthodes proposent des solutions pour remédier aux limites de sélection de paramètres des algorithmes neuronaux et de partitionnement. La combinaison des deux types de méthodes (hiérarchiques et neuronaux ou de partitionnement) est, sans aucun doute, parmi les méthodes de résolution les plus puissantes car elles exploitent la puissance de deux méthodes et les combinent en un seul algorithme. Nous exploitons cette hybridation pour résoudre le "big challenge" de la classification automatique qui est le choix du nombre de classes d'un ensemble de données ainsi que la meilleur partition.

Les réseaux connexionnistes permettent de créer de l'intelligence artificielle à l'aide du moteur essentiel de l'apprentissage. Celui-ci permet au réseau d'assimiler un traitement d'information à travers une fonction et de le reproduire pour les données qui lui seront ensuite présentées. Les réseaux de neurones reposent à présent sur des bases mathématiques solides qui permettent d'envisager des applications dans presque tous les domaines : la finance et la gestion en tant qu'un outil d'aide à la décision, l'archéologie, la médecine et les télécommunications, l'industrie et à grande échelle, la classification.

Les RNAs sont connus par leurs avantages divers qui leur donnent un pouvoir particulier. Parmi ces avantages figurent :

- La capacité de représentation de fonction (linéaire ou pas, simple ou complexe) et d'apprentissage à partir d'exemples représentatifs.
- La capacité de construction automatique du modèle.
- Résistance au bruit ou au manque de fiabilité des données.
- Comportement moins mauvais en cas de faible quantité de données.
- Simplicité de manipulation grâce à l'idée d'apprentissage qui est plus simple à comprendre que les complexités des statistiques multivariées.

Sur un autre volet, ces réseaux souffrent de quelques inconvénients qui influencent leur résultat. Ces inconvénients peuvent être résumés comme suit :

- Absence de méthode systématique permettant de définir les paramètres de l'architecture des réseaux comme : la meilleure topologie du réseau, le nombre de couches, le nombre de neurones à placer dans la (ou les) couche(s) cachée(s)...
- Le choix des valeurs initiales des poids du réseau et le réglage du pas d'apprentissage sont critiques et jouent un rôle important dans la vitesse de convergence.
- Le problème de sur apprentissage.

— Le codage (les valeurs des poids) de la connaissance acquise par un réseau de neurones est difficile à interprété pour l'utilisateur.

Comme nous le verrons dans la description des parties expérimentales de nos travaux de recherche, l'utilisation des réseaux de neurones en pratique nécessite une mise en œuvre rigoureuse. Nous nous intéressons alors à pallier aux faiblesses des RNAs, que nous venons d'exposer, en proposons des méthodes prometteuses dans les chapitres suivants.

1.8 Conclusion

En introduction de ce chapitre, nous avons évoqué un ensemble de notions et de concepts liés à l'apprentissage artificiel. Ensuite, nous avons introduit la thématique de classification avec ses différents types en plus des notions fondamentales liées à celle-ci. Trois approches de classification, approches neuronales déterministes, probabilistes et les méthodes de partitionnement dont chacune contient plusieurs méthodes, ont été par la suite exposées dans ce chapitre pour le problème général de la classification. Nous nous sommes limités aux méthodes neuronales les plus utilisées pour la classification. Pour une étude profonde des différentes méthodes, nous référons le lecteur à la thèse de Boubou Mounzer [34]. Il présente un tableau qui englobe un sommaire des méthodes de classification sauf les méthodes neuronales. Nous avons terminé ce chapitre par une discussion sur une autre catégorie de méthodes de classification ainsi que les points positifs et négatifs des RNAs.

Notre objectif était de donner une vision détaillée sur les méthodes que nous avons utilisées pour nos contributions. Aussi, Nous avons mis l'accent sur les différents avantages et inconvénients des méthodes exposées pour essayer de prendre en charge de résoudre les problèmes de ces méthodes.

Notons enfin, que les chapitres suivants de ce manuscrit seront consacrés aux contributions. Dans ces chapitres, Nous nous sommes attachés à résoudre les problèmes des méthodes de classification évoquées dans ce chapitre. Ces problèmes seront décrits avec plus de détails dans les chapitres suivants. Les méthodes proposées sont appliquées à des domaines réels. Le chapitre suivant traite le problème de sélection d'architecture du réseau probabiliste PrSOM en proposant la nouvelle méthode HPrSOM dédiée à la détermination du code-book optimal pour la compression du signal parole.

Chapitre 2

Sélection d'architecture neuronale du PrSOM pour la construction du code-book optimal du signal des chiffres arabes

2.1 Introduction

Comme nous avons introduit dans le chapitre précédent, la sélection de l'architecture neuronale et l'estimation de paramètres représentent un domaine de recherche qui occupé une partie importante des travaux de la littérature. La version probabiliste de la carte topologique est l'une des méthodes d'apprentissage qui n'a pas gagné un grand intérêt pour le problème de sélection d'architecture. Motivé par cette lacune, nous proposons, dans ce chapitre, une nouvelle méthode (H-PrSOM) itérative qui s'inspire du processus des méthodes hiérarchiques et des méthodes de "clustering ensemble". En effet, nous définissons une nouvelle méthode divisive qui consiste à diviser l'ensemble de données en sous ensembles disjoints. A l'aide de cette méthode, nous générons plusieurs initialisations possibles pour la carte probabiliste (nombre de neurones ainsi que leurs poids). A chaque itération, une fois initialisé par notre nouvelle méthode, l'algorithme PrSOM classique joue le rôle de quantificateur local qui cherche la meilleure quantification du signal parole à l'itération courante. Grâce à l'indice d'évaluation, toutes les quantifications sont évaluées pour sélectionner la meilleure quantification.

Nous organisons la présentation de ce chapitre en cinq sections. Pour commencer, nous décrivons les spécificités du signal parole. Ainsi nous signalons la nécessité de la compression de la parole avec l'augmentation de la quantité de données disponibles. Nous réservons la section suivante à la description de la quantification vectorielle (QV) comme l'une des méthodes de compression avec perte ainsi que la détermination du dictionnaire optimal avec les cartes topologiques. Suite à la capacité des cartes topologiques pour la QV,

nous décrivons le fondement de la carte probabiliste en détail. Nous introduisons dans la quatrième section la nouvelle approche H-PrSOM. Afin d'évaluer la performance de cette approche, nous effectuons des tests expérimentaux sur une base de données de signal parole des dix premiers chiffres arabes. Nous effectuons aussi une comparaison entre la version classique du PRSOM et la nouvelle méthode H-PrSOM.

Ce chapitre a fait l'objet d'un article soumis dans le journal intitulé "International Journal on Artificial Intelligence Tools (IJAIT)". Il fait aussi l'objet d'une communication orale effectuée dans le congrès international : AAFD δ SFC 2016 : Conférence Conjointe Francophone sur la Science des Données [100].

2.2 Compression de la parole

La parole représente une information qui est véhiculée par les ondes sonores ou même c'est un bruit qui a un sens. Elle est un moyen de communication par excellence qui occupe une position privilégiée comme vecteur d'information dans notre société humaine. Son traitement joue un rôle fondamental d'une importance croissante et connait une expansion fulgurante avec le développement des moyens et des techniques de communication : communiquer en temps réel n'importe où dans le monde entier et avec des moyens variés. Cela a crée une augmentation perpétuelle du nombre de données. Par conséquent, le stockage et le transfert de données est devenu grand problème. Dés lors, la compression de la parole constitue une tâche primordiale pour la réduction du débit de données à transmettre ou à stocker sans une perte cruciale de l'information. Cette technique consiste à convertir la parole humaine en une représentation codée et réduite tout en préservant l'information essentielle qu'elle porte. Le codage fourni peut être décodé par la suite pour produire une approximation du signal original.

Les techniques de compression se répartissent en deux familles [117][107] : les algorithmes réversibles ou sans perte, dits aussi sans bruit, et les algorithmes irréversibles, dits avec perte. Les premiers ont comme but de restituer à l'identique les données originales en représentant la redondance des caractères sous une forme réduite sans perdre la moindre information. Le signal résultant est le même que l'original. Les seconds éliminent des données non essentielles. Le signal résultant est similaire à l'original. En particulier pour la parole, le signal exhibe de grandes redondances dues au mécanisme physique du trait vocal. De plus, la perte d'information peut être acceptable à un certain niveau puisque les différences de phases ne sont pas perceptibles par l'oreille humaine pour la plupart des fréquences. Tout cela montre que les méthodes de compression avec perte sont les plus utilisées pour la compression de la parole.

La quantification vectorielle est la méthode de compression avec perte la plus connue et qui a déjà montré ses performances pour la compression de la parole et l'image [51][124].

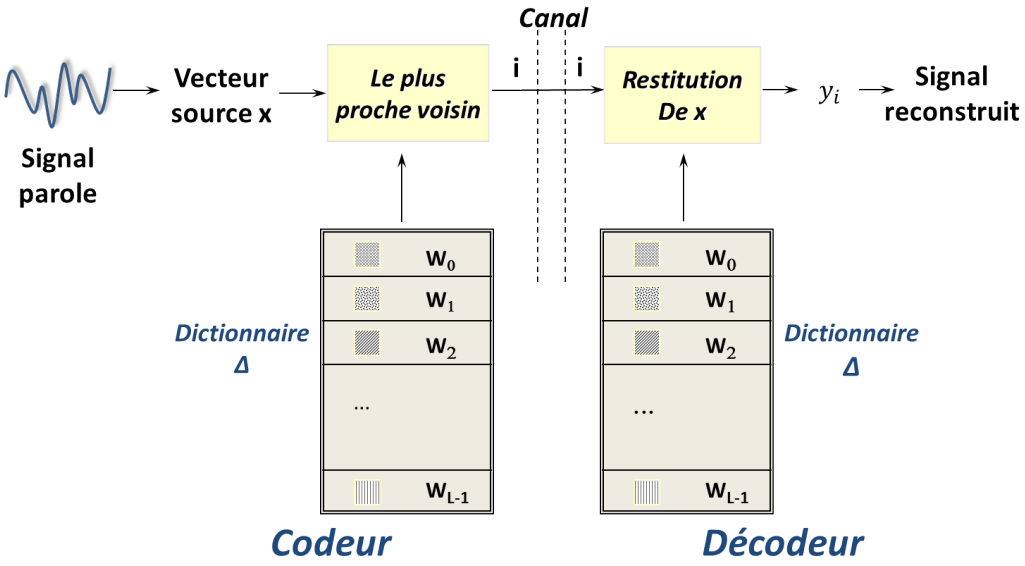

Figure 2.1 – Schéma général illustrant le principe de quantification vectorielle

2.3 Quantification vectorielle

La quantification consiste en l'approximation d'un signal d'amplitude continue par un signal d'amplitude discrète. La quantification vectorielle [89] consiste à quantifier un ensemble de valeurs conjointement, comme un seul bloc ou vecteur [149][89]. Plus précisément, cette technique représente tout vecteur x de dimension k par un autre vecteur y_i (appelé vecteurs représentants, vecteurs de reproduction ou code-vecteurs) de même dimension appartenant à un ensemble fini $\triangle$ de L vecteurs appelé dictionnaire.

Mathématiquement, la quantification vectorielle est définie comme une application, qu'on appelle Q, de R^k vers $\triangle$ (équation (2.1)).

$$\begin{array}{rccc} Q & : & R^k & \rightarrow & \triangle \\ & & x & \mapsto & Q(x) = y_i \end{array} \tag{2.1}$$

Le dictionnaire $\triangle$ est indexé par l'ensemble des indices $I = \{0, 1, ..., L-1\}$ et l'ensemble des vecteurs $y_i (i \in I)$. Nous écrivons alors : $\triangle = \{y_i \in R^k / i = 1, 2, ..., L\}$

La compression est fortement connectée à la classification car elle attribue un code à un ensemble de paramètres d'entrées. Cela est déterminé implicitement par l'application Q qui donne une partition de l'espace source R^k en L classes C_i associées à y_i et déterminées par : $C_i = \{x \in R^k / Q(x) = y_i\}$. Cette partition de R^k satisfait les deux contraintes de partitionnement ; séparation et cohérence.

Un quantificateur vectoriel se décompose généralement en deux fonctions séparées : l'encodeur et le décodeur (figure (2.1)). La première fonction joue le rôle le plus important de

la compression. Elle consiste à rechercher dans le dictionnaire construit $\triangle$ le vecteur-code y_i le plus proche du vecteur x du signal d'entrée. Cette recherche est faite pour chaque vecteur du signal d'entrée. La notion de proximité, encore appelée mesure de distorsion entre les vecteurs x et y_i, est modélisée par une fonction de distance entre le vecteur x et le vecteur reconstruit y_i.
Chaque vecteur y_i représente une région R_i (classe) qui est définie comme suit :

$$R_i = \left\{ x \in R^k / Q(x) = y_i, \;\; si \;\; d(x, y_i) \leq d(x, y_j), \forall j \neq i \right\} \tag{2.2}$$

Tous les vecteurs x appartenant à la même région R_i sont représentés par le même vecteur y_i du dictionnaire. C'est cette représentation qui élimine la redondance.
Jusqu'à maintenant on est arrivé à la création du dictionnaire de codes. L'étape de compression d'information est réalisée via la transmission ou le stockage seulement de l'indice du vecteur-code y_i minimisant le critère de distorsion au lieu du vecteur lui même. C'est à ce niveau que l'information est perdue et cela fait que la QV est une opération irréversible, le signal original ne pourra plus être restitué.
L'information codée est reçue par le décodeur qui réalise l'opération de décompression de l'information en reconstruisant le vecteur source. La réalisation de cette tâche nécessite la disposition d'une réplique du dictionnaire que le décodeur consulte afin de restituer le vecteur-code correspondant à l'indice reçu.

2.3.1 Détermination du dictionnaire optimal

Le succès d'un quantificateur vectoriel dépend du choix du dictionnaire : Si le code-book contient des blocs bien représentatifs, on aura une bonne qualité de compression. Par contre, les mots qui ne sont pas similaires aux blocs à coder ne permettront de reproduire que des vecteurs de faible qualité [142]. Par conséquent, le choix du dictionnaire optimal représente une tâche difficile de la quantification vectorielle qui consiste à optimiser la distorsion moyenne.
Pour une distribution statistique donnée de la source et un débit donné, le quantificateur optimal est celui qui minimise la distorsion moyenne [89][60] définit par :

$$D = E\left[D(X, Q(X))\right] = \int_{x \in R^k} d(x, Q(x)) f_X(x)\, \mathrm{d}x \tag{2.3}$$

X est un vecteur source aléatoire dans R^k, avec la fonction densité de probabilité $f_X(x)$ spécifiée et qui correspond à la fonction densité de probabilité conjointe des composantes X_i du vecteur X.
L'expression de la distorsion moyenne peut être donnée par :

$$D = \sum_{i=0}^{L-1} \int_{x \in R_i} d(x, y_i) f_X(x)\, \mathrm{d}x \tag{2.4}$$

Cette expression est due à la définition du quantificateur vectoriel ayant pour dictionnaire $\triangle$ et pour partition les classes R_i.
L'utilisation d'un ensemble de vecteurs d'apprentissage x_j (suffisamment grand et assez

représentatif de la statistique de la source à coder), donne une nouvelle estimation de l'opérateur de distorsion moyenne :

$$D = \sum_{i=0}^{L-1} \sum_{x_j \in R_i} d(x_j, y_i) \tag{2.5}$$

L'optimisation de la distorsion moyenne D est très complexe puisqu'elle consiste à chercher deux optimums en même temps : la meilleure partition et les meilleurs vecteurs-code qui minimisent la distorsion moyenne D. Dans la pratique et comme les méthodes de classification, la conception d'un quantificateur optimal est réalisée d'une manière itérative en vérifiant les deux conditions d'optimalité suivantes [89][60] :

— Pour un dictionnaire $\triangle$ donné, la partition optimale est celle qui vérifie la règle du plus proche voisin définit par :

$$R_i = \{x / d(x, y_i) \leq d(x, y_j), j \neq i, \forall j \in \{0, 1, 2, ..., L-1\}\}$$

— Pour une partition donnée, le vecteur représentant y_i doit minimiser la distorsion associée à la cellule R_i. Le vecteur y_i est donc le centroide de cette cellule.

L'élaboration du dictionnaire d'un quantificateur vectoriel se fait à partir d'une séquence d'apprentissage.

D'après ces critères d'optimalité, on constate qu'on est devant le problème de partitionnement qui consiste à trouver la partition optimale. Ce problème touche de nouveau la problèmatique autour de laquelle s'articulent nos travaux. Plusieurs méthodes existent pour générer des dictionnaires optimaux [89][149]. Cependant, nous rencontrons le problème de coût de stockage du dictionnaire (le nombre des vecteurs utilisés pour le codage). De plus, la plupart des techniques classiques ne sont pas optimales car elles utilisent des données corrélées entre elles.

Plusieurs auteurs ont proposé différentes solutions [71][155], mais les cartes topologiques restent les plus susceptibles à être utilisées comme quantificateur vectoriel de l'image [20] et la parole [74] grâce à leurs propriétés importantes [20]. La section suivante présente l'utilisation des cartes topologiques pour la quantification vectorielle.

2.3.2 Quantification vectorielle par la carte topologique

Nous constatons que la définition des vecteurs codes repose sur une classification plus ou moins adéquate des vecteurs à coder. Le but est alors d'incorporer une information de classification dans les vecteurs code en les triant pendant la construction du dictionnaire. La carte de kohonen représente une méthode neuronale de classification très connue pour la recherche du code-book. Nous avons vu dans le premier chapitre que les cartes de kohonen sont capables de réaliser une projection de l'espace d'entrées dans un ensemble fini de vecteurs de sortie suivant une procédure d'apprentissage non supervisé compétitif et auto-organisatrice. La construction du code-book par la carte passe par deux phases principales : la phase d'apprentissage et la phase de classification. La première phase a pour but de générer une partition de l'espace d'entrée en déterminant les meilleurs poids des neurones de la carte tout en gardant certaines propriétés topologiques. L'algorithme mis au point par Kohonen

permet de former, de façon autonome, des cartographies de propriétés données. Les poids de la carte finale trouvée après l'apprentissage représentent un codage des données d'entrées (code-book). La phase de classification repose sur le principe du plus proche voisin pour coder chaque entrée de l'ensemble de données par les vecteurs code de la carte de kohonen. Les modèles probabilistes en recherche d'informations sont importants parce qu'ils représentent une des tentatives les plus significatives pour donner une base théorique solide à la recherche d'informations. Dans ce travail nous proposons de générer le dictionnaire pour la compression du signal des chiffres arabes à travers la version probabiliste de la carte topologique qui donne une estimation de la fonction densité de probabilité d'un ensemble d'échantillons. Elle est utilisée pour la compression de la parole dans un nombre limité de travaux [153][74]. La section suivante est réservée à la description de la version probabiliste de la carte topologique [8] [191] [93].

2.4 Cartes topologiques probabilistes

Les cartes probabilistes appartiennent à la classe de méthodes de classification automatiques non supervisées. En général, ces méthodes visent à approximer la densité $p(z)$ des observations et par la suite donner une définition probabiliste des différentes partitions. Ainsi, une modélisation plus complexe de la carte SOM va permettre non seulement de proposer un modèle qui conserve les propriétés d'ordre mais aussi s'interprète comme un mélange de mélanges locaux de lois de probabilités. Celui-ci permet d'approximer les densités sous-jacentes des données. Ces modèles deviennent de plus en plus un outil populaire et utilisé avec succès puisqu'ils apportent une réponse rigoureuse, flexible et interprétable pour les multiples besoins de la classification.

Le PrSOM associe à chaque neurone c de la carte une fonction densité normale f_c. Cette fonction est complètement définie par deux paramètres : son vecteur référent de dimension n : $w_c = (w_c^1, w_c^2, ..., w_c^n)$ et sa matrice de variance-covariance Σ_c (une matrice carrée de dimension n et définie positive). Dans le cadre du modèle probabiliste, on se limite à la famille de matrices diagonales, définie par $\Sigma_c = \sigma_c^2 I$, où I est la matrice unité. La fonction f_c est définie par :

$$f_c(z) = \frac{1}{(2\pi)^{\frac{n}{2}}\sigma_c^n} exp(-\frac{\|z - w_c\|^2}{2\sigma_c^2}) \tag{2.6}$$

De cette façon, chaque neurone c du modèle est alors défini par le vecteur moyen w_c et le nombre positif σ_c.

2.4.1 Architecture de la carte probabiliste

Nous allons présenter le réseau correspondant au modèle probabiliste des cartes topologiques selon le formalisme neuronal. L'architecture du réseau comprend trois couches (figure (2.2)) :

- La couche d'entrée qui sert à la présentation des observations
- Deux couches C_1 et C_2 représentant une duplication de la carte C présentée pour le modèle déterministe des cartes auto-organisatrices. Elles sont de même taille et

munies de la même topologie que la carte C.

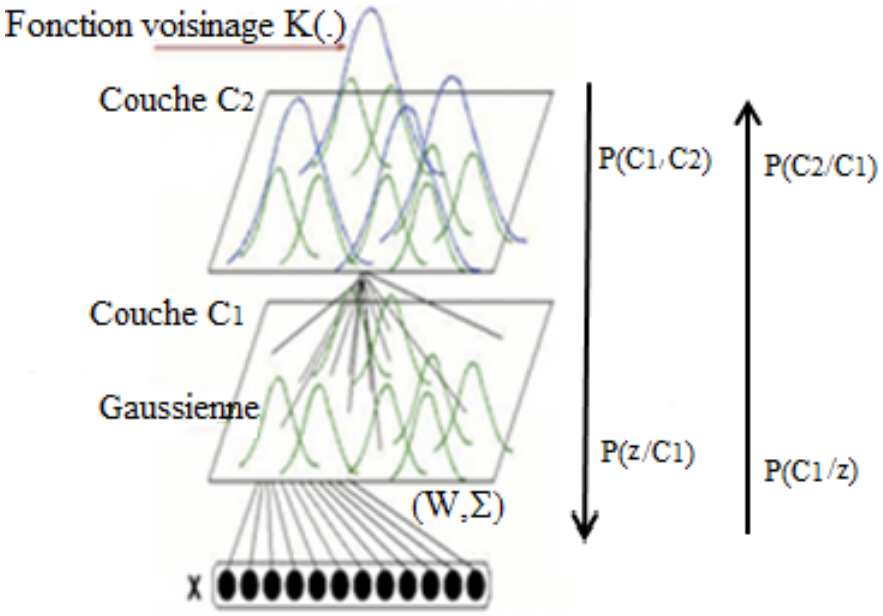

Figure 2.2 – Modélisation de la duplication de la carte auto-organisatrice sous forme d'un modèle de mélange de densités gaussiennes

Un neurone de C_1, noté c_1, représente une gaussienne de vecteur moyen w_c et de matrice de variance-covariance $\sigma_c^2 I$ alors qu'un neurone de C_2, noté c_2, représente un mélange de gaussiennes dont la densité est représentée par la relation (2.9). Les relations de voisinages entre les différents neurones sont définies dans cette version probabiliste grâce à la fonction de voisinage définie dans l'équation (2.10).

2.4.2 Fondement probabiliste

Le mélange de densités des cartes topologique a été introduit par LUTTRELL [156] en se basant sur le formalisme Bayésien ; il suppose qu'un phénomène de propagation probabiliste, dans les deux sens, se réalise à travers la duplication de la carte topologique (réseau à trois couches). Ce réseau permet alors de modéliser deux étapes de propagation de l'information entre les différents neurones des deux couches de traitement (figure (2.2)) :

- La première étape de propagation attribue à une observation z un neurone c_1 de la première couche avec la probabilité $p(c_1/z)$.
- La seconde étape attribue à tout neurone c_1 de la couche C_1 un neurone c_2 de la couche C_2 avec la probabilité $p(c_2/c_1)$.

On suppose en plus, pour simplifier le calcul, que le processus de propagation probabiliste vérifie la propriété de Markov suivante :

$$p(z/c_1, c_2) = p(z/c_1) \quad p(c_2/c_1, z) = p(c_2/c_1) \tag{2.7}$$

On a :

$$p(z) = \sum_{c_1,c_2} p(c_1, c_2, z) = \sum_{c_1,c_2} p(z/c_1)p(c_1/c_2)p(c_2) = \sum_{c_2} p(c_2) \sum_{c_1} p(c_1/c_2)p(z/c_1) \qquad (2.8)$$

En posant $p_{c_2}(z) = \sum_{c_1} p(c_1/c_2)p(z/c_1)$, $p_{c_2}(z)$ apparaît comme un mélange local de densités gaussiennes qui fait intervenir tous les neurones de la carte. On a alors :

$$p(z) = \sum_{c2} p(c_2)p_{c_2}(z) \qquad (2.9)$$

D'après l'équation (2.9), la probabilité de chaque observation z représente un mélange de probabilités. Les coefficients de ce mélange sont les probabilités $p(c_2)$ et les fonctions de densités relatives à chaque élément du mélange sont $p_{c_2}(z)$. La densité de probabilité de z est alors entièrement déterminée par l'architecture du réseau qui permet de donner une expression à la densité conditionnelle $p(c_1/c_2)$ et les tables de probabilités $p(z/c_1)$. Cette dernière est exprimée grâce à la loi normale qui représente chaque neurone de la carte C_1 par : $p(z/c_1) = f_{c_1}(z, w_{c_1}, \sigma_{c_1})$.

l'expression $p(c_1/c_2)$ représente la probabilité d'activation de la cellule c_1 de la couche C_1 connaissant la cellule c_2 de la couche C_2. Ces probabilités conditionnelles sont supposées connues et elles sont définies comme suit :

$$p\left(c_1/c_2\right) = \frac{K^T\left(\delta\left(c_1, c_2\right)\right)}{T_{c_2}} \qquad (2.10)$$

Où, $T_{c_2} = \sum_{r \in C_1} K^T\left(\delta\left(c_2, r\right)\right)$ est un terme normalisant pour obtenir des probabilités.

L'expression (2.10) permet l'introduction des relations de voisinage, donc de l'ordre topologique, dans le formalisme probabiliste de la carte.
Les coefficients à estimer sont alors les paramètres des probabilités a priori $p(c_2)$, et la densité conditionnelle des observations $p(z/c_1)$. Ce formalisme a déjà été utilisé dans [9] et a permis de définir le modèle PrSOM. Cette généralisation du modèle classique de Kohonen permet d'obtenir une quantification de l'espace des données ainsi qu'une estimation des densités locales.
Le réseau PrSOM proposé pour les données réelles [197] [7] utilise la modélisation probabiliste et cherche l'ensemble des vecteurs moyens $W = \{w_c; c \in C\}$ et les écarts-types $\sigma = \{\sigma_c; c \in C\}$ à l'aide de l'ensemble d'apprentissage A pendant la phase d'apprentissage.

2.4.3 Estimation des paramètres

L'estimation des paramètres par ce réseau s'obtient grâce à la maximisation de la vraisemblance de l'ensemble des observations. Ce résultat vient du fait que le modèle probabiliste exprime la densité des observations en fonction de lois de paramètres. Admettant les trois hypothèses suivantes :

- Les observations de l'ensemble A sont indépendantes.
- Chaque observation z_i est engendrée par le générateur $p_{\chi(z_i)}$ qui est associé au neurone $\chi(z_i)$.

— Les neurones c_2 de C_2 ont des probabilités a priori égales.
L'expression de la vraisemblance est exprimée alors comme suit :

$$p(z_1, z_2, ..., z_N/W, \sigma, \chi) = \Pi_{i=1}^{N} p_{\chi(z_i)}(z_i) \tag{2.11}$$

Comme le réseau SOM, l'utilisation de la fonction d'affectation χ permet d'affecter l'observation z_i à son générateur aléatoire (l'une des composantes du mélange). Cette fonction définit donc une partition de l'ensemble d'apprentissage A. L'objectif est de maximiser cette expression par rapport aux paramètres du modèle W, σ et la fonction d'affectation χ. D'une manière classique, maximiser cet objectif revient à minimiser l'opposé de la vraisemblance "classifiante" :

$$E(W, \sigma, \chi) = -ln(\Pi_{i=1}^{N} p_{\chi(z_i)}(z_i)) \tag{2.12}$$

Après un simple remplacement de chacune des expressions déjà citées, on aura :

$$E(W, \sigma, \chi) = -\sum_{i=1}^{N} ln \left[\sum_{r \in C} K_T(\delta(\chi(z_i), r)) f_r(z_i, w_r, \sigma_r) \right] \tag{2.13}$$

Plusieurs méthodes peuvent être utilisées pour atteindre l'optimum de cet objectif : algorithme de gradient, algorithme EM, formalisme des nuées dynamiques. Utilisant ce dernier, l'objectif est minimisé en alternant les deux phases de minimisation et d'affectation jusqu'à la convergence. Pendant la première phase d'affectation, la minimisation de E est faite par rapport à la fonction d'affectation χ. Les autres paramètres (W et σ) sont supposés constants et ils prennent les valeurs courantes. Cette étape permet d'obtenir une nouvelle partition de l'ensemble de données en affectant précisément chaque observation z à un neurone de la carte. Il est facile de voir que la fonction d'affectation qui minimise E est exprimée par :

$$\chi(z) = argmax_{c_2} p_{c_2}(z) \tag{2.14}$$

Elle consiste à affecter chaque observation z au neurone le plus probable selon la densité p_{c_2}. Dans la deuxième phase, dite de minimisation, on cherche alors à minimiser E par rapport à W et σ en supposant que la fonction d'affectation est fixée à la fonction d'affectation courante. Dans cette étape, les paramètres W et σ sont adaptés, comme la carte SOM, en annulant les dérivées partielles de la fonction E. Pour résoudre l'équation, on se base sur une procédure itérative utilisée par Duda Richard et al. [65]. Cette procédure suppose que, pour la i^{me} itération, la valeur initiale des paramètres est assez proche des vraies valeurs. On obtient alors les formules de mise à jour suivantes :

$$w_r^t = \frac{\sum_{i=1}^{N} z_i K\left(\delta\left(r, \chi^{t-1}\left(z_i\right)\right)\right) \frac{f_r\left(z_i, w_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(z_i)}(z_i)}}{\sum_{i=1}^{N} K\left(\delta\left(r, \chi^{t-1}\left(z_i\right)\right)\right) \frac{f_r\left(z_i, w_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(z_i)}(z_i)}} \tag{2.15}$$

$$\left(\sigma_r^t\right)^2 = \frac{\sum_{li=1}^{N} \left\| w_r^{t-1} - z_i \right\|^2 K\left(\delta\left(r, \chi^{t-1}\left(z_i\right)\right)\right) \frac{f_r\left(z_i, w_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(z_i)}(z_i)}}{n \sum_{i=1}^{N} K\left(\delta\left(r, \chi^{t-1}\left(z_i\right)\right)\right) \frac{f_r\left(z_i, w_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(z_i)}(z_i)}} \tag{2.16}$$

Nous résumons la méthode d'apprentissage du PrSOM dans l'algorithme (3).

Algorithm 3 Algorithme PRSOM avec T constant

Entrées: $T_{min}, T_{max}, T, t, Niter, W, \sigma$

Sorties: W, σ

Phase d'initialisation :

— Fixer les valeurs de T_{min}, T_{max}, $Niter$, la taille p de la carte, $t \leftarrow 0$

— Choisir la structure de la carte, initialiser W et σ

Tant Que Niter ou une stabilisation de W et σ ne sont pas atteint **faire**

Les W^{t-1} et σ^{t-1} étant calculés à l'itération précédente :

— Calculer la nouvelle valeur de T en appliquant la formule déjà vue pour SOM :

$$T = T_{max} * \left(\frac{T_{min}}{T_{max}}\right)^{\frac{t}{Niter-1}} \tag{2.17}$$

Pour cette valeur du paramètre T faire :

— Mise à jour des fonctions d'affectation associées à W^t et σ^t selon l'équation (2.14)

— Calcul des nouveaux paramètres W^t à l'aide des formules (2.15) et (2.16)

$t \leftarrow t + 1$

Fin Tant Que

2.5 Problématique

D'après la section précédente, nous constatons que l'algorithme d'apprentissage du PrSOM dépend de l'initialisation des paramètres à estimer (le nombre de neurones dans la carte, les poids et variances initiaux). Ainsi, les résultats obtenus sont influencés par les conditions initiales de l'architecture du réseau.

Le mauvais choix de ces paramètres, comme illustré sur la figure (2.3), augmente le risque de trouver après l'apprentissage trois types de solutions (figure (2.3)) :

— des solutions dégénérées ou singulières (neurones rouges) : lorsqu'il existe des neurones sur la carte qui ne sont affectés à aucune entrée.

— des solutions représentant une faible quantité d'information (neurones verts)

— des solutions représentant une grande quantité d'information (neurones bleus)

La compression de la parole à l'aide du réseau PRSOM est fortement affectée par le problème du choix de l'architecture. En effet, le nombre de neurones influence l'espace de stockage, car pour avoir un dictionnaire optimal il faut en contrepartie générer une carte optimale. De plus, la qualité des vecteurs codes, qui doivent représenter le maximum d'information des vecteurs à coder, dépend fortement de l'initialisation des paramètres de neurones de la carte.

D'après nos connaissances, il existe très peu de travaux qui se sont intéressés au problème de choix de l'architecture des cartes topologiques déterministes et probabilistes. Nous avons proposé deux travaux [77] [2] qui s'intéressent à ce problème pour la carte SOM. Les approches consistent à construire la carte d'une manière ascendante et évaluer à chaque itération le résultat trouvé pour sélectionner la meilleure carte. En-Naimani Zakariae et al. [74] proposent de modéliser le problème de choix de l'architecture du PrSOM sous forme d'un modèle d'optimisation non linéaire à variables mixtes sous contraintes linéaires. Le

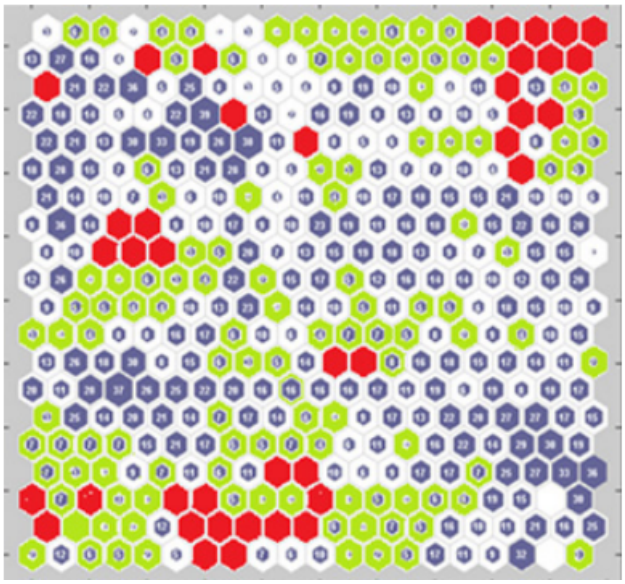

Figure 2.3 – Exemple illustratif des types de neurones de la carte obtenus après apprentissage [74]

modèle consiste à maximiser une fonction vraisemblance sous une contrainte qui assure l'existence d'au moins un neurone sur la carte en associant à chacun une variable binaire. Grâce à l'algorithme génétique, les auteurs trouvent la solution du modèle proposé.

A notre tour, pour remédier à ce problème, nous introduisons notre approche itérative qui permet de sélectionner la meilleure architecture (meilleur dictionnaire) parmi plusieurs autres générées selon une procédure bien déterminée. La section suivante présente une description du processus de notre approche proposée pour palier au problème de choix de l'architecture du PrSOM.

2.6 Description de l'approche H-PrSOM

Sous le grand objectif de la sélection de paramètres neuronaux de la carte topologique probabiliste, nous visons, dans ce travail, à réaliser les points suivants :

— Sélection de l'architecture optimale de la carte topologique probabiliste :
  - Nombre optimal de neurones de la carte probabiliste
  - Poids w_{ij} des neurones de la carte
  - Variances σ_{ij} des neurones de la carte

— Sélection du dictionnaire optimal pour la compression du signal parole

Ces objectifs sont atteints grâce à notre nouvelle approche H-PrSOM. Celle-ci représente une technique itérative qui génère plusieurs architectures en se basant sur une technique divisive bien déterminée pour sélectionner la meilleure architecture de la carte probabiliste. La méthode H-PrSOM est une méthode d'optimisation globale qui utilise l'apprentissage de la carte probabiliste comme une méthode d'optimisation locale. Dans chaque itération, le

processus de recherche d'une architecture possible passe par trois phases principales : phase de sélection de paramètres, phase d'apprentissage et phase d'optimisation (figure (2.5)). Ces étapes sont décrites dans les paragraphes suivants.

2.6.1 Phase de sélection de paramètres

L'objectif de cette première phase est de générer une architecture possible pour la carte probabiliste. Plus précisément, elle consiste à générer le nombre de neurones de la carte ainsi que leurs poids et leurs variances initiaux selon la technique que nous appelons « la technique de choix de la paire de données les plus loin ». En effet, cette technique commence par la recherche d'une paire de données x_1, x_2 dont la distance est maximale parmi toutes les paires possibles dans l'ensemble de données. La recherche de la paire x_1, x_2 se fait à l'aide de la formule suivante :

$$d(x_1, x_2) = max_{u,v \in A} d(u, v) \tag{2.18}$$

Pour mieux illustrer la technique de recherche de x_1 et x_2, nous présentons la figure (2.4).

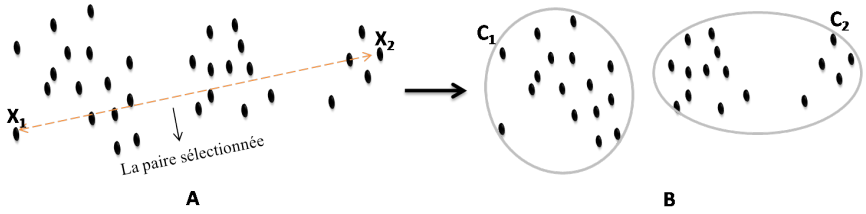

Figure 2.4 – Exemple illustratif de la technique du choix des centres initiaux

Ces deux points constituent alors les centres de deux groupes nommés G_1 et G_2. Ceux-ci sont construits selon les formules suivantes :

$$G_1 = \{y \in A / d(x_1, y) \leq d(x_2, y)\}$$

$$G_2 = \{y \in A / d(x_2, y) \leq d(x_1, y)\}$$

Le groupe G_1 est formé des points les plus proches à x_1 qu'à x_2 alors que G_2 est constitué des points les plus proches à x_2 qu'à x_1. La carte dupliquée est alors initialisée par deux neurones ; leurs poids et variances sont respectivement les centres et les poids des groupes G_1 et G_2.

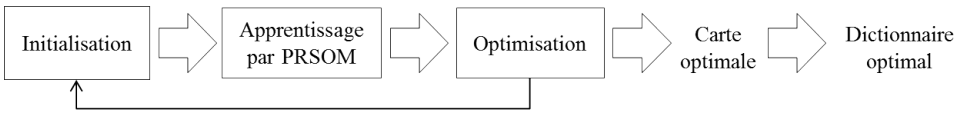

Figure 2.5 – Etapes principales de l'approche proposée

2.6.2 Phase d'apprentissage

Une fois l'architecture initiale est choisie dans la première phase, nous cherchons à stabiliser les poids et les variances des neurones de la carte pour converger vers une quantification optimale locale. La stabilisation se fait à travers le processus probabiliste d'apprentissage du PrSOM. Ce dernier joue le rôle d'un algorithme de recherche locale du quantificateur optimal.

2.6.3 Phase d'optimisation

A la fin de l'apprentissage, nous évaluons la qualité de la quantification trouvée pour la comparer à celle trouvée à l'itération suivante du processus itérative. L'évaluation des quantificateurs générés est faite selon l'indice PSNR et MSE que nous allons décrire dans la section suivante.

Dans les itérations suivantes, les mêmes étapes sont appliquées avec une augmentation du nombre de neurones par un, à chaque itération, en substituant un neurone de la carte par deux nouveaux neurones. Ces derniers sont les centres des nouveaux groupes G_1 et G_2 qui représentent le résultat de la division du groupe C_h selon la technique de choix de la paire de données les plus loin tel que :

$$G_1 = \{y \in G_h / d(x_1, y) \leq d(x_2, y)\}$$
$$G_2 = \{y \in G_h / d(x_2, y) \leq d(x_1, y)\}$$

Où C_h représente le groupe le plus hétérogène sélectionné parmi les groupes générés à la phase de sélection de paramètres selon l'équation (2.19).

$$h = argmax_k \frac{1}{2|G_k|} \sum_{i \in G_k} \sum_{\substack{j \in G_k \\ i \neq j}} d(x_i, x_j) \tag{2.19}$$

Le meilleur dictionnaire correspond alors à la meilleure carte trouvée par notre approche.

Pour valider cette approche et tester sa performance dans le domaine de la compression de la parole, nos simulations ont été effectuées en utilisant un corpus de chiffres arabes recueilli par le laboratoire d'automatique et de signaux de l'Université de Badji-Mokhtar - Annaba en Algérie [1]. 88 personnes (44 hommes et 44 femelles) de locuteurs natifs arabes ont été invités à prononcer les chiffres de 0 à 9 dix fois. En se basant sur le résultat de cette expérience, la base de données se compose alors de 8800 jetons (10 chiffres x 10 répétitions x 88 locuteurs). Nous signalons que les locuteurs sont indépendants.
Pour tester notre approche, nous divisons cette base de données en une base d'apprentissage contenant 75% de données et une autre de test contenant 25% de données.

2.7 Résultats expérimentaux

Avant d'exposer les résultats des expériences effectuées dans cette section, nous introduisons les outils d'évaluation des résultats de la compression de la parole obtenus par l'approche proposée. Pour cela, nous utilisons deux grandeurs : la première est appelée Peak Signal-to-Noise Ratio (PSNR) et exprimée en fonction de la deuxième appelée l'erreur quadratique moyenne (MSE). PSNR est une mesure de distorsion utilisée en compression de la parole ou de l'image. Elle consiste à quantifier la performance des codeurs en mesurant la qualité de reconstruction du signal compressé par rapport au signal original. Cette mesure est définie par :

$$PSNR = 10log_{10}\left(\frac{(nX^2)}{MSE}\right) \tag{2.20}$$

Tel que n est la taille du signal reconstruit, X est la valeur quadratique absolue maximale et MSE (Mean Squared Error), représente l'erreur quadratique moyenne et définie par :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{x}\left(i\right) - x\left(i\right)\right)^2 \tag{2.21}$$

Tel que $\widehat{x}$ et x représentent respectivement le signal parole original et le signal parole reconstruit.

La première expérience est réalisée dans le but de montrer la capacité de la méthode à trouver le nombre optimal de neurones, ainsi que la meilleure initialisation des poids.

Le tableau (2.1) illustre le processus de détermination de l'architecture adéquate du PrSOM pour la création du code-book nécessaire pour la compression du signal parole. Ce tableau comporte cinq colonnes : la première contient les chiffres arabes. La seconde indique quelques itérations du processus itératif. La troisième colonne est réservée au nombre de neurones de la carte généré dans chaque itération. Tandis que les deux dernières colonnes indiquent les valeurs du PSNR et MSE obtenues par la méthode proposée à chaque itération.

Le tableau (2.1) montre que le nombre nécessaire de neurones pour compresser le signal parole des dix chiffres arabes varie entre dix-neuf et vingt. Mais, dans la plupart des cas, vingt est le nombre de neurones approprié pour la construction du code-book optimal. En outre, les valeurs de MSE pour tous les exemples varient entre 0.2 et 0.4. Cette mesure atteint des valeurs minimales satisfaisantes qui correspondent à des architectures adéquates du PrSOM. Prenant le cas du chiffre un, la compression de son signal parole demande vingt neurones dans chaque carte. Ce résultat est associé à l'initialisation qui a fourni la plus haute valeur du PSNR (23,1438) et la plus faible valeur du MSE (0,3383).

Le tableau (2.1) atteste que l'approche proposée réussit à générer les neurones et leurs poids associés qui donnent l'architecture adéquate avec des erreurs satisfaisantes. Il faut signaler que l'approche proposée ne dépend d'aucun paramètre initial.
Nous remarquons aussi que le PSNR (resp. MSE) continue à augmenter (resp. diminuer)

Table 2.1 – Résultats de compression des dix chiffres arabes : architectures définies automatiquement par l'approche proposée et leurs PSNR et MSE associés

Ch.	Itér	Nb.N	PSNR	MSE	Ch.	Itér	Nb.N	PSNR	MSE
0	16	17	22.6530	0.3171	5	16	17	22.9039	0.3160
	17	18	22.7205	0.3125		17	18	22.9842	0.3111
	18	19	22.8164	0.3100		18	19	23.0807	0.3041
	19	20	23.0015	0.2902		19	20	23.0925	0.3040
	20	21	22.9436	0.2998		20	21	23.1528	0.2996
	21	22	22.8290	0.3065		21	22	23.1306	0.3017
1	16	17	22.8688	0.3601	6	16	17	22.9122	0.3502
	17	18	22.9240	0.3553		17	18	22.9811	0.3486
	18	19	22.9731	0.3516		18	19	23.0897	0.3419
	19	20	23.1438	0.3383		19	20	23.4305	0.3010
	20	21	23.0831	0.3422		20	21	23.2950	0.3123
	21	22	23.0337	0.3461		21	22	23.2019	0.3186
2	16	17	21.3350	0.2849	7	16	17	21.2315	0.3140
	17	18	21.3780	0.2825		17	18	21.9614	0.3081
	18	19	21.4485	0.2785		18	19	22.1318	0.3006
	19	20	21.4799	0.2765		19	20	22.3527	0.2939
	20	21	21.6536	0.2647		20	21	21.8945	0.3001
	21	22	21.6260	0.2672		21	22	21.6783	0.3041
3	16	17	21.1799	0.3116	8	16	17	20.8999	0.3316
	17	18	21.1763	0.3109		17	18	21.2513	0.3267
	18	19	21.2874	0.3030		18	19	21.8538	0.3132
	19	20	21.1574	0.3139		19	20	21.3474	0.3195
	20	21	21.1829	0.3112		20	21	21.1815	0.3212
	21	22	21.1085	0.3163		21	22	21.0085	0. 3273
4	16	17	20.7635	0.2825	9	16	17	20.9925	0.3109
	17	18	20.7612	0.2824		17	18	21.4485	0.3081
	18	19	20.8392	0.2775		18	19	21.8945	0.3001
	19	20	21.0023	0.2679		19	20	23.1023	0.2879
	20	21	20.9941	0.2681		20	21	22.9240	0.2945
	21	22	20.8982	0.2741		21	22	22.7205	0.2994

Table 2.2 – Génération arbitraire de plusieurs architectures de la carte pour le chiffre zéro

Chiffres	Taille de carte	PSNR	MSE
0	10	20.3654	0.5667
	15	20.6255	0.5345
	20	20.8164	0.5100
	25	20.9418	0.4986
	30	21.0063	0.4908
	moyenne	20.7510	0.5201

jusqu'à l'obtention d'une valeur maximale (resp. minimale) qui correspond au nombre approprié de neurones. Par conséquent, nous pouvons affirmer que l'approche proposée évite le phénomène de sur et sous apprentissage en choisissant le nombre adéquat de neurones de la carte.

Un deuxième tableau (2.2) est dressé dans cette section pour comparer notre approche à l'algorithme classique du PrSOM. Ce tableau contient un certain nombre d'initialisations arbitraires d'architecture du PrSOM avec une mesure du PSNR et MSE de chaque architecture pour la compression du signal zéro.

En comparant les résultats de compression listés dans le tableau (2.2) à ceux obtenus par l'approche proposée, nous concluons que cette dernière est capable de trouver les meilleurs vecteurs initiaux des neurones avec un nombre adéquat. La méthode HPrSOM surmonte le problème d'initialisation de l'architecture et améliore la qualité de l'algorithme PrSOM. En effet, la valeur maximale du PSNR réalisée par la version classique est de 21.0063 et correspond à 30 neurones. Cette valeur de PSNR est inférieure à la plus petite valeur trouvée par l'approche proposée.

A notre connaissance, nous n'avons pas trouvé des approches d'optimisation d'architecture du PRSOM ayant travaillé avec la base de données de chiffres arabes sur laquelle nous avons évalué notre approche. Pour cette raison, nous avons comparé notre méthode à la version classique du PRSOM.

Avant de clôturer cette section de résultats, nous présentons la figure (2.6). Cette dernière visualise le signal original du chiffre zéro et son signal décompressé après compression par l'approche H-PrSOM. Nous présentons cette figure dans le but de tester l'approximité du signal reconstruit de le signal original.

En examinant les deux signaux de la figure (2.6), nous concluons que la méthode proposée est capable de fournir un code-book adéquat qui donne approximativement le signal original.

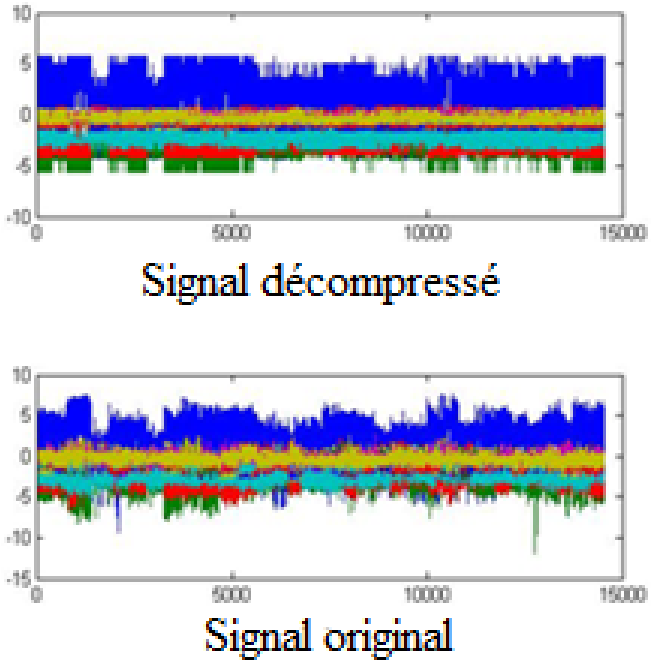

Figure 2.6 – Visualisation du signal original du chiffre zéro et son signal décompressé après compression par notre approche

Comme résultat de cette section, nous pouvons affirmer que cette nouvelle approche fournit des résultats satisfaisants et réalise les objectifs attendus qui sont les suivants :

— Identification de l'architecture appropriée de PRSOM
  - Nombre de neurones adéquat
  - Meilleurs poids initiaux

— Diminution du risque de sur-sous apprentissage
— Génération du code-book adéquat pour avoir un signal proche de l'original après le décodage

2.8 Conclusion

Dans ce chapitre, nous avons fixé comme objectif la sélection du modèle neuronale de la carte probabiliste PrSOM en introduisant une nouvelle approche itérative H-PrSOM avec une technique bien déterminée du choix des poids de neurones pour un apprentissage réussit. Le processus itératif de la méthode utilise l'algorithme PrSOM, dans chaque itération, comme une méthode de recherche locale après la phase de l'initialisation. Les modèles probabilistes sont les modèles les plus adéquats pour traiter des signales paroles. De plus, les cartes SOM représentent des méthodes de quantification vectorielle puissantes pour la compression de données. Pour ces deux raisons, nous avons choisi de valider notre approche proposée sur le domaine de la compression du signal parole de chiffres arabes.

Par conséquent, notre approche veille à choisir l'architecture adéquate qui donne le meilleur dictionnaire de données pour une meilleure compression du signal parole.

Les résultats numériques prouvent la capacité de l'approche proposée à atteindre les objectifs attendus. Les résultats numériques montrent que notre approche trouve des résultats pertinents et meilleurs que la version classique du PrSOM sans avoir besoin d'aucun paramètre initial. Grâce à ces résultats satisfaisants, nous croyons à la performance de la méthode H-PrSOM. En outre, nous visons à traiter autres types de données, comme les données continues et catégorielles.

La problématique de sélection d'architecture ne se limite pas seulement aux cartes topologiques mais aussi aux réseaux neuronaux de types multi-couches. Ce problème influence le processus d'apprentissage du PMC et par conséquent sa capacité de généralisation. Dans le chapitre suivant, nous présentons notre nouvelle modélisation mathématique du problème de sélection d'architecture neuronal du PMC.

Chapitre 3

Contribution à la sélection des modèles neuronaux de type PMC

3.1 Introduction

Bien que les PMCs ont connu un grand succès dans différents domaines (chimie, industrie, contrôle optimal, intelligence artificielle, etc) avec différentes applications (robotique, reconnaissance de forme et de parole, classification et prévision, optimisation, etc), leurs performances dépendent fortement des conditions initiales comme : les poids, le nombre de couches et de neurones cachés, l'algorithme d'apprentissage et la fonction de transfert. Ces paramètres sont traditionnellement choisis à l'aide d'un processus d'essai-erreur qui échoue souvent, à cause de son caractère aléatoire, à trouver les meilleurs paramètres possibles pour chaque problème spécifique. Dès lors, la proposition de nouvelles stratégies, de sélection d'architectures, plus structurées et plus intelligentes sont nécessaires pour améliorer l'efficacité des RNAs. Par conséquent, la sélection de la meilleure architecture a attiré l'attention de plusieurs chercheurs pendant plusieurs années.

Motivé par ce problème qui n'a pas jusqu'à maintenant une base théorique pour le résoudre, notre travail présenté dans ce chapitre s'articule autour de cette problématique. Pour cela, premièrement, nous proposons une modélisation mathématique non linéaire à variables mixtes sous contraintes dans le but de choisir le nombre de couches et de neurones adéquats avec la meilleure stabilisation des poids du PMC. Deuxièmement, l'optimisation de la fonction objectif du modèle sous contraintes est faite à l'aide de la fameuse stratégie d'évolution "Algorithme Génétique".

Nous consacrons ce chapitre à la présentation des éléments essentiels pour la résolution du problème d'architecture du réseau multi-couches. Nous entamons le chapitre par une description détaillée du fondement du perceptron multi-couches suivie de la position du problème de choix de l'architecture ainsi que les propositions des auteurs dans la littérature pour la

résolution de cette problématique. Nous présentons ensuite notre nouvelle méthode de résolution reposant sur une modélisation mathématique du problème de choix de l'architecture sous la forme d'un modèle non linéaire avec contraintes en définissant les différents éléments construisant le modèle proposé. Par suite, la section suivante est dédiée à la description de la démarche de résolution du modèle via l'algorithme génétique, après avoir donné une description générale des différentes étapes de l'algorithme. La dernière section présente une évaluation de la performance de la méthode d'optimisation d'architecture à l'aide de testes expérimentaux effectués sur des bases de données benchmarks pour la classification supervisée. Ce chapitre a fait l'objet d'une communication orale dans le congrès international : AAFD δ SFC 2016 : Conférence Conjointe Francophone sur la Science des Données.

Ce travail est soumis dans le journal international "Journal of Computer Science and Technology".

3.2 Réseaux Multi-couches

Dans le premier chapitre, nous avons vu qu'un neurone formel calcule une combinaison linéaire des variables d'entrées et la transforme ensuite par une fonction de transition qui peut être linéaire ou non-linéaire. En assemblant un certain nombre de neurones suivant une architecture en couches (section 3.2.1), on peut construire des modèles plus complexes capables de résoudre des problèmes de différents types. White [211] a proposé une interprétation probabiliste qui offre une présentation synthétique de nombreux aspects statistiques des réseaux.

3.2.1 Architecture du Perceptron Multi couches

Les réseaux multi-couches [105] [180] [31] sont les plus anciens réseaux de neurones et les plus populaires. Ce sont des réseaux acycliques (non-bouclé) et orienté. Cela signifie qu'il n y a pas de connexion vers l'arrière car le flux de l'information se propage en "avant" sans retour en arrière au contraire des réseaux récurrents. Un PMC se compose d'un nombre de couches, placées en parallèle, dont chacune regroupe un ensemble de neurones formels (figure (3.1)) (processeurs élémentaires) qui peut varier d'une couche à autre [47]. Ces réseaux sont totalement connectés : chaque neurone d'une couche inférieure est relié à tous les neurones de la couche supérieure tandis que les neurones d'une même couche ne peuvent pas communiquer entre eux (figure (3.1)).

La première couche (la rétine) du PMC est une couche virtuelle. Les neurones de cette couche servent seulement à présenter les attributs des données à la première couche cachée du réseau (figure (3.2)). Le nombre de neurones dans cette couche est déterminé apriori et égale au nombre d'attributs des données.

La couche de sortie (figure (3.3)) est la dernière couche du réseau qui correspond aux sorties du système. Comme la couche d'entrée, le nombre de neurones de cette couche est fixé apriori selon les sorties désirées du problème traité. Ces neurones effectuent le dernier calcul de la composition de fonctions. Les couches cachées se regroupent en parallèles entre la couche d'entrée et la couche de sortie. Les neurones de ces couches effectuent les calculs

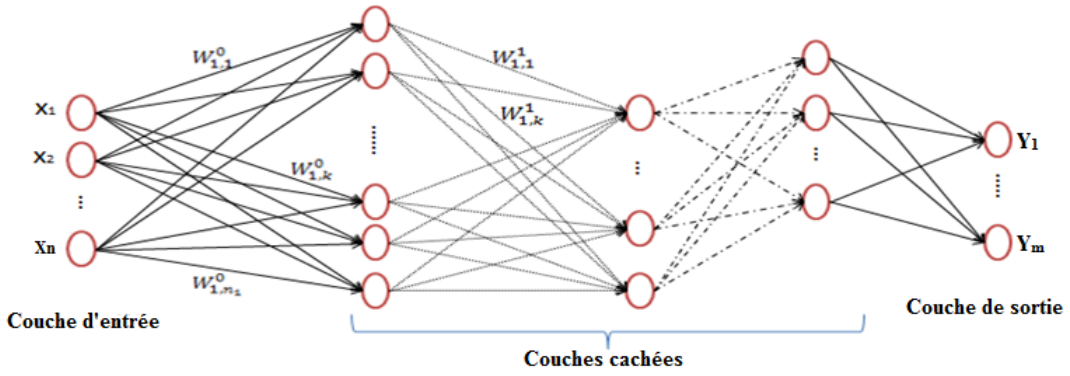

FIGURE 3.1 – Représentation graphique du réseau multi-couches général. Dans ce graphisme, l'information se propage de la gauche vers la droite

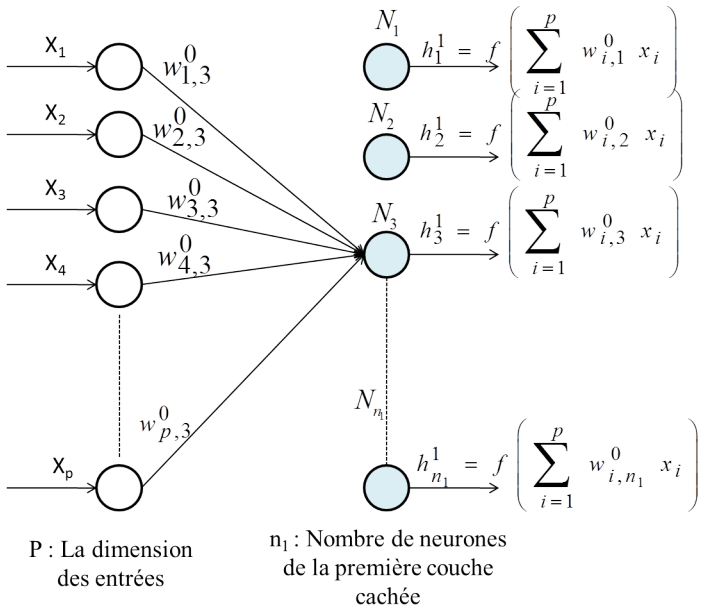

FIGURE 3.2 – Calcul des sorties de la première couche cachée

intermédiaires (figure (3.2)).

Les neurones de la première couche cachée reçoivent la somme pondérée des x_j localisés sur la "couche précédente" (les x_j sont les "entrées" du perceptron) (figure (3.2)). Chaque neurone calcul ensuite son état en transformant son entrée (la somme pondérée) à l'aide de la fonction de transfert non linéaire [218]. Cet état calculé est le signal de sortie de ce neurone qui se ramifie pour alimenter un nombre de neurones avals de la couche suivante. Ceci se répète pour chaque couche k à la différence que les entrées sont les sorties de la couche précédente (figure (3.4) et (3.3)).

Chaque neurone de la première couche cachée sépare l'espace des variables par un hy-

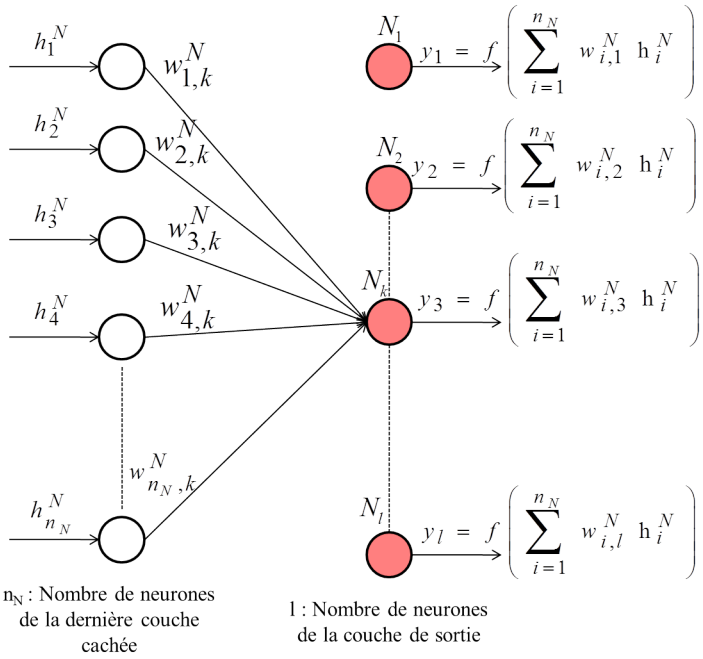

FIGURE 3.3 – Calcul des sorties du réseau

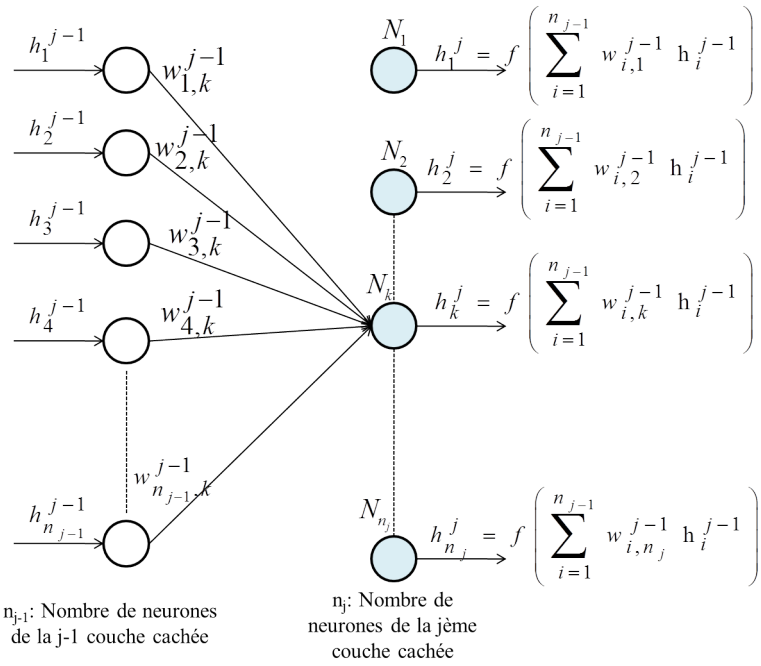

FIGURE 3.4 – Calcul des sorties de la j^{me} couche cachée

perplan. Les neurones de la (ou des) couche(s) suivante(s) réalisent une fonction logique quelconque de ces demi espaces.

les pondérations (poids) du réseau représentent la force de la connexion entre les neurones.

Plus une pondération $w_{i,k}^j$ est grande en valeur absolue, plus l'activité de h_k^j est corrélée à celle de h_i^{j-1} (positivement ou négativement suivant le signe de $w_{i,k}^j$).
Pour utiliser un PMC il faut premièrement choisir son architecture. Ce choix consiste à fixer les paramètres suivantes :

— Le nombre de couches
— Le nombre de neurones par couche
— La nature des différentes connexions entre les neurones
— La nature des neurones sur chaque couche

3.2.2 Fonction de transfert

Le choix de la fonction de transfert est critique dans la modélisation du réseau à couches. Ces réseaux qui sont dédiés à la séparation de données non-linéaires doivent utiliser des fonctions de transfert non-linéaires afin d'obtenir des modèles statistiques non-linéaires. La fonction d'activation ou de transfert est utilisée pour contrôler l'amplitude de la sortie de chaque neurone. Il existe de nombreuses fonctions candidates mais la classe de fonctions la plus fréquemment utilisée est la classe des fonctions écrasantes. Définissons d'abord c'est quoi une fonction écrasante.

Définition 6. *On dit que f est une fonction écrasante si :*
— *f est une fonction de R dans [0, 1]*
— *f est croissante*
— $\lim_{x\to-\infty} f(x) = 0$
— $\lim_{x\to+\infty} f(x) = 1$

La fonction logistique définie par $f(x) = \frac{1}{1+exp(-x)}$ est la fonction écrasante la plus populaire.

Une autre classe de fonctions qui est plus large est celle des fonctions sigmoïdes. Cette classe de méthodes ont la même définition que la classe des méthodes écrasantes sauf pour la première condition les bornes sont remplacées par deux réels A et B, tel que : $A < B$. On peut trouver parmi les fonctions sigmoïdes : la fonction logistique, la tangente hyperbolique ($tanh$) et l'arc-tangente ($arctan$).
Bien que pour certains auteurs, les fonctions écrasantes et sigmoïdes doivent être continues ou même différentiables, nous n'imposions pas ces restrictions mais les fonctions utilisées dans ce travail les respecteront.

3.2.3 Modélisation mathématique

Les PMCs sont définis mathématiquement comme suit :

Définition 7. *[145] Les perceptrons à n couches cachées de $\mathbb{R}^d$ dans $\mathbb{R}^a$ sont les fonctions de la forme :*

$$\psi_W(x) = \phi_n(W^{(n)}.\phi_{n-1}(W^{(n-1)}....\phi_1(W^{(1)}.x + b^{(1)}) + b^{(2)}) + ...) + b^{(n)}) \tag{3.1}$$

où x est dans $\mathbb{R}^d$ et :

$$\forall k \in \{1, ..., n\}, W^{(k)} \in M(n_k, n_{k-1}), b_k \in \mathbb{R}^{n_k}, n_1 = d, n_n = a$$

et où ϕ_k est une fonction de $\mathbb{R}^{n_k}$ dans lui même, qui à $y^{(k)} = (y_1^{(k)}, y_2^{(k)}, ..., y_{n_k}^{(k)})$ associe :

$$\phi_k(y^{(k)}) = (\varphi_k(y_1^{(k)}), \varphi_k(y_2^{(k)}), ..., \varphi_k(y_{n_k}^{(k)}))$$

avec φ_k est la fonction d'activation de la k^{ieme} couche du perceptron.

Les n_k sont le nombre de neurones de la couche k et l'ensemble $W = (W^{(k)}, b^{(k)}, 1 \leq k \leq n)$ est l'ensemble des paramètres du réseau. On appelle $W^{(k)}$ les poids et $b^{(k)}$ les seuils (ou biais) de la k^{ieme} couche de ψ_W.
Ces réseaux à couches possèdent une propriété fondamentale : l'approximation parcimonieuse. Le théorème suivant [145] montre cette propriété.

Théorème 1. *(Approximation universelle) Quelle que soit f une fonction $C^\propto$ bornée d'un compact de $\mathbb{R}^d$ dans $\mathbb{R}^a$, et quel que soit le réel ε strictement positif, il existe un entier c_2 et un jeu de paramètre $W = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$ associés au perceptron ψ_W tels que :*

$$\int_{x \in \mathbb{R}^d} \|f(x) - \psi_W(x)\| \, \mathrm{d}x < \varepsilon \tag{3.2}$$

Par la suite, Hornik et al. [111] prouve aussi par le théorème (2) que Les réseaux de neurones classiques avec une ou plusieurs couches cachées peuvent approcher arbitrairement bien toute fonction continue (le théorème prouve cette propriété pour la classe des fonctions Borel-mesurables qui contient strictement la classe des fonctions continues).

Théorème 2. *Les réseaux de neurones multi-couches à propagation avant avec une couche cachée contenant suffisamment de neurones cachés et une fonction de transfert écrasante arbitraire sont capables d'approcher, avec une précision arbitraire, n'importe quelle fonction Borel-mesurable d'un espace de dimension finie dans un espace de dimension finie. Dans ce sens, les réseaux de neurones multi-couches à propagation avant sont une classe d'approximateurs universels.*

Ces résultats ont montré que les PMCs constituent un outil universel pour l'approximation des fonctions. Par ailleurs ces deux théorèmes traduisent les points suivants :

- Le PMC représente une classe de fonctions intéressantes pour l'identification de fonctions.
- Une approximation à l'aide des PMCs est relativement peu gourmande en paramètres, en effet, à non linéarité égale, le nombre de paramètres d'un perceptron est moins élevé que celui d'un approximateur usuel.
- Les PMCs sont très efficaces grâce à la méthode d'apprentissage de rétro-propagation du gradient qui permet de construire une suite de poids $W(0), W(1), ...$ convergeant vers un minimum local de la distance entre ψ_W et la fonction à émuler f. Ce processus permet notamment de construire $(W^{(i)}, b^{(i)})(n)$ en fonction de $(W^{(i)}, b^{(i)})(n-1)$ et $(W^{(i+1)}, b^{(i+1)})(n-1)$ seulement.

Plusieurs autres résultats théoriques ont été établis concernant les capacités de ces approximateurs universels de fonctions [50][88][174][212]. Bien qu'il était prouvé théoriquement qu'un PMC à une seule couche cachée peut approximer toute fonction continue, plusieurs études ont montré l'intérêt de considérer des PMCs à deux couches cachés [150] [192]. Ces résultats théoriques ne donnent aucune information précise sur le choix de l'architecture du PMC.

3.2.4 Apprentissage du Perceptron Multi couches

L'apprentissage est la caractéristique la plus intéressante d'un réseau de neurones artificiels. Cette caractéristique consiste à modifier (corriger) les poids des connexions du réseau (initialiser aléatoirement) en fonction des données étiquetées (apprentissage supervisé) d'apprentissage, de telle sorte qu'après un certain temps d'entrainement il ait acquis une faculté de généralisation. L'objectif fondamental de l'apprentissage est de minimiser le risque suivant :

$$R_{th}(w) = E_X\left[L(h(x,w),y)\right] = \int_{X*Y} L(h(x,w),y)P(x,y)\mathrm{d}Y\,\mathrm{d}X \tag{3.3}$$

$E_X[.]$ est l'espérance mathématique sur les vecteurs aléatoires x et y. La fonction $L(h(x,w),y)$ donne le coût de perte entre l'étiquetage calculé $h(x,w)$ et désiré y. Cette fonction est dépendante de la tâche à accomplir. P est une loi de probabilité jointe sur X et Y. Elle correspond à la probabilité qu'une donnée x à étiqueter par y apparaisse en tant que donnée à classer. Cette quantité représente le risque théorique associé au problème. En cherchant à minimiser le risque $R_{th}(w)$, on approche le mieux possible au sens de ce risque la densité conditionnelle $p(d/x)$ par fonction de la machine d'apprentissage. On notera $h(x,w^*)$ toute fonction définie par : $w^* = argmin_W R_{th}(w)$

La difficulté du problème d'apprentissage vient du fait que la fonction de distribution de probabilité jointe $p(x,y)$ est inconnue. Ce risque ne peut alors être calculé car la loi de probabilité P n'est pas connue. Cela fait appel à une approximation de ce risque. Une technique courante est d'utiliser le risque empirique calculé à partir d'un ensemble des exemples d'apprentissage : $A = \left\{(x^1,y^1),(x^2,y^2),...,(x^N,y^N)\right\}$ (chaque paire x^n, y^n est supposée tirée indépendamment identiquement d'une population suivant la densité jointe $p(x,y)$).
Le risque empirique est défini comme suit :

$$R_{emp} = \frac{1}{N}\sum_{A} L(h(x,w),y) \tag{3.4}$$

Le risque empirique permet d'obtenir une approximation imparfaite du risque théorique. On approxime la fonction $h(w^*)$ qui minimise le risque théorique (3.3) par la fonction $h(w')$ au minimum du risque empirique (3.4) : $w' = argmin_w R_{emp}$. Cette approximation se base sur le principe d'induction ; On fait l'hypothèse que ce qui marche sur les exemples (minimisation du risque empirique), marche sur des données non vues (cachées) (minimisation du risque théorique). En d'autres termes, on dit que toute l'information dont on dispose se trouve dans un ensemble de taille fini N.

La valeur du risque théorique (3.3) au point w' s'appelle l'erreur de généralisation $R_{th}(w')$. Cet erreur dépend non seulement du modèle, mais aussi de l'ensemble d'apprentissage. Nous nous sommes intéressés dans nos travaux de recherche à ces deux types de problèmes.

Nous définissons la fonction L comme l'erreur quadratique entre la valeur désirée et la valeur calculée par le réseau en fonction des paramètres poids. Cela consiste à Trouver des poids permettant au réseau de réaliser une relation entrée-sortie spécifiée par des exemples de cette relation.

$$W_* = argmin \frac{1}{N} \sum_{l=1}^{N} (h(x_l, w) - y_l)^2 \tag{3.5}$$

L'optimisation de ce critère rencontre deux difficultés qui rend la recherche des poids optimaux NP-dur :

— L'existence de plusieurs minima locaux de la fonction de coût : le critère n'étant pas quadratique, elle ne possède pas un minimum unique.

— Le modèle n'est pas linéaire en ses paramètres : les équations obtenues en annulant le gradient du critère ne sont pas linéaires, ce qui complique l'estimation des paramètres.

Suite à ces difficultés, il n'y a pas de solution analytique au problème de minimisation d'où on est forcé à optimiser d'une manière itérative en cherchant un bon minimum local, ou même simplement une valeur suffisamment basse du critère. Ces méthodes itératives modifient les paramètres du modèle en fonction du gradient de la fonction de coût par rapport à ses paramètres. Chaque itération du processus d'optimisation (apprentissage) itératif nécessite donc la mise en œuvre de deux points bien distincts :

— L'évaluation du gradient de la fonction de coût.

— La modification des paramètres en fonction de ce gradient, afin d'approcher un minimum de la fonction de coût.

Différents algorithmes d'optimisation sont proposés, ils sont généralement basés sur l'évaluation du gradient pour la minimisation de la fonction objectif. L'algorithme appelé algorithme de rétro-propagation (RP) de gradient [185] [213] [209] est le plus connu dans le cadre de l'optimisation de la fonction coût du PMC et du gradient. Cet algorithme est devenu tellement populaire qu'il apparaît parfois comme synonyme d'apprentissage de réseaux de neurones. Son avantage majeur est la facilité à s'adapter au calcul du gradient de n'importe quelle fonction de coût dérivable. Cette méthode est fondée sur le calcul, à chaque itération, du gradient (comme son nom l'indique) de la fonction de coût par rapport aux paramètres W du modèle. Ce gradient est ensuite utilisé pour calculer une modification des paramètres. Son principe est de propager les erreurs déterminées à la sortie (figure (3.3)) vers les couches cachées car les valeurs désirées ne sont pas connues dans les neurones cachés. En d'autre terme, elle consiste à calculer l'erreur sur une connexion en fonction de l'erreur sur la couche suivante.

La méthode de rétro-propagation fonctionne alors sur deux phases pour chaque exemple de données : une phase de propagation et une autre de rétro-propagation.

— Phase de propagation : au cours de cette phase les variables correspondant à l'exemple présenté au réseau sont utilisées pour calculer les sorties et les potentiels de tous les neurones. L'information propage alors de l'entrée vers la sortie (figure (3.3)).

— Phase de rétro-propagation : au cours de cette deuxième phase, les erreurs de chaque

neurone du réseau sont calculées par évaluation des dérivées de l'erreur par rapport aux poids. Ces erreurs sont ensuite utilisées pour calculer la modification de chaque poids. Cette phase se compose à son tour de deux étapes : évaluation et modification.

— Évaluation du gradient : Cette étape consiste à évaluer l'erreur due à chaque connexion w_{ji} par le calcul de gradient $\frac{\delta E_p}{\delta w_{ji}}$. Ce gradient est calculé en fonction de l'erreur après la cellule j car E ne dépend du paramètre w_{ji} que par l'intermédiaire de la valeur de la sortie du neurone j, qui est elle-même en fonction uniquement de l'activation du neurone j ; on peut donc écrire :

$$\frac{\delta E_p}{\delta w_{ji}} = \frac{\delta E_p}{\delta a_j} * \frac{\delta a_j}{\delta w_{ji}} = \delta_j * z_i \tag{3.6}$$

Les erreurs de la couche de sortie diffèrent de celles des couches cachées alors on aura : Cas de la couche de sortie :

$$\delta_k = \frac{\delta E_p}{\delta a_k} = g\prime(a_k) * \frac{\delta E_p}{\delta y_k} = g\prime(a_k) * (u_k(x_p) - yk) \tag{3.7}$$

Cas de la couche cachée :

$$\delta_j = \frac{\delta E_p}{\delta a_j} = \sum_k \frac{\delta E_p}{\delta a_k}\frac{\delta a_k}{\delta a_j} = \sum_k \delta_k \frac{\delta a_k}{\delta z_j}\frac{\delta z_j}{\delta a_j} = g\prime(a_j)\sum_k w_{jk}\delta_k \tag{3.8}$$

— Modification des poids : Pour diminuer la fonction cout on ajuste les poids du réseau (initialisés aléatoirement). Le taux d'ajustement est déterminé par la dérivée de la fonction erreur en se basant sue la règle de delta :

$$\triangle_p\, w_{ji}(t) = -\eta * \frac{\delta E_p}{\delta w_{ji}}(t) \tag{3.9}$$

Le η représente le taux d'apprentissage. Le poids de la connexion entre les neurones i et j de l'entrée p est modifié alors comme suit :

$$w_{ji}(t+1) = w_{ji}(t) + \triangle_p\, w_{ji}(t) \tag{3.10}$$

L'algorithme d'apprentissage du réseau PMC avec la méthode de rétro-propagation est résumé dans Algorithme (4) (La fonction d'activation est la fonction signoide).

3.2.5 Pouvoir et faiblesse du Perceptron Multi couches

Le PMC est classé plus performant que les méthodes classiques de l'intelligence artificielle. Ce succès a des clés :

— Bon pouvoir de représentation : ce pouvoir est du premièrement à la propriété parcimonieuse et d'approximation universelle si on utilise suffisamment de neurones et/ou de couches. Deuxièmement, ce réseau est applicable aux problèmes à séparabilité linéaire et non-linéaire. troisièmement, la méthode d'apprentissage qui se repose sur la descente de gradient est applicable dans la plupart des cas.
— Convergence vers une solution
— Bonne capacité de généralisation en général.

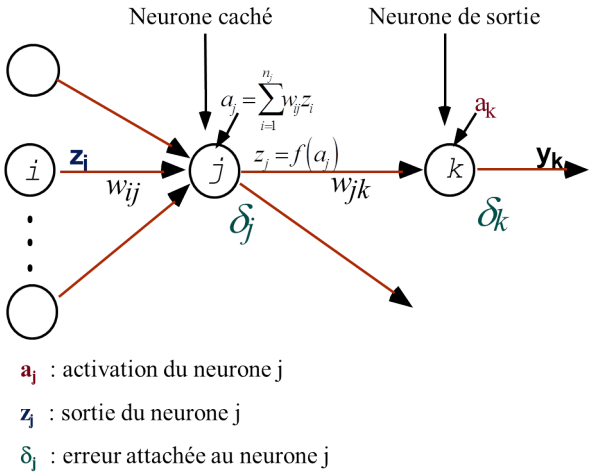

Figure 3.5 – Description de l'activation et la sotie d'un neurone

— Domaines d'application très vaste : Ces réseaux demandent seulement un ensemble représentatif des données et n'exige pas de connaissance profonde des relations d'entrées-sorties.

Bien que les réseaux à couches ont connus un grand succès mais ils souffrent de quelques points faibles :

— PMC est une boite noire : La relation entre les entrées et les sorties est seulement fournie sous forme (x, y). En outre, c'est difficile de dériver une explication causale des résultats et les neurones cachés avec les poids obtenus n'ont pas de sens sémantique. En effet, ce sont des paramètres opérationnels plutôt que des traits caractéristiques du domaine.
— Le pouvoir de généralisation n'est pas garanti même si l'erreur d'apprentissage est réduite à 0.
— Risque de convergence vers un minimum local.
— l'apprentissage peut être lent surtout avec des données complexes.
— Problème de choix de l'architecture du réseau.
— Problème de sur et sous apprentissage du à plusieurs causes.
— La plupart des paramètres peuvent seulement être déterminés empiriquement.

3.3 Problématique du choix de l'architecture

Le choix de l'architecture des RNAs présente une difficulté majeur et une nécessité pour les chercheurs. Face à une tâche à résoudre par un RNA, on doit premièrement faire des choix d'architecture (nombre de neurones cachés, nombre de couches cachées et leur interconnexion) non évidentes et bien importantes. En effet, toutes les propriétés et résultats théoriques sur les réseaux connexionnistes (leur puissance de calcul ou leur faculté de généralisation) ne tiennent que si l'on utilise l'architecture idéale (ou au moins suffisante et nécessaire).

Algorithm 4 Algorithme de rétro-propagation

ENTRÉES: $Niter, W$

SORTIES: W

— Initialiser tous les poids à de petites valeurs aléatoires, t=0

— Normaliser les données d'entrainement

Tant Que le nombre d'itération n'est pas atteint **faire**

— Permuter aléatoirement les données d'entrainement

pour Pour chaque donnée d'entrainement n **faire**

— Calculer les sorties observées en propageant les entrées vers l'avant

— Ajuster les poids en rétro-propageant l'erreur observée :

$$w_{ji}(n) = w_{ji}(n-1) + \Delta w_{ji}(n) = w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{3.11}$$

$$\delta_j(n) = \begin{cases} e_j(n) y_j(n)[1 - y_j(n)] & \text{Si} j \in \text{couche de sortie} \\ y_j(n)\left[1 - y_j(n)\right] \sum_k \delta_k(n) w_{kj}(n) & \text{Si} j \in \text{couche cachée} \end{cases} \tag{3.12}$$

Avec $0 \leq \eta \leq 1$ représentant le taux d'apprentissage et $y_i(n)$ représentant soit la sortie du neurone i sur la couche précédente, si celui-ci existe, soit l'entrée i autrement.

$t \leftarrow t + 1$

Fin pour

Fin Tant Que

Particulièrement pour le PMC, ce choix reste difficile car la précision, vitesse et la généralisation dépendent fortement des paramètres utilisés par le réseau. Même s'il était prouvé théoriquement qu'une seule couche avec suffisamment de neurone suffisait, il n'existe pas de résultat théorique sur le nombre minimum de neurones cachés nécessaires. En plus, il s'est avéré que le faite d'avoir des couches cachées multiples avec moins de neurones au total peuvent apprendre plus vite, pour la même performance. Par conséquent, aucun résultat théorique ne permet d'avoir une idée précise sur le nombre de couches et neurones dans chaque couche nécessaires pour approximer une fonction donnée.

Le nombre de couches cachées et de neurones par couche cachée conditionnent directement le nombre de paramètres à estimer et donc la complexité du modèle. Ils participent à la recherche d'un bon compromis biais/variance c'est-à-dire l'équilibre entre qualité d'apprentissage et qualité de prévision. En outre, l'ajout ou le manque de couches ou de neurones cachés peut créer des anomalies dans les résultats du réseau. En effet, de trop petites valeurs de paramètres auront une grande perte sur l'ensemble d'entrainement : cas de sous apprentissage (figure (3.6)). Alors plus on va augmenter le nombre de paramètres, plus le modèle généré va coller aux données. En contrepartie, Un réseau de neurones avec trop de couches ou de neurones cachés va apprendre par cœurs les données de test et donc créer un modèle faux : cas de sur apprentissage (figure (3.6)). Réellement, les données d'apprentissage sont bruitées ou représentent des valeurs approximatives. Donc si on oblige le réseau (par l'ajout de neurones ou de couches cachées) à répondre de façon quasi parfaite relativement à ces exemples, on peut obtenir un réseau biaisé par des valeurs erronées. En effet, la figure (3.7) présente des couples $(x_i, f(x_i))$ situés sur une droite d'équation $y = ax + b$, mais bruités de

sorte que les points ne soient pas exactement sur la droite. Le réseau répond $ax + b$ pour toute valeur de x présentée dans la figure à gauche. Par contre Le graphique de droite n'a pas su généraliser les données car chaque couple $(x_i, f(x_i))$ positionné en dehors de la droite va influencer la décision.

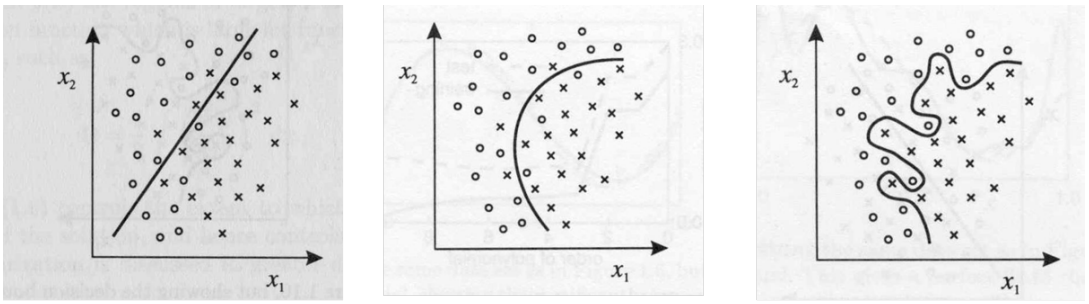

FIGURE 3.6 – Illustration des deux problèmes d'apprentissage

Ces problèmes peuvent aussi être rencontrés à cause du test d'arrêt de l'apprentissage puisqu'il n'est pas précisé par l'algorithme. Dans ce cas, on se contente de s'arrêter dès que l'erreur estimée passe sous un seuil prédéfini. En effet, le critère d'arrêt dépend de l'erreur observée et mesurée sur l'ensemble d'apprentissage et non de l'erreur réelle puisqu'on est intéressé par la généralisation et non pas l'erreur d'apprentissage.

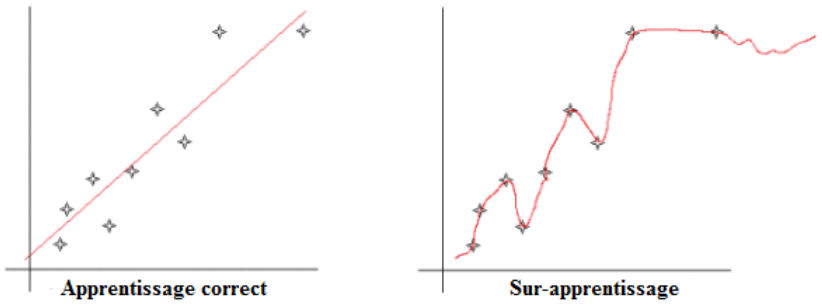

FIGURE 3.7 – Illustration du problème de sur apprentissage

Dans la plupart du temps, pour le PMC, le choix de l'architecture se fait de manière ad hoc ou avec des heuristiques simples en essayant diverses architectures pour un problème donné et en calculant l'erreur de généralisation pour chacune sur un ensemble de validation. Bien qu'il existe une recherche exhaustive dans ce domaine, aucune méthode n'est connue pour déterminer l'architecture optimale pour un problème donné. Les travaux de la littérature dédiés à ce problème peuvent être catégorisés comme suit :

Les auteurs de [131], [160], [25] cherchent la meilleure combinaison du nombre de couches et neurones cachés, le choix des facteurs d'entrée, les paramètres de l'algorithme d'apprentissage, etc., en présentant des méthodes statistiques qui sont utilisées pour

étudier l'effet des paramètres du réseau et choisir les valeurs appropriées pour eux selon la performance du modèle.

Autres travaux [82], [178], [157], [21], [118], [125], [190] commencent par une architecture initiale minimale (resp. maximale ou complète) et ajoutent (resp. suppriment) des couches et, ou des connexions au fur et à mesure de l'apprentissage en utilisant un critère spécifié a priori pour évaluer la performance du réseau après modifications. Cette catégorie de méthodes est appelée algorithmes constructifs et / ou d'élagage. Supprimer (afin d'éliminer les poids ou neurones inutiles) ou ajouter des neurones est basé sur les règles suivantes :

— Si l'apprentissage est lent ou l'erreur quadratique moyenne n'est pas minimale, alors les neurones sont ajoutés au RNA ;
— Si l'erreur quadratique moyenne n'est pas modifiée en changeant la valeur du neurone ou lorsque les valeurs de poids qui sont associés aux neurones demeurent constantes pendant plusieurs itérations lors de l'apprentissage, ces neurones seront éliminés ;

La convergence vers l'optimum global n'est pas garantie par ces méthodes alors elles peuvent être bloquées au minimum local. Aussi les méthodes constructifs rencontrent souvent le problème de la sur-spécialisation.
Depuis peu, on commence à utiliser des méthodes d'optimisation [23], [42],[80], [78] [216], [11], [152] pour chercher l'architecture idéale. Ces travaux proposent l'utilisation des algorithmes génétiques pour optimiser l'architecture du perceptron multi-couches en variant le nombre de couches et de neurones cachés par l'application des opérateurs génétiques. Différentes architectures sont évaluées selon une fonction objective.

3.4 Description de la nouvelle modélisation mathématique

Fondée en partie sur le perceptron multi-couches, notre approche, baptisée OPT-MLP (perceptron multi-couches optimal), se propose de supprimer des couches inutiles de la fin d'un réseau de tailles maximale afin d'avoir un réseau optimal capable de reconnaitre correctement les formes. L'idée de la suppression des couches et des neurones de la fin du réseau provient du fait de la simplicité de liaison.

L'architecture optimale, fournie par la méthode proposée, évite le problème de sous ou sur apprentissage qui est très courant en utilisant une architecture définie arbitrairement ou par tâtonnement. Pour élaborer ce modèle, nous avons modélisé le problème du choix de l'architecture mathématiquement avec une fonction objective non linaire, des contraintes linéaires et des variables mixtes. Ce modèle cherche le nombre idéal de couches et de neurones dans chaque couche ainsi que la meilleure stabilisation des poids du réseau.

Pour donner une description détaillée du modèle proposé, nous définissons premièrement les notations utilisées pour la modélisation du choix de l'architecture du PMC :

— X : Ensemble des entrées du réseau ;
— Y : Ensemble des sorties calculées par le PMC ;

— D : Ensemble des sorties désirées ;
— W : poids du réseau ;
— V : Vecteur de valeurs binaires associées aux couches cachées du réseau
— Q : Nombre des entrées ;
— n_i : Nombre de neurones de la couche i ;
— N_{max} : Nombre maximal de couche cachées ;
— n_{max} : Nombre maximal de neurones cachés ;
— N_{opt} : Nombre optimal de couches cachées ;
— n_{opt} : Nombre optimal de neurones cachés ;
— $w^s_{j,k}$: Poids de la connexion entre le j^{ieme} neurone de la couche s et le k^{ieme} neurone de la couche $s+1$;
— y^s_j : Sortie du j^{ieme} neurone de la couche s ;

Nous définissons par la suite les différentes entités du modèle proposé :

3.4.1 Variables du modèle proposé

Les variables du modèle sont des variables mixtes (binaires et réelles). Comme nous cherchons l'architecture adéquate du réseau avec une stabilisation des poids, nous avons défini trois variables : deux variables binaires v_s et $u_{s,j}$ qui sont respectivement affectées à chaque couche s et chaque neurone j de la couche s et une troisième variable $w^s_{j,k} \in \mathbb{R}$ de type réel affectée aux connexions entre les neurones du réseau. Les deux premières variables sont définies comme suit :

$$v_s = \begin{cases} 1 & \text{si s est la dernière couche cachée} \\ 0 & \text{sinon} \end{cases}$$

$$u_{s,j} = \begin{cases} 1 & \text{si le neurone j exist dans la couche s} \\ 0 & \text{sinon} \end{cases}$$

Ces deux variables jouent un rôle de décision dans le modèle proposé. En effet, la première décide si les couches qui se situent à la fin du réseau participeront à l'apprentissage ou non. la seconde joue approximativement le même rôle en décidant si le neurone j restera sur la couche s et participera à l'apprentissage ou non.

3.4.2 Fonction objectif

La fonction objectif du problème d'optimisation proposé pour le choix de l'architecture du PMC est l'erreur empirique que le réseau vise à minimiser durant l'apprentissage :

$$\|f(X,U,V,W) - D\|^2 \tag{3.13}$$

Avec :

$$f(X,U,V,W) = (y_1, y_2, ..., y_{n_{N_{max}+1}}) \tag{3.14}$$

La nouveauté réside dans l'introduction des variables binaires que nous avons définies précédemment dans le calcul des sorties du réseau. Nous calculons la sortie du neurone k de la

couche de sortie comme suit :
$\forall k \in \{1, ..., n_{N_{max}+1}\}$ nous avons :

$$y_k = \sum_{s=2}^{N_{max}} g\left(\sum_{j=1}^{n_{N_s}} w_{j,k}^{s}\left[g\left(\sum_{l=1}^{n_{s-1}} w_{l,j}^{s-1} h_l^{s-1} u_{s-1,l}\right) v_s\right] u_{s,j}\right) + g\left(\sum_{j=1}^{n_1} w_{j,k}^{1}\left[g\left(\sum_{l=1}^{n_0} w_{l,j}^{0} x_l\right) v_1\right] u_{1,j}\right) \tag{3.15}$$

g est la fonction d'activation.

$g\left(\sum_{l=1}^{n_{s-1}} w_{l,j}^{s-1} h_l^{s-1} u_{s-1,l}\right)$ représente la sortie du j_{eme} neurone de la couche s. Nous multiplions la sommation pondérée par $u_{s-1,l}$ qui décide la participation ou l'absence du l_{eme} neurone dans la couche $s-1$. Cette sortie est aussi multipliée par v_s pour ne préserver que seule dernière couche cachée. Celui ci représente une entrée, pour la couche s, pondéré par les poids des connexions entre la couche s et $s-1$ et les valeurs $u_{s,j}$ associés à chaque neurone j. Nous avons ajouté, dans une somme séparée, le terme associé aux sorties de la première couche cachée car il utilise toutes les entrées du réseau.

L'introduction des variables de décisions dans le calcul de la sortie global du réseau veille à jouer le rôle de décision si des couches et des neurones vont participer au calcul de la sortie ou non.

3.4.3 Contraintes du modèle

La fonction objectif est définie sous deux contraintes :

la première assure que le réseau contient au moins une couche cachée. Cette contrainte assure aussi la présence d'une seule couche finale dans le réseau Nous définissons cette contrainte comme :

$$\sum_{s=1}^{N_{max}} v_s = 1 \tag{3.16}$$

La seconde contrainte garantie que chaque couche cachée contient au moins un neurone. Nous définissons celle-ci comme :

$$\sum_{j=1}^{n_s} u_{s,j} \geq 1 \forall s \in \{1, ..., N_{max}\} \tag{3.17}$$

Nous arrivons à la formulation finale du programme mathématique proposé qui est écrit sous la forme d'une fonction objectif non linéaire (quadratique), avec contraintes et a variable mixtes :

$$\begin{cases} \min \|f(X,U,V,W) - D\|^2 & \\ \sum_{s=0}^{N_{max}} v_s = 1 & \\ \sum_{j=0}^{n_{max}} u_{s,j} \geq 1 & \text{pour chaque couche cachée s} \\ v_s \in \{0,1\} & s = 1,...,N_{max} \\ u_{s,j} \in \{0,1\} & s = 1,...,N_{max} \quad j = 1,...,n_s \\ W = (w_{i,j}^k)_{\substack{1 \leq i \leq n_k \\ 1 \leq j \leq n_{k+1} \\ 1 \leq k \leq N}} & \end{cases} \tag{3.18}$$

La modélisation proposée consiste à déterminer l'optimum de la fonction objectif avec contraintes sur un ensemble fini de choix souvent très grand [17]. L'obtention de l'optimum (architecture optimale) est liée à la bonne résolution du programme mathématique proposé. Cette problématique entre dans le cadre de l'optimisation combinatoire qui occupe une place très importante en recherche opérationnelle, mathématiques appliquées et informatique et autre secteurs. Cette discipline fait alors l'objet de plusieurs études motivées par la difficulté des problèmes d'optimisation [70] [173] et par ses nombreuses applications issues de la réalité [18] qui sont modélisables sous la forme d'un problème d'optimisation combinatoire. La plupart de ces problèmes sont NP-difficiles et ne possèdent pas de solution algorithmique efficace valable pour toutes les données [26]. En effet, le nombre de solutions possibles augmente d'une manière exponentiel par rapport à la taille du problème et par conséquent le temps de calcul pour chercher la solution optimale devient critique. Cela a donné naissance à plusieurs méthodes de résolution, de nature différente, en recherche opérationnelle et en intelligence artificielle.

On distingue deux grandes classes de méthodes : les méthodes exactes et les méthodes approchées [61]. La première catégorie (Outer Approximation (OA) [85], Branch-and-Bound (BB) [176], Extended Cutting Plane [210]) est conçue pour résoudre des problèmes de taille raisonnable et permet d'obtenir, à la convergence, la solution exacte du problème d'optimisation considéré avec une garantie absolue [175]. Malgré les progrès réalisés, avec le développement du web et l'augmentation de la masse de données jour après jour, les méthodes de cette classe sont limitées par la taille importante des applications [46] [96] [168] [90] ; ce qui affecte négativement le temps de calcul pour atteindre l'optimum. En revanche les méthodes approchées constituent une alternative très importante pour traiter les problèmes de grande taille [62] mais ne garantissent pas de trouver une solution exacte. Elles fournissent des solutions de bonnes qualités en un temps raisonnable.

Les méta-heuristiques représentent une nouvelle génération de méthodes approchées puissantes et générales [99] [67] [146] caractérisée par leur capacité à s'échapper des optima locaux par la dégradation de la fonction objectif au cours de leur progression et l'utilisation d'une population de solutions. Une autre caractéristique très importante réside dans leur adaptation à une large classe de problèmes [164][79] [78] [73]. Il y a quatre principes généraux sur les quels reposent les méta-heuristiques : Voisinage, diversification, intensification et mémoire. Les méta-heuristiques représentent une gamme très riche de méthodes de résolution de problèmes d'optimisation ; on cite par exemple : Récuit simulé, Recherche tabou, Colonie de fourmis, essaim de particules, Algorithme génétique (AG). Dans ce travail, nous mettons

l'accent sur les algorithmes génétiques que nous avons choisis pour résoudre notre modèle mathématique de sélection d'architecture. La description de cet algorithme ainsi que son adaptation pour la résolution de notre problème seront développées ci après.

3.5 Approche de résolution par les Algorithmes génétiques

3.5.1 Généralités sur les Algorithmes génétiques

L'apparition des algorithmes génétiques comme méthode d'optimisation stochastique était en 1975 par John Holland [110]. David Goldberg a largement contribué par la suite à l'enrichissement de ces algorithmes [91]. Les AGs ont été largement utilisés pour l'optimisation de différents problèmes depuis leur apparition [52]. Les principes fondamentaux de ces algorithmes s'inspirent et préserve même le vocabulaire de la génétique et la théorie de l'évolution naturelle chez l'être humain (individu, population, génotype, gène, parent, enfant, reproduction, croisement, mutation, génération).
Ils consistent à produire un cycle de reproduction d'un ensemble (population) de solutions possibles (individus) pour l'échange de l'information grâce aux opérateurs d'évolution et en reposant sur le principe Darwinien (survival of the fittest).
Le fonctionnement d'un algorithme génétique est caractérisé par sa simplicité : on part d'une population d'individus initiaux (souvent arbitrairement choisie) dont l'adaptation à leur entourage est évaluée par la suite à l'aide de leur performance relative (fitness) liée à l'objectif du problème. Une nouvelle population des individus survis (les autres disparaissent car ils sont les moins adaptés) est crée à la base de ces performances en utilisant des opérateurs génétiques simples : la sélection, le croisement et la mutation. Ce cycle est répété jusqu'à la satisfaction d'un critère d'arrêt et avoir une solution satisfaisante. Les membres de la dernière génération sont habituellement très différents de leurs ancêtres car ils possèdent de l'information génétique à travers l'héritage génétique. Ce dernier est très important car il permet à la population d'être adaptée et donc répondre au critère d'optimisation.
La figure (3.8) illustre les principales étapes d'un AG. Chacune de ces étapes est décrit soigneusement dans la suite :

- **Codage** : la façon avec laquelle on code les individus est très importante pour la réussite de l'optimisation par AG. Cette première étape consiste à modéliser chaque point de l'espace de recherche sous forme d'une structure de données spécifique appelé génotype ou ensemble de chromosomes. Ce codage caractérisera dans la suite de l'algorithme chaque individu de la population. Il existe différents types de codage comme : codage binaire et le codage réel. Le choix du type de codage dépend de l'objectif du problème traité.
- **Génération de la population initiale** : Le point de départ de l'évolution de la population au cous du cycle de reproduction est la génération des dispositifs de départ. Le bon choix de ces derniers conditionne fortement la rapidité de l'algorithme. Toutefois, l'initialisation aléatoire de la population initiale est plus simple à réaliser puisqu'elle génère les valeurs des gènes au hasard selon une distribution uniforme. Néanmoins, le faite de guidée la génération initiale vers des sous domaines intéressants

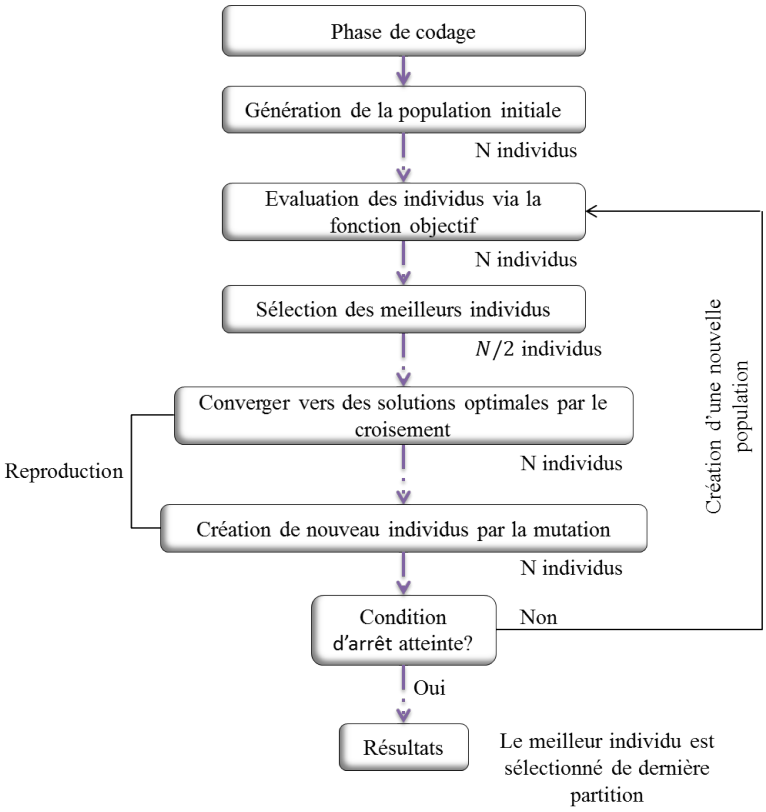

Figure 3.8 – Différent étapes d'un algorithme génétique

de l'espace de recherche peut être fructueux pour les étapes suivantes. Par exemple si on a un problème d'optimisation sous contraintes, il est préférable de générer des éléments réalisables. Une taille ainsi qu'une diversité assez importants de la population initiale garantissent un parcours de l'espace d'état dans un temps limité.

— **Fitness** : La fitness est une fonction d'évaluation définit pour mesurer la performance des individus de la population afin de juger leur qualité d'adaptation et les comparés aux autres. La définition de cette fonction ne suit aucune règle, elle peut être quelconque mais elle est généralement liée au formalisme du critère d'optimisation.

— **Sélection** : Cette phase repose sur la fonction d'évaluation pour sélectionner statistiquement les meilleurs individus d'une population et éliminer les mauvais pendant le passage d'une génération à une autre. Cet opérateur de reproduction vise également à donner une chance aux mauvais éléments car ils peuvent, par les autres opérateurs, engendrer des enfants pertinents. Différents techniques de sélection sont proposées dans la littérature, on cite par exemple : sélection uniforme, sélection par tournoi, sélection par roulette. Pour plus de détail sur ces techniques on réfère le lecteur à [61].

— **Croisement** : Cet opérateur joue un rôle très important pour garantir l'explosion de l'espace de recherche et assurer une diversité de la population en manipulant la structure des chromosomes de deux individus appelés parents afin de générer deux enfants. La combinaison de deux parties des parents est réalisée sous l'espérance qu'un des deux enfants au moins héritera de bons gènes de leurs ancêtres et par conséquent sera mieux adapté qu'eux. Le croisement n'enrichit pas seulement les solutions possibles

dans la population mais aussi il dirige l'algorithme vers la convergence. Il existe plusieurs méthodes de croisement par exemple le croisement uniforme, en un point, ou en multiples points.

— **Mutation** : Si le croisement intensifie la recherche pour la convergence en creusant dans des zones spécifiques, la mutation transporte cette recherche vers une nouvelle région pour assurer la diversification et éviter une convergence prématurée de l'algorithme vers un optimum local. En addition, elle évite la génération d'une population uniforme incapable d'évaluer durant le processus de l'algorithme. Pour cela, il est nécessaire que le taux de croisement soit plus élevé que la mutation pour éviter l'exploration aléatoire de l'espace de solutions. Pratiquement, l'opérateur de mutation opère sur l'information génétique de l'individu sélectionnés pour cette étape en modifiant un ou plusieurs de ses gènes. Un exemple illustratif du principe de mutation est montré dans la figure (3.9).

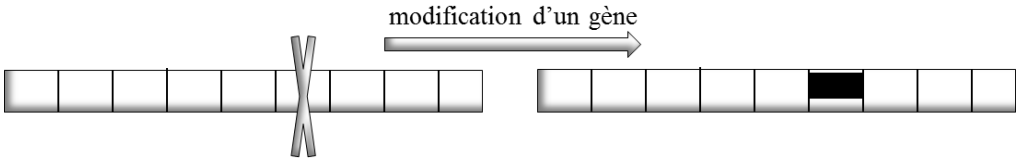

Figure 3.9 – Example illustrant la modification d'un gène pour la mutation

— **Remplacement** : Cette étape intervient à la fin de chaque itération de l'algorithme pour déterminer parmi les individus résultant de la phase de reproduction et leurs parents, ceux qui vont passer à la nouvelle population. Parmi les techniques de remplacement on trouve une appelée : Remplacement stationnaire qui remplace toujours les parents par les enfants sous l'hypothèse que ces derniers sont plus performants que leurs parents. En revanche, Un deuxième exemple de remplacement consiste à ne conserver que les meilleurs individus parmi les parents et leurs enfants. Ce type est appelé remplacement élitiste.

— **Test d'arrêt** : Comme tout algorithme itératif, le test d'arrêt joue un rôle important pour mettre fin au processus d'apprentissage et indiquer la solution qui est suffisamment approchée de l'optimum. Il faut bien choisir ce critère car un arrêt prématuré ne donne pas de bonne optimalité. On peut citer des exemples de test d'arrêt comme : après un nombre fixé de générations, lorsque l'évolution de la population est ralentie ou lorsque la fitness d'un individu dépasse un certain seuil fixé au départ.

Comme nous avons cité précédemment, un grand avantage de l'AG est son adaptation à la nature du problème. Le paragraphe suivant présente en détail l'adaptation de l'AG à notre problème d'optimisation.

3.5.2 Adaptation des Algorithmes génétiques au modèle proposé

Arrivons à ce stade, nous exposons notre adaptation de l'algorithme génétique à la modélisation mathématique que nous avons proposée précédemment. Nous définissons en premier lieu, le type de codage utilisé, suivie par la génération de la population initiale. La fonction d'évaluation des individus utilisée, ainsi que les deux opérateurs d'évolution de

la population sont ensuite présentés en détail. Enfin, l'étape de remplacement est mise en œuvre afin de déterminer les individus participant à l'itération suivante.

Expliquons tout d'abord la raison pour laquelle nous avons choisi l'AG comme méthode d'optimisation :

— La possibilité de tomber sur un minimum local est réduite parce que la procédure d'échantillonnage de l'AG commence à partir d'une population initiale
— Le cycle de reproduction est effectuée à l'aide des opérateurs génétiques d'une manière stochastique et pas déterministe
— Le résultat de l'AG est indépendant de la continuité de la fonction objectif ou l'existence de sa dérivée, parce que l'algorithme fonctionne sur une forme codée des paramètres

Définissons alors les différentes entités de l'AG adapté au problème d'optimisation d'architecture du PMC :

3.5.2.1 Codage

Le codage des individus (solutions possibles du problème) de la population doit refléter efficacement l'architecture d'un réseau multi-couches avec nombre de couches, nombre de neurones dans chaque couche et poids de connexions à l'aide des variables de la modélisation (figure (3.18)). Pour cela, la représentation choisie contient trois chromosomes : un vecteur, une matrice et un cube. Les deux premiers chromosomes sont codés selon un codage binaire car ils présentent des variables de décision et la troisième selon un codage réel car elle est une variable réelle. Le vecteur représente les couches cachés présentes dans le réseau tel que le 1 indique la dernière couche cachée selon la définition de la variable v_i. Chaque ligne de la matrice représente les neurones présents dans chaque couche cachée du réseau selon la définition de la variable u_i. Quant au troisième chromosome, chaque matrice du cube représente les connexions entre les neurones de deux couches successives dans le réseau.
A titre d'exemple, la figure (3.10) présente une modélisation (codage) d'un individu de la population. La figure illustre deux chromosomes U et V de l'individu qui correspondent respectivement aux neurones et couches du réseau. Cet exemple représente un réseau contenant cinq couches : la première, deuxième et quatrième contiennent trois neurones, dans le troisième on a quatre et dans la cinquième on a cinq.
La modélisation qu'on a adaptée prend en considération la simplicité et la clarté du codage dans le but de simplifier la manipulation des individus par les opérateurs génétiques dans la suite de l'algorithme.

3.5.2.2 Génération des individus initiaux

L'initialisation de la population représente un point de départ très important pour l'initialisation de l'AG. Le choix du point de départ représente généralement un défi pour les chercheurs. Dans ce travail nous générons un ensemble d'individus initiaux aléatoirement selon une distribution uniforme. En addition, pour contribuer à la réussite de l'algorithme, nous guidons l'initialisation de la population vers des sous domaines intéressants par la génération d'individus satisfaisant les contraintes du problème (figure (3.18)).

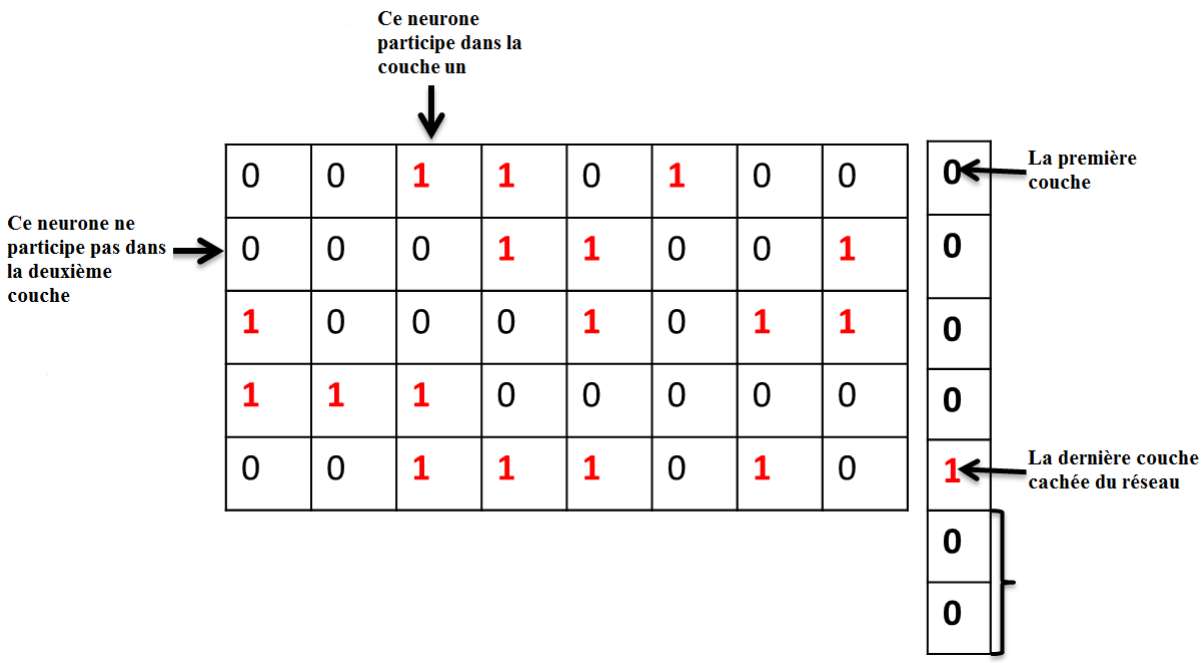

FIGURE 3.10 – Exemple du codage adapté pour l'approche OPT-MLP d'un individu de la population

3.5.2.3 Fitness

Nous définissons la fonction d'évaluation $fit(i)$ (équation (3.19)), associée à chaque individu i de la population. Cette fonction retournera une valeur numérique indiquant la performance de l'individu ainsi que sa chance d'être sélectionné pour les étapes suivantes. La Fitness (3.19) représente l'inverse de l'erreur du réseau i, indiquée dans l'équation (3.13), après apprentissage par rétro-propagation de gradient.

$$fit(i) = \frac{1}{1 + E(i)} \tag{3.19}$$

Les individus ayant une grande fitness auront une grande performance.

3.5.2.4 Sélection

Dans ce travail, la sélection des individus, qui vont participer par la suite à la mutation et le croisement, est faite aléatoirement selon une distribution uniforme.

3.5.2.5 Croisement

La première étape d'évolution de la population aura lieu à ce niveau à l'aide de l'opérateur de croisement. La recherche de la convergence est élaborer par une technique classique (croisement en un point) en combinant des parties de deux parents choisis dans la

phase de sélection. Choisissons un point de croisement différent pour chaque chromosome, la permutation des parties parentales donne naissance à deux enfants : la première partie du parent 1 jusqu'au point de croisement et la seconde du parent 2 partent à l'enfant 1 et le reste à l'enfant 2.

Pour donner une visualisation de la technique adaptée pour le croisement, nous présentons l'exemple (3.11).

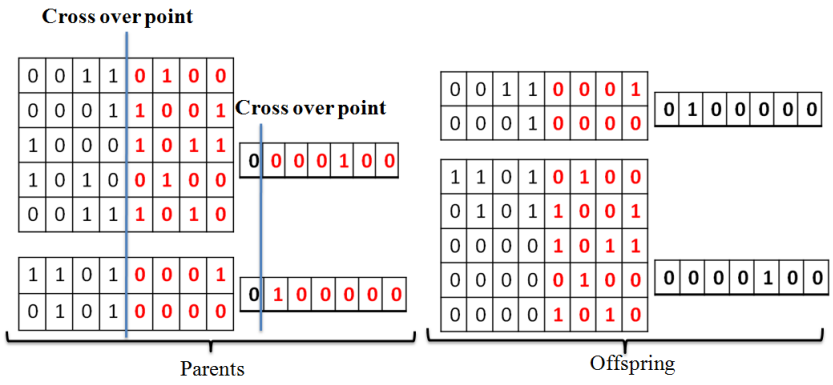

Figure 3.11 – Exemple illustratif de la manière avec laquelle deux parents de la population sont croisés à l'aide du croisement en un seul point mais différent pour chaque chromosomes

3.5.2.6 Mutation

Afin de diversifier notre population et éviter la convergence prématurée, les parents sélectionnés pour la mutation à l'aide d'une probabilité p_m, sont perturbés par des petits changement de leurs informations génétiques. Le principe de ce changement consiste à permuter deux gènes choisis au hasard du chromosome V et deux autres de chaque ligne du chromosome U. La figure (3.12) présente un exemple de la technique choisie pour la mutation.

3.5.2.7 Remplacement

Pour la création de la nouvelle population, nous utilisons le principe de la méthode d'élitisme en conservant les meilleurs individus parmi les anciens et reproduits.

Dans la section suivante, nous présentons une simulation et examination de la méthode proposée dans le but d'illustrer ses avantages et la comparer à autres méthodes de littérature.

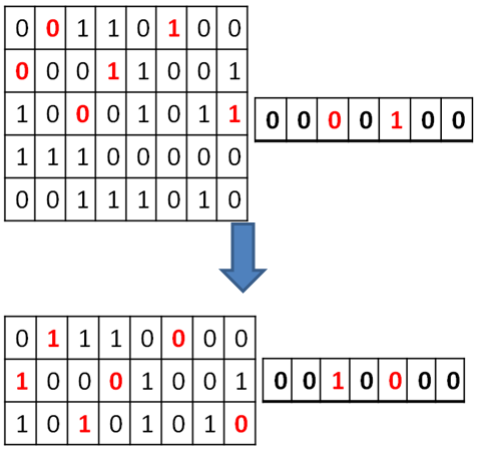

Figure 3.12 – Description de la technique utilisée pour la mutation d'un individu de la population

3.6 Résultats expérimentaux

Dans cette section nous visons à montrer la capacité de notre méthode à réduire l'architecture d'un PMC et donner des résultats satisfaisants en terme de classification. La performance de la méthode est alors examinée via des expériences réalisées en utilisant cinq bases de données (Iris, Pima Indian Diabetes, Wine, Hayes-Roth and Glass) connues dans la littérature de la classification. Ces données sont issues de l'université de California-Irvine (UCI) machine learning repository [1]. Le tableau (3.1) liste les caractéristiques des bases de données utilisées : elles sont de divers types, couvrent des exemples de différentes tailles et différent nombres de classes et d'attributs.

Table 3.1 – Caractéristiques des bases de données utilisées pour la validation de notre modèle

Base de données	**Taille**	**Données d'apprentissage**	**Données de test**	**Nombre d'attributs**	**Nombre de classes**
Glass	214	171	43	9	6
Iris	150	120	30	4	3
Wine	178	142	36	13	3
Pima	768	614	154	8	2
Hayes-Roth	132	106	26	4	3

Le tableau (3.1) montre le nombre total d'instances dans chaque base de données, avec

le nombre de données utilisées pour la phase d'apprentissage et la phase de tests, le nombre d'attributs et le nombre de classes. Ces deux nombre déterminent respectivement le nombre de neurone dans la couche d'entrée du PMC et celui des neurones de la couche de sortie.

Quatre expériences ont été élaborées dans ce travail pour donner plus de fiabilité au résultats. la base de données est divisée sur deux : 80% pour la base d'apprentissage et 20% pour le test.

L'objectif de la première simulation est de montrer la capacité de notre méthode à optimiser le nombre de couches cachées et des unités cachées pour les cinq bases de données. Le tableau (3.2) présente le réseau optimisé trouvé par la méthode proposée pour chaque base de données. En plus, ce tableau présente l'erreur associée à la fonction objectif du modèle (3.18) pour la phase d'apprentissage et de test.

Table 3.2 – L'architecture optimale obtenue par OPT-MLP pour les cinq bases de données

Bases de données	$\{\mathbf{N_{opt}}, \mathbf{n_{opt}}\}$	**Erreur d'apprentissage**	**Erreur de test**
IRIS	$\{1, 6\}$	0.047	0.005
GLASS	$\{1, 11\}$	0.448	0.441
WINE	$\{1, 7\}$	0.09	0.13
Hayes-Roth	$\{1, 8\}$	0.28	0.44
Pima Indian diabetes	$\{2, (7, 5)\}$	0.32	0.37

Le tableau (3.2) indique clairement que la méthode OPT-MLP est capable d'optimiser le nombre de couches et de neurones cachés du perceptron. Ces paramètres trouvés sont suffisants pour donner une erreur satisfaisante de classification pour toutes la bases utilisées. Dans ce contexte, nous remarquons que l'erreur de test est inférieure à celle d'apprentissage pour IRIS et GLASS. De plus ce tableau montre qu'une seule couche est suffisante pour séparer les classes de données sauf PIMA qui a besoin de deux couches pour une bonne séparation de données.

Le premier objectif de ce travail est l'optimisation d'architecture, alors une comparaison avec des méthodes qui ont le même objectif sera bénéfique. Nous allons comparer avec un travail récent (AOM) [80] qui a un principe semblable à celui développé dans ce travail. Les auteurs de [80] ont validé leur méthode sur la base de données IRIS et ils ont optimisé le nombre de neurones dans deux couches cachées. Les résultats de la comparaison sont stockés dans le tableau (3.3).

— $n_{i,max}$: le nombre initial de neurones dans la couche i
— $n_{i,opt}$: le nombre optimal de neurones dans la couche i

Le tableau de comparaison montre que l'approche OPT-MLP est plus performante en terme d'optimisation en comparaison avec l'approche AOM. En effet, le réseau optimal trouvé

Table 3.3 – Résultats de l'approche OPT-MLP en comparaison avec l'approche AOM pour la base de données IRIS

$n_{1,max}, n_{2,max}$	$n_{1,opt}, n_{2,opt}$	
	OPT-MLP	**AOM**
7.7	0.4	3.4
7.7	0.4	3.4
9.9	0.4	3.3
10.10	0.7	4.4

par OPT-MLP utilise une seule couche qui contient un nombre plus petit de neurones tandis que l'approche AOM utilise deux couches qui contient un nombre de neurones plus grand.

Autre comparaison avec des méthodes désignées pour l'optimisation de l'architecture du perceptron [97][141][113][118][40][114] a été effectuée pour évaluer l'efficacité de notre approche. Les résultats sont rangés dans le tableau (3.4). On note que cette comparaison est validée sur la base IRIS avec 222 données d'apprentissages et 78 données de test.

Table 3.4 – Résultats obtenus par l'approche OPT-MLP en comparaison avec six méthodes d'optimisation de l'architecture pour la base IRIS

Algorithmes	**Nbr de C.C**	**Nbr de N.C**	**Taux de reconnaissance (%)**
OPT-MLP	1	14	100
CP-NN	1	19 ± 2	95.57 ± 1.03
EFAST FNN	1	21±2	93.81±1.32
AGPNNC	1	24±2	93.27±1.32
AMGA	1	23±2	94.33±2.22
MPANN-HN	1	100	94.52±1.46
EI-ELM	1	121±8	95.52±2.16

— Nbr de C.C : Nombre de couches cachées
— Nbr de C.C : Nombre de neurones cachés

A partir du tableau (3.4) nous constatons que toutes les méthodes utilisent une seule couche pour séparer les données de la base d'IRIS. Par ailleurs, notre approche fournit le meilleur nombre de neurones cachés ainsi que le meilleur et l'optimal taux de classification en comparaison avec les méthodes citées dans ce tableau.

D'après les expériences précédentes, notre approche fournit des résultats satisfaisants, en terme d'optimisation. Cela prouvent la puissance et la performance de l'approche OPT-MLP.

La comparaison suivante sera effectuée dans le but d'illustrer les capacités classifiantes du réseau optimal trouvé par notre approche. Nous comparons le taux de reconnaissance des classes trouvées par notre approche à celui trouvé par douze méthodes de classification : QSTAR algorithm (Q*), Incremental Decision Tree Induction software (ITI), CN2 software (CN2), C4.5, C4.5 rules, 5-Nearest Neighbor (K5), Learning Vector Quantization (LVQ), Radial Basis Function (RBF) network, Nevada back propagation (NEV-prop), Linear Machine Decision Tree (LMDT) and Oblique Classifier (OC1). Ces classifieurs sont proposés dans [109], [72], [116].

Nous définissons la grandeur A selon laquelle nous comparons nos résultats à ceux des autres méthodes dans le tableau (3.5).

$$A = \frac{\text{Nobre total des objets bien classés}}{\text{Nombre total des objets}} \times 100 \tag{3.20}$$

Table 3.5 – Résultats de classification obtenus par l'approche OPT-MLP en comparaison avec douze classifieurs

Approches	**Iris**	**Glass**	**Wine**	**Pima**	**Hayes-Roth**
OPT-MLP	100	72.09	94.44	74.02	73.07
LMDT	95.45	60.59	95.4	73.51	78.94
OC1	93.89	57.72	87.31	50	63.73
LVQ	92.55	60.69	68.9	71.28	52.26
Q*	92.1	74.78	74.35	68.5	74.84
K5	91.94	69.09	69.49	71.37	55.21
CN2	91.92	70.23	91.09	72.19	75.21
C4.5	91.6	70.23	91.09	71.02	70.03
C4.5rule	91.58	67.96	91.9	71.55	75.95
ITI	91.25	67.49	91.09	73.16	78.6
NEV-Prop	90.34	44.08	95.41	68.52	78.91
RBF	85.64	69.54	67.87	70.57	70.03

Les résultats de comparaison listés dans le tableau (3.5) traduisent la performance du réseau optimal en classification. Plus précisément, notre approche, OPT-MLP, a obtenu le meilleur taux de classification (100 %) pour la base de données IRIS et le meilleur résultat (74.02%) supérieur à tout les autres classifieurs pour Pima Indian Diabetes. De même pour la base GLASS, l'approche OPT-MLP donne un taux de reconnaissance de 72.09%

supérieur à celui obtenu par toutes les autres approches sauf Q* qui atteint 74.78%.
Concernant la base de données WINE, Notre approche a réussi à bien classer 94.44% les données. Avec ce résultat notre approche a dépassé aussi les autres classifieurs sauf les approches LMDT et NEV-Prop avec une faible différence de 1%. Pour la dernière base de validation (Hayes-Roth), nous pouvons constater que l'approche OPT-MLP reconnait les classes de 73.07% de données. Avec ce résultat, notre approche arrive à dépasser les approches suivantes : OC1, LVQ, K5, C4.5 et RBF.

Avant de clôturer cette section, nous mentionnons les remarques suivantes :
— Le modèle développé dans ce travail peut être appliqué par d'autres versions du perceptron multi-couches (comme le réseau HMLP [177], ELM [195], PNN [194]) pour bénéficier des avantages de l'approche proposée.
— Il est difficile d'effectuer des comparaisons en terme de temps de calcul en raison de la différence des outils utilisés par chercheurs comme : système d'exploitation, configuration informatique, langage de programmation, etc.

3.7 Conclusion

Suite à un état de l'art sur la problématique de sélection de l'architecture optimale du PMC, nous avons présenté notre nouvelle approche (OPT-MLP) visant à optimiser le nombre de couches et de neurones d'un réseau maximal en supprimant les unités inutiles de la fin du réseau. Cet objectif est réalisé grâce à une nouvelle modélisation mathématique non linéaire à variables mixtes sous contraintes dans le but de choisir l'architecture adéquate avec la meilleure stabilisation des poids du PMC grâce à la méthode de rétro-propagation de gradient. La résolution du modèle a été élaborée par l'adaptation des algorithmes génétiques. Afin d'évaluer la performance de l'approche OPT-MLP, des expériences en terme d'optimisation et de classification ont été réalisées en utilisant des bases de données benchmarks issues de "UCI machine learning repository". Les résultats de ces expériences montrent que l'approche OPT-MLP a prouvé une bonne capacité à optimiser les paramètres du réseau et avoir un taux de reconnaissance satisfaisant.

Les résultats sont très satisfaisants et comparables à ceux trouvés dans la littérature. Cela nous encourage à creuser davantage dans ce domaine et s'attendre à des améliorations considérables par la proposition de nouveaux opérateurs génétiques, l'ajout de nouveaux paramètres dans le modèle et l'amélioration de la phase d'apprentissage par de nouvelles méthodes moins couteux en temps de calcul.

Nous avons détaillé dans ce chapitre les caractéristiques du PMC dans le domaine de la classification supervisée. Grâce à celles-ci, nous nous sommes intéressés à adapter ce type de réseau à la classification non supervisée. Cet objectif est réalisé à l'aide d'une nouvelle méthode s'appuyant sur l'algorithme de k-means. Ce travail fait l'objet du chapitre suivant.

Chapitre 4

Contribution à la classification automatique par le réseau PMC et une nouvelle version de k-moyennes

4.1 Introduction

Dans le même cadre de sélection de paramètres, nous proposons une nouvelle méthode hybride qui vise à adapter le PMC à la classification non supervisée. Cet objectif est réalisé à l'aide d'une nouvelle approche de k-means qui consiste à générer une série de partitions pour sélectionner la meilleure. Celle ci va étiqueter les données sans avoir aucune information a priori. Baptisée H-kmeans, cette méthode est capable de surmonter le problème de sélection de paramètres initiaux de l'algorithme de k-means, chercher le nombre de clusters adéquat, ainsi que la meilleure partition. Ces résultats sont exploités par la suite, grâce aux indices d'évaluation des données de la partition, pour choisir les meilleures données de chaque classe, qui vont participer à l'apprentissage du PMC.

Pour présenter ce travail, nous commençons par détailler le fondement de la méthode de k-means et positionner sa problématique dans les sections 2 et 3. Un bref état de l'art sur les méthodes dédiées à la résolution du problème posé ainsi que les méthodes qui combinent l'algorithme de k-means avec celui du PMC, sera donné dans la section 4. Par la suite, nous proposons notre approche hybride en trois étapes : clustering, sélection et apprentissage. Plus précisément, dans la première étape, nous présentons la nouvelle méthode H-kmeans. Dans la deuxième méthode, nous expliquons la méthode de sélection des objets pertinents qui représenteront les entrées étiquetées du PMC. La dernière étape traite l'apprentissage du réseau multi-couches par les données sélectionnées. Afin de montrer l'efficacité de cette approche, nous exposons la simulation à l'aide d'une étude comparative avec d'autres méthodes, en utilisant un ensemble de bases de données benchmarks. Enfin, nous concluons et discutons d'éventuelles extensions et futures suggestions d'amélioration.

Le travail présenté dans ce chapitre a fait l'objet d'un article publié dans le journal international : "WSEAS TRANSACTIONS on COMPUTERS" [101].

4.2 Méthode k-moyennes

La méthode de centres mobiles (k-moyennes) due à Forgy [86] est la plus ancienne et la plus simple méthode de partitionnement qui se base sur l'erreur quadratique [159] comme critère d'évaluation de la partition. L'objectif général de la méthode est d'obtenir la partition optimale qui, pour un nombre k fixé de classes, minimise l'erreur quadratique. Pour atteindre cet objectif, Le principe de cette méthode est de définir a priori k centres arbitraires $c_1, \ldots, c_k$ et de regrouper les données autour du plus proche centre. Géométriquement, cette technique revient à partager l'espace des données en k zones définies par les plans médiateurs des segments $[c_i, c_j]$. Par itération, de nouveaux centres (référents) sont calculés à partir des classes qui viennent d'être formées. Par la suite, de nouvelles partitions sont générées par la technique de plus proche centre de chaque donnée jusqu'à avoir la partition optimale.

4.2.1 Fondement mathématique de k-means

La quantification vectorielle recherchée par la méthode des k-moyennes est représentée par l'ensemble des vecteurs référents W et par la fonction d'affectation $\aleph$ qui minimisent $I(W, \aleph)$ sur l'ensemble d'apprentissage A. Puisqu'on minimise les vecteurs référents et la fonction d'affectation, il faut faire apparaître la fonction d'affectation $\aleph$. La quantité $I(W, \aleph)$ que l'on cherche à minimiser s'écrit alors sous la forme :

$$I(W, \aleph) = \sum_{z_i \in A} \|z_i - W_{\aleph(z_i)}\|^2 = \sum_{c} \sum_{z_i \in P_c \cap A} \|z_i - W_c\|^2 \tag{4.1}$$

$$I(W, \aleph) = \sum_{c} \sum_{z_i \in A} \|z_i - W_c\|^2 = \sum_{c} I_c \tag{4.2}$$

L'inertie I_c représente l'erreur de quantification obtenue si l'on remplace chaque observation de P_c par son référent w_c. Alors la quantité $I(W, \aleph)$ que l'on cherche à minimiser représente la somme des inerties locales I_c. Le processus de minimisation de la fonction coût par la méthode de k-moyennes consiste à optimiser deux variables alternativement. En effet, L'algorithme procède d'une manière itérative et chaque itération comporte deux phases. La première phase minimise $I(W, \aleph)$ par rapport au paramètre $\aleph$ en supposant les valeurs des centres fixées aux valeurs calculées précédemment. Dans cette étape, on génère une nouvelle partition en calculant la nouvelle fonction d'affectation $\aleph$. La seconde phase suppose que la fonction d'affectation est fixée à la valeur qui vient d'être calculée et minimise la fonction $I(W, \aleph)$ par rapport au paramètre W. Cette procédure fait décroître la valeur de $I(W, \aleph)$ à chaque itération.

4.2.2 Phases d'apprentissage

La description d'une itération de l'apprentissage de la méthode proposée est résumée dans les étapes suivantes :

4.2.2.1 Phase d'affectation

C'est la première phase de l'itération qui consiste à minimiser $I(W, \aleph)$ par rapport à la fonction d'affectation $\aleph$, et fixer la première variable (l'ensemble des centres W). La minimisation par rapport à la partition s'obtient en affectant chaque observation z au référent W_c selon la nouvelle fonction d'affectation $\aleph$.

$$\aleph(z_i) = argmin_{r=1,...,p} \|z_i - W_r\|^2 \tag{4.3}$$

Cette nouvelle fonction d'affectation $\aleph$ définit une nouvelle partition P de l'ensemble de données, où chaque observation z_i est affectée au plus proche référent W_c. Cette partition est formée par les Voronois des référents W_c (figure (4.1)). Ce type d'affectation réduit le terme correspondant à z_i dans la fonction de coût $I(W, \aleph)$.

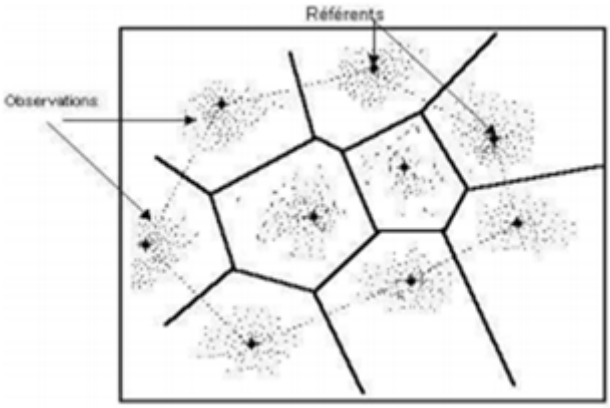

Figure 4.1 – Diagramme de Voronoi (tiré de [139]) : définition d'une partition telle que chaque sous ensemble de la partition est associé à l'un des référents

4.2.2.2 Phase de minimisation

A la seconde phase de l'itération, la fonction coût décroit à nouveau en fonction de l'ensemble des vecteurs référents et en fixant la deuxième variable (fonction d'affectation). La fonction coût $I(W, \aleph)$ est quadratique et convexe par rapport à W alors elle atteint le minimum global pour $\frac{\partial I}{\partial W} = 0$.

$$\frac{\partial I}{\partial W} = 0 = \left[\frac{\partial I}{\partial W_1}, \frac{\partial I}{\partial W_2}, ..., \frac{\partial I}{\partial W_p}\right]^p \tag{4.4}$$

Le calcul du vecteur gradient associé à tout référent W_c permet d'obtenir un ensemble d'équations vectorielles :

$$\forall c \qquad \sum_{\substack{z_i \in A \\ \aleph(z_i)=c}} (z_i - w_c) = 0 \tag{4.5}$$

Ces équations définissent les p nouveaux centres, et la mise à jour au cours de l'algorithme se fait selon cette formule :

$$W_c = \frac{\sum_{z_i \in P_c \cap A} z_i}{n_c} \tag{4.6}$$

Où n_c représente le nombre d'éléments de la classe C de la partition P. D'après cette formule les référents w_c sont alors les centres de gravité des observations $P_c \cap A$.

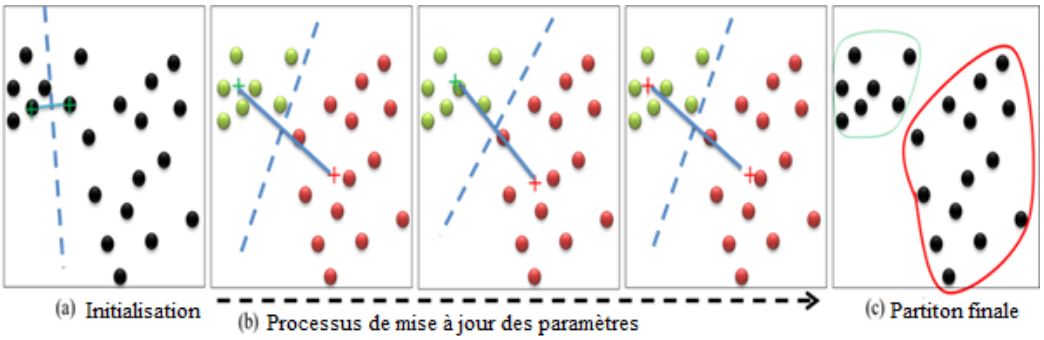

FIGURE 4.2 – : Illustration des étapes de la méthode de k-moyennes de l'étape (a) d'initialisation des centres des classes, à l'étape (c), de convergence en passant par des itérations de mises à jour et de réaffectation des données

Pour cet algorithme, Il existe une preuve de convergence citée dans le livre d'apprentissage statistique [63].

L'algorithme de la méthode "k-moyennes" [103] [158] est résumé dans l'algorithme (5). On peut utiliser la méthode du gradient stochastique, qui recalcule les référents chaque fois qu'une observation z_i est présentée.
Les étapes de la méthode de k-moyennes sont présentées sous forme d'un schéma illustratif dans la figure (4.3).

4.3 Problématique

Grâce à ses avantages, la méthode de k-moyennes a connu un grand succès et elle est largement utilisée dans plusieurs domaines et différentes applications : [172] [151] [147] [3] [163] [127]. En effet, cette méthode est caractérisée par sa facilité d'implémentation, sa complexité linéaire ($O(nkl)$ avec n le nombre de données, k le nombre de classes et l le nombre d'itérations nécessaire pour la convergence de l'algorithme), son indépendance à l'ordre de présentation des données. Aussi, le fait de comparer les données à leurs centres

associés réduit le temps de compilation de l'algorithme. La convergence est garantie par cette méthode en un nombre fini d'itérations.

Algorithm 5 Algorithme de k-moyennes

Entrées: $k, P, P', Niter$

Sorties: P

Phase d'initialisation : choisir les k référents initiaux d'une manière aléatoire

Tant Que l'inertie intra-classe ne s'est pas stabilisée ou $Niter$ n'est pas atteint **faire**

— Phase d'affectation : les k référents étant connus, générer une nouvelle partition P' en misant à jour la fonction d'affectation $\aleph^t$ associée à W^{t-1} : on affecte chaque objet à la classe dont le centre est le plus proche (équation (4.3))

— Phase de minimisation : calcul des nouveaux référents W^t (centres de gravité des classes de la nouvelle partition P') en appliquant l'équation (4.6)

$P \leftarrow P', t \leftarrow t+1$

Fin Tant Que

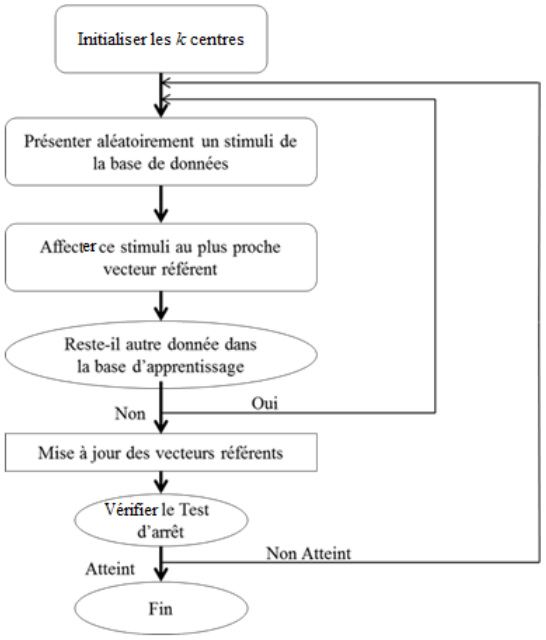

Figure 4.3 – Schéma de l'algorithme d'apprentissage de la méthode de k-moyennes

Malgré ces avantages, la méthode souffre d'une sensibilité au bruit et aux points aberrants qui influencent la valeur de la moyenne. Cette méthode n'est appliquée qu'aux données numériques où la moyenne est définie. Le problème majeur de cette méthode est la sélection des paramètres initiaux. En effet, la partition finale (optimum de la fonction (4.1)) trouvée par l'algorithme est très dépendante au choix des centres initiaux. Pour illustrer ce problème, nous présentons la figure (4.4), où nous donnons deux exemples possibles pour l'initialisation de l'algorithme de k-means.

D'après les deux cas traités dans la figure (4.4), nous constatons que si les centres initiaux ne sont pas proprement choisis (cas 2), l'algorithme tombe dans un minimum

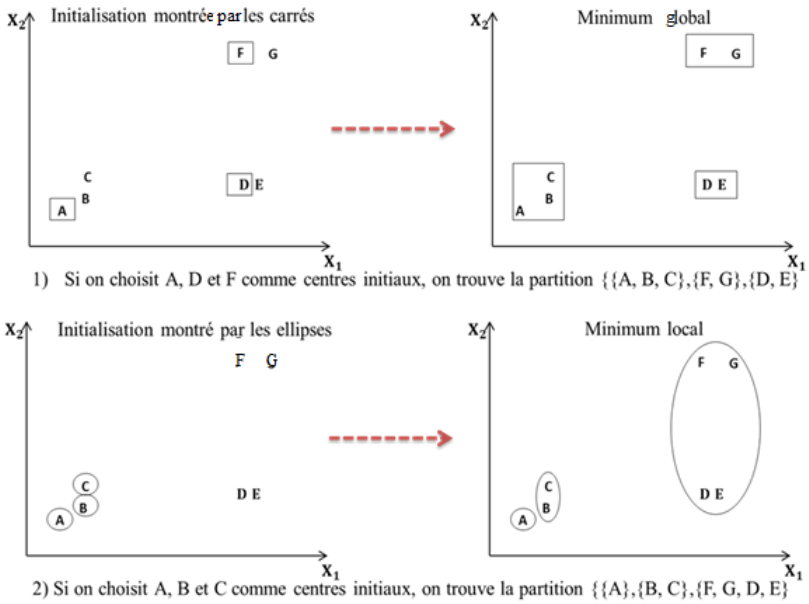

Figure 4.4 – Exemple (inspiré de [122]) d'illustration du problème d'initialisation de l'algorithme de k-moyennes

local qui ne donne pas une bonne classification. Alors qu'un choix approprié (cas 1) guide l'algorithme de k-moyennes vers le minimum global qui donne la meilleure classification.

Un deuxième problème majeur est le choix du nombre de centres k. D'après la figure (4.4), nous constatons que même si on donne le nombre exacte pour k, on aura pas la bonne solution si les centres ne sont pas bien choisis.

D'après ces remarques, nous devons s'intéresser à la sélection de ces deux paramètres.

Les perceptrons multi-couches sont les réseaux de neurones les plus couramment utilisés grâce à leurs avantages qu'on a traité précédemment. A l'aide des couches cachées, les informations issues de la couche d'entrée sont traitées et transmises vers la couche de sortie. Au cours de l'apprentissage, les poids du réseau sont ajustés afin de minimiser l'erreur quadratique entre la sortie calculée par le réseau et la sortie désirée selon l'algorithme d'apprentissage adopté. Bien que ce type de réseaux de neurones représente un classifieur universel qui joue un rôle très important en classification supervisée, le résultat qu'il fournit peu être affecté par le choix des données de l'apprentissage. La quantité et la qualité de ces données sont très importantes pour une bonne généralisation, tel que l'erreur de l'apprentissage diminue avec l'augmentation de la taille de l'échantillon d'apprentissage. Aussi, le bon choix des données d'apprentissage donne de bons résultats tel que les données présentant des anomalies, ou qui sont irrégulières, perturbent le processus d'apprentissage. Par conséquent, cela améliore les capacités d'apprentissage d'un réseau neuronal multi-couches et réduit le temps de calcul et les ressources requis par le processus d'apprentissage.

4.4 État de l'art

Plusieurs variantes de la méthode de k-moyennes [6] ont été reportées dans la littérature pour remédier aux limites de cet algorithme :

— Global k-means [148] : c'est une approche itérative qui consiste dans chaque itération à définir un nouveau centre selon une procédure de recherche déterministe globale, et tourner k-means avec les i+1 centres calculés. Elle commence par calculer le centre de gravité $\bar{x} = \frac{a}{n}\sum_{i=1}^{n} x_i$ de l'ensemble de données prévues pour la classification. Celui-ci est défini comme premier centre. A l'itération i ($i \in 1, \ldots, k-1$), N exécutions de k-means sont effectuées en considérant chacun des N données un candidat à être le $i+1$ centre. Global k-means ne dépend pas des conditions initiales. Cependant elle est couteuse en terme de calcul pour les données de grande taille, puisque elle nécessite $N(k-1)$ tournées de k-means sur l'ensemble des données. Pour cette raison, les auteurs proposent quelques modifications de la méthode pour réduire le temps de calcul par : la suppression des données les plus proches aux centres calculés dans l'itération précédente et l'utilisation de l'inégalité triangulaire qui permet d'esquiver les calculs non nécessaires.

— k-means++ [12] : Cette version propose une amélioration probabiliste de la version originale au niveau du calcul des vecteurs référents des classes. En effet, le premier centre est choisi aléatoirement alors que le i^{me} centre $x_{\prime}$ est sélectionné parmi l'ensemble de données avec une probabilité égale à :

$$\frac{md(x_{\prime})^2}{\sum_{j=1}^{n} md(x_j)^2} \tag{4.7}$$

Où $md\,(x_{\prime})$ est la distance minimale de $x_{\prime}$ aux $i-1$ centres sélectionnés précédemment. Cette méthode a une complexité de $\theta(log(k))$. Une modification de cette méthode appelée « greedy k-means++ » a été proposée dans le but d'éviter le choix de deux centres proches entre eux. Pour cela, en premier lieu, la méthode cherche $log(k)$ centres d'une manière probabiliste dans chaque itération. En deuxième lieu, elle sélectionne le vecteur référent qui minimise la fonction coût de clustering.

— K-médoides : Cet algorithme, proposé par [130], permet de classifier des données de façon plus robuste en cherchant à diminuer la sensibilité de l'algorithme k-moyennes aux valeurs atypiques dans l'ensemble de données. Cette sensibilité augmente avec l'utilisation de l'erreur quadratique. Pour diminuer cette sensibilité, la méthode vise à choisir soigneusement les représentants des k classes appelés médoïdes. Ce dernier représente l'observation d'une classe qui minimise la moyenne des distances ou dissimilarités aux autres observations de la classe en utilisant un critère d'erreur absolue E définit par :

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j| \tag{4.8}$$

L'algorithme fonctionne de manière analogue à celui de Mac Queen. À chaque itération, un médoïde est mis en concurrence avec un autre individu aléatoire. Si l'échange

améliore le critère, cet individu devient le nouveau médoïde. Cependant, une différence majeur avec l'algorithme k-moyennes est qu'un médoïde fait partie des données et permet donc de partitionner des matrices de dis-similarités. La méthode k-médoïdes cherche à alléger la sensibilité aux points aberrants mais en contrepartie, elle est limitée par le nombre d'observations (matrice de dis-similarités à stocker) et le temps de calcul (la complexité de l'algorithme dans chaque itération est $O(k(n-k)^2)$).

L'algorithme de k-means est présent dans la majorité des problèmes de clustering [169][214][217][56]. De même pour le réseau neuronal PMC qui a été utilisé dans de nombreux problèmes avec différents degrés de complexité et dans différents domaines d'application [35][161] [49][208]. La combinaison de ces deux algorithmes est bénéfique pour plusieurs travaux dans différents domaines.

Venkatesh et Kumar [203] utilisent un PMC pour la classification des images satellites multi-spectrales afin d'identifier les régions d'intérêt tels que l'eau, le sol, la forêt, la terre, etc. L'utilisation du résultat de clustering des images par l'algorithme de k-means comme entrée du PMC était avantageuse. D'ailleurs, l'étude présentée dans ce travail montre que l'hybridation PMC-k-means conduit à une classification plus précise et plus rapide des images multi-spectrales. De même, nous citons le travail d'Orhan et al. [171] où les auteurs utilisent la sortie de l'algorithme de k-means comme une entrée du réseau neuronal « PMC ». Ce dernier est introduit comme un mécanisme d'aide à la décision de diagnostic dans le traitement de l'épilepsie. Une autre application de la combinaison entre le PMC et l'algorithme de k-means, pour profiter de leurs avantages, est présentée par Al-Mohair et al. [3]. L'objectif de cette combinaison est la détection des régions de la peau avec précision en exploitant les informations de couleur et de texture. Un autre modèle, présenté par Faraoun et Boukelif [83], vise à améliorer les capacités d'apprentissage du PMC, et réduire son intensité de calcul par la réduction de l'ensemble d'apprentissage présenté au réseau multi-couches. Cet objectif est réalisé grâce à l'algorithme de k-means qui est utilisé en premier lieu pour la classification de données et la sélection des plus pertinents. Nous terminons par le travail de Azimi et al. [16] qui propose une nouvelle méthode de clustering basée sur l'algorithme de k-means, appelée TB k-means, pour surmonter le problème d'initialisation des paramètres de cet algorithme. Cette méthode est utilisée dans le but de trouver une bonne partition de données d'irradiation solaire. Les clusters qui forment la partition trouvée vont par la suite fournir les entrées du PMC après leur pré-traitement. Le réseau de neurones, à son tour, prévoit les radiations solaires pour chaque classe de données, qui vont contribuer à la prévision finale.

Bien que la combinaison du PMC et de k-means est avantageuse, mais il y a des problèmes qui se posent. Dans le but de résoudre ces problèmes, nous adaptons par la suite le réseau multi-couches pour la classification non supervisée à l'aide d'une nouvelle méthode de clustering reposant sur la méthode de k-means comme méthode d'optimisation locale.

4.5 Description de l'hybridation proposée

La proposition de cette nouvelle méthode vise à atteindre les objectifs suivants :

— Développer un package de clustering automatique
— Surmonter le problème d'initialisation de paramètres de k-means

— Sélectionner automatiquement le nombre de clustres
— Étiqueter les données d'apprentissage
— Adapter le perceptron multi-couches au problème de clustering
— Améliorer la capacité de généralisation du PMC

Pour atteindre ces objectifs, nous effectuons un processus itératif qui repose sur l'utilisation de trois techniques principales : une heuristique de prétraitement, l'algorithme de K-means et l'algorithme du PMC. Le processus de l'approche proposée se compose de trois phases principales. L'objectif de la première est la sélection de paramètres (paramètres initiaux de k-means, nombre de classes, partition optimale). Cela est réalisé grâce à l'approche H-kmeans (figure (4.5)) qui s'inspire du processus hiérarchique appliqué dans le deuxième chapitre pour la carte topologique probabiliste. Le modèle étant sélectionné, la deuxième phase consiste à étiqueter un nombre d'objets en sélectionnant les objets bien classés de chaque classe et en associant à chacun une étiquète de sa classe. La dernière phase lance l'apprentissage du PMC.

Plus précisément, le procédé est décrit avec plus de détail dans les paragraphes qui suivent.

4.5.1 Phase de sélection de paramètres

Dans cette phase, nous proposons une nouvelle méthode visant à surmonter le problème d'initialisation de l'algorithme de k-means. En plus, elle propose une nouvelle alternative pour sélectionner le nombre de classes de l'ensemble de données ainsi que la meilleure partition. Cette méthode constitue une méthode d'optimisation globale qui ne dépend d'aucun paramètre initial et emploie l'algorithme de k-means comme une méthode locale de la recherche d'optimum (partition optimale). La méthode H-kmeans utilise un processus itératif de génération de plusieurs solutions possibles pour sélectionner la meilleure grâce à des critères d'optimisation. En effet, au lieu de choisir d'une manière arbitraire les valeurs initiales des centres de clusters, comme est le cas avec la plupart des algorithmes de clustering, l'approche proposée veille à les choisir convenablement en procédant d'une manière incrémentale, et en tentant d'ajouter, à chaque étape, un nouveau centre de cluster pour avoir une nouvelle partition. La partition trouvée à chaque étape est évaluée pour sélectionner la meilleure partition parmi l'ensemble des partitions générées. Le processus général de cette approche est décrit dans la figure (4.5).

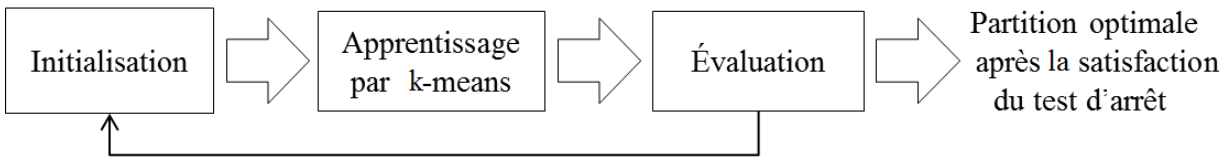

Figure 4.5 – Etapes principales de l'approche de clustering

Plus précisément, pour résoudre un problème de clustering sans savoir le nombre de clusters désiré, la méthode se déroule comme suit :

En premier lieu, l'ensemble de données est divisé en deux sous-ensembles G_1 et G_2 selon la technique de choix de la paire de données les plus loin définie dans le chapitre (2) par la méthode H-PrSOM. Les centres de G_1 et G_2 constituent les points de départ de l'algorithme de k-moyennes (initialisation). Une fois les centres sont initialisés, l'apprentissage par l'algorithme de k-means produit la partition $P^1 = \{C_1, C_2\}$. Celle-ci sera évaluée selon les deux critères d'évaluation F et R^2 que nous avons vus dans le premier chapitre.

Les étapes suivantes de cette phase sont élaborées de la même manière que la première étape, sauf que le choix de la paire de données se fait dans la classe la plus hétérogène C_h de la partition P^i de l'étape i au lieu de l'ensemble de données total. L'hétérogénéité de la classe est mesurée à l'aide du critère basé sur l'inertie des classes. Après la division de la classe C_h qui se fait selon le critère de choix de la paire de données, cette classe est substituée dans la partition P^i par les deux nouveaux groupes résultants de la division.

Chaque étape contient alors les principales tâches suivantes :

— Substitution de la classe hétérogène par deux nouveaux groupes (sauf dans la première étape)
— Calcul des centres des groupes
— Application de l'algorithme de k-moyennes
— Évaluation de la partition obtenue

Le processus est arrêté lorsque toutes les classes ont un cardinal inférieur à un seuil relativement faible ou l'hétérogénéité des classes est assez faible. Ainsi la meilleure partition est retenue. Nous présentons le diagramme récapitulatif de la méthode de clustering développé dans cette section (figure (4.6)).

En ce qui pourrait être une préoccupation pour la complexité de calcul, la méthode nécessite M ($M < N$) exécutions de l'algorithme de k-means. En fonction des ressources disponibles et les valeurs de N (nombre de données), l'algorithme peut être une approche intéressante puisque la performance de la méthode est excellente grâce à la qualité des résultats trouvés et la vitesse de l'algorithme de k-means qui représente une qualité majeur.

4.5.2 Phase d'étiquetage de données

L'évaluation des partitions générées par l'approche proposée dans la première phase fournit la sélection de la meilleure partition. Cette dernière donne le nombre de classes des données ainsi que les classes obtenues. De chaque classe, nous sélectionnons un nombre de données bien classées à l'aide de l'indice « silhouette » qui permet de déterminer les données qui sont bien classées (indice proche de 1), mal classées (indice proche de -1), et ceux qui se trouvent sur la frontière entre deux classes (indice proche de 0). Ces données représentent un échantillon discriminant qui réalisera la phase de l'apprentissage par le PMC.

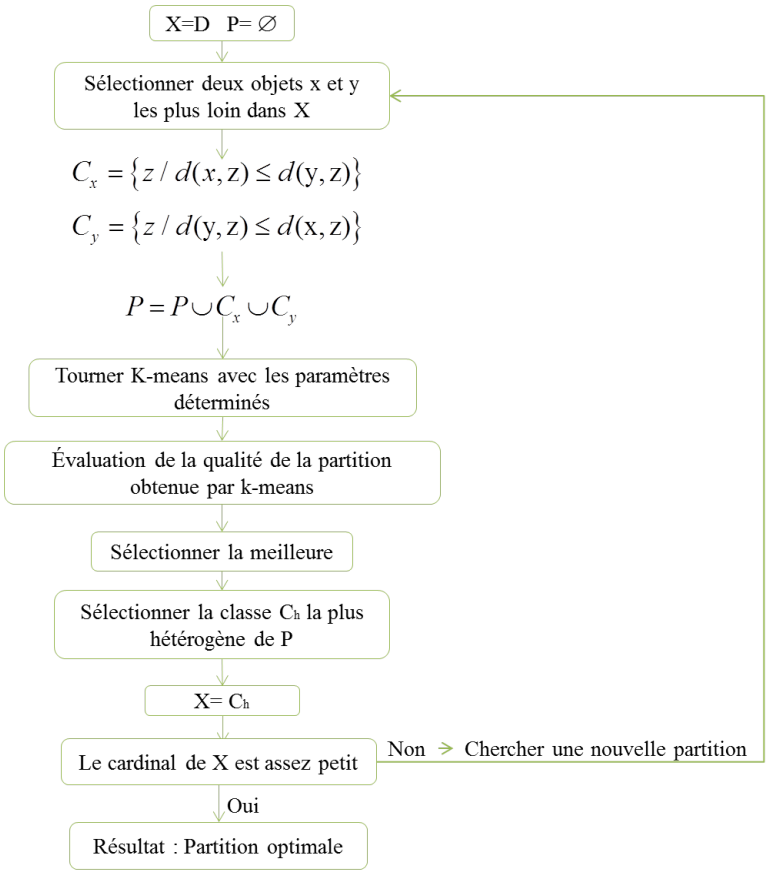

Figure 4.6 – Diagramme de la méthode H-kmeans

4.5.3 Phase d'apprentissage

Cette phase exploite le rendement des premières. En effet, le perceptron multi-couches cherche la généralisation à travers les données d'apprentissage trouvées précédemment. Le nombre de neurones de la couche d'entrée du PMC dépend de la dimension des données, alors que celui de la couche de sortie égal au nombre de classes déterminé par l'approche proposée dans la première phase. Le nombre de couches et neurones cachés sont initialisés arbitrairement.

Une illustration des étapes de la méthode globale de l'hybridation proposée est présentée dans la figure (4.7).

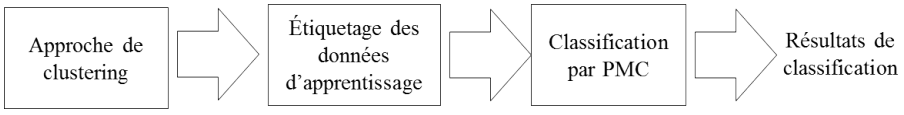

Figure 4.7 – Schéma illustratif des étapes de la l'approche globale

Pour tester la performance et la validité de la méthode proposée dans le domaine de la

Table 4.1 – Description des données utilisées pour la validation de notre approche

Base de données	Nombre d'objet	Nombre d'attributs	Nombre de classes
IRIS	150	4	3
WINE	178	13	3
SOYBEAN	47	35	4
HAYES-ROTh	132	5	3
GLASS	214	10	6

classification automatique, nous présentons dans les sections suivantes les différentes expériences effectués.

4.6 Résultats expérimentaux

Dans cette section, nous cherchons à valider la performance de la combinaison proposée et montrer la capacité du PMC adapté pour classifier des données non étiquetées. Pour se faire, nous testons l'approche proposée sur cinq bases de données benchmarks (IRIS, WINE, SOYBEAN, HAYES-ROTH, GLASS) issues de UCI machine repository. Nous présentons les résultats numériques de notre approche illustrés sur des tableaux et figures, ainsi que des comparaisons avec d'autres méthodes de la littérature.

Dans le tableau (4.1), nous donnons quelques informations sur le nombre de données, le nombre d'attributs et le nombre de classes dans chaque jeu de données utilisé. Pour montrer la performance et l'efficacité de notre approche, nous avons utilisé des bases de données différentes en terme de taille et de nombre de classes. Pour avoir plus de détails sur la nature des données utilisées, nous renvoyons le lecteur à l'annexe A.

4.6.1 Évaluation de l'approche de clustering proposée

En premier lieu, nous examinons la capacité de la méthode de clustering proposée, avant l'apprentissage du PMC, à déterminer le nombre de clusters à l'aide de l'indice R^2. Les tableaux (4.2), (4.3), (4.4), (4.5), (4.6) et leurs figures associées montrent les premières itérations du processus de détermination du nombre de classes approprié ainsi que les valeurs des indices F et R^2 renvoyant la qualité de la partition.

Ces tableaux et courbes montrent que la méthode proposée a pu trouver le nombre exact de classes pour toutes les bases de données utilisées. En effet, les valeurs de l'indice R^2 correspondant au nombre exact de classes sont 0,1357 pour IRIS, 0,0796 pour wine, 0,1098 pour Soybean, 0,1078 pour Hayes-Roth et 0,1098 pour Glass. Ces valeurs sont déterminées

Table 4.2 – Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de IRIS

Nombre de classes	R^2	F
1	0.1194	9.8936
2	0.1315	5.4518
3	0.1362	3.7317
4	0.1357	2.7468
5	0.1381	2.2102
6	0.1390	1.8292
7	0.1399	1.5566
8	0.1420	1.3649
9	0.1424	1.1995
10	0.1426	1.0644
11	0.1430	0.9560
12	0.1431	0.8630

selon la règle de l'indice R^2 qui consiste à choisir le nombre de classes associé à la première différence assez petite. Ce résultat très intéressant montre l'efficacité de l'approche à trouver le nombre de classes, la chose qui pose un grand problème dans le domaine de la classification non supervisée.

4.6.2 Évaluation de l'hybridation proposée

Cette section évalue l'adaptation du PMC au problème de classification automatique basée sur la nouvelle méthode de clustering proposée.

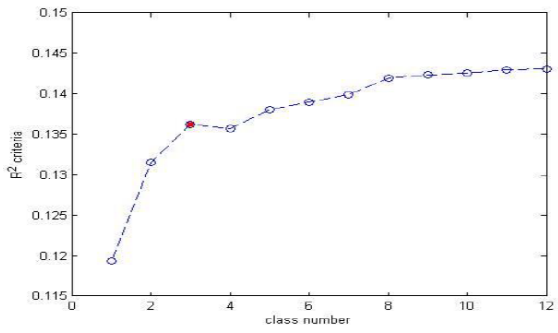

Figure 4.8 – Variation de R^2 en fonction du nombre de classes pour la base IRIS

Table 4.3 – Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de WINE

Nombre de classes	R^2	F
1	0.0521	4.0084
2	0.0742	2.8857
3	0.0796	2.0474
4	0.0773	1.4666
5	0.0787	1.1795
6	0.0798	0.9823
7	0.0797	0.8286

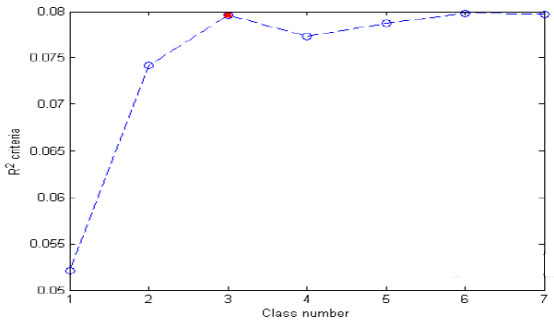

Figure 4.9 – Variation de R^2 en fonction du nombre de classes pour la base WINE

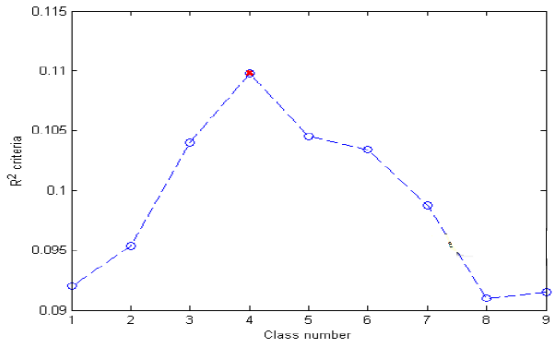

Figure 4.10 – Variation de R^2 en fonction du nombre de classes pour la base SOYBEAN

Table 4.4 – Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de SOYBEAN

Nombre de classes	R^2	F
1	0.0920	7.3948
2	0.0954	3.7946
3	0.1040	2.7458
4	0.1098	2.1579
5	0.1045	1.6112
6	0.1034	1.3077
7	0.0987	1.0482
8	0.0910	0.8260
9	0.0915	0.7278

Table 4.5 – Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de HAYES-ROTh

Nombre de classes	R^2	F
1	0.0865	5.9665
2	0.0999	3.9960
3	0.1978	2.8604
4	0.1094	2.1500
5	0.1148	1.7889
6	0.1066	1.3518
7	0.1067	1.1427

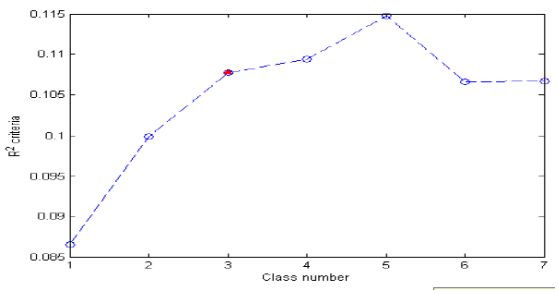

Figure 4.11 – Variation de R^2 en fonction du nombre de classes pour la base HAYES-ROTh

Table 4.6 – Résultats de variation des indices R^2 et F en fonction du nombre de classes pour les données de GLASS

Nombre de classes	R^2	F
1	0.1035	7.2694
2	0.1107	4.4805
3	0.1112	2.9623
4	0.1098	2.1574
5	0.1110	1.7233
6	0.1098	1.3972
7	0.1098	1.1809
8	0.1048	0.9655
9	0.1036	0.8350
10	0.1029	0.7343
11	0.1053	0.6741
12	0.1034	0.5957
13	0.0697	0.5022
14	0.0973	0.4618
15	0.0946	0.4109
16	0.0950	0.3804
17	0.0945	0.3499

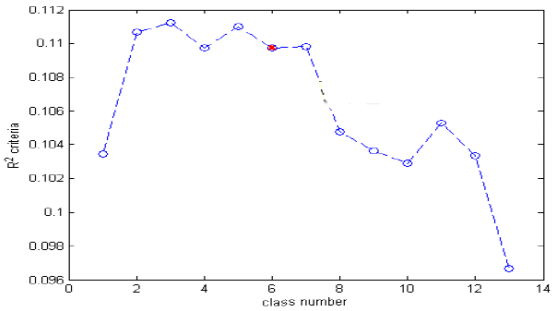

Figure 4.12 – Variation de R^2 en fonction du nombre de classes pour la base GLASS

Le tableau (4.7) présente le taux de reconnaissance des cinq bases de données obtenu par le PMC selon la variation de l'indice silhouette qui détermine la base d'apprentissage. Cet indice calcul le taux d'adaptation des données à leurs classes en indiquant si une donnée est très bien

classée ou non. Nous considérons cet indice comme un seuil de sélection des données pour l'apprentissage. La première colonne du tableau (4.7) décrit les bases de données utilisées ; la deuxième spécifie la variation du seuil de silhouette, alors que la troisième présente le taux de reconnaissance atteint pour chaque seuil.

En variant le seuil de sélection des données bien classées, on calcule le taux de clustering associé. Par exemple, pour un seuil de 0.6, de chaque classe appartenant à la meilleure partition obtenue par l'approche de clustering, nous sélectionnons les données ayant une silhouette supérieure ou égale à 0.6. Cette variation a pour but de sélectionner les meilleures données de chaque classe qui vont participer à l'apprentissage du PMC.

Table 4.7 – Résultats de classification des données selon la variation du seuil de similarité

Base de données	Seuil de Similarite	Taux de Classification
IRIS	0.65	82.66
	0.7	89.33
	0.8	89.33
	0.9	93.33
WINE	0.5	73.33
	0.6	74
	0.7	78.88
	0.8	78.88
	0.9	80
	0.94	80
SOYBEAN	0.4	34
	0.6	72
	0.7	84.5
	0.85	89.36
HAYES-ROTh	0.3	81
	0.4	81.30
	0.6	84
	0.7	89.23
	0.8	93.84
GLASS	0.4	79.43
	0.6	87.5
	0.8	91.2

En examinant les résultats du tableau (4.7), le choix du seuil a un grand impact sur la

qualité de la classification. L'augmentation du seuil donne une bonne classification par le réseau PMC. Notre approche atteint en général des taux de reconnaissance très satisfaisants pour toutes les bases de données en variant le seuil de silhouette, sauf pour la base de Soybean pour un seuil de 0.4.

Pour positionner notre approche par rapport aux méthodes de la littérature, nous effectuons des comparaisons avec des méthodes de classification différentes. Le tableau (4.8) illustre les résultats de comparaison effectuée entre notre approche et l'algorithme classique de k-means, les réseaux de neurones RBF et LVQ ainsi que l'approche VD. Cette dernière utilise le principe du diagramme de Voronoi pour surmonter le problème d'initialisation de k-means en connaissant le nombre de classes [179].

Table 4.8 – Comparaison de l'approche proposée avec des méthodes neuronales et des méthodes de clustering

Base de Données	Notre Approche	k-means	VD	LVQ	RBF
IRIS	93.33	85.33	88.67	92.55	85.64
WINE	81.46	70.78	90.44	68.9	67.87
SOYBEAN	89.36	76.6	82.98	–	–
HAYES-ROTh	93.4	–	–	52.26	70.03
GLASS	91.2	–	–	60.69	69.54

Le tableau (4.8) montre que l'approche proposée conduit à des résultats satisfaisants. Nous remarquons, d'après ce tableau, que les meilleurs résultats trouvés par notre approche sont supérieurs à ceux obtenus par les autres algorithmes (K-means, RBF, LVQ et VD) sur les bases de données IRIS, Glass et Hayes-Roth. A l'aide de notre approche, nous avons obtenu la meilleure classification pour les données de Soybean en comparaison avec l'algorithme de k-means et celui de VD. De même nous pouvons constater que notre approche donne également un résultat meilleur que celui de l'algorithme de k-means, RBF et LVQ sur la base de données Wine, tandis que l'approche VD dépasse notre approche. Il faut signaler que notre approche fonctionne d'une manière non supervisée sans avoir aucune information au préalable sur les données ou le nombre de clusters.

D'après les tableaux et les figures présentés dans cette section, nous remarquons que l'approche proposée fournie des résultats très satisfaisants et compétitifs à celle de la littérature. Par conséquent, l'approche proposée représente une nouvelle méthode avantageuse dans le domaine de la classification non supervisée et l'apprentissage.

4.7 Conclusion

Bien que la classification supervisée présente une tâche très importante, voir obligatoire, pour plusieurs domaines, la majorité des problèmes nécessitent la classification des données non étiquetées (clustering). Le PMC représente un bon classifieur universel qui appartient à la famille des RNAs. Notre contribution vise à bénéficier des avantages de ce classificateur universel pour le clustering. Cet objectif est réalisé en deux phases : la première consiste à étiqueter un ensemble d'apprentissage qui va servir dans la deuxième phase à l'apprentissage du PMC pour la généralisation du modèle. Étiqueter un ensemble de données sans avoir aucune information consiste à trouver le nombre de classes et affecter chaque donnée à la classe dont elle appartient ; cela revient à chercher la partition optimale de l'ensemble de données. L'algorithme de k-means est la méthode la plus connue qui a prouvé son succès en classification non supervisée. Cependant, le problème d'initialisation et du nombre de classes diminuent son efficacité. Pour cela, nous avons proposé une approche itérative utilisant l'algorithme de k-means comme méthode de recherche locale de la partition optimale à l'aide des critères d'évaluation de la classification. Le PMC adapté au clustering est expérimenté sur des bases de données Benchmarks de différents types et comparé à des méthodes de clustering classiques et autres de type RNAs. Les résultats fournis prouvent que notre approche réalise les objectifs envisagés et donne des résultats très satisfaisants et compétitifs à ceux de la littérature. Cependant, l'approche proposée peut être considérée comme un package d'un algorithme de clustering sans paramètres grâce à la détection automatique du nombre de classes et l'ensemble d'apprentissage. Ces résultats nous encouragent à appliquer cette approche sur un PMC d'architecture optimale et ses variantes, et à traiter les problèmes de clustering dans des domaines plus complexes comme le big data.

Comme la compression de la parole, la réduction de dimensionnalité des données est devenue une nécessité pour réduire le temps de calcul, et améliorer la qualité du traitement de données. Dans le chapitre suivant, nous présentons une nouvelle méthode s'appuyant sur l'algorithme de k-means pour la sélection des caractéristiques pertinentes des données.

Chapitre 5

Nouvelle méthode de clustering pour la Sélection de caractéristiques pertinentes

5.1 Introduction

L'énorme quantité des services de Web et le nombre croissant d'informations, des articles et des produits misent en ligne, la quantité de données est en pleine explosion dans le monde entier. En outre, cette énorme quantité de données est qualifiée de données de grande dimension. L'analyse de grands ensembles de données est un problème urgent d'une grande importance pratique. Précisément, les principales préoccupations sont dirigées vers la réduction de la haute dimensionnalité de l'espace des caractéristiques à cause de la complexité de calcul et des considérations de précision. Par conséquent, une variété de méthodes ont été initialement misent en place dans la littérature pour sélectionner un nombre optimal de caractéristiques.

Généralement, la réduction de dimension et la classification partagent certaines objectifs comme : l'analyse de la structure des données, extraction des informations pertinentes et la découverte de la structure intrinsèque des données. Dans ce chapitre, nous proposons une méthode originale pour la réduction de la dimensionnalité basée sur la classification non supervisée des caractéristiques des données pour extraire la meilleure partition. Nous nous attendons à ce que la méthode de clustering, appliqué aux caractéristiques des données, extrait des informations pertinentes. Grâce à la nouvelle technique d'initialisation des paramètres de l'algorithme de k-means présentée dans ce chapitre, nous tentons de générer une nouvelle méthode de clustering surmontant les faiblesses de cet algorithme afin d'avoir des résultats pertinents pour la classification des caractéristiques.

Nous commençons la présentation de ce chapitre par une introduction sur la thématique

de réduction de la dimension suivie de deux sections contenant un survol des méthodes d'extraction de caractéristiques et de sélection de variables. Dans la quatrième section, nous présentons notre nouvelle méthode de réduction. Nos résultats seront examinées dans la dernière section de ce chapitre à l'aide des tests expérimentaux via trois bases de données benchmarks.

Ce chapitre a fait l'objet d'une communication orale présentée dans le congrès international : «The 3nd IEEE International Conference on Logistics Opeartions Management - GOL'2016» et publiée dans IEEE Xplore [84].

5.2 Réduction de la dimension

Bien que le stockage de l'information ne pose plus de problème aujourd'hui, la quantité de données ainsi que leurs dimensions sont en perpétuelle augmentation. Cela montre que les variables qui génèrent ces données peuvent être très nombreuses et peuvent fournir la même information (variables redondantes), la chose qui nous pousse à trouver l'information utile et rejeter le redondant. Dans certains cas, l'information de certaines caractéristiques des données peut être déduite grâce à d'autres variables. En plus du risque fort de la présence de données bruitées, ces données ne font qu'alourdir les calculs et limiter la performance des analyseurs automatiques [94]. Dans ce contexte, des analyses théoriques et des études expérimentales ont montré la faiblesse de nombreux algorithmes et techniques statistiques en présence de variables non pertinentes ou redondantes [140][95].

Au-delà des aspects de diminution de stockage et de complexité de calcul, la réduction de dimensionnalité (RD) vise à faciliter la visualisation et la compréhension des données, réduire le temps d'apprentissage et d'utilisation, identifier les facteurs pertinents et plus précisément améliorer les performances de la classification.

Les formes représentées dans de grandes dimensions engendrent un problème connu sous le nom du phénomène de l'espace vide. Les travaux [197][144] le caractérisent en énonçant quelques propriétés inattendues dans des espaces de grandes dimensions et apparait contre l'intuition humaine. Il concerne la variation du volume de la boule unité en fonction de la dimension de l'espace [36]. Ce volume est donné par la relation suivante :

$$V(p) = \frac{\pi^{\frac{p}{2}}}{\Gamma(\frac{p}{2}+1)} \tag{5.1}$$

p représente la dimension des données et Γ la fonction gamma usuelle. On déduit de cette formule que le volume d'une sphère inscrit dans un cube tend vers zéro quand la dimension augmente. En outre, lorsque la dimension est très grande, la plupart des données sont situées près de la couche périphérique et les distances entre elles deviennent les mêmes, d'où la difficulté d'évaluer la ressemblance entre ces données, par suite leur traitement devient délicat.

Ce phénomène est introduit aussi par Bellman [24] sous l'expression : malédiction de

la dimensionnalité, révélant le problème de la grande dimension surtout quand le nombre d'observations est limité. Il spécifiait que, pour une estimation précise d'une distribution de données de grande dimensionnalité, le nombre d'échantillons devrait être important. Puisque le nombre d'observations est souvent très limité et il est parfois impossible de pouvoir en obtenir autant que nécessaire, ce résultat constitue un obstacle majeur dans l'apprentissage artificiel qui est souvent contraint à manipuler des ensembles d'observations réduites.

Pour diminuer l'influence de ces difficultés, les chercheurs utilisent la réduction de la dimensionnalité comme un moyen incontournable dans le processus de reconnaissance de formes pour conditionner le processus de classification. Ainsi, deux approches, illustrées dans les figures (5.1) et (5.2), sont principalement utilisées pour réduire la dimension :

— la sélection de variables [186], qui consiste à réduire la dimensionnalité de l'espace d'entrée en éliminant les variables non pertinentes de l'espace d'entrée. En d'autre termes, cette technique vise à former un sous-ensemble de variables en ne préservant que l'information utile.

— l'extraction de caractéristiques [207][38], qui utilise toutes les variables initiales et applique une transformation sur ces variables, afin d'obtenir une nouvelle représentation plus synthétique.

Ces deux catégories ainsi que les méthodes associées à chacune seront détaillées avec des descriptions dans les sections suivantes.

5.3 Extraction de caractéristiques

Ce type d'approches part du principe que l'information, contenue dans un grand nombre de variables, peut être représentée par un nombre plus faible de caractéristiques non visibles directement. La construction de ces nouvelles caractéristiques (figure (5.1)) est fondée sur une transformation des variables originales qui peut être linéaire ou non linéaire. Les méthodes utilisées pour l'extraction de caractéristiques sont très variées.

Nous rappelons ici brièvement les principes des méthodes linéaires et non linéaires qui ont fait l'objet de nombreuses études.

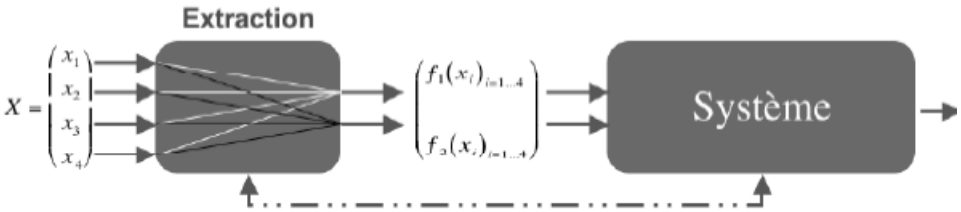

Figure 5.1 – Processus général de la technique d'extraction de caractéristiques (reprise de [106])

5.3.1 Méthodes linéaires

Les méthodes linéaires traitent les données linéairement séparables. Parmi ces méthodes, l'analyse en composantes principales (ACP) [126] et la méthode de positionnement multidimensionnel (Multi-dimensional Scaling) MDS [189] sont les plus utilisées.

L'ACP est définie par hotelling [112] comme une projection orthogonale des observations dans un espace de faible dimension préservant au mieux la distribution des observations et en utilisant les vecteurs propres de la matrice associée à ces données. Cette méthode veille à ce que la variance des observations projetées soit maximisée. La technique de projection permet d'apporter une réponse aux problèmes des variables redondantes telle que les caractéristiques de la nouvelle représentation ne sont donc pas corrélées. De nombreux ouvrages décrivent précisément cette analyse et quelques extensions, comme [126][188][32].

L'objectif de la deuxième méthode MDS [200][189][33] reste identique à l'ACP, mais sa façon de présentation des données est différente. Cette technique consiste aussi à trouver une projection dans un espace de faible dimension mais en préservant au mieux les distances entre chaque paire d'observations au lieu de leurs distribution. La méthode MDS utilise une matrice contenant les distances, similarités ou dis-similarités entre les observations pour reconstituer une carte des observations en cherchant une représentation dans un espace euclidien.

La facilité d'implémentation est une qualité qui caractérise ces méthodes de projection linéaires et les rend néanmoins très populaires, la chose qui justifie leur large utilisation. Cependant, comme indiqué auparavant, ces méthodes ne peuvent pas détecter des structures ou des relations non linéaires présentes dans les données, ce qui oblige, selon les applications, à utiliser d'autres approches appelées non linéaires.

5.3.2 Méthodes non linéaires

Une première voie consiste à adapter les méthodes linéaires au cas non linéaire. Nous trouvons la méthode ACP-kernilisée comme exemple de celles-ci. Elle utilise au lieu du produit scalaire classique, un autre défini à partir d'une fonction noyau. Ainsi, dans le nouvel espace, l'ACP est effectuée en utilisant les corrélations d'ordre supérieur. Une autre variante consiste à appliquer localement l'analyse en composantes principales (c'est-à-dire dans un voisinage de chaque point). Il existe aussi des méthodes qui consistent à plonger les données dans une variété non linéaire géodésique, puis à effectuer la méthode MDS en utilisant la distance géodésique [144].

Généralement, deux approches non linéaires permettent de réaliser une extraction de caractéristiques : les approches algébriques et les approches neuronales.

5.3.2.1 Méthodes algébriques

On trouve plusieurs méthodes de réduction non linéaire basées sur des aspects géométriques. Parmi celles-ci on trouve : Locally Linear Embedding (LLE) [184] , isometric feature mapping (isomap) [196], nonlinear mapping (NLM) [187].

La méthode LLE tente à projeter les observations des entrées dans un espace de plus faible dimension, en considérant que les observations, globalement non linéaires, sont localement linéaires, amenant ainsi à conserver les configurations locales. Dans le cas de la méthode isomap, les auteurs utilisent la méthode MDS en définissant la matrice de dis-similarité entre les observations en termes de distance géodésique [144]. Cette distance représente le plus court chemin entre deux observation en passant par d'autres observations. Ce chemin est obtenu à l'aide d'un graphe liant chaque observation à ses k plus proches voisins.

5.3.2.2 Méthodes neuronales

Les approches neuronales représentent un autre type de méthodes de réduction de la dimensionnalité. Parmi ce type de méthodes, les réseaux de Kohonen, que nous avons évoqués dans le premier chapitre, sont les plus utilisés. L'algorithme d'apprentissage de ce réseau vise à effectuer une quantification vectorielle de l'espace d'entrée (partitionnement en plusieurs clusters) et une projection non linéaire des observations originales dans un espace discret de très faible dimension (carte ou grille). Comme nous avons défini dans le premier chapitre, la taille de la grille est prédéfinie au début de l'algorithme (généralement rectangulaire ou hexagonale). A travers l'apprentissage, cette grille doit aboutir à une représentation discrète de l'espace d'entrée où chaque neurone appartenant à cette grille (l'espace de projection) est lié à l'espace des observations par un vecteur référent. Pendant l'apprentissage, ces vecteurs s'adaptent à la distribution des observations, en conservant la topologie de la carte. La conservation de la notion de topologie joue un rôle important à garder la notion de proximité entre les neurones. Ainsi, deux neurones proches sur la grille doivent avoir leurs vecteurs référents proches dans l'espace des observations.
Autres méthodes ont été proposées pour améliorer l'algorithme SOM et s'affranchir de quelques faiblesses. Parmi ces méthodes on trouve, generative topographic mapping (GTM) et curvilinear component analysis (CCA).

5.4 Sélection de variables

Nous avons déjà montré la nécessité de la réduction de la dimensionnalité d'un problème en évoquant les problèmes causés par des variables redondantes qui sont repérables par des corrélations élevées. Ces variables peuvent être considérées comme superflues, dans le sens qu'aucune information supplémentaire n'apparaît en les ajoutant. En revanche, autres variables contribuent fortement dans l'information globale portée par les données. Par suite, l'élimination des variables non pertinentes n'influence pas significativement l'ensemble d'information. Dés lors, l'objectif de la sélection de variables est de choisir, parmi toutes les variables originales, un sous ensemble de variables pertinentes (figure (5.2)). En réduisant

le nombre de variables à recueillir, ces méthodes réduisent implicitement l'espace d'apprentissage. La recherche ou l'identification des variables pertinentes est alors une tâche très importante et complexe. Cette complexité découle du fait qu'elle requiert deux éléments principaux :

— Le critère d'évaluation, qui doit permettre d'estimer si un sous-ensemble de variables est meilleur qu'un autre.
— la procédure de recherche, qui permet de chercher les sous-ensembles candidats de variables.

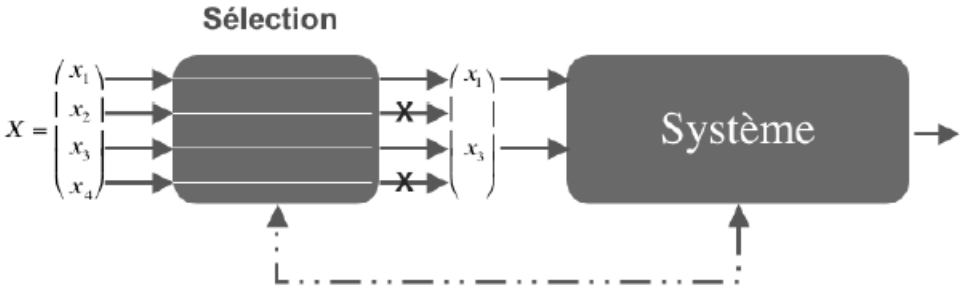

Figure 5.2 – Principe général de la technique de sélection de variables (reprise de [106])

À l'instar de l'extraction de caractéristiques, il existe plusieurs méthodes de sélection de variables. Ces méthodes peuvent être catégorisées comme suit :

5.4.1 Méthodes à base d'apprentissage connexionniste

Ces méthodes reposent sur le concept d'apprentissage pour converger à un minimum local. Le processus de ces méthodes consiste à associer, à chaque variable, une valeur qui dépend des paramètres et de la structure du réseau pour mesurer la pertinence de la variable. A la fin de chaque itération de l'apprentissage, les variables sont ordonnées selon leur pertinence dans le but de supprimer les moins pertinentes. Par suite, la prochaine itération est effectuée avec moins de variables.

5.4.2 Méthodes à base de modèles d'optimisation

Ce type de méthodes consistent à modéliser le problème de sélection de variables pertinentes sous forme d'un modèle mathématique.
Nous adoptons la notation suivante, S est l'ensemble des données à traiter et V est celui des variables qui génèrent les composantes de ces données. On note $P(V)$ l'ensemble des sous parties de V. la partie A de V représente le sous ensemble de variables pertinentes cherché. L'ensemble A est choisi de telle manière à réaliser un gain considérable qui ne doit pas différer significativement de celui réalisé par l'ensemble de toutes les variables.

En définissant une fonction f qui associe à chaque partie de V le gain et la corrélation entre ses éléments, la tâche de sélection peut être modélisée en termes d'un problème

d'optimisation [94] sous forme du modèle suivant :

$$\begin{cases} minf(x) \\ sc. \\ g_i(x) \leqslant 0, i = 1, ..., n \\ x \in D \\ D \subset \mathbb{R}^n \end{cases} \tag{5.2}$$

Nous pouvons citer quelques contraintes de ce modèle comme : exiger que le cardinal de la partie A ne doit pas dépasser un certain seuil ou que la corrélation entre les variables de cette partie soit minimale.

5.4.3 Méthodes à base de classification

La classification est un moyen qui est très utilisé dans la sélection de variables pertinentes. Son utilisation est soit pour évaluer l'importance des variables soit pour classifier les variables dans certains groupes. Ces derniers sont ensuite exploités pour extraire les variables les plus importantes [170].

5.5 Approche proposée pour la réduction de la dimensionnalité

Les contributions que nous avons proposées dans les chapitres précédents ont justifié l'importance de l'étude itérative de la structure des données au lieu de l'utilisation d'un algorithme classique de clustering. Elle permet dans le cas de l'algorithme K-moyennes et du PrSOM de réaliser les objectifs suivants :

— Diluer le problème d'initialisation des paramètres qui influence négativement la qualité de la classification.
— Déterminer du nombre optimal de classes.

Dans cette section, nous allons présenter en détail les différentes étapes de la méthode proposée pour la réduction de la dimensionnalité. Après avoir donné la structure générale de la méthode de réduction proposée, nous introduisons le nouvel algorithme de clustering qui vise à proposer une nouvelle amélioration de l'algorithme k-means [103] et par suite nous exposons la technique générale qui exploite l'algorithme de clustering pour la sélection des caractéristiques pertinentes.

Les méthodes de clustering ont l'objectif de regrouper les données partageant plus ou moins la même information. En se basant sur cette caractéristique, nous proposons dans ce travail de regrouper les caractéristiques des données (figure (5.3)) à l'aide d'une nouvelle méthode de clustering visant à surmonter les faiblesses de l'algorithme k-means.
A l'aide de cette méthode, nous visons à :

— Détecter l'information redondante

— Représenter chaque cluster de caractéristiques par l'élément le plus représentatif du cluster

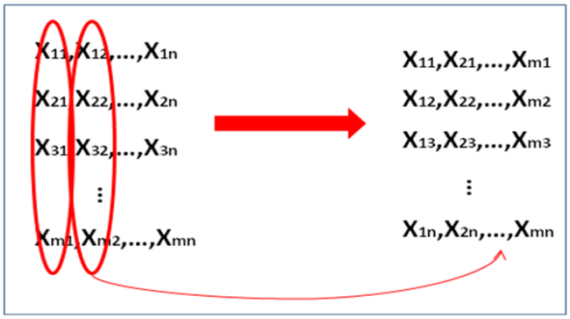

Figure 5.3 – Technique de construction de la nouvelle base de données

Dans les paragraphes suivantes, nous décrivons le processus des étapes de la méthode clustering.

5.5.1 Approche de clustering

La classification non supervisée (clustering) est un outil important pour la réduction de la dimensionnalité qui devient une tâche très importante de nos jours. Intéressés par ce sujet, nous proposons une méthode originale de clustering qui vise à tirer profit de l'algorithme de k-moyennes et surmonter ses faiblesses. Être l'algorithme le plus simple, le plus rapide et efficace pour le traitement de grandes quantités de données, n'empêche pas que la performence de k-means est influencée par le choix des paramètres initiaux : le nombre k et les centres initiaux. Comme nous avons expliqué dans le chapitre 3, le mauvais choix des paramètres initiaux peut bloqué l'algorithme de k-means sur un minimum local. La détermination de ces paramètres est une tâche difficile qui a été traitée par de nombreux chercheurs [43]. A notre tour, nous avons proposé deux travaux [76][101] qui s'intéressent à cette problématique.

Dans notre travail, nous visons à surmonter le problème d'initialisation de l'algorithme de k-means en proposant une nouvelle méthode capable de classifier un ensemble de données sans avoir aucune connaissance préalable sur la structure de données ou le nombre de classes.

La méthode représente un processus itératif qui génère plusieurs partitions en utilisant l'algorithme de k-means comme un algorithme de recherche local dans chaque itération. Celle ci comporte trois phases principales : phase d'initialisation, phase d'optimisation et phase d'évaluation. Dans la première, nous proposons une technique d'initialisation qui vise à chercher les centres de k-means et le nombre k. Pour se faire, la technique se base sur un paramètre bien déterminé qui contrôle la similarité entre les données. Le résultat de la phase d'initialisation, une fois trouvé, est utilisé par k-means dans la deuxième phase pour lancer le processus de recherche de la partition optimale de l'itération courante. Quant à la phase d'évaluation, nous visons à évaluer la qualité de la partition trouvée par k-means à l'aide d'un indice d'évaluation. Le processus général de la méthode proposée consiste à répété ces trois étapes jusqu'à satisfaire la condition d'arrêt que nous fixons au début du

processus de la méthode.

De plus amples détails sur chaque étape de la méthode de clustering seront exposés dans les paragraphes suivants.

5.5.1.1 Phase d'initialisation

Dans cette phase, nous cherchons le nombre k et les centres initiaux. Pour se faire, nous déterminons en premier lieu un intervalle de variation d'un paramètre qu'on nome ϵ. Ce paramètre varie entre la plus petite distance, nommée a telle que $a = min_{x,y \in D} d(x, y)$, et la plus grande distance, nommée b telle que $b = max_{x,y \in D} d(x, y)$. Un schéma illustratif de la technique de détermination de a et b est présenté dans la figure (5.4).

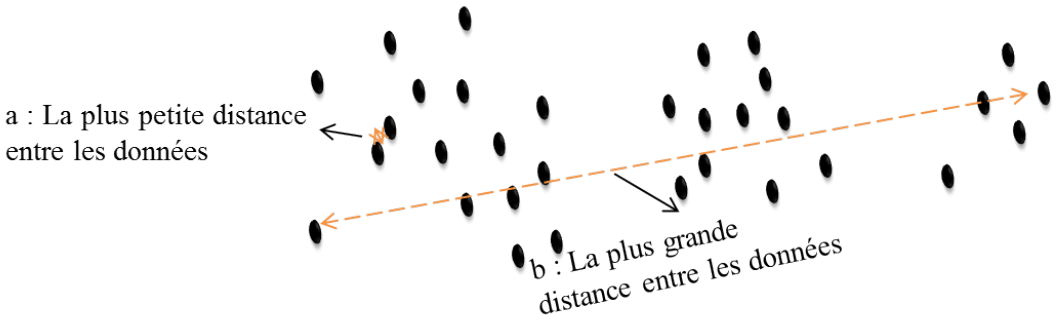

Figure 5.4 – Technique de détermination de l'interval de variation de ϵ

C'est difficile de déterminer exactement à quelle valeur de similarité on dit que deux données sont assez similaires pour appartenir à la même partition. Pour cette raison, nous avons introduit le paramètre ϵ dans la méthode. En variant ce paramètre dans un intervalle bien déterminé, ϵ joue un rôle important pour déterminer le meilleur taux de similarité entre données qui donnera la meilleure partition.

Pour chaque x de l'ensemble D de données, nous générons l'ensemble E_x suivant la règle :

$$E_x = \{y \in D / d(x, y) < \epsilon\} \tag{5.3}$$

Ces groupes représentent des boules de rayon ϵ. Le groupe le plus homogène E_h est sélectionné et puis retiré de l'ensemble de données. Cette étape représente la première itération de la phase d'initialisation.

Dans la deuxième itération, le même processus est répété pour chaque élément du nouveau ensemble de données $X = D / E_h$. Les groupes E_x sont générés selon l'équation (5.3) et le plus homogène est sélectionné et puis retiré de l'ensemble X pour que la prochaine itération s'applique aux éléments restants (X / E_h). Cette technique est répétée jusqu'à ce que l'ensemble X soit vide. De cette manière, nous sélectionnons les groupes compacts (les plus homogènes) de l'ensemble de données pour extraire leurs représentants.

5.5.1.2 Phase d'optimisation

Le nombre des groupes E_h, sélectionnés dans la phase précédente, ainsi que leurs centres représentent respectivement le nombre k et les centres initiaux pour l'algorithme de k-means. Une fois initialisé, cet algorithme se charge de trouver la partition optimale de l'itération courante. Pour cela, l'algorithme de k-means représente, à chaque itération, un algorithme de recherche local pour trouver la partition optimale de l'itération courante et qui sera par la suite comparée aux autres partitions.

5.5.1.3 Phase d'évaluation

La méthode proposée consiste à générer un ensemble de partitions et sélectionner la meilleure. Cela pose le grand problème de choix de la partition qui consiste à évaluer chaque partition pour déterminer celle qui donne le meilleur résultat. Dans ce travail, nous utilisons l'indice de silhouette pour sélectionner la meilleure partition en évaluant la qualité de chaque partition dans chaque itération.

A la fin de ces trois étapes, le paramètre ϵ est incrémenté pour entamer une nouvelle itération en augmentant le seuil de similarité entre donnée ; ϵ. Le schéma général décrivant la méthode de clustering est montré dans la figure (5.5).

Après avoir trouvé la meilleure partition de l'ensemble des caractéristiques à l'aide de la méthode de clustering proposée, l'étape de réduction de la dimensionnalité est entamée.

5.5.2 Étape de réduction de la dimension

En se basant sur le fait que les caractéristiques situées dans le même cluster partagent la même information, nous choisissons l'objet représentant (ou centre) de chaque cluster, de la partition optimale trouvée dans l'étape précédente, pour représenter ces voisins. Les centres des classes représentent un sous ensemble S de l'ensemble des caractéristiques des données. Les caractéristiques n'appartenant pas à S seront éliminer de l'ensemble de données. De cette manière, nous remédions au problème de la redondance des caractéristiques en supprimant les caractéristiques qui partagent plus ou moins la même information.

Les idées principales du processus proposé pour la réduction de la dimensionnalité peuvent être résumés dans la figure (5.6) ainsi que les points suivants :

— Les objets classés dans la même classe partagent plus ou moins la même information
— Classifier l'ensemble des caractéristiques, sans avoir aucune information, pour ne sélectionner que l'information pertinente
— La tâche de clustering est effectué grâce à la méthode qui surmonte le problème d'initialisation de k-means

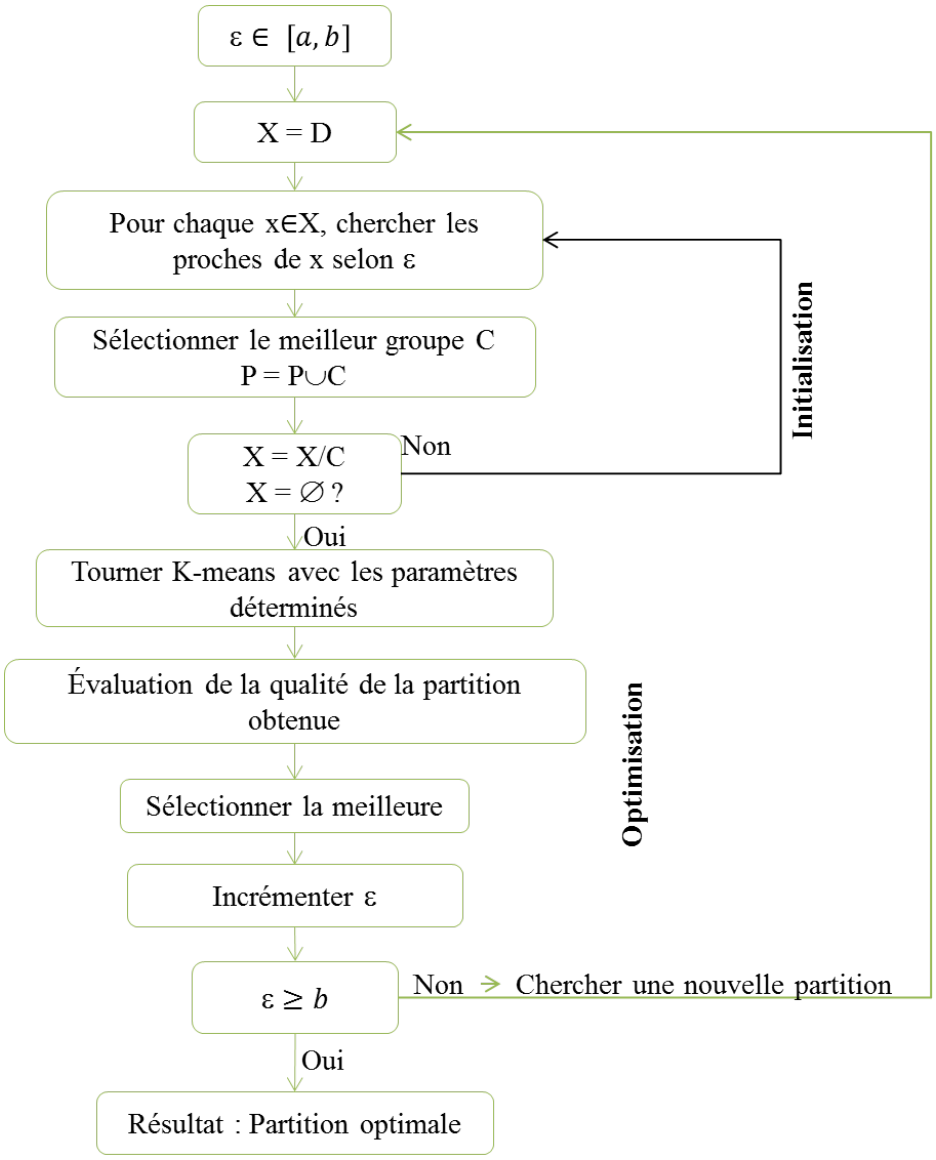

Figure 5.5 – Schéma général de la méthode de clustering proposée

5.6 Validation de la méthode proposée

Dans cette section, nous visons à évaluer l'efficacité et la performance de la méthode proposée pour la réduction de la dimentionnalité. Pour se faire, nous commençons par une description des trois bases de données disponibles sur UCI Machine Learning Repository (tableau (5.1)) utilisées pour la validation de la méthode de réduction proposée. Ces bases de données contiennent différents nombres de caractéristiques, d'instances et de classes. Nous présentons par la suite les résultats des simulations pour estimer le nombre de caractéristiques pertinentes pour chaque base de données. C'est dans cette veine que deux expériences sont présentées dans deux tableaux différents afin de :

— Évaluer la qualité de la classification des caractéristiques par la nouvelle méthode de

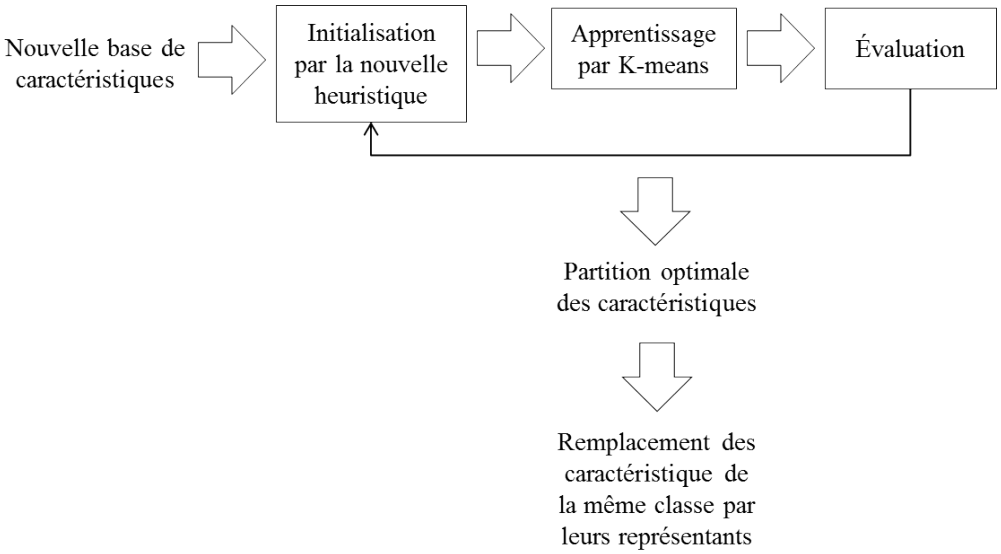

Figure 5.6 – Illustration schématique de la méthode proposée

clustering
— Évaluer l'optimalité du nombre de clusters trouvé
— Évaluer la capacité de l'approche de réduction à fournir de nouvelles variables réduites

Table 5.1 – Description de quelques information sur les bases de données Benchmarks utilisées

Bases de données	**Nbr de caractéristiques**	**Nbr d'instances**	**Nbr de classes**
Soybean	35	47	4
Ionosphere	34	351	2
Wine	13	178	3

La première expérience est effectuée dans le but de tester si le remplacement des caractéristiques classifiées dans le même cluster par leurs représentants va nous couter une perte d'information ou élimination de la redondance avec la préservation de l'information pertinente. Le tableau (5.2) stocke les résultats de clustering (nombre de classes et la variation de l'indice R) obtenus par l'algorithme proposé avant et après la réduction de la dimensionnalité.

En examinant le tableau (5.2), la méthode proposée réussit à éliminer les informations non pertinentes et bruitées. Nous avons réussi à trouver le nombre exacte de classes et améliorer le taux de classification après l'élimination de l'information bruité. En effet, pour les données de Soybean, la méthode parvient à éliminer les données bruitées qui empêchent

Table 5.2 – Résultats de clustering avant et après la réduction de la dimensionnalité

Bases de données	**Avant réduction**		**Après réduction**	
	Taux de reconnaissance	**Nbr de clusters**	**Taux de reconnaissance**	**Nbr de clusters**
Soybean	78.72	3	78.72	4
Ionosphere	71.22	2	76.63	2
Wine	65.73	2	65.73	2

la méthode de clustering à déterminer le nombre exact de clusters avant la réduction. Quant aux données d'Ionosphere, le nombre exact de clusters est déterminé avant et après la réduction de la dimensionnalité en plus de l'amélioration significative (5%) dans le taux de reconnaissance après la réduction. Cependant, notre méthode ne parvient pas à reconnaitre les trois clusters de Wine.

Par conséquent, d'après les résultats du tableau (5.2), nous pouvons bien évidemment voir la capacité de la méthode proposée à contribuer à la bonne réduction des données qui participe à l'amélioration de la classification. Ces résultats prouvent que l'élimination de la redondance et du bruit améliore la qualité du clustering soit pour le nombre de clusters ou le taux de reconnaissance. Cela confirme à nouveau la nécessité de réduction de dimension pour le pré-traitement des données.

Le deuxième tableau complète les résultats du premier en présentant le nombre de données réduites ainsi que le taux de réduction.

Table 5.3 – Résultats de réduction obtenus par l'approche proposée

Bases de données	**Nombre de caractéristiques**	**Taux de réduction**
Soybean	11	68.5
Ionosphere	2	94.11
Wine	2	81.8

A partir des résultats listés dans le tableau (5.3), nous constatons que le taux de réduction est élevé pour les données de Ionosphere et Wine, tandis que pour les données de Soybean, le taux est un peu réduit.

5.7 Conclusion

La réduction de la dimensionnalité des données est essentielle dans plusieurs domaines de l'apprentissage automatique et l'analyse des données. Dans ce chapitre, nous avons proposé pour ceci une nouvelle méthode pour réduire la taille des données. Le processus de la méthode proposée comprend trois tâches principales : (1) transposition des données, (2) le regroupement des caractéristiques de données et (3) la suppression des informations redondantes. En premier lieu, pour avoir un nombre optimal ou même adéquat de caractéristiques ainsi qu'une bonne partition sans avoir aucune information à priori, nous avons proposé la méthode de clustering qui tire sa force de son indépendance des paramètres initiaux qui influencent négativement le résultats de la plupart des méthodes de clustering proposées dans la littérature. En deuxième lieu, nous nous sommes basés sur le fait que les données classifiées dans la même classe partagent plus ou moins la même information afin de ne garder qu'un représentant de chaque classe et par suite éliminer la redondance. Grâce au processus de classification non supervisée de la méthode proposée, indépendamment de toute information préalable ou paramètre initial, notre contribution pour la réduction de la dimensionnalité a démontré, à l'aide des résultats expérimentaux, sa force et son efficacité.

Encourager par ces résultats, une direction possible pour les travaux futures est d'investir la méthode de réduction dans les systèmes de recommandation qui représentent un problème complexe de données volumineuses en raison de l'énorme quantité de données et de leur structure difficile.

Arrivons à la fin de cette thèse, nous présentons dans ce qui suit la conclusion générale pour la récapitulation des différents travaux présentés précédemment.

Conclusion Générale

A cause des avancées des outils informatiques, le volume d'information disponible est devenu si important. Par conséquent, l'accès à l'information pertinente, le traitement et le transfert des données dans un temps raisonnable sont parmi les grands problèmes auxquels on se confronte dans le domaine de la recherche d'information. Dès lors, les recherches dans le domaine de l'apprentissage artificiel se sont fortement développées en s'intéressant à la classification comme outil essentiel dans le traitement de l'information. Les travaux présentés dans ce manuscrit de thèse s'inscrivent dans ce contexte, où nous avons étudié des algorithmes d'apprentissage artificiel dans le cadre de la classification supervisée et non supervisée pour des applications dans des domaines réels.

Dans cette dernière partie, nous récapitulons les points clés détaillés dans ce mémoire, ainsi que les principales remarques évoquées lors des travaux réalisés pour résoudre la problématique posée. Pour conclure, nous donnons quelques directions pour de futures recherches.

Résumé

Dans le cadre de cette thèse, nous avons essayé d'apporter des solutions à la double problématique de sélection de paramètres et du nombre de classes pour les méthodes d'apprentissage automatique déterministes et probabilistes. Plus précisément, nos études se sont intéressées à déterminer les paramètres initiaux pour la méthode de k-means, l'architecture initiale pour les réseaux connexionistes PMC et PrSOM ainsi que le nombre de classes et la sélection des données d'apprentissage misent en œuvre pour la classification. Les méthodes proposées ont été présentées afin de réaliser un système complet de discrimination qui ne dépend pas des paramètres initiaux.

Le déroulement des étapes de notre travail et les résultats obtenus sont les suivants :

Dans le premier chapitre, nous avons commencé par définir le cadre général et les modes

de l'apprentissage artificiel. Nous avons introduit par la suite les deux axes principaux de ce mémoire : Classification et apprentissage artificiel. Un panorama servant à la description du processus de base d'un ensemble de méthodes de classification probabilistes neuronales et de partitionnement a été présenté. A l'image de ce premier chapitre, nous avons aussi signalé les problématiques de la détermination du nombre de classes, des paramètres initiaux et de la sélection d'architecture pour les RNAs.

Après avoir posé les problématiques liées aux méthodes d'apprentissage, nous avons détaillé chaque problématique pour proposer des solutions dans les quatre chapitres suivants qui illustrent nos contributions.

Le deuxième chapitre a proposé une solution pour le problème de sélection d'architecture de la carte topologique probabiliste. La nouvelle méthode a été utilisée pour la compression du signal des chiffres arabes. Restons dans le même cadre de sélection d'architectures neuronales, le troisième chapitre a présenté une nouvelle modélisation mathématique résolue par les AGs pour la sélection de l'architecture optimale du PMC. Le quatrième chapitre a traité le problème de sélection de paramètres pour l'algorithme de k-means ainsi que la sélection des données d'apprentissage pour le PMC. La nouvelle méthode proposée a été utilisée en amont pour le PMC afin d'adapter ce dernier à la classification automatique. Le dernier chapitre était consacré à la présentation d'une nouvelle méthode pour la sélection des paramètres de l'algorithme de k-means. Cette méthode a été appliquée dans le domaine de la réduction de dimensionnalité des données.

En s'intéressant à la problématique de sélection de paramètres et des architectures neuronales dont souffrent la majorité des méthodes de classification, nous avons pu traiter différentes thématiques : classification, clustering, parole et réduction de dimensionnalité.

Principales contributions

Dans ce manuscrit de thèse, nous avons présenté quatre contributions qui visent à sélectionner les paramètres des méthodes d'apprentissage artificiel (PrSOM, PMC, k-means) ainsi que l'application à des domaines réels (classification, compression de la parole, réduction de dimensionalité). Les apports originaux engagés sur notre problématique peuvent se répartir comme suit :

- Optimisation de l'architecture de la carte probabiliste PrSOM par une nouvelle méthode d'optimisation globale, H-PrSOM, permettant la sélection du nombre adéquat des neurones de la carte probabiliste ainsi que leurs poids et variances associés (chapitre 2).
- Présentation d'un outil générique capable de générer un dictionnaire optimal pour la compression de la parole (chapitre 2).
- Nouvelle modélisation mathématique du problème de sélection d'architecture neuronale du PMC avec la résolution par la méta-heuristique d'optimisation AG (chapitre 3).
- Enrichissement de la gamme de méthodes de clustering par la méthode H-kmeans et Hb-kmeans (chapitre 4 et 5).

— Nouvelle hybridation alliant les points forts des k-means et des réseaux multi-couches au profit de la classification automatique (chapitre 4).
— Sélection des paramètres de k-means, des données d'apprentissage du PMC, du nombre de classes et des caractéristiques pertinentes des données (chapitre 4 et 5).
— Réduction de la dimensionnalité à travers la classification des caractéristiques de données (chapitre 5).
— Toutes les approches proposées permettent de sélectionner les paramètres d'une manière automatique sans avoir aucune information à priori sur les données.

Ces apports originaux montrent la force et la richesse des thématiques et domaines traitées dans ce travail de thèse.

Perspectives de recherche

Durant nos années de recherche, nous avons fixé comme vision la sélection des paramètres des méthodes d'apprentissage artificiel dédiées à la classification. Suite aux apports originaux réalisés dans cette thèse, nous voyons, pour le moment, quelques directions dans lesquelles ce travail pourrait être poursuivi :

— Vue la diversité des méthodes traitées et des problématiques résolus, plusieurs comparaisons et collaborations entre ces méthodes peuvent être élaborées.
— La recherche d'une méthode d'extraction de représentants des classes. Cela va permettre d'améliorer la qualité de données représentatives des classes.
— La recherche d'autres critères plus pertinents pour l'évaluation des partitions.
— Continuité de notre travail pour les systèmes de recommandation vue qu'il demande la plupart des apports trouvés dans notre parcours de recherche.

Annexes

Description des Instances utilisées pour la validation des méthodes proposées

Pour mieux illustrer les ensembles de données utilisés dans le présent document, nous décrivons brièvement chacun dans les paragraphes suivants.

IRIS

Nous utilisons les fameux échantillons Iris extraites de [1]. Chaque échantillon de données IRIS comporte quatre attributs, à savoir ; longueur sépales en cm, la largeur des sépales en cm, longueur de pétale en cm, la largeur des pétales en cm. L'ensemble de données Iris se compose de trois classes cibles représentant les trois types de plantes Iris à savoir, Iris Setosa, Iris versicolor et Iris Virginica.

GLASS

L'ensemble de données GLASS avec 214 échantillons est choisi aussi de [1]. L'ensemble de données contient six types qui sont définis en termes de leur oxyde. Les échantillons contiennent neuf caractéristiques d'entrée, à savoir ; numéro d'identification, RI : indice de réfraction, Na : sodium (unité de mesure : pour cent en poids de l'oxyde correspondant, sont des attributs 4-10), Mg : magnésium, Al : aluminium, Si : silicium, K : potassium, Ca : calcium, Ba : baryum et Fe : fer.

Hayes-Roth

Hayes-Roth est le troisième jeu de données choisi pour la configuration des expériences. Les 132 cas dans ces données décrivent cinq attributs numériques [1] à savoir : passe-temps,

l'âge, le niveau d'instruction, l'état matrimonial et le nom de la personne. L'ensemble de données Hayes-Roth est utilisé pour classer les personnes dans l'une des trois classes conceptuelles.

Wine

Le quatrième jeu de données que nous avons utilisé pour la validation des expériences est le résultat d'une analyse chimique des vins cultivés dans la même région en Italie, mais provenant de trois cultivateurs différents. L'ensemble de données de vin contient 178 échantillons classés dans la classe 1, classe 2 et classe 3. Chaque échantillon d'entrée a 13 attributs, l'analyse détermine les quantités des 13 constituants (ie l'alcool, l'acide malique, cendres, alcalinité des cendres, Magnésium, totale phénols, flavonoïdes, des phénols Nonflavanoid, proanthocyanins, intensité de la couleur, la teinte, la DO280 / OD315 des vins dilués et Proline) trouvés dans chacun des trois types de vins.

Pima Indian diabetes

Pour plus de précision, les Indiens Pima diabète est sélectionné de [1] pour tester notre méthode. Il contient 768 échantillons où chacun a huit attributs à savoir ; nombre de fois enceintes, la concentration du plasma glucose à 2h dans un test de tolérance au glucose par voie orale, la pression artérielle diastolique (mm Hg), l'épaisseur du triceps pli cutané (mm), le sérum d'insuline 2-Hour (mu U / ml), indice de masse corporelle, le diabète fonction de pedigree et de l'âge. L'institut national du diabète et des maladies digestives et rénales avait pris ces informations provenant de patients femelles de moins de 21 ans du patrimoine Pima indien. L'ensemble de données a deux classes 1 et 0 interprétés respectivement comme testé positif pour le diabète ou non.

Autres critères d'évaluation de classification

Autres indices d'évaluations

Indice de Dunn

Pour identifier les clusters compacts et bien séparés, on définit l'indice de Dunn [68] comme le rapport entre la plus petite dis-similarité inter-classe d_{min} (i.e. entre deux individus de deux classes différentes) et la plus grande dis-similarité intra-classe s_{max} (i.e. entre deux individus de la même classe). En effet, l'objectif principal de cet indice est de maximiser la dis-similarité inter-classe et de minimiser la dis-similarité intra-classe.

Cet indice est défini par :

$$Dunn(P) = \frac{d_{min}}{s_{max}} \tag{4}$$

Indice de Dunn généralisé

L'indice de Dunn généralisé [30] a été introduit après qu'il ait été démontré que l'indice de Dunn que nous venons de définir était non robuste. En effet, il dépend uniquement d'un nombre réduit d'individus de la population, et des liens établis entre eux. Ainsi, il est sensible à tout changement qui intervient dans la structure des clusters ainsi qu'aux points aberrants. Les modifications apportées à cet indice interviennent dans le calcul de la dis-similarité interclasse et intra-classe. L'indice de Dunn généralisé est reconnu comme l'un des indices les plus appropriés pour l'évaluation de la qualité d'une partition donnée, car il fournit un bon compromis entre la maximisation de la dis-similarité interclasse et la minimisation de la dis-similarité intra-classe de la partition.

$$DG(P) = \frac{\min_i \{\min_{j \neq i} \{d_a(C_i, C_j)\}\}}{\max_h s_a(C_h)} \tag{5}$$

Avec :

$$\forall C_i \in P : s_a(C_i) = \frac{1}{|C_i|(|C_i|-1)} \sum_{u=1}^{|C_i|} \sum_{q=1}^{|C_i|} d(X_u, X_q) \tag{6}$$

$$\forall C_i, C_j \in P : d_a(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{u=1}^{|C_i|} \sum_{q=1}^{|C_j|} d(X_u, X_q) \tag{7}$$

où d représente la mesure de dis-similarité définie sur l'ensemble d'individus X et $|C_i|$ le cardinal (nombre d'individus de la classe C_i.

La partition P produisant la plus grande valeur de $DG(P)$ correspondra à la meilleure classification.

Entropie

Cet indexe utilise la distribution des objets à l'intérieur de chaque classe et il est défini comme suit :

$$E_j = \sum_{1}^{n_j} p_i j log p_{ij} \tag{8}$$

L'entropie d'un ensemble de classes est définie par la somme des entropies de toutes les classes, c'est à dire : $E = \sum_1^N E_j$ Avec $p_i j = X_j(x_i)$ où X_j est la distribution à l'intérieur de la classe C_j et x_i un de ses éléments.

Indice de pureté

La pureté d'une classe C_j est définie par l'équation suivante :

$$P_j = \frac{1}{n_j} \max_i n_j^i \tag{9}$$

Où n_j^i est le nombre des objets de la classe C_j qu'il partage avec la classe C_i. La pureté de l'ensemble des classes est définie comme une somme pondérée de pureté de ses classes c'est-à-dire :

$$Purt = \sum_{1}^{N} \frac{n_j}{n} P_j \tag{10}$$

Où n_j est le cardinal de la classe C_j et N le nombre total des classes.

Critères de sélection de modèles

Critère AIC

Le critère AIC (An Information Criterion) s'applique aux modèles estimés par une méthode du maximum de vraisemblance : les analyses de variance, les régressions linéaires multiples, les régressions logistiques et de Poisson peuvent rentrer dans ce cadre. Le critère AIC est défini par :

$$AIC = -2logL + 2k \tag{11}$$

où L est la vraisemblance maximisée et k le nombre de paramètres dans le modèle.
Avec ce critère, la déviance du modèle $-2log(L)$ est pénalisée par 2 fois le nombre de paramètres.

L'AIC représente donc un compromis entre le biais, diminuant avec le nombre de paramètres, et la parcimonie, volonté de décrire les données avec le plus petit nombre de paramètres possibles.

- La rigueur voudrait que tous les modèles comparés dérivent tous d'un même « complet » inclus dans la liste des modèles comparés.
- Il est nécessaire de vérifier que les conditions d'utilisation du modèle complet et de celui sélectionné sont remplies.
- Le meilleur modèle est celui possédant l'AIC le plus faible.

Critère BIC

Le critère BIC (Bayesian information criterion) est défini par :

$$BIC = -2log(L) + klog(n). \tag{12}$$

Il est plus parcimonieux que le critère AIC puisqu'il pénalise plus le nombre de variables présentent de le modèle.

Ripley [181] souligne que l'AIC a été introduit pour retenir des variables pertinentes lors de prévisions, et que le critère BIC vise la sélection de variables statistiquement significative dans le modèle.

Bibliographie

[1] Uci machine learning repository.

[2] Abdelatif, E.-s., Fidae, H., and Mohamed, E. Optimization of the organized kohonen map by a new model of preprocessing phase and application in clustering. *Journal of Emerging Technologies in Web Intelligence 6*, 1 (2014), 80–84.

[3] Al-Mohair, H. K., Saleh, J. M., and Suandi, S. A. Hybrid human skin detection using neural network and k-means clustering technique. *Applied Soft Computing 33* (2015), 337–347.

[4] Almajali, S., Sharieh, A., and Qutiashat, M. Arabic speech recognition using som-lva neural network. *ADVANCES IN MODELLING AND ANALYSIS-B- 44*, 3/4 (2001), 1A–16A.

[5] Anderberg, M. R. Cluster analysis for applications. monographs and textbooks on probability and mathematical statistics, 1973.

[6] Anderberg, M. R. *Cluster analysis for applications : probability and mathematical statistics : a series of monographs and textbooks*, vol. 19. Academic press, 2014.

[7] Anouar, F. *Modélisation probabilistes des cartes auto-organisées. Application en classification et en régression*. PhD thesis, 1996.

[8] Anouar, F., Badran, F., and Thiria, S. Self organizing map, a probabilistic approach. In *Proceedings of WSOM* (1997), vol. 97, pp. 4–6.

[9] Anouar, F., Badran, F., and Thiria, S. Probabilistic self-organizing map and radial basis function networks. *Neurocomputing 20*, 1 (1998), 83–96.

[10] Arciniegas, I., Daniel, B., and Embrechts, M. J. Exploring financial crises data with self-organizing maps (som). In *Advances in Self-Organising Maps*. Springer, 2001, pp. 39–46.

[11] Arifovic, J., and Gencay, R. Using genetic algorithms to select architecture of a feedforward artificial neural network. *Physica A : Statistical mechanics and its applications 289*, 3 (2001), 574–594.

[12] Arthur, D., and Vassilvitskii, S. k-means++ : The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), Society for Industrial and Applied Mathematics, pp. 1027–1035.

[13] Audouze, K., Ros, F., Pintore, M., and Chretien, J. Prediction of odours of aliphatic alcohols and carbonylated compounds using fuzzy partition and self organising maps (som). *Analusis 28*, 7 (2000), 625–632.

[14] Azami, S. B. Z., and Feng, G. Robust vector quantizer design using self-organizing neural networks. *Signal processing 80*, 7 (2000), 1289–1298.

[15] Azcarraga, A. P., and Yap Jr, T. N. Som-based methodology for building large text archives. In *Database Systems for Advanced Applications, 2001. Proceedings. Seventh International Conference on* (2001), IEEE, pp. 66–73.

[16] Azimi, R., Ghayekhloo, M., and Ghofrani, M. A hybrid method based on a new clustering technique and multilayer perceptron neural networks for hourly solar radiation forecasting. *Energy Conversion and Management 118* (2016), 331–344.

[17] Bäck, T., Fogel, D., and Michalewicz, Z. Handbook of evolutionary computation, 1997.

[18] Bäck, T., Fogel, D. B., and Michalewicz, Z. *Evolutionary computation 1 : basic algorithms and operators*, vol. 1. CRC Press, 2000.

[19] Backer, E. *Computer-assisted reasoning in cluster analysis.* Prentice Hall International (UK) Ltd., 1995.

[20] BAIL, L., and MITICHE, A. Quantification vectorielle d'images par le réseau neuronal de kohonen. 529–539.

[21] Balkin, S. D., and Ord, J. K. Automatic neural network modeling for univariate time series. *International Journal of Forecasting 16*, 4 (2000), 509–515.

[22] Bazarghan, M., and Gupta, R. Automated classification of sloan digital sky survey (sdss) stellar spectra using artificial neural networks. *Astrophysics and Space Science 315*, 1-4 (2008), 201–210.

[23] Bebis, G., Georgiopoulos, M., and Kasparis, T. Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization. *Neurocomputing 17*, 3 (1997), 167–194.

[24] Bellman, R. Adaptative control processes, 1961.

[25] Benardos, P., and Vosniakos, G. C. Prediction of surface roughness in cnc face milling using neural networks and taguchi's design of experiments. *Robotics and Computer-Integrated Manufacturing 18*, 5 (2002), 343–354.

[26] Benders, J. F. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik 4*, 1 (1962), 238–252.

[27] Bennani, Y. Apprentissage connexionniste.

[28] Bennani, Y. Apprentissage par réseaux de neurones artificiels.

[29] Benzécri, J.-P., et al. *L'analyse des données*, vol. 2. Dunod Paris, 1973.

[30] Bezdek, J. C., and Pal, N. R. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 28*, 3 (1998), 301–315.

[31] Bishop, C. M. *Neural networks for pattern recognition.* Oxford university press, 1995.

[32] Bishop, C. M. *Pattern recognition.* Springer, 2006.

[33] Borg, I., and Groenen, P. J. *Modern multidimensional scaling : Theory and applications.* Springer Science & Business Media, 2005.

[34] Boubou, M. *Contribution aux méthodes de classification non supervisée via des approches prétopologiques et d'agrégation d'opinions.* PhD thesis, Université Claude Bernard-Lyon I, 2007.

[35] Bourlard, H. A., and Morgan, N. *Connectionist speech recognition : a hybrid approach*, vol. 247. Springer Science & Business Media, 2012.

[36] Bouveyron, C. *Modélisation et classification des données de grande dimension.* PhD thesis, UNIVERSITÉ JOSEPH FOURIER, 2006.

[37] Breiman, L., Friedman, J., Olshen, R., and Stone, C. Isodata, a novel method of data analysis and classification. Tech. rep., Technical report, Stanford Research Institute, 1965.

[38] Burges, C. J. Geometric methods for feature extraction and dimensional reduction-a guided tour. In *Data mining and knowledge discovery handbook.* Springer, 2009, pp. 53–82.

[39] Burrascano, P. Learning vector quantization for the probabilistic neural network. *IEEE transactions on neural networks/a publication of the IEEE Neural Networks Council 2*, 4 (1991), 458.

[40] Caballero, J. C. F., Martínez, F. J., Hervás, C., and Gutiérrez, P. A. Sensitivity versus accuracy in multiclass problems using memetic pareto evolutionary neural networks. *IEEE Transactions on Neural Networks 21*, 5 (2010), 750–770.

[41] Caliński, T., and Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods 3*, 1 (1974), 1–27.

[42] Castillo, P. A., Merelo, J., Prieto, A., Rivas, V., and Romero, G. G-prop : Global optimization of multilayer perceptrons using gas. *Neurocomputing 35*, 1 (2000), 149–163.

[43] Celebi, M. E., Kingravi, H. A., and Vela, P. A. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications 40*, 1 (2013), 200–210.

[44] Celeux, G., Diday, E., Govaert, G., Lechevallier, Y., and Ralambondrainy, H. Classification automatique des données. 1989. *Bordas, Paris*.

[45] ChandraShekar, B., and Shoba, G. Classification of documents using kohonen's self-organizing map. *International Journal of Computer Theory and Engineering 1*, 5 (2009), 610–613.

[46] Chang, C.-T. On the mixed integer signomial programming problems. *Applied mathematics and computation 170*, 2 (2005), 1436–1451.

[47] Chentouf, R. *Construction de réseaux de neurones multicouches pour l'approximation.* PhD thesis, 1997.

[48] Cornuéjols, A., and Miclet, L. *Apprentissage artificiel : concepts et algorithmes.* Editions Eyrolles, 2011.

[49] Cortez, P., Rocha, M., and Neves, J. Evolving time series forecasting arma models. *Journal of Heuristics 10*, 4 (2004), 415–429.

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

[51] Das, A., and Gersho, A. Variable dimension spectral coding of speech at 2400 bps and below with phonetic classification. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on* (1995), vol. 1, IEEE, pp. 492–495.

[52] De Jong, K. Genetic algorithms : a 30 year perspective. *Perspectives on Adaptation in Natural and Artificial Systems 11* (2005).

[53] Delbimbo, A., Landi, L., and Santini, S. Three-dimensional planar-faced object classification with kohonen maps. *Optical Engineering 32*, 6 (1993), 1222–1234.

[54] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)* (1977), 1–38.

[55] Deng, Y.-q., and Wang, P.-m. Predicting the shrinkage of thermal insulation mortar by probabilistic neural networks. *Journal of Zhejiang University SCIENCE A 11*, 3 (2010), 212–222.

[56] Di Piazza, A., Di Piazza, M. C., Ragusa, A., and Vitale, G. Environmental data processing by clustering methods for energy forecast and planning. *Renewable energy 36*, 3 (2011), 1063–1074.

[57] Diday, E. The dynamic clusters method in nonhierarchical clustering. *International Journal of Computer & Information Sciences 2*, 1 (1973), 61–88.

[58] Diday, E., Celeux, G., Govaert, G., Lechevallier, Y., and Ralambondrainy, H. Classification automatique des données. *Dunod, Paris 49* (1989), 80.

[59] Diday, E., and Simon, J. Clustering analysis. In *Digital pattern recognition*. Springer, 1976, pp. 47–94.

[60] Djamah, M. *Codage échelonnable à granularité fine de la parole utilisant la quantification vectorielle arborescente*. PhD thesis, Université du Québec, Institut national de la recherche scientifique, 2011.

[61] Douiri, S. M., Elbernoussi, S., and Lakhbab, H. Cours des méthodes de résolution exactes heuristiques et métaheuristiques. *Université Mohamed V, Faculté des sciences de Rabat* (2009).

[62] Dréo, J., Pétrowski, A., Siarry, P., and Taillard, E. *Métaheuristiques pour l'optimisation difficile*. Eyrolles, 2003.

[63] Dreyfus, G., Martinez, J.-M., Samuelides, M., Gordon, M. B., Badran, F., and Thiria, S. *Apprentissage statistique : Réseaux de neurones-Cartes topologiques-Machines à vecteurs supports*. Editions Eyrolles, 2011.

[64] Dreyfus, G., Martinez, J.-M., Samuelides, M., Gordon, M. B., Badran, F., Thiria, S., and Hérault, L. Réseaux de neurones-méthodologie et applications. *Réseaux de neurones-Méthodologie et applications* (2002).

[65] Duda, R. O., Hart, P. E., et al. *Pattern classification and scene analysis*, vol. 3. Wiley New York, 1973.

[66] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern classification*. John Wiley & Sons, 2012.

[67] Duflo, M. Algorithmes stochastiques, volume 23 of mathématiques & applications (berlin)[mathematics & applications], 1996.

[68] Dunn, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters.

[69] Duran, B. S., and Odell, P. L. *Cluster analysis : a survey*, vol. 100. Springer Science & Business Media, 2013.

[70] Duran, M. A., and Grossmann, I. E. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming 36*, 3 (1986), 307–339.

[71] Edelman, G. M., and Finkel, L. H. Neuronal group selection in the cerebral cortex. *Dynamic aspects of neocortical function* (1984), 653–695.

[72] Eklund, P. W., and Hoang, A. A performance survey of public domain supervised machine learning algorithms. *Australian Journal of Intelligent Information Systems. v9 i1* (2002), 1–47.

[73] En-Naimani, Z., Lazaar, M., and Ettaouil, M. Hybrid system of optimal self-organizing maps and hidden markov model for arabic digits recognition. *WSEAS Trans. Syst 13* (2014), 606–616.

[74] En-Naimani, Z., Lazaar, M., and Ettaouil, M. Architecture optimization model for the probabilistic self-organizing maps and speech compression. *International Journal of Computational Intelligence and Applications* (2016), 165–7.

[75] Es-safi, A. *Contribution à l'analyse de données et à la fouille documentaire : Amélioration de certains algorithmes et proposition de nouveaux modèles*. PhD thesis, Faculty of sciences and technologies, 2014.

[76] ETTAOUIL, M., ABDELATIF, E., and HARCHLI, F. Improving the performance of k-means algorithm using an automatic choice of suitable code vectors and optimal number of clusters. *Journal of Theoretical & Applied Information Technology 56*, 3 (2013).

[77] Ettaouil, M., Essafi, A., and Harchli, F. Optimizing the architecture of kohonen map and classification. *Journal of Computing 4* (2012), 31–38.

[78] Ettaouil, M., and Ghanou, Y. Neural architectures optimization and genetic algorithms. *Wseas Transactions On Computer 8*, 3 (2009), 526–537.

[79] Ettaouil, M., Ghanou, Y., El Moutaouakil, K., and Lazaar, M. Image medical compression by a new architecture optimization model for the kohonen networks. *International Journal of Computer Theory and Engineering 3*, 2 (2011), 204–210.

[80] Ettaouil, M., Lazaar, M., and Ghanou, Y. Architecture optimization model for the multilayer perceptron and clustering. *Journal of Theoretical and Applied Information Technology 47* (2013), 64–72.

[81] Everitt, B. Cluster analysis. 1993. *Edward Arnold and Halsted Press,* (1993).

[82] Fahlman, S. E., and Lebiere, C. The cascade-correlation learning architecture.

[83] Faraoun, K., and Boukelif, A. Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering 1*, 10 (2007), 3138–3145.

[84] Fidae, H., Abdelatif, E.-s., and Mohamed, E. Original approach for reduction of high dimensionality in unsupervised learning. In *Logistics Operations Management (GOL), 2016 3rd International Conference on* (2016), IEEE, pp. 1–4.

[85] Fletcher, R., and Leyffer, S. Solving mixed integer nonlinear programs by outer approximation. *Mathematical programming 66*, 1-3 (1994), 327–349.

[86] Forgy, E. W. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics 21* (1965), 768–769.

[87] Fraley, C., and Raftery, A. E. How many clusters ? which clustering method ? answers via model-based cluster analysis. *The computer journal 41*, 8 (1998), 578–588.

[88] Funahashi, K.-I. On the approximate realization of continuous mappings by neural networks. *Neural networks 2*, 3 (1989), 183–192.

[89] Gersho, A., and Gray, R. M. *Vector quantization and signal compression*, vol. 159. Springer Science & Business Media, 2012.

[90] Ghorbel, S., Jemaa, M. B., and Chtourou, M. Object-based video compression using neural networks. *IJCSI International Journal of Computer Science Issues 8*, 4 (2011).

[91] Golberg, D. E. Genetic algorithms in search, optimization, and machine learning. *Addion wesley 1989* (1989), 102.

[92] Gordon, A. D. Classification, (chapman & hall/crc monographs on statistics & applied probability).

[93] Govaert, G. *Classification binaire et modèle*. PhD thesis, INRIA, 1988.

[94] Guérif, S. *Réduction de dimension en apprentissage numérique non supervisé*. PhD thesis, Paris 13, 2006.

[95] Gutierrez-Osuna, R. Pattern analysis for machine olfaction : a review. *IEEE Sensors journal 2*, 3 (2002), 189–202.

[96] Hammami, N., and Bedda, M. Improved tree model for arabic speech recognition. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on* (2010), vol. 5, IEEE, pp. 521–526.

[97] Han, H.-G., and Qiao, J.-F. A structure optimisation algorithm for feedforward neural network construction. *Neurocomputing 99* (2013), 347–357.

[98] Han, J., Kamber, M., and Pei, J. *Data mining : concepts and techniques*. Elsevier, 2011.

[99] Hao, J.-K., Galinier, P., and Habib, M. Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'intelligence artificielle 13*, 2 (1999), 283–324.

[100] HARCHLI, F., En-naimani, z., Es-SAFI, A., and ETTAOUIL, M. Prsom. In *AAFD, SFC 2016 : Conférence Conjointe Francophone sur la Science des Données* (2016).

[101] HARCHLI, F., JOUDAR, N.-e., Es-SAFI, A., and ETTAOUIL, M. Adaptation of multilayer perceptron neural network to unsupervised clustering using a developed version of k-means algorithm. *WSEAS TRANSACTIONS on COMPUTERS 15* (2016), 103–116.

[102] Hartigan, J. A. Clustering algorithms.

[103] Hartigan, J. A., and Wong, M. A. Algorithm as 136 : A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28*, 1 (1979), 100–108.

[104] Haton, J.-P., and Haton, M.-C. *L'intelligence artificielle.* Presses universitaires de France, 1989.

[105] Haykin, S. *Neural networks, a Comprehensive Foundation.* Preditice Hall, 1996.

[106] HBL, P. M. S. M. *Réduction de dimension en Apprentissage Numérique Non Supervisé.* PhD thesis, Université Paris 13, 2006.

[107] Hernandez, J. Algorithmes d'acquisition, compression et restitution de la parole à vitesse variable. étude et mise en place. *graduation project* (1995).

[108] Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. *Boltzmann machines : Constraint satisfaction networks that learn.* Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984.

[109] Hoang, A. Supervised classifier performance on the uci database. *University of Adelaide* (1997).

[110] Holland, J. H. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[111] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks 2*, 5 (1989), 359–366.

[112] Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology 24*, 6 (1933), 417–441.

[113] Hsu, C.-F. Adaptive growing-and-pruning neural network control for a linear piezoelectric ceramic motor. *Engineering Applications of Artificial Intelligence 21*, 8 (2008), 1153–1163.

[114] Huang, G.-B., and Chen, L. Enhanced random search based incremental extreme learning machine. *Neurocomputing 71*, 16 (2008), 3460–3468.

[115] Huang, Z. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery 2*, 3 (1998), 283–304.

[116] Isa, N. A. M., and Mamat, W. M. F. W. Clustered-hybrid multilayer perceptron network for pattern recognition application. *Applied Soft Computing 11*, 1 (2011), 1457–1466.

[117] Isar, A., Cubitchi, A., and Nafornita, M. Algorithmes et techniques de compression. *Editura Orizonturi Universitare, Timisoara* (2002).

[118] Islam, M. M., and Murase, K. A new algorithm to design compact two-hidden-layer artificial neural networks. *Neural Networks 14*, 9 (2001), 1265–1278.

[119] Jain, A. K. Data clustering : 50 years beyond k-means. *Pattern recognition letters 31*, 8 (2010), 651–666.

[120] Jain, A. K., and Dubes, R. C. *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

[121] Jain, A. K., and Maheswari, S. Survey of recent clustering techniques in data mining. *J Curr Comput Sci Technol 3* (2013).

[122] Jain, A. K., Murty, M. N., and Flynn, P. J. Data clustering : a review. *ACM computing surveys (CSUR) 31*, 3 (1999), 264–323.

[123] Jambu, M. *Méthodes de base de l'analyse des données*. Eyrolles, 1999.

[124] Jiang, W., Wu, X., and Ng, W. Y. Vq index coding for high-fidelity medical image compression. In *Image Processing, 1997. Proceedings., International Conference on* (1997), vol. 3, IEEE, pp. 678–681.

[125] Jiang, X., and Wah, A. H. K. S. Constructing and training feed-forward neural networks for pattern classification. *Pattern recognition 36*, 4 (2003), 853–867.

[126] Jolliffe, I. *Principal component analysis*. Wiley Online Library, 2002.

[127] Kapageridis, I. Variable lag variography using k-means clustering. *Computers & Geosciences 85* (2015), 49–63.

[128] Kaski, S., Kangas, J., and Kohonen, T. Bibliography of self-organizing map (som) papers : 1981–1997. *Neural computing surveys 1*, 3&4 (1998), 1–176.

[129] Kauffman, L., and Rousseeuw, P. Finding groups in data. *An introduction to cluster analysis. New York : John Willey & Sons* (1990).

[130] Kaufman, L., and Rousseeuw, P. J. *Finding groups in data : an introduction to cluster analysis*, vol. 344. John Wiley & Sons, 2009.

[131] Khaw, J. F., Lim, B., and Lim, L. E. Optimal design of neural networks using the taguchi method. *Neurocomputing 7*, 3 (1995), 225–245.

[132] Kim, D. K., Kim, D. H., Chang, S. K., and Chang, S. K. Modified probabilistic neural network considering heterogeneous probabilistic density functions in the design of breakwater. *KSCE Journal of Civil Engineering 11*, 2 (2007), 65–71.

[133] Kogan, J. *Introduction to clustering large and high-dimensional data*. Cambridge University Press, 2007.

[134] Kohonen, T. Automatic formation of topological maps of patterns in a self-organizing system.

[135] Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological cybernetics 43*, 1 (1982), 59–69.

[136] Kohonen, T. Self-organizing maps, 2001.

[137] Kohonen, T. *Self-organization and associative memory*, vol. 8. Springer Science & Business Media, 2012.

[138] Krishnan, T., and McLachlan, G. The em algorithm and extensions. *Wiley 1*, 997 (1997), 58–60.

[139] Ladjal, H. Classification automatique et cartes auto-organisatrices.

[140] Langley, P. *Elements of machine learning*. Morgan Kaufmann, 1996.

[141] Lauret, P., Fock, E., and Mara, T. A. A node pruning algorithm based on a fourier amplitude sensitivity test method. *IEEE Transactions on Neural Networks 17*, 2 (2006), 273–293.

[142] Lazaar, M. *Contribution à la sélection des modèles d'architectures dans les réseaux de neurones artificiels et apprentissage : Traitement automatique de la parole*. PhD thesis, Université SMBA, Faculté de sciences et techniques, 2011.

[143] LeCun, Y., and Bengio, Y. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks 3361*, 10 (1995), 1995.

[144] Lee, J. A., Lendasse, A., and Verleysen, M. Nonlinear projection with curvilinear distances : Isomap versus curvilinear distance analysis. *Neurocomputing 57* (2004), 49–76.

[145] Lehalle, C.-A. *Le contrôle non linéaire par réseaux de neurones formels : les perceptrons affines par morceaux*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2005.

[146] Lenstra, J. K. *Local search in combinatorial optimization*. Princeton University Press, 1997.

[147] Li, H., He, H., and Wen, Y. Dynamic particle swarm optimization and k-means clustering algorithm for image segmentation. *Optik-International Journal for Light and Electron Optics 126*, 24 (2015), 4817–4822.

[148] Likas, A., Vlassis, N., and Verbeek, J. J. The global k-means clustering algorithm. *Pattern recognition 36*, 2 (2003), 451–461.

[149] Linde, Y., Buzo, A., and Gray, R. An algorithm for vector quantizer design. *IEEE Transactions on communications 28*, 1 (1980), 84–95.

[150] Lippmann, R. An introduction to computing with neural nets. *IEEE Assp magazine 4*, 2 (1987), 4–22.

[151] Liu, H., and Yu, X. Application research of k-means clustering algorithm in image retrieval system. In *Proceedings of the Second Symposium International Computer Science and Computational Technology (ISCSCT'09), Huangshan, PR China* (2009), pp. 274–277.

[152] Liu, Y., and Yao, X. Evolving modular neural networks which generalise well. In *Evolutionary Computation, 1997., IEEE International Conference on* (1997), IEEE, pp. 605–610.

[153] López-Rubio, E. Probabilistic self-organizing maps for qualitative data. *Neural Networks 23*, 10 (2010), 1208–1225.

[154] Lotfi, A., Mezzoug, K., and Benyettou, A. Rotated kernel neural networks for radar target detection in background noise. *Journal of Applied Sciences(Faisalabad) 10*, 13 (2010), 1331–1335.

[155] Luttrell, S. Self-organising multilayer topographic mappings. In *Neural Networks, 1988., IEEE International Conference on* (1988), IEEE, pp. 93–100.

[156] Luttrell, S. P. A bayesian analysis of self-organizing maps. *Neural Computation 6*, 5 (1994), 767–794.

[157] Ma, L., and Khorasani, K. A new strategy for adaptively constructing multilayer feedforward neural networks. *Neurocomputing 51* (2003), 361–385.

[158] MacKay, D. An example inference task : clustering. *Information theory, inference and learning algorithms* (2003), 284–292.

[159] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA., pp. 281–297.

[160] Maier, H. R., and Dandy, G. C. The effect of internal parameters and geometry on the performance of back-propagation neural networks : an empirical study. *Environmental Modelling & Software 13*, 2 (1998), 193–209.

[161] Masters, T. *Signal and image processing with neural networks : a C++ sourcebook*. John Wiley & Sons, Inc., 1994.

[162] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[163] Merino, M., Gómez, I. M., and Molina, A. J. Envelopment filter and k-means for the detection of qrs waveforms in electrocardiogram. *Medical engineering & physics 37*, 6 (2015), 605–609.

[164] Messaoud, E., Alaoui, A. E. H., and Boukachour, J. Hybridation de l'algorithme de colonie de fourmis avec l'algorithme de recherche à grand voisinage pour la résolution du vrptw statique et dynamique. *INFOR : Information Systems and Operational Research 51*, 1 (2013), 41–51.

[165] Minsky, M., and Papert, S. Perceptron : an introduction to computational geometry. *The MIT Press, Cambridge, expanded edition 19*, 88 (1969), 2.

[166] Mirdehghani, M., and Monadjemi, S. A. Web pages aesthetic evaluation using low-level visual features. *Proceedings of World Academy of Science : Engineering & Technology 37* (2009), 811–814.

[167] Mitchell, T. M. Machine learning. 1997. *Burr Ridge, IL : McGraw Hill 45* (1997), 995.

[168] Nagih, A., and Plateau, G. Problèmes fractionnaires : tour d'horizon sur les applications et méthodes de résolution. *RAIRO-Operations Research 33*, 4 (1999), 383–419.

[169] Ng, H., Ong, S., Foong, K., Goh, P., and Nowinski, W. Medical image segmentation using k-means clustering and improved watershed algorithm. In *2006 IEEE Southwest Symposium on Image Analysis and Interpretation* (2006), IEEE, pp. 61–65.

[170] Okada, R., and Soatto, S. Relevant feature selection for human pose estimation and localization in cluttered images. In *European Conference on Computer Vision* (2008), Springer, pp. 434–445.

[171] Orhan, U., Hekim, M., and Ozer, M. Eeg signals classification using the k-means clustering and a multilayer perceptron neural network model. *Expert Systems with Applications 38*, 10 (2011), 13475–13481.

[172] Oyelade, O., Oladipupo, O., and Obagbuwa, I. Application of k means clustering algorithm for prediction of students academic performance. *International Journal of Computer Science and Information Security 7*, 1 (2010), 292–295.

[173] Papadimitriou, S., Mavroudi, S., Vladutu, L., Pavlides, G., and Bezerianos, A. The supervised network self-organizing map for classification of large data sets. *Applied Intelligence 16*, 3 (2002), 185–203.

[174] Poggio, T., and Girosi, F. Networks for approximation and learning. *Proceedings of the IEEE 78*, 9 (1990), 1481–1497.

[175] Puchinger, J., and Raidl, G. R. Combining metaheuristics and exact algorithms in combinatorial optimization : A survey and classification. In *International Work-Conference on the Interplay Between Natural and Artificial Computation* (2005), Springer, pp. 41–53.

[176] Quesada, I., and Grossmann, I. E. An lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers & chemical engineering 16*, 10-11 (1992), 937–947.

[177] Raji, U., Mashor, M., Ali, A., Adom, A., and Sadullah, A. Hmlp, mlp and recurrent networks for carbon monoxide concentrations forecasting : A comparison studies. *WSEAS Transactions on Systems 4*, 6 (2005), 812–820.

[178] Rathbun, T. F., Rogers, S. K., DeSimio, M. P., and Oxley, M. E. Mlp iterative construction algorithm. In *AeroSense'97* (1997), International Society for Optics and Photonics, pp. 2–10.

[179] Reddy, D., Jana, P. K., and Member, I. S. Initialization for k-means clustering using voronoi diagram. *Procedia Technology 4* (2012), 395–400.

[180] Reed, R. D., and Marks, R. J. *Neural smithing : supervised learning in feedforward artificial neural networks.* Mit Press, 1998.

[181] Ripley, B. D. *Pattern recognition and neural networks.* Cambridge university press, 2007.

[182] Rosenblatt, F. Principles of neurodynamics.

[183] Rousseeuw, P. J. Silhouettes : a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics 20* (1987), 53–65.

[184] Roweis, S. T., and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326.

[185] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Cognitive modeling 5*, 3 (1988), 1.

[186] Saeys, Y., Inza, I., and Larrañaga, P. A review of feature selection techniques in bioinformatics. *bioinformatics 23*, 19 (2007), 2507–2517.

[187] Sammon, J. W. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers 18*, 5 (1969), 401–409.

[188] Saporta, G. *Probabilités, analyse des données et statistique.* Editions Technip, 2006.

[189] Shepard, R. N. The analysis of proximities : Multidimensional scaling with an unknown distance function. i. *Psychometrika 27*, 2 (1962), 125–140.

[190] Sinha, M., Kumar, K., and Kalra, P. K. Some new neural network architectures with improved learning schemes. *Soft computing 4*, 4 (2000), 214–223.

[191] Sokal, R., Gaul, W., Opitz, O., and Schader, M. *Data analysis : scientific modeling and practical application.* Springer Science & Business Media, 2012.

[192] Sontag, E. D. Feedback stabilization using two-hidden-layer nets. *IEEE Transactions on neural networks 3*, 6 (1992), 981–990.

[193] Späth, H., Bull, U., Hanson, O., and Meek, B. L. Cluster analysis algorithms for data reduction and classification of objects.

[194] Specht, D. F. Probabilistic neural networks. *Neural networks 3*, 1 (1990), 109–118.

[195] Tang, J., Deng, C., and Huang, G.-B. Extreme learning machine for multilayer perceptron. *IEEE transactions on neural networks and learning systems 27*, 4 (2016), 809–821.

[196] Tenenbaum, J. B., De Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science 290*, 5500 (2000), 2319–2323.

[197] Thiria, S., Lechevallier, Y., Gascuel, O., and Canu, S. *Statistique et méthodes neuronales*, vol. 4. Dunod Paris, 1997.

[198] Thiria, S., Mejia, C., Badran, F., and Crepon, M. A neural network approach for modeling nonlinear transfer functions : Application for wind retrieval from space-borne scatterometer data. *Journal of Geophysical Research : Oceans 98*, C12 (1993), 22827–22841.

[199] Thorndike, E. L. "animal intelligence". *Nature 58* (1898), 390.

[200] Torgerson, W. S. Multidimensional scaling : I. theory and method. *Psychometrika 17*, 4 (1952), 401–419.

[201] Torres Moreno, J. M. *Apprentissage et généralisation par des réseaux de neurones : étude des nouveaux algorithmes constructifs*. PhD thesis, 1997.

[202] Touzet, C. Reseaux de neurones artificiels : introduction au connexionnisme ({A} rtificial neural nets : introduction to connectionism).

[203] Venkatesh, Y., and Raja, S. K. On the classification of multispectral satellite images using the multilayer perceptron. *Pattern Recognition 36*, 9 (2003), 2161–2175.

[204] Vercauteren, L., Sieben, G., and Praet, M. The classification of brain tumours by a topological map.

[205] Von der Malsburg, C. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik 14*, 2 (1973), 85–100.

[206] Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. Constrained k-means clustering with background knowledge. In *ICML* (2001), vol. 1, pp. 577–584.

[207] Wang, X., and Paliwal, K. K. Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition. *Pattern recognition 36*, 10 (2003), 2429–2439.

[208] Weigend, A. S. *Time series prediction : forescasting the future and understanding the past*. No. 04 ; QA280, T5. 1994.

[209] Werbos, P. *Beyond regression : New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.

[210] Westerlund, T., and Pettersson, F. *A cutting plane method for solving convex MINLP problems*. Åbo Akademi, 1994.

[211] White, H. Learning in artificial neural networks : A statistical perspective. *Neural computation 1*, 4 (1989), 425–464.

[212] White, H. Connectionist nonparametric regression : Multilayer feedforward networks can learn arbitrary mappings. *Neural networks 3*, 5 (1990), 535–549.

[213] Williams, D. R. G. H. R., and Hinton, G. Learning representations by back-propagating errors. *Nature 323* (1986), 533–536.

[214] Wu, M.-N., Lin, C.-C., and Chang, C.-C. Brain tumor detection using color-based k-means clustering segmentation. In *Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007. Third International Conference on* (2007), vol. 2, IEEE, pp. 245–250.

[215] Y. Bengio, R. Ducharme, P. V. e. C. J. A neural probabilistic language model. *Machine Learning Research 3* (2003), 1137–1155.

[216] Yao, X., and Liu, Y. A new evolutionary system for evolving artificial neural networks. *IEEE transactions on neural networks 8*, 3 (1997), 694–713.

[217] Younus, Z. S., Mohamad, D., Saba, T., Alkawaz, M. H., Rehman, A., Al-Rodhaan, M., and Al-Dhelaan, A. Content-based image retrieval using pso and k-means clustering algorithm. *Arabian Journal of Geosciences 8*, 8 (2015), 6211–6224.

[218] Zhang, J., Walter, G. G., Miao, Y., and Lee, W. N. W. Wavelet neural networks for function learning. *IEEE Transactions on Signal Processing 43*, 6 (1995), 1485–1497.