N° d'ordre : 70 / 2015

UNIVERSITE SULTAN MOULAY SLIMANE
Faculté des Sciences et Techniques
Béni-Mellal

Centre d'Etudes Doctorales « Sciences et Techniques »
Formation Doctorale « Mathématiques et Physique Appliquées »

THÈSE

Présentée par

Said GOUNANE

Pour l'obtention du grade de

Docteur

Spécialité : Informatique

Reconnaissance des caractères manuscrits Tifinagh par logique floue et modèles de langage N-grammes

Soutenue le 9 mai 2015 devant la commission d'examen :

Pr. Mustafa JOURHMANE	Professeur à la FST – USMS - Béni Mellal	Président
Pr. Jilali ANTARI	Professeur à la FP – UIZ - Taroudant	Rapporteur
Pr. Abdellatif HAIR	Professeur à la FST – USMS - Béni Mellal	Rapporteur
Pr. Cherki DAOUI	Professeur à la FST – USMS - Béni Mellal	Rapporteur
Pr. Mohamed SABRI	Professeur à la FST – USMS - Béni Mellal	Examinateur
Pr. Mohamed FAKIR	Professeur à la FST – USMS - Béni Mellal	Directeur de thèse
Pr. Belaid BOUIKHALENE	Professeur à la FP – USMS - Béni Mellal	Co- Directeur de thèse

Résumé

Dans la littérature, on trouve une panoplie d'algorithmes et de techniques pour la conception des systèmes de reconnaissance des caractères manuscrits. Le paramétrage de ces systèmes se base sur un ensemble d'algorithmes d'entraînement. Ainsi, ces systèmes peuvent être considérés comme des boîtes noires, ce qui rend leur maintenance une tâche fastidieuse voir même impossible. En outre, une discrimination des caractères basée uniquement sur les connaissances acquises de leurs différentes formes est imparfaite. En effet, dans certains cas, même un être humain n'arrive pas à classer correctement un caractère manuscrit. Ainsi, on a souvent recours au contexte, donc à la position de ce caractère dans le mot et parfois même la position du mot dans la phrase entière pour décider la classe exacte du caractère en question.

Ainsi, l'objectif de ce travail de thèse est la mise en œuvre d'un système de reconnaissance automatique hors ligne des caractères manuscrits Tifinagh, transparent, interprétable et maniable par un être humain. Pour aboutir à ce but, nous avons développé un système basé essentiellement sur : (1) une méthode d'extraction des caractéristiques sous forme d'une combinaison linéaire de la méthode des densités de pixels et la méthode des distances des centres de gravité; (2) deux algorithmes de classification connues par leur simplicité d'implémentation et de paramétrage; l'algorithme des c-moyennes floues et l'algorithme des k-plus proches voisins flous; et (3) les modèles de langage N-grammes pour représenter les connaissances linguistiques de la langue Amazighe.

La phase d'entrainement de ce système se fait en deux étapes : (1) A partir d'une large base d'exemples constituée de 8847 images de caractères manuscrits Tifinagh isolés, développée au laboratoire de Traitement de l'Information et d'Aide à la Décision (TIAD), l'algorithme des c-moyennes floues est utilisé pour éliminer les données redondantes et les intrus et génère ainsi un nombre réduit des prototypes les plus représentatifs possibles; (2) à partir d'un large corpus constitué de 133178 instances des mots Amazighe, on estime les probabilités de tous les N-grammes possibles.

Ainsi, pour une image d'un mot inconnu, l'algorithme des k-plus proches voisins flous génère un ensemble de propositions pour chacun des caractères de ce mot, muni de leurs degrés d'appartenance aux différentes classes. Ensuite, à partir de ces propositions, on construit les

différents mots possibles avec leur poids. Le modèle N-gramme du langage Amazighe est ensuite utilisé pour attribuer une probabilité à chacun de ces mots. Le produit de cette probabilité et le poids affecté par l'algorithme des k-plus proches voisins sera le facteur clé pour le choix de la séquence des caractères la plus adéquate au mot en question.

Enfin, pour tester notre système nous avons construit une autre base de test composée d'un ensemble d'images des mots Amazighe en Tifinagh contenant 3965 caractères. Avec ce système, nous avons obtenu des résultats meilleurs que ceux obtenus par l'algorithme des k-plus proches voisins flous, les perceptrons multicouches ou les SVM.

Mots-clés : Reconnaissance de formes, caractères manuscrits Tifinagh, distances des centres de gravité, densité de pixels, c-moyennes floues, k-plus proches voisins flous, modèles N-grammes, SVM, Réseaux de neurones, MMC.

Abstract

There exists many algorithms and techniques dealing with the design of systems for pattern recognition problems and handwritten character recognition. The parameters adjustment for these systems is usually a result of training algorithms using collected data. These systems can be considered as black boxes; consequently, matching between these parameters and their real roles in the system is not an easy task. Furthermore, with no context, even a Human been can misclassify some handwritten characters. Therefore, the use of the entire word or even the entire sentence can support the decision about these characters.

The goal of our work is to design a handwritten Tifinagh character recognition system, transparent, maintainable and interpretable by human being. To achieve this goal, our system is based essentially on: (1) A features extraction method based on the pixel density method and the distances of gravity centers method; (2) the fuzzy c-means algorithm and the fuzzy k-nearest neighbors. These two algorithms are well known for their ease of implementation; and (3) N-grams language models, to implements informations about the Amazigh words structures.

The learning process is made up of two steps: (1) From a large dataset of 8847 isolated handwritten characters images, developed in the laboratory of Information Processing and Decision Support (TIAD), the fuzzy c-means algorithm assigns a reduced number of prototypes for each class and eliminate outliers and redundant data; (2) a large corpus containing 133178 Amazigh words is used to estimate N-grams probabilities.

For an unknown word image, the fuzzy k-nearest neighbors algorithm proposes a set of candidates and their membership degrees to different classes. From these candidates we then construct all possible weighted words. The N-gram model assigns a probability for each of the constructed words. The most appropriate word is then the one that has the greatest product of weight assigned by the fuzzy k-nearest neighbors and the probability given by the N-gram model.

Finally, to test the performance of our system, we made another database of word images containing 3965 characters. Results obtained using our approach were considerably higher than those obtained using the fuzzy k-nearest neighbors algorithm alone, multilayer perceptron and support vector machines.

Keywords : Pattern recognition, handwritten Tifinagh characters, gravity center distance, pixel density, fuzzy c-means, fuzzy k-nearest neighbors, Bi-gram model, SVM, ANN, HMM.

Table des matières

Liste des tableaux

Liste des figures

Liste des Abréviations

AdaBoost : Adaptive Boosting
ADALINE : ADAptive LINear NEuron
ADL : Analyse discriminante Linéaire
ACP : Analyse en Composantes Principales
ANFIS : Adaptive Neuro-Fuzzy Inference System
ASCII : American Standard Code for Information Interchange
CMF : C-Moyennes Floues
DCG : Distance du Centre de Gravité
DP : Densité des Pixels
SIF : Systèmes d'inférence floue
k-PPV : k Plus Proches Voisins
k-PPVF : k-plus proches voisins Flous
LMS : Least Mean Square
MMC : Modèles de Markov Cachés
OCR : Optical Character Recognition
PDA : Personal Digital Assistance
PMC : Perceptron Multicouche
RdF : Reconnaissance de formes
RBF : Radial Basis Function (Fonction de Base Radiale)
RNA : Réseaux de Neurones Artificiels
SOM : Self Organized Map
SRF : Système de Reconnaissance de formes
SVM : Support Vector Machines
TALN : Traitement Automatique du Langage Naturel

À mes parents

À ma femme et mes enfants

À mes frères et sœurs

Remerciements

Les travaux présentés dans ce manuscrit ont été effectués au Laboratoire du Traitement de l'Information et Aide à la Décision (TIAD) de la Faculté des Sciences Techniques, Université Sultan Moulay Slimane, Beni Mellal.

Mes recherches ont été réalisées sous la direction de Monsieur Mohamed FAKIR, à qui je tiens à exprimer toute ma reconnaissance pour m'avoir accueillie, pour ses précieux conseils et pour la confiance qu'il m'a accordée. Il convient de remercier également, mon co-encadrant Monsieur Belaid BOUIKHALENE, sa sympathie, son aide et ses encouragements m'ont aidé à mener à bien mes travaux de thèse.

Il convient de remercier également Monsieur Mostafa JOURHMANE, professeur à l'université Sultan Moulay Slimane qui m'a fait l'honneur de présider le jury de la thèse. Je témoigne toute ma gratitude à Monsieur Jilali ANTARI professeur à l'université Ibn Zohr pour avoir accepté de rapporter ce travail. Mes remerciements s'adressent également, à Monsieur Abdellatif HAIR et à Monsieur Cherki DAOUI professeur à l'université Sultan Moulay Slimane, qui m'ont fait l'honneur d'être des rapporteurs de cette thèse et pour l'intérêt qu'ils ont démontré pour mon travail.

Mes vifs remerciements s'adressent aussi à Monsieur Mohamed SABRI, professeur à l'université Sultan Moulay Slimane, pour avoir accepté le rôle d'examinateurs et pour ces remarques avisées.

Je remercie également mes amis : Ouahiba Benmasouda et Youssef Ben Taleb, pour leur soutien amical et professionnel.

Je pense naturellement aux personnes qui m'ont côtoyé durant ces années et je tiens à remercier chaleureusement : mes parents, pour m'avoir toujours encouragé à poursuivre mes études et m'avoir permis d'arriver là où j'en suis. Mon épouse Nadia, qui depuis les prémisses m'accompagne au quotidien et tolère mes défauts. Ma famille et ma belle-famille pour leurs aides, leurs soutiens et leurs encouragements.

INTRODUCTION

L'homme a toujours cherché à améliorer ses conditions de vie. Avec l'apparition de l'ordinateur puis les assistantes personnelles (PDA), les smart phones, etc. L'homme a pu automatiser la majorité de ses tâches quotidiennes dans tous les domaines. Il se trouve alors entouré d'un ensemble de systèmes "intelligents" mais avec des modes d'interaction très réduits et très rigides au contraire de l'homme, caractérisé par une souplesse et une variété de modes d'interactions ; écriture, parole, geste, etc.

Cette automatisation est présente dans le domaine de la reconnaissance de la parole, la commande vocale, la dictée automatique, la traduction en temps réel, la reconnaissance de l'écriture, l'identification des montant dans les chèques bancaires, l'archivage de documents, la lecture et reconnaissance d'adresses pour le tri automatique des courriers et la détection et reconnaissance de visage, etc.

A cause de la sensibilité des capteurs et des médias utilisés pour acquérir et transmettre les informations, les informations qui sont finalement traitées diffèrent très souvent des originales [1] [2]. Ainsi, leur traitement nécessite la mise en œuvre des systèmes complexes d'analyse de perception et de décision. Malgré la croissance de la puissance de calcul, la complexité de ces systèmes est un facteur limitant lors de leur implémentation dans les différents systèmes informatiques, en particulier les systèmes nomades qui ne possèdent que de faibles ressources informatiques. Par conséquent, l'élaboration de ces systèmes nécessite presque toujours un compromis entre les fonctionnalités souhaitées et les capacités que possède la machine hôte.

Dans cette thèse nous nous intéressons plus particulièrement aux systèmes de reconnaissance des caractères. En général, dans ce domaine mais aussi dans des sous-domaines plus précis comme celui de la reconnaissance de formes manuscrites Tifinagh, qui est à l'origine de notre démarche, le but est d'arriver à concevoir un système qui consiste à analyser une forme présentée en entrée afin de l'identifier automatiquement. De nombreuses approches génériques ont été élaborées en s'appuyant sur différents cadres d'études tels que les statistiques, la théorie

des probabilités, l'optimisation convexe, etc. Mais aucune d'elles n'est réellement la solution pour tous les problèmes. Elles possèdent chacune leurs avantages et leurs inconvénients qui les rendent plus ou moins exactes dans un contexte donné. Les concepteurs doivent donc souvent adapter les systèmes existants ou en créer de nouveaux en fonction de leurs besoins et des contraintes de leurs applications. Cette tâche s'avère fastidieuse parce que les classiffieurs ont été souvent considérés comme des boîtes noires qui, à une entrée, devraient faire correspondre une sortie. En conséquence de quoi il est souvent difficile de les paramétrer correctement et de comprendre leurs comportements inappropriés dans certains cas.

Nous pensons donc qu'il est nécessaire pour un classiffieur donné de pouvoir identifier les éléments qui sont à l'origine de défaillances afin d'essayer de les corriger. Pour cela le système ne doit plus être considéré comme une boîte noire, mais plutôt comme un ensemble de modules et de connaissances que le concepteur peut interpréter, manipuler et comprendre. Ainsi le classiffieur doit être suffisamment générique pour pouvoir s'adapter à différents types de problèmes de complexités variées. Pour obtenir de bonnes performances, il doit être fiable et robuste face aux imprécisions et à la variabilité des formes. En vue de son usage dans différents contextes applicatifs, on souhaite qu'il soit compact afin de faciliter son intégration dans des systèmes moins puissants. En fin, pour faciliter son optimisation et son adaptation aux différents contextes d'utilisation, on souhaite que l'architecture soit modulaire et compréhensible pour le concepteur.

La thèse est organisée comme suit :

Dans le chapitre 1, nous présentons les concepts élémentaires de la reconnaissance de formes. L'ensemble de l'exposé s'appuie sur la notion de connaissances dans un classiffieur, leurs liens avec les données et leur intérêt pour le concepteur. Ensuite, Nous présentons un survol des différents algorithmes et techniques de classification supervisées et non supervisées. A la fin de ce chapitre, nous présentons une étude comparative des performances de ces algorithmes dans différents cadres d'application.

Le chapitre 2 sera entièrement consacré à la présentation des différentes techniques de classification par la logique floue. Dans la première partie de ce chapitre nous allons présenter la théorie des sous-ensembles flous qui est à la base de tout processus flou. Après nous

présenterons les systèmes d'inférences flous qui sont au cœur de tout système flou. La dernière partie présente les différents algorithmes flous utilisés dans les systèmes de reconnaissance de formes. Ce chapitre sera clôturé par une présentation d'une comparaison des performances de ces algorithmes dans divers applications.

Le chapitre 3 présente l'état de l'art dans la reconnaissance des caractères manuscrits en général et les manuscrits Tifinagh en particulier. Les différents résultats obtenus dans ce domaine par différentes méthodes de classifications comme les réseaux de neurones, les k-plus proches voisins, les réseaux Bayésiens, les modèles de Markov cachés,… etc. Ensuite, nous présentons une nouvelle approche pour l'extraction des caractéristiques réalisée à l'aide d'une combinaison linéaire de la méthode des densités des pixels et la méthode des distances des centres de gravité. L'algorithme des c-moyennes floues est introduit pour produire un ensemble réduit de prototypes utilisés par l'algorithme des k-plus proches voisins flous dans la phase de classification. Une application de ce processus est présentée à la fin de ce chapitre.

Le chapitre 4 traite le sujet du traitement automatique du langage naturel. Dans ce chapitre, on s'intéresse uniquement aux techniques de modélisation d'un langage. Dans la première partie de ce chapitre, on traite le problème de la détection et correction des fautes d'orthographe en utilisant la distance d'édition minimale " Minimum edit distance " et le modèle du canal bruité " Noisy chanel model ". La deuxième partie comporte une présentation des modèles N-grammes, les bi-grammes en particulier, que nous avons utilisés pour la modélisation du langage Amazighe à l'aide d'un corpus que nous avons élaboré au sein du laboratoire (TIAD). On termine ce chapitre par quelques méthodes utilisées pour l'évaluation d'un modèle de langage comme l'entropie et la perplexité.

Le chapitre 5 présente une nouvelle approche pour la reconnaissance des caractères manuscrits Tifinagh, par l'algorithme du k-plus proches voisins flous combiné avec le modèle de langage N-gramme, afin d'introduire des connaissances relatives à la langue Amazighe lors de la reconnaissance. Le système développé dans cette approche est alimenté par un mot manuscrit entier au lieu d'un caractère isolé et donne en sortie les différents caractères qui le composent. Pour cette fin, nous avons construit une base de données des caractères manuscrits Tifinagh et un corpus pour la langue Amazighe constitué d'un ensemble de texte écrit en Tifinagh extrait de plusieurs sites Amazighe. Notre système de reconnaissance a été validé par

un ensemble d'expérimentations permettant de mettre en évidence l'intérêt de son architecture structurée autour des connaissances intrinsèques et discriminantes. Les résultats de ces tests sont reportés à la fin de ce chapitre.

Finalement nous terminons ce mémoire par une conclusion générale et des perspectives.

PREMIÈRE PARTIE

ÉTUDE BIBLIOGRAPHIQUE

CHAPITRE 1. GENERALITES SUR LA RECONNAISSANCE DE FORMES

I. INTRODUCTION

La reconnaissance de formes est une discipline qui a pour but la classification d'objets en un nombre de classes. Ces objets peuvent être des images, des signaux ou tout autre type de mesure. Avant les années 60, la reconnaissance de formes était essentiellement le fruit de longues années de recherches théoriques en statistiques. Après et avec les progrès en informatique, des applications pratiques de reconnaissance de formes ont vu le jour. Vu le succès de ces applications, les recherches ont été de plus en plus approfondies pour couvrir tous les aspects de notre quotidien. De nos jours, la reconnaissance de formes est devenue une partie intégrante dans la majorité des systèmes intelligents pour la prise de décision [3].

La vision par ordinateur est une application de grande importance de la reconnaissance de formes. Les images sont prises par une caméra, puis analysées pour produire une description de ces images afin de prendre une décision. La vision par ordinateur est bien appliquée dans l'industrie pour la détection des imperfections dans les chaînes de production où les produits présentant des défauts de fabrication sont automatiquement détectés et rejetés. Les images prises par la caméra sont donc analysées en lignes puis classées en objets "défectueux" ou "non défectueux". Puis une décision est prise pour garder ou rejeter ces objets.

La reconnaissance des caractères est aussi un domaine où la reconnaissance de formes a son mot à dire. Une panoplie de systèmes OCR (Optical Character Recognition) existe déjà dans le marché et utilisé presque par tout le monde. Un système OCR se compose en général de deux parties, un capteur (scanner) pour transformer une page d'un livre en une image numérique, puis une application pour segmenter l'image en lignes puis en caractères pour ensuite alimenter le système de reconnaissance de formes pour classer chacun de ces caractères. Le document est

ainsi stocker en ASCII ou UNICODE au lieu d'une image, ce qui réduit la taille du document et permet d'y effectuer des traitements ultérieurs. Dans le domaine de la reconnaissance des caractères, on distingue entre la reconnaissance des caractères imprimés et la reconnaissance des manuscrits. Les manuscrits représentent plus de difficulté que les caractères imprimés, car les manuscrits dépendent énormément du scripteur, ce qui rend la reconnaissance très ambigüe même par un être humain. Malgré ces difficultés, plusieurs applications de reconnaissance automatique des caractères manuscrits sont commercialisées. Dans les banques, elles sont utilisées pour la détection automatique des montants des chèques ainsi que l'authentification des signatures. Dans les postes, la reconnaissance automatique des caractères manuscrits est utilisée pour classer les courriers selon leurs destinations.

En médecine, le diagnostic assisté par ordinateur est une autre forme d'application de reconnaissance de formes. Un médecin est souvent confronté à interpréter déférents types de données médicales comme les rayon-X, les images ultrason, les électrocardiogrammes. Parfois ces données ne sont pas aisément interprétables ou identifiables par un spécialiste et demandent plus de vigilance et de concentration d'où le besoin d'une seconde interprétation pour confirmer ou non celle du médecin [4].

Comme la parole est l'outil de communication le plus naturel utilisé par l'être humain, la mise au point d'une machine capable d'interpréter le langage humain est un but de grande importance. Dans ce sens, plusieurs recherches dans le domaine de la reconnaissance de formes se sont orientées vers la reconnaissance de la parole [5] [6]. Parmi les principales applications de la reconnaissance de la parole on trouve la transformation de la parole en texte en utilisant un microphone. La parole est ainsi transformée en texte ASCII ou Unicode.

Dans n'importe quel système de reconnaissance de formes, un certain nombre de composantes et de techniques doit être introduits ; il faut d'abord acquérir des informations du monde extérieur, subir ces informations à un prétraitement afin d'éliminer toute sorte de bruit, extraire et sélectionner uniquement les informations pertinentes et discriminantes, classification et enfin tester et valider les résultats obtenus. Dans ce chapitre nous allons présenter en détail, chacune de ces techniques et composantes. Plus d'importance sera accordée à l'extraction et la sélection des caractéristiques et aux algorithmes de classification.

II. COMPOSANTES D'UN SYSTEME DE RECONNAISSANCE DE FORMES

Un système de reconnaissance de formes est toujours associé à un classifieur et un algorithme d'apprentissage. En réalité, un système de reconnaissance de formes se compose de plusieurs blocks où le classifieur et l'algorithme d'apprentissage constituent un de ces composantes (Figure 1).

II.1. Acquisition des données

Un bon système de reconnaissance de formes nécessite un nombre de données suffisant pour son entraînement et son évaluation. En plus, ces données doivent être suffisamment représentatives. En général, il n'y a aucune règle qui spécifie exactement le nombre des données.

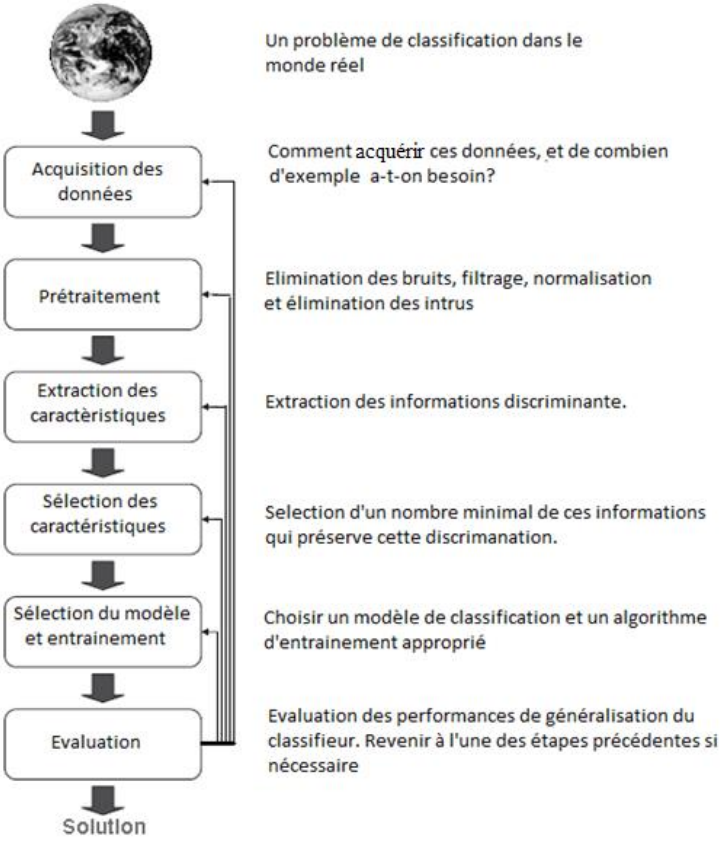


Figure 1: Composantes d'un système de reconnaissance de formes

Dans quelques ouvrages, on trouve que le nombre de données doit être au moins dix fois le nombre de paramètres à ajuster pour le classifieur [7]. Les données doivent être représentatives, pour assurer que les données d'entraînement couvrent la totalité des cas possibles qu'un classifieur peut rencontrer. Cette dernière condition n'est pas aisément satisfaite vu qu'on ne peut pas s'assurer que ces données couvrent d'une façon équilibrée tout l'espace sur lequel le problème de classification est défini.

II.2. Prétraitement

Après l'acquisition, le prétraitement est une étape essentielle. Son but est de conditionner les données par la réduction des bruits. Si par exemple l'environnement de mesure introduit des bruits de hautes (basses) fréquences, un filtre passe-haut (passe-bas) doit être utilisé. Le prétraitement concerne aussi la normalisation des données afin que les éléments d'un vecteur caractéristique aient une moyenne nulle et une variance égale à 1. Cette normalisation s'effectue suivant l'équation suivante :

$$x'_i = \frac{x_i - \mu_i}{\sigma_i} \qquad (1)$$

Où x_i est le i^{éme} élément du vecteur caractéristique de dimension n, μ_i la moyenne et σ_i la déviation standard.

On divise alors par une constante pour que ces éléments soient dans l'intervalle [-1 1] :

$$x' = \frac{x}{\sqrt{\sum_{i=1}^{n} x_i^2}} \qquad (2)$$

L'élimination des intrus (outliers) est une autre étape du prétraitement. Ces intrus peuvent être identifiés visuellement dans le cas des espaces de dimension n ≤ 3. Pour n > 3 la distance de Mahalanobis peut être utilisée. Cette distance se définie comme suit :

$$M_i = (x - \mu_i)^T \sum_i^{-1} (x - \mu_i) \qquad (3)$$

Où μ_i est le centre de la classe i et Σ_i est la matrice de covariance. Si cette distance dépasse un seuil prédéterminé, alors le vecteur x est un intrus.

II.3. Extraction des caractéristiques

Le but de cette phase est d'extraire un nombre réduit et significatif des caractéristiques d'un objet pour le mieux classer et le distinguer des autres objets. En plus, ces caractéristiques doivent être invariantes aux différentes transformations qui peuvent affecter cet objet comme la rotation, la translation, le changement de taille…etc.

L'extraction des caractéristiques est souvent obtenue à l'aide d'un ensemble de transformations mathématiques des données, comme la méthode de l'analyse en composantes principales (ACP) (Principal Component Analysis PCA) ou l'analyse discriminante linéaire (ADL) (Linear Discriminent Analysis LDA) [2].

II.3.a. Analyse en composantes principales (ACP)

Développée par Pearson en 1901 puis généralisée par Karhunen-Loéve en 1963 [8]. L'ACP est l'une des transformations les plus utilisées pour la réduction des dimensions dans la le domaine de la reconnaissance de formes.

Le principe de cette technique est de projeter les données sur un autre espace de dimension réduite, tout en gardant les informations les plus significatives et discriminantes dans les données originales. On calcule les valeurs propres et les vecteurs propres de la matrice de covariance. Les vecteurs propres s'appellent les axes principaux et les projections des données sur ces axes s'appellent les composantes principales. On ne garde que les axes correspondants aux valeurs propres élevées (Figure 2).

Bien que l'ACP produise une meilleure représentation des données dans un espace de dimensions réduites, rien ne garantit que ces données soient séparables en classes comme leurs représentations dans l'espace d'origine, car l'ACP ne prend pas en considération les informations relatives à l'appartenance aux différentes classes.

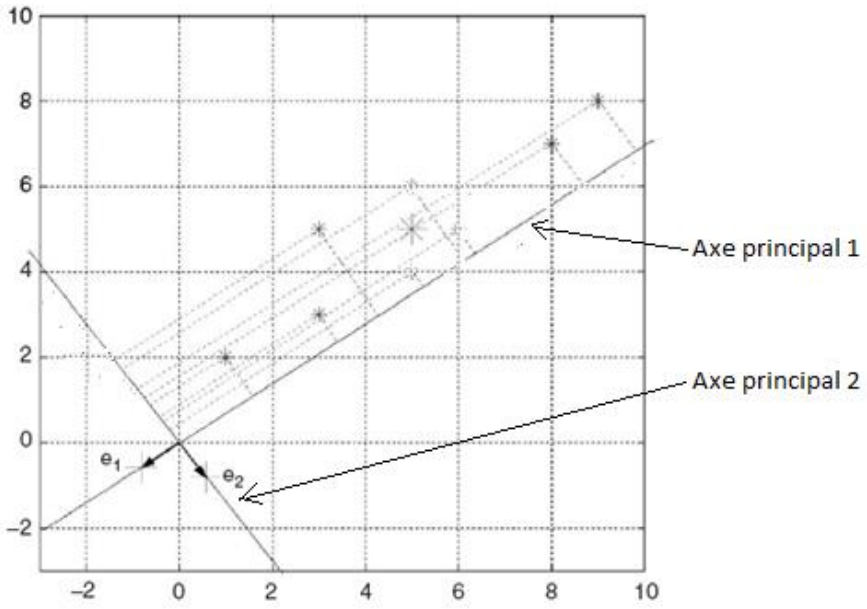


Figure 2: Analyse en composantes principales [2]

II.3.b. Analyse discriminante linéaire (ADL)

L'ADL permet de maximiser la distance entre les différentes classes tout en minimisant la distance entre les différents éléments de la même classe dans le nouvel espace de dimension réduite.

Soit N est le nombre de classes dans un problème de classification donné. Soit m_i et n_i la moyenne et le nombre respectivement des éléments dans la i^{éme} classe C_i. Soit m la moyenne de tous les éléments des N classes. Soient les matrices suivantes :

$$S_W = \sum_{i=1}^{N} \sum_{x \in C_i} (x - m_i).(x - m_i)^T \quad (4)$$

$$S_B = \sum_{i=1}^{N} n_i (m - m_i).(m - m_i)^T \quad (5)$$

Soit la matrice W obtenue par la maximisation du critère suivant :

$$J(W) = \frac{W^T S_B W}{W^T S_W W} \quad (6)$$

La transformation à effectuer est alors la projection suivante :

$$y = W^T . x \quad (7)$$

Les colonnes de la matrice W sont les vecteurs propres et les valeurs propres associées sont calculées à l'aide de l'équation suivante :

$$S_W^{-1} S_B w_i = \lambda_i w_i \quad (8)$$

Comme dans la méthode précédente, on ne garde que les axes correspondants aux valeurs propres élevées.

On peut constater facilement que le nombre de colonnes de la matrice W, d'où le nombre de vecteurs propres, est égale au nombre de classe N. d'où le maximum nombre d'axes dans le nouvel espace de dimensions réduites est N-1. Cette contrainte présente un problème majeur lorsqu'on passe d'un espace de grande dimension N, à un autre de dimension N-1, on risque de perdre des informations utiles pour la classification.

II.4. Sélection des caractéristiques

La sélection des caractéristiques consiste à choisir parmi les n caractéristiques, obtenues lors de la phase d'extraction, les m caractéristiques représentant les informations les plus discriminantes. Cette étape est très dépendante du classifieur choisi, car ces m caractéristiques sélectionnées doivent aboutir à des bonnes performances de généralisation de ce dernier. Les caractéristiques sélectionnées pour un réseau de neurones par exemple peuvent être différentes de celles choisie pour les réseaux Bayésiens ou pour la méthode des k-plus proches voisins [2].

Une méthode intuitive consiste à tester tous les sous-ensembles possibles des caractéristiques sur le classifieur choisi et de choisir celui qui présente les meilleures performances de généralisation. En revanche, malgré sa simplicité de cette méthode, elle est très coûteuse en calcul, car dans un espace de dimension n, le nombre de combinaisons possibles

est $\sum_{m=1}^{n-1} C_n^m$. Donc pour un vecteur caractéristique de taille n=11 par exemple, nous avons 184756 sous ensemble possible, ce qui ne rend pas la tâche facile pour des espaces de grandes dimensions.

Pour éviter ce problème, plusieurs algorithmes de recherche plus efficace existent. On peut citer par exemple, l'algorithme de recherche en profondeur d'abord (Depth-First Search), l'algorithme de recherche en largeur d'abord (Breath-First Search) et l'algorithme d'exploration par escalade (Hill-Climbing Search) [9] …etc. L'algorithme d'exploration par escalade consiste à tester une seule caractéristique à la fois sur le classifieur et de garder celle qui présente les meilleures performances. On teste une autre fois une caractéristique combinée avec celle choisie lors de la phase précédente. On continue ainsi jusqu'à ce qu'aucune caractéristique additionnelle ne parvienne à améliorer les performances du classifieur (Figure 3).

Une autre variante du même algorithme [9] consiste à commencer par l'utilisation de la totalité des caractéristiques, éliminer une et voir si les performances du classifieur ne sont pas détériorées, alors la caractéristique est éliminée définitivement, sinon, on la garde et on passe à la caractéristique suivante (Figure 4).

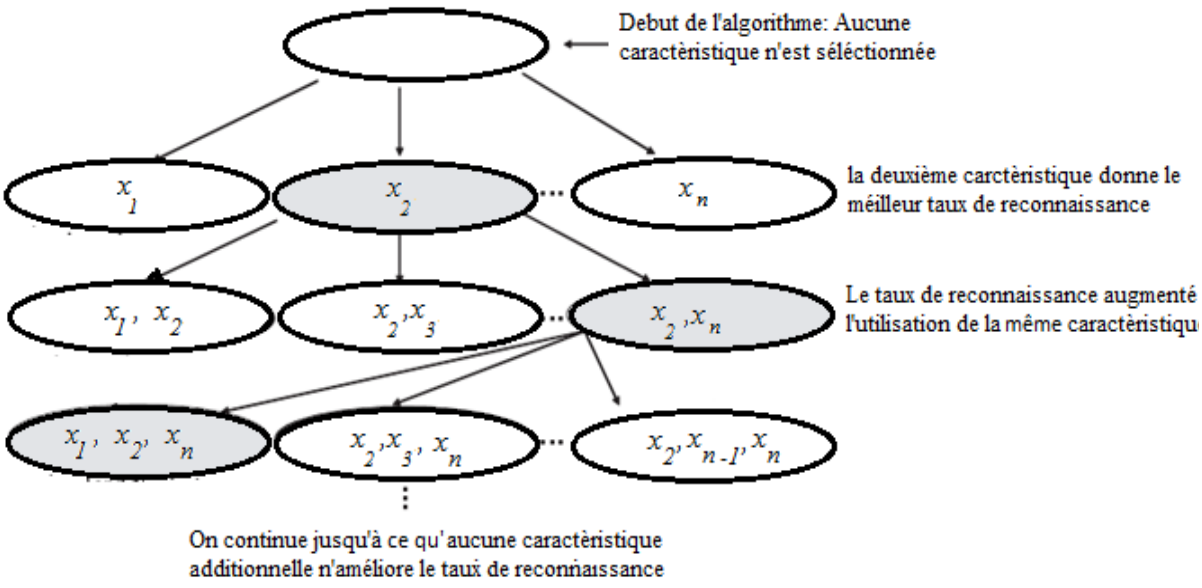


Figure 3: Algorithme d'exploration par escalade (Forward search hill climb)

II.5. Classifieurs

Après l'acquisition, le prétraitement des données, l'extraction et la sélection des caractéristiques les plus représentatives, vient l'étape du choix du classifieur et de son algorithme d'entraînement. Un classifieur peut être vu comme un problème d'approximation de fonction qui, à chaque entrée, va associer une sortie adéquate. Dans ce sens, les entrées et les sorties doivent être convenablement représentées sous forme numérique. Par suite une panoplie d'algorithmes peut être utilisée comme les réseaux de neurones, les réseaux Bayésiens, les modèles de Markov Cachés, las machines à vecteurs support ….etc.

II.5.a. Réseaux Bayésiens

Principe

Dans cette méthode, les données sont supposées être issues d'une distribution de probabilités. Chaque élément possède une probabilité d'appartenance à une classe. Ces probabilités peuvent être soient des fonctions connues ou bien des fonctions estimées à partir d'un ensemble de données d'entraînement.

Soit $x = (x_1, x_2, \dots, x_n)$ un élément à affecter à une classe parmi N classes c_1, $c_2, \dots, c_N$. Un vecteur caractéristique x appartenant à la classe c_i est considéré comme une

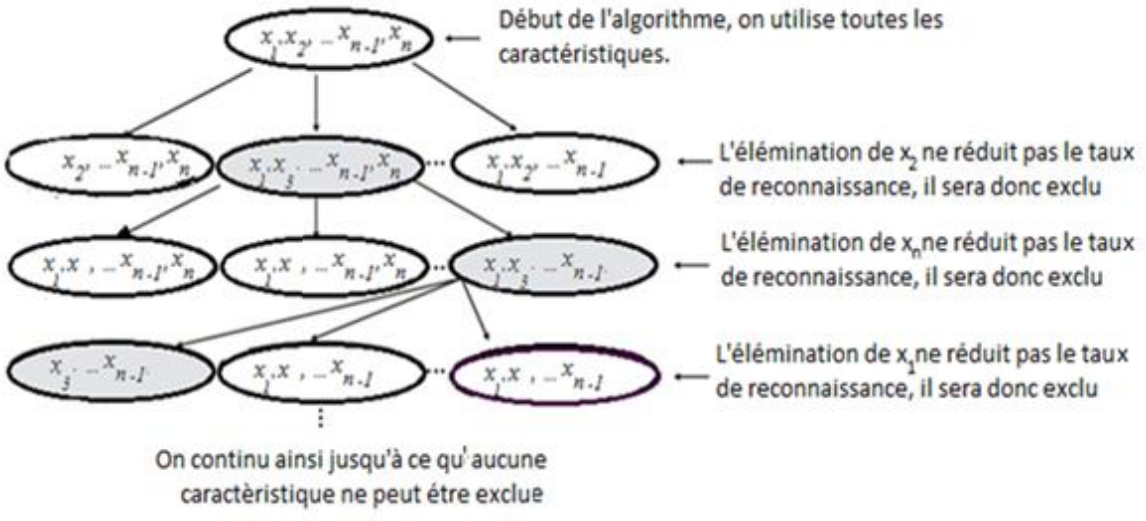


Figure 4: Une variante de l'algorithme d'exploration par escalade (Backward search hill climb)

observation issue aléatoirement d'une distribution de probabilités conditionnelle sur c_i ; $P(x|c_i)$. Si toutes les classes ont la même probabilité, alors le vecteur x est affecté à la classe produisant la plus grande valeur de $P(x|c_i)$. Sinon, on utilise la probabilité à postériori $P(c_i|x)$. Cette probabilité est calculée à l'aide du théorème de Bayes comme suit :

$$P(c_i|x) = \frac{P(x|c_i)P(c_i)}{P(x)} \quad (9)$$

Où $P(c_i)$ est la probabilité à priori, qui est la probabilité d'avoir la classe c_i sans prendre en considération l'observation x. Cette probabilité est souvent déduite par expérience ou bien à travers un ensemble de données d'entraînement.

Pour une observation x, la probabilité à postériori est alors calculée pour chaque classe. Le vecteur x, est affecté à la classe produisant la probabilité à postériori maximal. Si x appartient à la classe c_i, alors:

$$P(c_i|x) > P(c_j|x), \forall j = 1, .., i-1, i+1, \dots N \quad (10)$$

Autrement dit, la classe à laquelle x sera affecté est:

$$c = \underset{c_i}{\operatorname{argmax}} P(c_i|x) = \underset{c_i}{\operatorname{argmax}} \frac{P(x|c_i)P(c_i)}{P(x)} \quad (11)$$

Puisque $P(x)$ est la même pour toutes les classes alors:

$$c = \underset{c_i}{\operatorname{argmax}} P(x|c_i)P(c_i) \quad (12)$$

Dans cette équation, reste un terme à évaluer ; $P(x|c_i)$. L'estimation de ce terme n'est pas évidente, surtout dans des espaces de grandes dimensions, ce qui rend ce type de classifieur inutile pour la majorité des problèmes de classification. Ce classifieur n'est alors utilisé que lorsque :

1. La fonction $P(x|c_i)$ est connue a priori.

2. La forme de la fonction $P(x|c_i)$ est connue mais pas ses paramètres qu'on peut estimer en utilisant la méthode du maximum de vraisemblance (Maximum Liklehood Estimation).
3. On cherche à comparer les performances d'un classifieur avec le classifieur de Bayes sur des données générées artificiellement.

Réseaux Bayésiens naïfs

Le problème majeur dans la mise en pratique du classifieur de Bayes est l'évaluation du terme $P(x|c_i)$ dans des espaces de grandes dimensions. Pour remédier à ce problème, une solution très pratique est adoptée. Cette solution consiste à supposer que les éléments d'un vecteur caractéristique x sont conditionnellement indépendants, autrement dit, si $x = (x_1, x_2, \dots, x_n)$ alors nous avons :

$$P(x|c_i) = \prod_{j=1}^{n} P(x_j|c_i) \qquad (13)$$

L'équation (12) se transforme alors en l'équation suivante:

$$c = \underset{c_i}{\operatorname{argmax}}\, P(c_i) \prod_{j=1}^{n} P(x_j|c_i) \qquad (14)$$

L'avantage majeur de cette technique est de passer d'une estimation d'une fonction de distribution de probabilités multivariable $P(x|c_i)$ à une fonction monovariable $P(x_j|c_i)$ plus simple à évaluer.

En pratique, malgré que les éléments d'un vecteur caractéristique ne soient pas réellement conditionnellement indépendants, la méthode des réseaux Bayésiens naïfs a montré de grandes performances par rapport à celles des réseaux de neurone artificiels [2] [10].

II.5.b. Modèles de Markov Cachés (MMC)

Les modèles de Markov cachés sont une extension des chaines de Markov. Un MMC est caractérisé par un processus aléatoire modélisant l'état du système et qui est observé à l'aide d'une émission aléatoire. Un tel système est dit un système doublement stochastique.

Dans le cas discret, un MMC est bien définit par les paramètres suivants :

1. $A = \{a_{ij}\}$, la matrice de la chaine de Markov, où a_{ij} est la probabilité de la transition de l'état i à l'état j, avec $i, j \in \{1, 2, \dots, N\}$ où N est le nombre des états dans la chaine.
2. $B = \{b_i(k)\}$, la matrice de distribution de sortie, où $b_i(k)$ est la probabilité de produire le symbole k quand le processus de Markov est dans l'état *i*, avec $k \in \{1, 2, \dots, M\}$.
3. $\prod = \{\pi_i\}$, la probabilité que le processus de Markov commence à l'état *i*. Dans la plupart des cas, on suppose que le processus commence à l'état 1, par suite $\pi_1 = 1 \; et \; \pi_i = 0 \; \forall i \neq 1$

La représentation compacte d'un MMC est : $\lambda = (\prod, A, B)$.

Le développement d'un MMC passe toujours par la résolution de trois problèmes ; L'évaluation, le décodage et l'entrainement.

Evaluation

Pour une séquence d'observations donnée $O = o_1, o_2, \dots, o_T$, le calcule de $P(O|\lambda)$ est donnée par :

$$P(O|\lambda) = \sum_S P(O|S, \lambda)P(S) \qquad (15)$$

Où S est l'ensemble de toutes les transitions possibles.

Sachant que dans un MMC les observations sont conditionnellement indépendantes et qu'un état ne dépend que de l'état qui le précède, l'équation (15) devient :

$$P(O|\lambda) = \sum_S a_{s_0 s_1} b_{s_1}(o_1) a_{s_1 s_2} b_{s_2}(o_2) \dots a_{s_{T-1} s_T} b_{s_T}(o_T) \qquad (16)$$

Comme le nombre de séquence possible est N^T, le calcul de cette probabilité n'est pas évident. Pour simplifier ce calcul, on introduit la probabilité $\alpha_t(i) = P(o_1, \; o_2, \dots, o_t, \; s_t = i|\lambda)$. Ensuite, on calcul la probabilité $P(O|\lambda)$ dans un processus itératif décrit par l'algorithme suivant :

Algorithme : Evaluation

1. *Initialisation* :

$$\alpha_1(1) = 1$$

2. *Récurrence :*

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(o_t)$$

3. *Terminaison :*

$$P(O|\lambda) = \alpha_T(N)$$

Décodage

Le but du décodage est de trouver la séquence des états S^* correspandant à la sequence des observations O tel que :

$$S^* = \underset{S}{\operatorname{argmax}}\, P(O, S|\lambda) \qquad (17)$$

Par l'introduction de la probabilité partielle de Viterbi $\delta_t(i)$ définit par :

$$\delta_t(i) = \max_{s_1, s_2, \ldots, s_{t-1}} P(o_1, o_2, \ldots, o_t, s_1, s_2, \ldots, s_{t-1}, s_t = i|\lambda) \qquad (18)$$

Le calcul de S^* est donné par l'algorithme suivant :

Algorithme de Viterbi: Décodage

1. Initialisation :

$$\delta_1(1) = 1$$

$$B_1(1) = 1$$

2. Récurrence :

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(j) a_{ij}\right] b_j(o_t)$$

$$B_t(j) = \underset{1 \le i \le N}{argmax} \left[\delta_{t-1}(j) a_{ij}\right]$$

3. Terminaison :

$$P(S^*) = \delta_T(N)$$

$$B_T(N) = \underset{1 \le i \le N}{argmax} [\delta_{t-1}(N) a_{iN}]$$

4. Retour en arrière :

$$s_t^* = B_{t+1}(s_{t+1}^*)$$

Où $B_t(i)$ représente la trace du méilleur chemain partiel amenant à l'état i à l'instant t.

Entrainement

Avant la phase d'évaluation ou de décodage, les paramètres et la topologie d'un HMM doivent être préalablement déterminés. D'abord, grâce à une bonne connaissance du problème, la topologie du MMC est mise au point. Ensuite, les paramètres $\lambda = (\prod, A, B)$ sont estimés à l'aide d'un ensemble de données d'entrainement sous forme de plusieurs séquences d'observations. La technique la plus utilisée pour estimer ces paramètres est l'algorithme de Baum-Welsh. Le but de cet algorithme est de trouver un modèle λ qui maximise $P(O|\lambda)$. D'abord, on initialise aléatoirement le modèle λ. On utilise alors la fonction $Q(\lambda, \lambda')$ défini par :

$$Q(\lambda, \lambda') = \sum_S P(O, S|\lambda) \log(P(O, S|\lambda')) \qquad (19)$$

Cette fonction vérifie la condition suivante :

$$si\ Q(\lambda, \lambda') \geq Q(\lambda, \lambda)\ Alors\ P(O, S|\lambda') \geq P(O, S|\lambda) \qquad (20)$$

D'où, la maximisation de $Q(\lambda, \lambda')$ suivant λ implique la maximisation de $P(O, S|\lambda)$.

Pour un ensemble de R séquences d'observation supposées indépendantes, l'entrainement d'un MMC maximise le terme suivant :

$$L = \prod_{r=1}^{R} P(O^r, S|\lambda) \qquad (21)$$

Où O^r est la $r^{ième}$ séquence d'observations. A l'aide du terme $\alpha_t(i)$ définie précédemment et par l'introduction du terme $\beta_t(i) = P(o_{t+1},\ o_{t+2}, \dots, o_T|s_t = i, \lambda)$, la ré-estimation des paramètres du MMC discret sont calculés par :

$$\overline{a_{ij}} = \frac{\sum_{r=1}^{R} \frac{1}{P(O^r|\lambda)} \sum_{t=1}^{T} \sum_{i=1}^{N} \alpha_{t-1}^r(i) a_{ij} b_j(o_t) \beta_t^r(j)}{\sum_{r=1}^{R} \frac{1}{P(O^r|\lambda)} \sum_{t=1}^{T} \sum_{i=1}^{N} \alpha_{t-1}^r(i) \beta_t^r(j)} \qquad (22)$$

$$\overline{b_j}(k) = \frac{\sum_{r=1}^{R} \frac{1}{P(O^r|\lambda)} \sum_{t=1}^{T} \sum_{i=1}^{N} \delta(o_t^r = v_k) \alpha_{t-1}^r(i) a_{ij} b_j(o_t) \beta_t^r(j)}{\sum_{r=1}^{R} \frac{1}{P(O^r|\lambda)} \sum_{t=1}^{T} \sum_{i=1}^{N} \alpha_{t-1}^r(i) a_{ij} b_j \beta_t^r(j)} \qquad (23)$$

Avec

$$\delta(o_t^r = v_k) = \begin{cases} 1 & si\ o_t^r = v_k \\ 0 & sinon \end{cases}$$

II.5.c. k-plus proches voisins (k-PPV)

L'algorithme des k-PPV est l'un des algorithmes les plus utilisés en classification. Il est trivial, simple à implémenter et donne de bons résultats. Son principe est très simple ; pour un vecteurs caractéristique donné x, on cherche parmi les vecteurs d'entraînement les k vecteurs les plus proches de x, le vecteur x est alors affecté à la classe possédant plus de représentants parmi ces k plus proches. L'algorithme est le suivant :

Algorithme des k-PPV

---

1. *Parmi les vecteurs d'exemples d'entrainement, en utilisant une distance comme la distance euclidienne, choisir un k impaire puis identifier les k-plus proches voisins du vecteur x.*

2. *Parmi ces k plus proches voisins, k_i éléments appartient à la classe C_i.*
3. *x est affecté à la classe C_i correspondante à la plus grande valeur k_i.*

Dans le cas k=1, on parle du plus proche voisin, le vecteur x sera affecté à la classe possédant l'élément le plus proche de x. Le choix d'un k plus grand a l'avantage de lisser les limites de décisions, mais en contrepartie, il représente une grande complexité dans les calculs.

La figure (Figure 5) illustre cet algorithme avec k=11. Le vecteur inconnu x présenté au k plus proche voisins est affecté à la classe 1, car parmi les 11 plus proches, 6 appartiennent à la classe C_4, 3 à la classe C_3, 2 à la classe C_2 et aucun n'appartient à la classe C_1.

Dans cet algorithme, on remarque l'absence d'une phase d'entraînement, car tous les calculs sont faits lors de la phase de classification ; le vecteurs x est comparé à la totalité des données d'entraînement.

Malgré sa structure simple, en présence d'un nombre suffisant de données d'entraînement, l'algorithme des k-PPV est un algorithme de très bonne qualité et peut même être comparable aux autres techniques comme les réseaux Bayésiens [2] [11].

II.5.d. Réseaux de neurones

Le perceptron multicouche (PMC) et les fonctions à base radiale (RBF) sont les structures de réseaux de neurones les plus utilisées en classification supervisée, parmi une

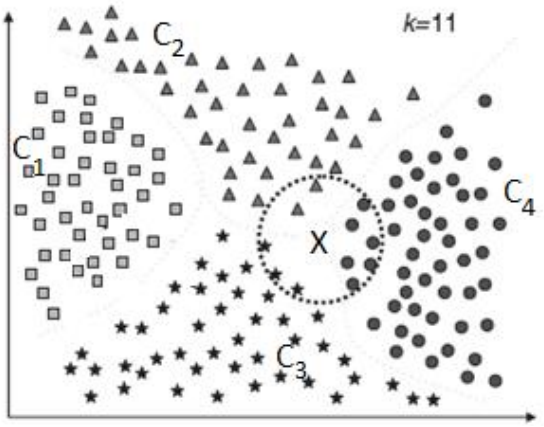


Figure 5: k-plus proches voisins (figure extraite de [2])

multitude de structures. Ces structures ont pu attirer une grande part d'intérêt dans la recherche et ont été utilisées dans une multitude d'applications dans divers domaines. Il a été aussi démontré que ces deux types de réseaux de neurones sont des approximateurs universels [12] [13] grâce à leurs habilité d'approximer n'importe quelle fonction de n'importe quelle dimension avec une précision arbitraire, à condition de leur présenter un nombre de données d'entraînement suffisant et adéquat et un bon choix de leur structure.

Modèle de neurones et réseaux

L'idée des réseaux de neurones artificiels est inspirée des réseaux de neurones biologiques. Un réseau de neurones biologique est un ensemble de neurones massivement interconnectés. L'unité principale d'un réseau de neurone est le neurone. Ce dernier se compose de quatre parties principales :

1. Les dendrites : Ils forment un maillage de récepteurs nerveux qui permettent d'acheminer des signaux électriques en provenance d'autres neurones vers le corps du neurone.
2. Le soma : Le corps de la cellule. Il agit comme une espèce d'intégrateur en accumulant des charges électriques. Lorsque le neurone devient suffisamment excité, il engendre un potentiel électrique qui se propage à travers son axone pour exciter d'autres neurones.
3. L'axone : Sert à transmettre les informations vers d'autres neurones ;
4. Les synapses : Une interface entre l'axone du neurone et les dendrites des autres neurones.

La fonction précise d'un réseau de neurones biologique est essentiellement déterminée par l'arrangement spatial des neurones et de leur axone, ainsi que la qualité des connexions synaptiques individuelles.

Un modèle d'un neurone artificiel est montré dans la figure ci-dessous (Figure 6). Il se compose essentiellement d'un ensemble d'entrées x_i muni d'un poids w_i pour chacune, une unité de calcul représentant le corps du neurone et une seule sortie.

La taille du vecteur d'entrée est souvent augmentée en ajoutant une autre entrée $x_0 = -1$ de poids b. Cette entrée supplémentaire s'appelle un biais. La sortie du neurone *j* est la somme

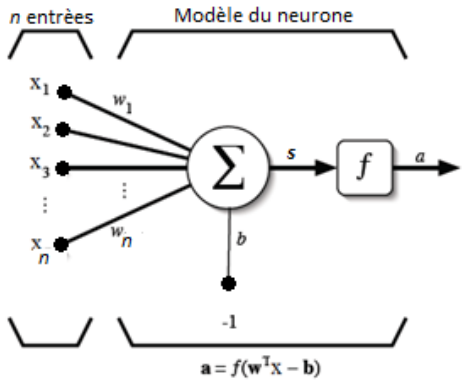


Figure 6: Structure d'un neurone artificiel

des x_i pondérés par les w_i qui alimente l'entrée d'une fonction d'activation *f*. Cette sortie est donnée par l'équation suivante :

$$a = f(s) = f(W^T X - b) = f\left(\sum_{i=0}^{n} w_{ji} x_i\right) \qquad (24)$$

Il existe plusieurs possibilités pour le choix de la fonction d'activation. Le tableau (Tableau 1) illustre un ensemble d'exemple des fonctions d'activation utilisables.

Comme son nom l'indique, la fonction seuil applique un seuil sur son entrée. Pour une entrée négative ne dépassant pas le seuil la fonction retourne la valeur 0. Alors que pour une entrée positive ou nulle, dépassant le seuil, la fonction retourne 1. Cette fonction est illustrée dans la figure (Figure 7(a)). Le biais dans l'équation (24) détermine l'emplacement du seuil où la fonction passe de 0 à 1. Dans le contexte des réseaux de neurones, cette fonction permet de prendre des décisions binaires (vraie ou faux).

La fonction linéaire est très simple, elle affecte directement son entrée à sa sortie. Cette fonction est illustrée à la figure (Figure 7(b)).

La fonction d'activation tangente hyperbolique a la même forme que la fonction sigmoïde, sauf que la fonction sigmoïde prend ses valeurs entre 0 et 1, alors que la tangente hyperbolique prend ses valeurs entre -1 et 1. Le paramètre β dans la fonction d'activation sigmoïde et la fonction d'activation tangente hyperbolique, détermine l'allure de ces deux

Tableau 1 : Fonctions d'activation

Nom de la fonction	Relation d'entrée/sortie	Icône	Nom Matlab
seuil	$a = 0 \quad \text{si } n < 0$ $a = 1 \quad \text{si } n \geq 0$		hardlim
seuil symétrique	$a = -1 \quad \text{si } n < 0$ $a = 1 \quad \text{si } n \geq 0$		hardlims
linéaire	$a = n$		purelin
linéaire saturée	$a = 0 \quad \text{si } n < 0$ $a = n \quad \text{si } 0 \leq n \leq 1$ $a = 1 \quad \text{si } n > 1$		satlin
linéaire saturée symétrique	$a = -1 \quad \text{si } n < -1$ $a = n \quad \text{si } -1 \leq n \leq 1$ $a = 1 \quad \text{si } n > 1$		satlins
linéaire positive	$a = 0 \quad \text{si } n < 0$ $a = n \quad \text{si } n \geq 0$		poslin
sigmoïde	$a = \frac{1}{1+\exp^{-n}}$		logsig
tangente hyperbolique	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
compétitive	$a = 1 \quad \text{si } n \text{ maximum}$ $a = 0 \quad \text{autrement}$	C	compet

fonctions en passant d'une fonction seuil pour β=1 à une fonction linéaire pour β=0 (Figure 7(c)).

Rosenblat a montré qu'un perceptron peut apprendre à produire n'importe quelle séparation linéaire, en utilisant un algorithme d'entraînement où les poids sont aléatoirement déterminés puis modifiés itérativement par l'équation suivante :

$$w(t+1) = w(t) + \Delta w \quad avec \;\; \Delta w = \eta \varepsilon_i x_i \qquad (25)$$

Où η est le taux d'apprentissage et ε_i l'erreur réalisé pour l'entrée x_i.

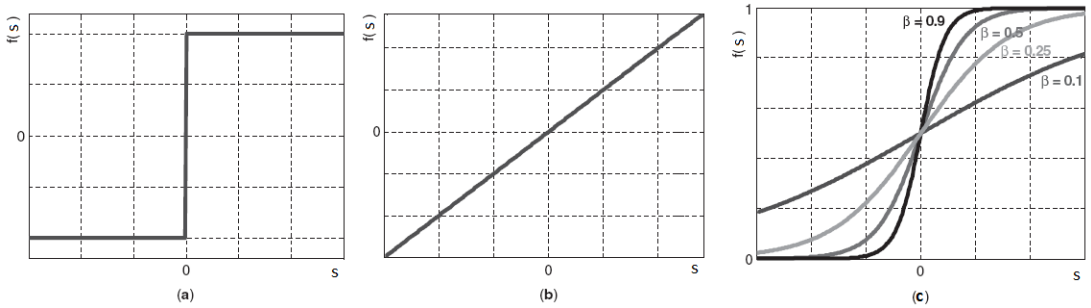


Figure 7 : Fonctions d'activation (a) seuil (b) linéaire (c) sigmoïde.

Un seul perceptron n'a pas de grande utilité puisque qu'il ne converge pas dans le cas des problèmes non linéairement séparables. En revanche, une combinaison appropriée de plusieurs perceptrons est d'une puissance considérable pour ce genre de problèmes. Une structure bien connue dans le domaine, appelée le perceptron multicouche (PMC), où les neurones sont structurés sous forme de couches et l'information passe d'une couche à l'autre jusqu'à ce qu'elle attient sa destination, a bien montré ses performances de grandes qualités.

Architecture d'un réseau de neurone

En général, un réseau de neurones est un maillage de plusieurs neurones organisés en couches. Chaque couche contient R neurones connectés aux n entrées. On dit alors que la couche est totalement connectée. Un poids w_{ij} est associé à chacune des connexions (Figure 9(a)). L'ensemble des poids d'une couche forme une matrice W de dimension $S \times R$:

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{R1} & \cdots & w_{Sn} \end{bmatrix} \qquad (26)$$

Dans cette matrice, le premier indice désigne le numéro de neurone sur la couche, alors que le deuxième indice désigne le numéro de l'entrée. Ainsi, w_{ij} spécifie le poids de la connexion qui relie le neurone i à son entrée j. Si de plus on prend $b = [b_1, b_2 \dots b_R]^T$, s= $[s_1, s_2 \dots s_R]^T$ et $a = [a_1, a_2 \dots a_R]^T$, alors nous obtiendrons une représentation simplifiée plus compacte illustrée dans la figure ci-dessous (Figure 9).

Pour construire un réseau de neurone, il suffit de combiner plusieurs couches comme montre la figure (Figure 8). Dans le cas général $R^1 \neq R^2 \neq R^3$ et chaque couche possède sa propre matrice de poids W^k où k est l'indice de la couche. Ainsi que ces vecteurs propres $b^k, n^k \; et \; a^k$.

En théorie, on peut enfiler autant de couches que l'on veut. Pour chaque couche (sauf celle d'entré et de sortie), on peut fixer un nombre quelconque de neurones. En pratique, il n'est pas souhaitable d'utiliser trop de neurones.

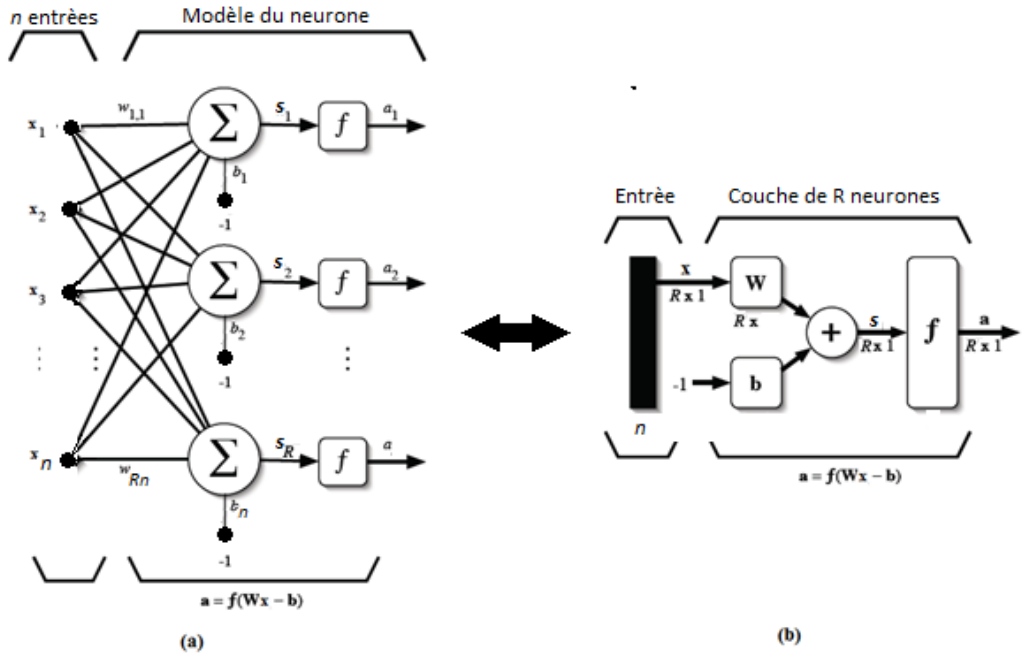


Figure 9: Une couche d'un réseau de neurones (a) et sa représentation matricielle (b)

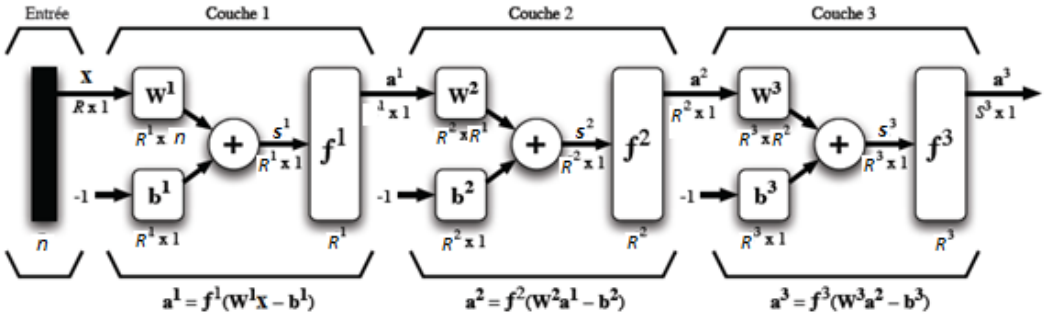


Figure 8: Représentation matricielle d'un perceptron multicouche

Les réseaux multicouches sont plus puissants que les réseaux simples à une seule couche. Avec une couche cachée munie d'une fonction d'activation sigmoïde, on peut entraîner un réseau à produire une approximation de la plupart des fonctions, avec une précision arbitraire. Sauf dans des cas précis, les réseaux de neurones artificiels exploitent plus qu'une couche cachée.

Entraîner un réseau de neurones consiste à modifier les valeurs de ses poids et de ses biais pour qu'il réalise la fonction entrées/sorties désirée. Dans la littérature, il existe plusieurs

algorithmes d'entraînement [14], comme l'algorithme de la descente du gradient, la méthode de quasi-newton, les algorithmes génétiques, etc.

Le nombre de couches et le nombre de neurones par couche dépend du problème que l'on veut résoudre. Par exemple, si dans un problème, le nombre d'entrées est *5* et le nombre de sortie est *3*, alors $n=5$ et $R^m = 3$, où m est l'indice de la couche de sortie. Pour le choix de la fonction d'activation, si l'on désire produire des sorties binaires, alors une fonction seuil pour la couche de sortie sera souhaitable.

Perceptron multicouche (PMC)

Un perceptron multicouche est un réseau de neurone de type propagation vers l'avant (feed-forward). Ça signifie que l'information circule dans un seul sens, de l'entrée vers la sortie. L'apprentissage d'un PMC est de type supervisé par correction des erreurs. Le signal d'erreur est rétro-propagé vers les entrées afin de mettre à jour les poids des synaptiques.

- ***Perceptron simple***

Un perceptron simple est illustré dans la figure ci-dessous (Figure 10). Il s'agit d'une seule couche de S neurones totalement connectés sur le vecteur entrée x de dimension $n \times 1$. La fonction d'activation utilisée pour chaque neurone est la fonction seuil. La sortie obtenue est alors le vecteur de dimension $R \times 1$, $a = [a_1, a_2 \dots a_R]^T$ avec :

$$a_i = \begin{cases} +1 \; si \; s_i \geq 0 \\ -1 \quad sinon \end{cases} \qquad (27)$$

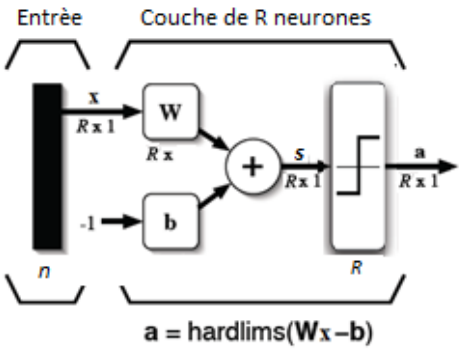


Figure 10: Représentation matricielle d'un perceptron simple

Avec cette structure, chaque neurone possède son propre vecteur de poids et son propre biais. On se retrouve alors avec *R* frontières de décisions linéaires, chacune découpe l'espace d'entrée en deux régions. Par conséquence, chaque neurone d'un perceptron simple permet de résoudre un problème de classification à deux classes linéairement séparables. A condition de bien déterminer les poids et le biais.

Pour déterminer les poids et le biais, on commence par leur affecter des valeurs aléatoires, puis pour chaque exemple d'entrée/sortie (x, d), on effectue la mise à jour suivante :

$$W(t+1) = W(t) + \Delta W \quad (28)$$

$$\Delta W = ex^T \quad (29)$$

$$b(t+1) = b(t) + \Delta b \quad (30)$$

$$\Delta b = -e \quad (31)$$

Où $e = d - a$ est le vecteur d'erreurs que l'on observe en sortie pour le stimulus *x* et *d* la sortie désirée.

Il a été démontré que cette méthode converge toujours vers une solution en un nombre fini d'itérations si certaines hypothèses sont vérifiées :

1. Le problème est linéairement séparable ;
2. Les poids ne sont mis à jour que lorsqu'un stimulus d'entrée est mal classé ;
3. Il existe une borne supérieure sur la norme des vecteurs de poids ;

- ***Réseau ADALINE***

Une autre alternative d'un perceptron simple consiste à utiliser des fonctions d'activation linéaire (Figure 11). Ce réseau s'appelle un réseau ADALINE (Adaptive Linear Neuron) [15]. Ce réseau souffre des mêmes limitations que celles d'un perceptron simple. Cependant son algorithme d'apprentissage est beaucoup plus puissant. Cet algorithme s'appelle la règle LMS (Least Mean Square) [16]. Cette règle a l'avantage de minimiser l'erreur quadratique moyenne

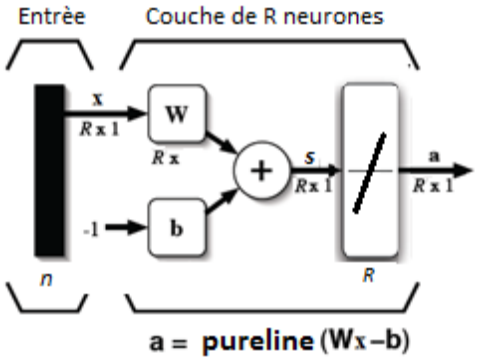


Figure 11: Représentation matricielle d'un réseau ADALINE

de sorte que la frontière de décision a tendance à se retrouver aussi loin que possible des exemples.

Pour la règle LMS, nommée aussi la règle de Windrow-Hoff, la mise à jour à appliquer est la suivante :

$$\Delta W = 2\eta e(t)x^T \quad (32)$$

$$\Delta b = -2\eta e(t) \quad (33)$$

Où η est le taux d'apprentissage. Si ce paramètre est très restreint, la méthode de LMS garanti la convergence vers un minimum global. En pratique, on fixe η le plus grand possible pour converger le plus rapidement possible. En voulant aller trop vite, les pas sont plus grands mais peuvent avoir tendance à osciller.

- ***Perceptron multicouche***

La première couche s'appelle la couche d'entrée constituée de n+1 neurones, où chaque neurone possède une entrée qui correspond à un élément du vecteur caractéristique de taille n. Le nœud restant possède une entrée de valeur égale à *-1*. La fonction d'activation des neurones de la couche d'entrée est l'identité. Ainsi, cette couche ne fait aucun traitement sur les données. Elle ne fait que les présenter à la couche suivante. Les autres couches (sauf la dernière couche) qui suivent la couche d'entrée s'appellent les couches cachées. Il a été montré qu'avec une seule couche cachée, un PMC peut produire des classes de n'importe quelle forme. Le nombre de neurones (nœuds) dans ces couches peut varier d'une couche à une autre et chaque nœud dans

une couche est totalement connecté aux nœuds de la couche précédente à travers des poids W_{ij} et aux neurones de la couche suivante à travers des poids W_{jk}. Dans les neurones des couches cachées on utilise des fonctions d'activation non-linéaire, généralement des fonctions d'activation de type sigmoïde. La dernière couche s'appelle la couche de sortie, elle est constituée d'un nombre de nœud exactement égal à celui des classes existantes dans le problème de classification.

- ***Rétro-propagation des erreurs***

Après avoir choisi le nombre de couches cachées et le nombre de neurones dans chaque couche ainsi que les fonctions d'activation, la tâche suivante consiste à ajuster les poids W en minimisant le critère suivant :

$$J(W) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{c}(d_{ik} - a_{ik})^2 \qquad (34)$$

Où *W* est la totalité des poids dans le réseau, N le nombre des données d'entraînement, c le nombre de classes, d_{ik} le k^{ème} élément de la sortie désiré, a_{ik} la sortie du k^{ème} neurone de la sortie du réseau.

Un algorithme bien connu qui permet de réaliser cette minimisation, appelé la rétro-propagation des erreurs. Dans cet algorithme la mise à jour des poids est calculée comme suivant :

$$w(t+1) = w(t) + \Delta w \; avec \; \Delta w = -\eta \frac{\partial J(W)}{\partial W} \qquad (35)$$

L'algorithme de la rétro-propagation des erreurs commence par une initialisation des poids du réseau. Ensuite, on calcule la sortie du réseau pour un élément des données d'entraînement. Puisqu'on connait la sortie désirée, la mise à jour des poids de sortie sont alors facilement calculées à l'aide de l'équation suivante :

$$\Delta w_{kj} = \eta \delta_k x_j = \eta(d_k - \mathrm{a}_k) f'(s_k)\mathrm{x}_j \qquad (36)$$

Le terme

$$\delta_k = (d_k - a_k) f'(s_k) \quad (37)$$

est appelé la sensitivité du neurone k.

En utilisant la fonction sigmoïde comme une fonction d'activation nous avons :

$$f'(s_k) = f(s_k).\left(1 - f(s_k)\right) \quad (38)$$

Pour la mise à jour des poids des neurones des couches cachées on procède de la même façon. Le problème avec les couches cachées est qu'on ne connaît pas les sorties désirées. On fait alors une rétro-propagation des erreurs de la couche de sortie vers la couche cachée de la manière suivante :

$$\Delta w_{ji} = -\eta \frac{\partial J}{\partial W_{ji}} = \eta \left[\sum_{k=1}^{c} \delta_k w_{kj}\right] f'(s_j) x_i \quad (39)$$

Le terme

$$\delta_j = \left[\sum_{k=1}^{c} \delta_k w_{kj}\right] f'(s_j) \quad (40)$$

est la sensitivité du neurone *j* de la couche cachée.

L'équation (32) devient :

$$\Delta w_{ji} = \eta . \delta_j . x_i \quad (41)$$

Cartes auto-organisatrices

Les cartes auto-organisatrices (Self Organizing Map SOM), ont été développées par Teuvo Kohonen [17] vers les années 80. Ce réseau est un cas spécial des réseaux de neurones, son apprentissage est compétitif non supervisé.

Dans les réseaux SOM, on trouve une couche d'entrée et une couche de sortie sans aucune couche cachée. Les neurones de la couche de sortie sont arrangés dans une carte à une

ou deux dimensions (ou même plus) afin de définir des relations de voisinage entre les neurones. La figure (Figure 12) montre une forme carrée de voisinage qui est la plus souvent utilisée.

L'algorithme d'apprentissage d'un réseau de Kohonen est de type compétitif [18]. Il commence d'abord par une initialisation aléatoire des poids synaptiques. Après, l'algorithme se déroule en trois étapes essentielles ; la compétition, la coopération et enfin l'adaptation des poids synaptiques.

- ***Compétition***

Le but dans cette étape est de déterminer dans la carte le neurone gagnant, pour une entrée x à un instant t. Soit $x = (x_1, x_2, \dots, x_n)$ le vecteur présenté à l'entrée du réseau et $w_j = (w_{j1}, w_{j2}, \dots, w_{jn}), j = 1,2, \dots, N$, les poids synaptiques entre l'entrée x et le neurone j. L'indice du neurone gagnant est alors déterminé par l'équation suivante :

$$i(x) = \underset{j}{\operatorname{argmax}}(w_j - x) \qquad (42)$$

Cette équation montre que l'indice du neurone gagnant est celui du vecteur des poids le plus proche du vecteur x.

- ***Coopération***

En neurobiologie, le neurone actif (gagnant) a tendance à exciter ses neurones voisins. De la même façon, dans les cartes auto-organisatrices, on définit une fonction de voisinage

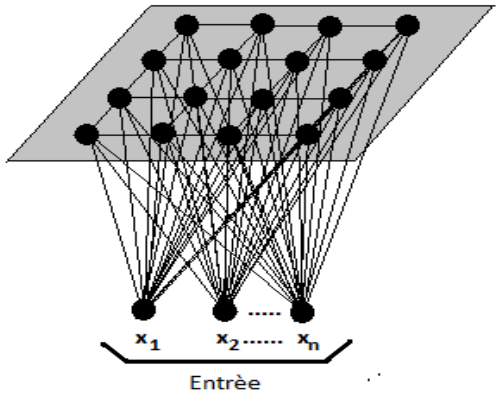


Figure 12: Structure d'une SOM

$h_{j,i}(t)$ autour du neurone gagnant d'indice j (Figure 13). Cette fonction est souvent de forme gaussienne et satisfait les conditions suivantes :

1. $h_{j,j}(t) = 1 \; \forall j, \forall t$ est symétrique.
2. $h_{j,i}(t) < h_{j,k}(t) \; si \; d_{ji} < d_{jk}$, où d_{ji} est la distance entre neurone d'indice j et le neurone d'indice i.
3. $h_{j,i}(t) > h_{j,i}(t') \; pour \; t' > t$

Une fonction typique qui satisfait ces conditions peut être définie comme suit :

$$h_{j,i}(t) = e^{\left(\frac{d_{i,j}^2}{2\sigma^2(t)}\right)} \qquad (43)$$

avec

$$\sigma(t) = \sigma_0 e^{\left(\frac{-t}{\tau}\right)} \qquad (44)$$

Où τ est une constante de temps.

La figure (Figure 14) montre une carte à deux dimensions avec le neurone gagnant d'indice 13, les poids des neurones immédiatement voisins (8, 12, 18, 14) sont les plus affectés lors de l'étape de modification des poids, ensuite les neurones (7,17, 19, 9) puis les neurones (3, 11, 23, 15). Les poids des neurones les plus loin restent intacts.

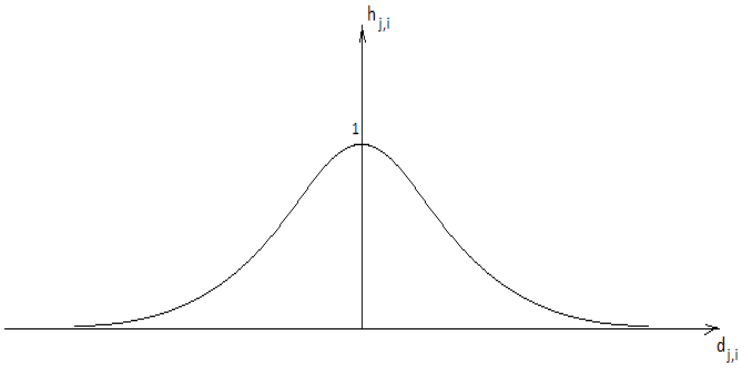


Figure 13: Fonction topologique

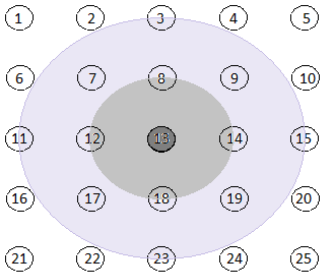


Figure 14 : Topologie de voisinage, les neurones les plus proche du neurone gagnant (neurone d'indice 13) sont plus excités

- ***Adaptation des poids synaptiques***

Par définition, pour entrainer un réseau de neurones d'une manière non supervisée, il faut que la modification des poids soit en relation avec le vecteur d'entrée. Cette modification est décrite dans l'équation suivante :

$$w_j(t+1) = w_j(t) + \eta(t) h_{j,i}(t)(x - w_j(t)) \qquad (45)$$

Où $\eta(t)$ est le taux d'apprentissage. $h_{j,i}(t)$ est la fonction de voisinage.

Algorithme: Cartes auto-organisatrices

---

1. *Initialiser aléatoirement les poids synaptiques.*
2. *Répéter T fois*
    a. *Choisir aléatoirement un exemple x(t).*
    b. *Déterminer le neurone gagnant à l'aide de l'équation* (42)
    c. *Mettre à jour les poids synaptiques en utilisant l'équation* (45)
    d. *t=t+1 et revenir à l'étape 2*

Taux d'apprentissage

Le taux d'apprentissage détermine le pas effectué à chaque étape lors de la recherche de l'erreur minimale. En théorie, le taux d'apprentissage affect uniquement le temps de convergence. En pratique, des grandes valeurs de ce taux peuvent impliquer une divergence du système car on peut sauter les minimums. Un choix de petits taux impliquera des temps de convergences trop longues. Un choix typique du taux de convergence est souvent situé dans l'intervalle [0 1].

La forme exacte du taux d'apprentissage n'a pas d'importance, elle peut être linéaire, exponentielle ou inversement proportionnelle du temps. En particulier, elle doit avoir une valeur initiale η_0 et décroit avec le temps. Ces critères peuvent être satisfaits par l'expression suivante :

$$\eta(t) = \eta_0 e^{\left(\frac{k}{\tau}\right)} \qquad (46)$$

Où τ est une constante du temps.

II.5.e. Machines à vecteurs supports (SVM)

Les machines à vecteurs supports sont des méthodes de classification binaire, introduites par Vladimir Vapnik [19]. L'idée est de trouver parmi le nombre infini des hyperplans séparateurs de deux classes, l'hyperplan qui maximise la marge entre ces deux classes (Figure 15). L'équation de cet hyperplans est donnée par :

$$W.x + b = 0 \qquad (47)$$

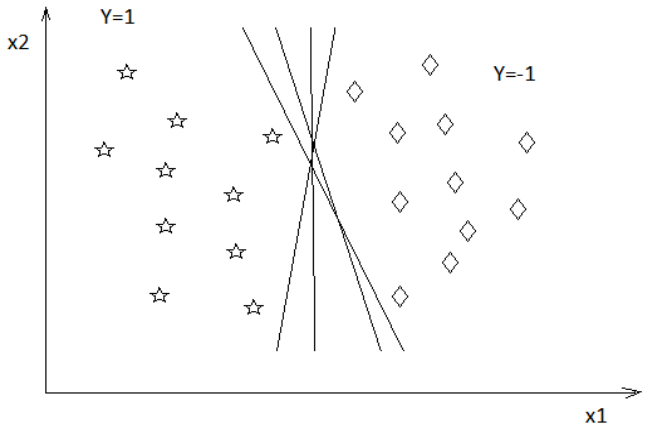


Figure 15: Infinité d'hyperplans separateurs

A la présence d'un ensemble d'exemples (x_i, y_i) Le but est donc de :

1. Maximiser la marge : $2/\|W\|$
2. Garder une séparation correcte, autrement dit :

   Pour les x positifs (y=1) : W.x+b≥1

   Pour les x négatifs (y=-1) : W.x+b≤ -1

Ces conditions peuvent être formulées sous forme du problème d'optimisation quadratique suivant :

$$\begin{cases} \min\limits_{W,b} \dfrac{\|W\|^2}{2} \\ y_i(W.x_i + b) \geq 1 \end{cases} \quad (48)$$

La solution de ce problème est donnée par :

$$W = \sum_i \alpha_i y_i x_i \quad (49)$$

Les α_i sont des poids calculés par entraînement, les x_i possédants des poids non nuls s'appellent les vecteurs supports (Figure 16).

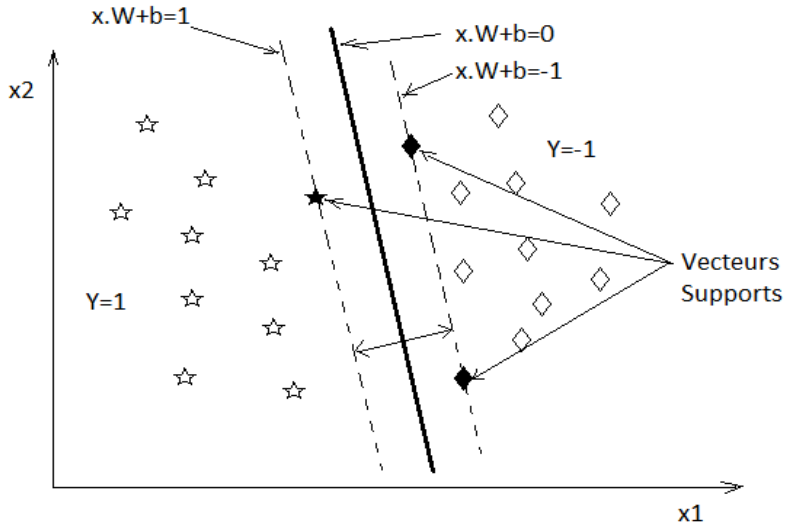


Figure 16: Les vecteurs supports

Pour les vecteurs supports nous avons :

$$b = y_i - Wx_i \quad (50)$$

En utilisant les équations (47) et (49), L'équation de l'hyperplan séparateur est :

$$W.x + b = \sum_i \alpha_i y_i x_i \,.x + b = 0 \quad (51)$$

On remarque alors, que pour classer un vecteur inconnu, il suffit de calculer son produit scalaire avec les vecteurs supports.

Noyau(Kernel)

Si les exemples d'entraînement ne sont pas linéairement séparables, il suffit de passer à un autre espace de dimension plus grande à l'aide d'une transformation, par exemple $\varphi(x) = (x, x^2)$. Dans ce nouvel espace, l'équation de l'hyperplan est donnée par :

$$\sum_i \alpha_i y_i \varphi(x_i).\varphi(x) + b = 0 \quad (52)$$

Au lieu de définir la transformation, il suffit de choisir une fonction *K(x,y)* et supposer qu'il existe une transformation φ telle que :

$$k(x, y) = \varphi(x).\varphi(y) \quad (53)$$

Cette fonction s'appelle un Noyau (Kernel). On n'est pas alors supposé connaitre le nouvel espace, on s'intéresse uniquement au produit scalaire dans cet espace représenté par le Noyau *K*. Pour qu'une fonction *K(x,y)* soit un noyau valide, elle doit vérifier certaines conditions, nommées les conditions de Mercer.

Exemple (Gaussian Kernel)

La particularité de ce kernel est qu'il définit un produit scalaire dans un espace de dimension infinie. Il est défini par l'équation suivante :

$$k(x,y) = exp\left(\frac{\|x-y\|^2}{\sigma^2}\right) \quad (54)$$

Cas multi-classes:

Comme nous l'avons déjà mentionné, les machines à vecteurs supports sont réalisables uniquement pour des problèmes de classification binaires (y=1, y=-1). Pour pouvoir utiliser cette méthode dans des problèmes multi-calasses (N classes), on utilise une combinaison de plusieurs SVM binaires. Dans ce sens deux stratégies sont envisageables :

1. Un contre tous : Elle consiste à entrainer N SVM, une classe contre les autres. Lors de la reconnaissance, un vecteur inconnu est affecté à la classe associée à la SVM qui répond le mieux.
2. Un contre un : on entraine une SVM pour chaque paire de classes. Lors de la reconnaissance, un vecteur inconnu est affecté à la classe identifiée par la majorité des SVM binaires.

II.5.f. Combinaison des classifieurs

A partir des années 1990, la combinaison des classifieurs est devenue un important axe de recherche [1] [20] [21] [22]. Plusieurs méthodes de combinaison ont été proposées, et l'application de ces méthodes dans différents problèmes concrets a prouvé l'efficacité de cette approche. Rahman [23] a catégorisé ces méthodes en méthodes parallèles et méthodes séquentielles. Les méthodes parallèles sont souvent utilisées pour améliorer la précision du classifieur, alors que les méthodes séquentielles accélèrent la classification dans des problèmes caractérisés par un nombre de classes considérablement élevé. Pour une étude plus rigoureuse du problème de combinaison des classifieurs le lecteur est invité à lire l'ouvrage de Kuncheva [24].

Formulation du problème

Soit un problème de classification en N classe $C_i, i \in \Lambda = \{1, 2, \dots, N\}$. Pour un vecteur inconnu x, un classifieur (noté e) affect un index $j \in \Lambda \cup \{N+1\}$, si $j = N+1$ alors le classifieur e n'a pas arrivé à classer ce vecteur sinon $x \in C_j$. Sans tenir compte de la structure

interne d'un classifieur *e*, le type d'information qu'il fournit en sortie (*e(x)=y*), peut-être catégoriser selon trois niveaux [20]:

1. Niveau abstraction (Abstract level): La sortie du classifieur *e* est l'unique étiquette de la classe de *x*.
2. Niveau ordonnancement (Rank level): La sortie de *e* est l'ordre d'appartenance de *x* aux différentes classes.
3. Niveau mesure (Measurment level): La sortie de *e* est le degré (ou la probabilité) d'appartenance de *x* aux différentes classes.

Il est évident que le niveau mesure offre plus d'information de classification que les deux autres niveaux. A partir des degrés d'appartenance on peut obtenir l'ordre d'appartenance aux différentes classes et finalement aboutir au niveau abstraction. Ainsi, la combinaison des classifieurs dépend fortement du niveau d'abstraction offert en sortie par ces classifieurs. Le fait que le troisième niveau est plus riche en information que les deux autres, les classifieurs supportant ce type de sortie sont souvent placés comme des processus de classification intermédiaires [20]. Ainsi, le problème de combinaison des classifieurs peut être traité selon trois méthodes:

1. Pour un ensemble de *K* classifieurs $e_{k=1,2,\dots,K}$ du premier niveau, et pour un vecteur inconnu *x* chaque e_k propose une sortie $e_k(x) = j_k$. Il faut alors utiliser ces sorties pour construire un classifieur E qui affect à *x* une classe $\mathrm{j} \in \Lambda \cup \{\mathrm{N}+1\}$.
2. Pour des classifieurs du second niveau, chaque e_k propose un sous ensemble ordonné $e_k(x) = L_k \subseteq \Lambda$. Le but est construire un classifieur E de tel sorte que $E(x) = j \in \Lambda \cup \{N+1\}$.
3. Pour des classifieurs du troisième niveau, chaque e_k propose un vecteur $v_k = (v_k(1), v_k(2), \dots, v_k(N))^T$. A partir de ces *K* vecteurs il faut construire un classifier E de tel sorte que $E(x) = j \in \Lambda \cup \{N+1\}$.

Les méthodes du premier type sont plus génériques, car les classifieurs du second et troisième niveau peuvent être considéré comme des classifieurs du premier niveau. Alors que ceux du troisième types nécessites l'utilisation des classifieurs du troisième niveau uniquement.

Dans la suite de cette partie nous présentons quelques méthodes de combinaison rencontrées dans la littérature.

Méthode du Vote majoritaire

Cette méthode est applicable pour une combinaison des classifieurs du premier niveau. Soit $d_k = (d_{k1}\ ,\ d_{k2}, \dots, d_{kN})$ avec $d_{ki} = 1$ si $e_k(x) = i$ et $d_{ki} = 0$ si $e_k(x) \neq i$, Les vote pour chaque classe sont donnés par la formule suivante:

$$y_i = \sum_{k=1}^{K} d_{ki} \ \ avec\ i = 1, \dots, N \qquad (55)$$

Le vecteur *x* est alors affecté à la classe *i* si $\max_i y_i \geq\ K/2$, sinon il sera rejeter (la classe N+1).

Méthode de la moyenne

Cette méthode est applicable pour la combinaison des classifieurs du troisième niveau. Soit $v_k = (v_k(1), v_k(2), \dots, v_k(N))^T$ la sortie d'un classifieur ($e_k(x) = v_k$). Alors la sortie de la combinaison $E(x) = (E_1(x), \dots, E_N(x))$ est donnée par la formule suivante:

$$E_i(x) = \frac{1}{K}\sum_{k=1}^{K} v_k(i) \ \ avec\ i = 1, \dots, N \qquad (56)$$

Le vecteur *x* est de classe *j* si :

$$j = \operatorname{argmax}_i E_i(x) \ \ avec\ i = 1, \dots, N \qquad (57)$$

Méthode de Combinaison pondérée

Dans cette méthode, le score y_i généré par la combinaison des classifieurs est une moyenne pondérée des scores $v_k(i)$ des *K* classifieurs :

$$y_i = \sum_{k=1}^{K} w_{ki} v_k(i) \ \ avec\ i = 1, \dots, N \qquad (58)$$

$$\sum_{k=1}^{K} w_{ki} = 1.$$

Dans certains cas on peut utiliser des poids indépendants des classes : $w_{ki} = w_k$.

Plusieurs méthodes ont été proposées pour la détermination de ces poids [1] à l'aide d'un ensemble d'exemples de validation. Avec un choix adéquat de ces poids, cette méthode aboutis à des résultats meilleur que la méthode de la moyenne [1].

Méthode du Boosting

Boosting est une méthode générale pour l'amélioration des performances de classification par une combinaison séquentielle de plusieurs classifieurs. Dans cette méthode, le premier classifieur est entrainé avec les exemples originaux, le classifieur suivant est entrainés avec les mêmes exemples pondérés selon les résultats de classification obtenus par le classifieur qui le précède, de telle sorte que les exemples mal classés possèdent des poids plus importants. Pour la détermination de ces poids, un premier algorithme nommé AdaBoost, a été proposé par Freund et Schapire [25]. Cet algorithme est opérationnel uniquement dans le cas des problèmes de classification binaires. Une variante multi-classes de cet algorithme nommé AdaBoost.M1, a été proposée par les mêmes auteurs [26]. Cet algorithme est présenté dans le listing suivant :

Algorithme (AdaBoost.M1)

---

1. Exemples d'entrainement

$$(x_i, y_i)_{i=1\ldots M}\ ,\ y_i \in \Lambda = \{1, \ldots, N\}$$

2. Initialisation des poids

$$D_1(i) = \frac{1}{M}, i = 1 \ldots M$$

3. Pour $k = 1 \ldots K$

- *Entrainer le classifieur* e_k *avec les exemples pondérés par* D_k
- *Calculer l'erreur des poids par:*

$$\varepsilon_k = \sum_{i:\, e_k(x_i) \neq y_i} D_k(i)$$

- *Si* $\varepsilon_k > 0.5$ *on sort de la boucle.*
- *Choisir* $\beta_k = \frac{\varepsilon_k}{1-\varepsilon_k}$
- *Mise à jour des poids:*

$$D_{k+1}(i) = \frac{D_k(i)}{Z_k} \cdot \begin{cases} \beta_k & si \;\; e_k(x_i) = y_i \\ 1 & sinon \end{cases}$$

Où Z_k est un facteur de normalisation.

4. Sortie finale du classifieur:

$$E(x) = \underset{y \in \Lambda}{argmax} \sum_{k:\, e_k(x)=y} log \frac{1}{\beta_k}$$

III. ÉVALUATION DES PERFORMANCES

III.1. Evaluation

Après la mise au point du classifieur et son entraînement, il faut évaluer ses performances de généralisation à l'aide d'un autre ensemble de données non utilisées lors de la phase d'entraînement.

Pour cela, l'une des méthodes la plus simple et la plus utilisée est de subdiviser les données d'entraînement en deux parties ; la première pour l'entraînement et la seconde pour l'évaluation. Cette méthode présente quelques points faibles, à savoir :

1. Après la subdivision de l'ensemble d'exemples, il se peut que les données significatives soient présentes dans une partition et absentes dans une autre.
2. Le taux d'exemples utilisés pour le test peut affecter énormément le taux de reconnaissance. Si ce taux est grand, le taux d'exemple utilisé pour l'entraînement sera petit, ce qui entrainera un faible taux de reconnaissance.

La validation croisée vient pour remédier à ces problèmes. Dans cet approche les données sont subdivisées en k (k>2) portions. Parmi ces k portions, une est utilisée pour la validation, les autres k-1 portions pour l'entraînement. Cette procédure est répétée k fois, à chaque fois on utilise une différente portion des données pour la validation. Le taux reconnaissance est alors la moyenne de taux reconnaissance obtenu à chaque étape (Figure 17). Cette approche présente l'avantage d'utiliser tous les exemples pour l'entraînement et pour la validation mais pas en même temps.

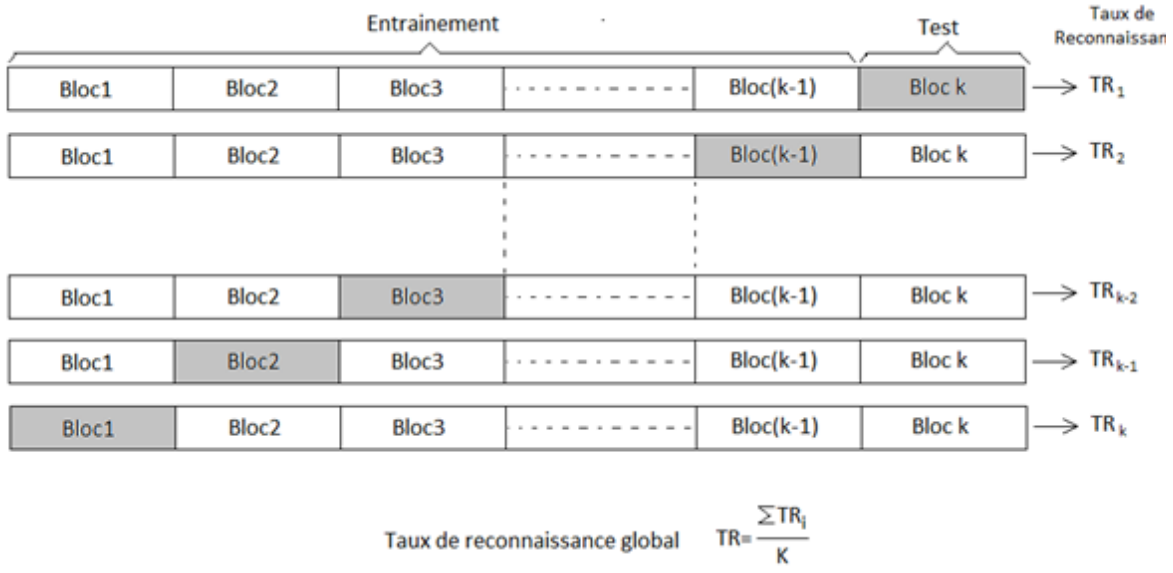


Figure 17: Validation croisée

Le choix du paramètre k dépend de la taille des données. Si k est très petit, le nombre des partitions sera réduit, par conséquence on n'aura pas assez de données pour l'entraînement. Si k est très grand, le nombre de partition est alors augmenté ce qui engendre un large nombre de données d'entraînement mais un nombre réduit de données de test. Généralement, pour un ensemble de données suffisamment large, k est choisi entre 5 et 10 [2]. Autrement, k peut être plus grand pour garantir suffisamment de données pour l'entraînement.

III.2. Comparaison des classifieurs

Dans une étude publiée par Liu et al [21], dans le cadre de reconnaissance automatique de caractères, une comparaison des performances de deux grandes familles de classifieurs a été présentée ; les classifieurs statistiques comme l'algorithme des K-PPV, les réseaux Bayésiens, les MMCs et les classifieurs discriminants comme les réseaux de neurones, les SVM et les RBF. Cette comparaison s'articule autour de quatre critères: complexité et flexibilité d'entrainement ; précision de classification ; complexité temporelle et spatiale ; Capacité de rejet.

1. **Complexité et flexibilité d'entrainement**: Le temps d'entrainement des classifieurs statistiques est linéaire avec le nombre des classes. L'ajout d'une nouvelle classe ou bien du nouvel ensemble d'exemple d'entraînement se fait d'une façon simple et directe. Pour la famille des classifieurs discriminants, le temps d'entrainement est proportionnel au

carrée du nombre des classes et l'ajout d'une nouvelle classe ou un nouvel ensemble d'entrainement nécessite un réentrainement avec la totalité des exemples.

2. **Précision de classification**: Souvent, avec un ensemble d'exemple d'entrainement suffisant, les classifieurs discriminant sont plus performent que les classifieurs statistique. Mais, avec un ensemble d'exemple réduit, les classifieurs statistiques sont plus performants.
3. **Complexité temporelle et spatiale**: Les classifieurs discriminants possèdent peut de paramètres. Ainsi, ces classifieurs sont moins exigeant en mémoire et en temps d'exécution.
4. **Capacité de rejet**: Les classifieurs discriminants sont mieux placés pour le rejet des entrées ambigües, mais pas pour le rejet des intrus. Alors que les classifieurs statistiques sont favorisés pour le rejet des intrus.

Plusieurs travaux ont traité le problème de comparaison des performances des classifieurs, soit d'une façon théorique [27], soit empirique [28] ou bien dans le cadre d'un problème particulier [21] [29] [30] [31]. La suite de cette partie expose quelques résultats comparatifs dans divers domaines.

Pour différentes valeurs des paramètres des différents algorithmes dans différents problèmes de classification et avec différentes bases d'entrainement, Caruana et al. [28] ont obtenu un taux de reconnaissance moyen de 80.9% pour l'algorithme des K-PPV, 84.2% par les RNA, 65.4% par les réseaux Bayésiens et 85.2% par les SVM.

Liu et al. [29] ont comparé les performances de plusieurs algorithmes dans le cadre de la reconnaissance des chiffres en utilisant différentes base d'entrainement. Les taux d'erreur obtenus sont listés dans (Tableau 2). Dans le même contexte, Russell et al. [31], en utilisant la base NIST, ont obtenus un taux d'erreur de 2.4% par les K-PPV (avec k=3), 1.6% par un PMC avec une seule couche cachée et 1.1% par les SVM.

Ainsi, la comparaison entre différents algorithmes de classification reste un problème d'une grande importance, car leurs performances change selon le problème et selon la base d'entrainement.

Tableau 2: Taux d'erreur en (%) pour différents Algorithmes dans des différentes base

	K-PPV	PMC	RBF	LQV	SVM-poly	SVM-rbf
CENPARMI	*2.630*	*2.015*	*1.780*	*2.00*	*1.510*	*1.235*
CEDAR	*1.310*	*1.052*	*1.079*	*1.106*	*0.871*	*0.781*
MNIST	*1.597*	*1.065*	*0.968*	*1.285*	*0.90*	*0.823*

IV. CONCLUSION

Dans ce chapitre, nous avons décrit les différentes composantes d'un système de reconnaissance de formes. La grande partie de cette présentation a abordé la description d'un ensemble d'algorithmes de classification. Nous avons donné une vue générale sur les classifieurs de Bayes, les MMCs, l'algorithme du k-PPV, les réseaux de neurones et les SVM. Nous avons aussi présenté quelques techniques de combinaison des classifieurs. Le test, la validation et la comparaison des performances des classifieurs sont d'une grande importance. De ce fait, nous avons consacré la dernière partie de ce chapitre à la description des méthodes de test et de validation les plus utilisées dans le domaine ainsi qu'une brève comparaison des performances des différents classifieurs.

Nous n'avons cité dans ce chapitre que les algorithmes de classification, les plus utilisés dans la littérature. En réalité il existe d'autres algorithmes et méthodes de classification. Le chapitre suivant sera entièrement consacré à la description des techniques de classification basées sur la logique floue.

CHAPITRE 2. LOGIQUE FLOUE POUR LA RECONNAISSANCE DE FORMES

I. INTRODUCTION

La logique floue est introduite dans plusieurs domaines d'applications ; en modélisation floue, contrôle flou, data mining, traitement d'image et en reconnaissance de formes [32].

En reconnaissance de formes (classification), la logique floue est devenue une partie indispensable. Cette forte présence est expliquée par le fait que cette logique peut décrire convenablement la structure des données, simplifier la communication avec l'utilisateur et peut conduire à une modélisation par block de façon intuitive. Le concept des sous-ensembles flous est très attractif par son habilité de qualifier l'appartenance partielle d'un élément à un ensemble. Ce concept est de grande importance dans plusieurs problèmes de classification rencontrés dans notre quotidien [32].

La logique floue a été d'abord proposée pour représenter des données incertaines [33]. Dans la théorie des ensembles classiques, un élément appartient ou non à un ensemble donné. Par contre, dans la logique floue, un élément peut appartenir à plusieurs ensembles, mais avec un degré d'appartenance à chaque ensemble. Ceci est mathématiquement représenté à l'aide des sous-ensembles flous [33].

Dans ce chapitre, nous présentons les bases fondamentales de la logique floue. Ensuite, une description de différentes techniques de classification floues sera abordée. Dans une seconde étape, on abordera les différents types de combinaison des réseaux de neurones et logique floue pour la classification. On donnera à la fin de ce chapitre, quelques résultats comparatifs entre ces classifieurs.

II. DEFINITIONS

II.1. Sous-ensemble flou

Soit X un ensemble dénombrable ou non. Un sous-ensemble flou A de X est caractérisé par la fonction d'appartenance suivante :

$$\mu_A \ : \ X \longrightarrow [0,1] \qquad (59)$$

Pour un élément $x \in X$, $\mu_A(x) \in [0,1]$ est le degré d'appartenance de x au sous-ensemble flou A. L'ensemble X est appelé ensemble ou univers de discours. Dans le cas particulier où la fonction d'appartenance μ_A prend ses valeurs dans {0,1}, le sous-ensemble flou A est un sous-ensemble classique, c'est-à-dire qu'un élément x appartient ($\mu_A = 1$) ou non ($\mu_A = 0$) au sous-ensemble A.

Il existe différents types de fonctions d'appartenance (Figure 18)

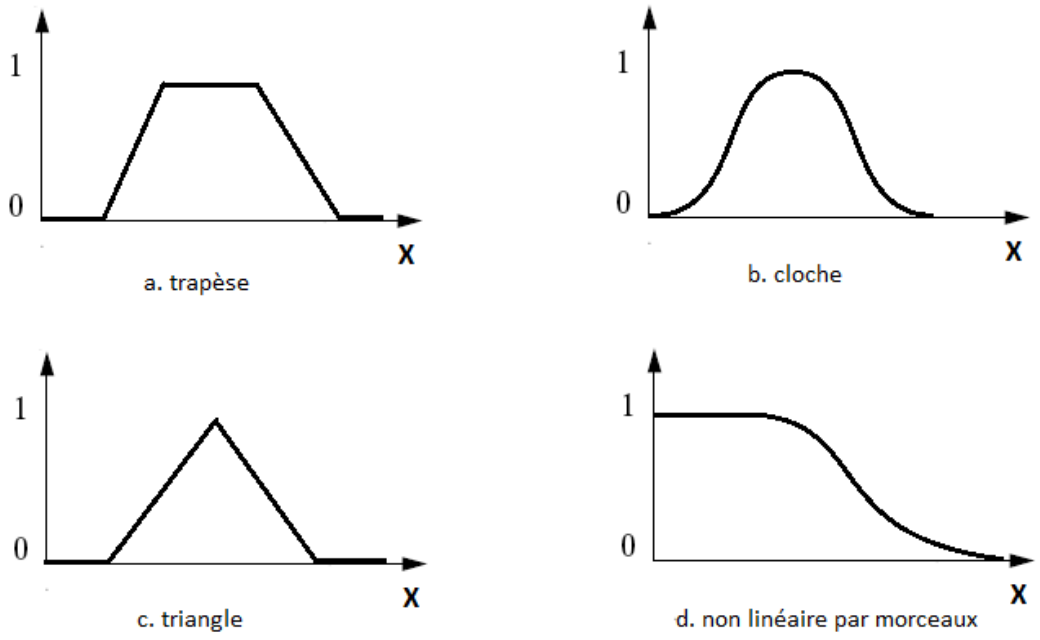


Figure 18 : Différents types de fonction d'appartenance

II.2. Support

On appelle support de A, noté Supp(A), l'ensemble des éléments de X pour lesquels le degré d'appartenance à A n'est pas nul :

$$Supp(A) = \{x \in X \mid \mu_A(x) > 0\} \quad (60)$$

II.3. Noyau

On appelle noyau de A, noté Noy(A), l'ensemble des éléments de X pour lesquels le degré d'appartenance à A est égal à 1 :

$$Noy(A) = \{x \in X \mid \mu_A(x) = 1\} \quad (61)$$

II.4. Hauteur

On appelle hauteur de A, noté h(A), la plus grande valeur prise par la fonction d'appartenance associée à A :

$$h(A) = \sup_{x \in X}(\mu_A(x)) \quad (62)$$

Le sous-ensemble A est dit normalisé si h(A)=1.

II.5. α-coupe

On appelle une α-coupe de A, noté A_α , l'ensemble des éléments de X pour lesquels le degré d'appartenance à A est supérieur ou égal à α :

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\} \quad (63)$$

II.6. Opérations sur les sous-ensembles flous

Comme pour les sous-ensembles classiques, on peut définir des opérations ensemblistes sur les sous-ensembles flous, comme l'égalité, l'union, l'intersection et le complément.

Soient A et B deux sous-ensembles flous de l'univers de discours X, on généralise les opérations ensemblistes sur A et B de comme suit :

II.6.a. Egalité

Deux sous-ensembles flous A et B de X sont égaux si leurs fonctions d'appartenance sont identiques sur l'univers de discours X :

$$A = B : \qquad \forall\, x \epsilon X, \mu_A(x) = \mu_B(x) \tag{64}$$

II.6.b. Complément

Le complément A^c d'un sous-ensemble flou A de X est le sous-ensemble flou de X définie par la fonction caractéristique suivante :

$$A^c: \qquad \forall\, x \epsilon X, \mu_{A^c}(x) = 1 - \mu_A(x) \tag{65}$$

II.6.c. Union

L'union de deux sous-ensembles flous A et B de X est un sous-ensemble flou de X composé des éléments auxquels on affecte le plus grand des degrés d'appartenance à A et à B :

$$C = A \cup B: \quad \forall\, x \epsilon X, \mu_C(x) = \max(\mu_A(x), \mu_B(x)) \tag{66}$$

II.6.d. Intersection

L'intersection de deux sous-ensembles flous A et B de X est un sous-ensemble flou de X composé des éléments de X auxquels on affecte le plus petit des degrés d'appartenance à A et à B :

$$C = A \cap B: \forall\, x \epsilon X, \mu_C(x) = \min(\mu_A(x), \mu_B(x)) \tag{67}$$

II.6.e. T-Norme et T-conorme

La définition d'une généralisation de l'union et de l'intersection de deux sous-ensembles flous n'est pas unique. Zadeh [33] a proposé, en premier, d'utiliser les opérateurs Min pour

l'intersection et l'opérateur Max pour l'union des sous-ensembles flous. Après la proposition de Zadeh, une multitude d'opérateurs ont été proposés. Ces opérateurs sont regroupés en deux familles : les T-normes pour définir les opérateurs d'intersection et les T-conormes pour définir les opérateurs d'union.

T-norme

$T: [0,1] \times [0,1] \longrightarrow [0,1]$ est une fonction de type T-norme si et seulement si pour chaque élément x, y et z de [0, 1], elle vérifie les propriétés suivantes :

P1. 1 est un élément neutre : $T(x,1) = x$

P2. Isotonie : $x \leq y, u \leq v \Rightarrow T(x,u) \leq T(y,v)$

P3. Commutativité : $T(x,y) = T(y,x)$

P4. Associativité : $T\big(x, T(y,z)\big) = T(T(x,y),z)$

T-conorme

$T: [0,1] \times [0,1] \longrightarrow [0,1]$ est une fonction de type T-conorme si et seulement si pour chaque élément x, y et z de [0, 1], elle vérifie les propriétés (P2), (P3) et (P4) plus la propriété suivante :

P5. 0 est un élément neutre : $T(x,0) = x$

En utilisant les T-normes et les T-conormes, il est possible de donner différentes formes aux opérateurs d'union et d'intersection (Tableau 3).

Tableau 3 : Exemples des T-normes et T-conormes

Nom	T-norme	T-conorme
Zadeh	$\boldsymbol{Min(x,y)}$	$\boldsymbol{Max(x,y)}$
Probabiliste	$\boldsymbol{x.y}$	$\boldsymbol{x+y-x.y}$
Lukasiewicz	$\boldsymbol{Max(x+y,0)}$	$\boldsymbol{Min(x+y,1)}$

Il faut noter que le choix d'un opérateur dépend du problème traité.

II.7. Produit cartésien et projection

II.7.a. Produit cartésien

Dans le domaine de la reconnaissance de formes, la caractérisation d'une forme se formalise par rapport à plusieurs caractéristiques ou attributs. Ces attributs appartiennent à plusieurs espaces.

Ainsi, si on veut travailler sur plusieurs univers de discours simultanément, il faut avoir une manière de concevoir un univers de discours globale. Cette conception est possible grâce au produit cartésien des différents univers de discours. La définition de ce produit est la suivante :

Le produit cartésien de n sous-ensembles flous $A_1, A_2, \ldots, A_n$, définis respectivement sur les univers de discours $X_1, X_2, \ldots, X_n$, est un sous-ensemble flou $A = A_1 \times A_2 \times \ldots \times A_n$, défini sur un univers de discours X, décrit par la fonction caractéristique suivante :

$$\forall\, x = (x_1, x_2, \ldots, x_n) \epsilon X, \mu_A(x) = \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n)) \qquad (68)$$

II.7.b. Projection

Inversement au produit cartésien, si on dispose de connaissances globales dans un univers de discours global X, et on veut extraire les informations relatives aux univers de discours qui composent X, on réalise alors la projection de X sur chacun de ces univers de discours qui le composent. La définition de la projection est la suivante :

La projection d'un sous-ensembles flous A de $X = X_1 \times X_2 \times \ldots \times X_n$ sur $X_a \times X_b \times \ldots \times X_m$, est un sous-ensemble flou noté proj(A), défini sur $X_a \times X_b \times \ldots \times X_m$, , décrit par la fonction caractéristique suivante :

$$\forall\, x = (x_a, x_b, \ldots, x_m) \epsilon X_a \times X_b \ldots X_m\,, \mu_A(x) = \sup_{1 \le i \le n, i \neq a, \ldots, i \neq m} \mu_A((\ldots, x_i \ldots)) \qquad (69)$$

III. SYSTEMES D'INFERENCES FLOUS

III.1. Introduction

Dans le but de s'approcher du raisonnement humain, caractérisé par sa souplesse vis-à-vis des connaissances imprécises et floues L. A. Zadeh [33] a introduit le concept du raisonnement flou. Ce raisonnement utilise le concept des sous-ensembles flous. Ce qui lui permet de prendre en compte des informations imprécises des classes aux frontières, qui ne sont pas nettes et qui peuvent se chevaucher, ainsi que la propriété d'appartenance d'un élément à plusieurs classes avec des degrés d'appartenance différents. En plus, le raisonnement flou prend en compte aussi des connaissances symboliques ou qualitatives.

En général, un système d'inférence flou se compose de propositions floues, règles floues et de l'agrégation des règles floues.

III.2. Règles floues

Une règle floue correspond à la mise en relation de deux propositions floues par une implication :

SI *x est A* ***alors*** *y est B*

Où A et B sont des sous-ensembles flous

La première partie de la règle (x est A) s'appelle prémisse et la seconde partie de la règle (y est B) s'appelle conclusion.

Une proposition floue peut être sous forme d'une combinaison de plusieurs propositions primitives (x est A), à l'aide des opérateurs 'et' et 'ou' qu'on peut ensuite définir par les T-normes et les T-conormes. Par exemple :

$$p : x_1 \text{ est } A_1 \text{ et } x_2 \text{ est } A_2$$

La proposition *p* est définie par sa fonction caractéristique suivante :

$$\mu_p : E_1 \times E_2 \longrightarrow [0,1]$$
$$(x_1, x_2) \longrightarrow T(\mu_{A_1}(x_1), \mu_{A_1}(x_2)) \qquad (70)$$

On peut alors construire des règles floues plus complexes dont la partie prémisse et la partie conclusion correspondent à une combinaison de propositions, par exemple :

Si *x_1 est A_1* ***et*** *x_2 est A_2* ***Alors*** *y_1 est B_1 et y_2 est B_2*

III.3. Système d'inférences floues

D'une manière générale, un système d'inférences floues se compose de plusieurs règles floues :

r_i : ***Si*** *x_1 est M_{i1}* ***et*** *x_2 est M_{i2}* ***et****... **et** x_n est M_{in}* ***Alors*** *y_1 est B_{i1}* ***et*** *y_2 est B_{i2}* ***et****... **et** y_c est B_{ic}*

Chacune des règles de ce système est évaluée à l'aide d'un autre opérateur (Alors) appelé l'implication floue I :

$$\mu_{B_i}(x_1, x_2, .. x_n) = I\left(T\left(\mu_{M_{i1}}(x_1), \dots, \mu_{M_{in}}(x_n)\right), T\left(\mu_{B_{i1}}(y_1), \dots, \mu_{B_{ic}}(y_c)\right)\right) \qquad (71)$$

Plusieurs opérateurs d'implications floues peuvent être utilisés pour l'évaluation des règles floues (Tableau 4).

Tableau 4: Exemples d'implications floues

Nom	I(A,B)
Mamdani	$\mathbf{min}(\boldsymbol{\mu_A}, \boldsymbol{\mu_B})$
Larsen	$\boldsymbol{\mu_A}.\boldsymbol{\mu_B}$
Lukasiewicz	$\mathbf{min}(\mathbf{1} - \boldsymbol{\mu_A} + \boldsymbol{\mu_B}, \mathbf{1})$

IV. CLASSIFICATION FLOUE NON SUPERVISEE

IV.1. C-Moyennes floues

L'algorithme des c-moyennes floues est la variante floue des c-moyennes classiques. Cet algorithme a été proposé par Dunn [34]. Il constitue la base de nombreuses approches de classification nommées classifications floues adaptatives.

IV.1.a. Principe

A l'opposé de l'algorithme c-moyenne classique, l'algorithme des c-moyennes floues effectue une partition floue caractérisée par une matrice $U = [\mu_{ij}]$ de dimension C×N nommé matrice de C-partition floue. Cette matrice est définie par Ruspini [35] comme suit :

$$\begin{cases} \mu_{ij} \in [0,1] & \forall\, i,j \\ 0 < \sum_{j=1}^{N} \mu_{ij} < N & \forall i \\ \sum_{i=1}^{C} \mu_{ij} = 1 & \forall j \end{cases} \qquad (72)$$

Où C est le nombre de classes, N est le nombre d'éléments à classer, et μ_{ij} est le degré d'appartenance du j^éme élément à la i^éme classe.

Cet algorithme, comme les autres algorithmes de classification, cherche à minimiser une fonction objective. Cette fonction est définie comme suit :

$$J_{P,U,D} = \sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^{m} d^2(x_j, P_i) \qquad (73)$$

Dans cette équation, qui est une généralisation de l'équation définie pour l'algorithme des c-moyennes classique est caractérisée par deux paramètres :

1. La distance $d^2(x_j, P_i)$, entre le vecteur caractéristique (vecteur d'attributs) x_j et le prototype P_i. Cette distance est définie en général de la façon suivante :

$$d^2(x_j, P_i) = (x_j - P_i)^T A(x_j - P_i) \quad (74)$$

Où A est une matrice définie positive de dimension n×n.

Cette distance détermine la forme des classes obtenues. En général, ces classes sont de formes ellipsoïdales. La distance Euclidienne est un cas particulier où A=I. Dans ce cas les formes de classe obtenues sont des sphères.

2. Le paramètre $m \in [1, \infty]$ détermine la netteté du partitionnement flou. Quand m tend vers 1, la partition obtenue tend vers une partition nette comme celle obtenue par la méthode des c-moyennes classique. Quand m tend vers l'infini, la partition est de plus en plus floue et donc les classes se chevauchent d'avantage.
3. Le choix de ce paramètre n'est pas défini par une méthode précise, dans la plupart des d'application en trouve m=2 [36] [37].
4. Il faut noter que pour des contraintes de convergence de l'algorithme, le théorème de Tuker [38] suggère de prendre $m \geq N/(N-2)$, où N est le nombre des données.

D'après Bazdek [39], Les éléments de la matrice *U* assurant la minimisation du critère $J_{P,U,D}$ vérifient les équations suivantes :

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{d^2(x_j, P_i)}{d^2(x_j, P_k)} \right)^{\frac{1}{m-1}}}, \qquad si\ I_j = \emptyset \quad (75)$$

$$\left. \begin{array}{ll} \mu_{ij} = 0, & i \notin I_j \\ \sum_{i \in I_j} \mu_{ij} = 1\ , & i \in I_j \end{array} \right\}, \qquad si\ I_j \neq \emptyset \quad (76)$$

$$Avec \quad I_j = \left\{ i \ / \ 1 \leq i \leq C;\ d^2(x_j, P_i) = 0 \right\}$$

Contrairement à l'algorithme de c-moyennes classique qui utilise uniquement les éléments d'une classe supposée de même poids, l'algorithme des c-moyenne floues utilise toutes les données pondérées par des valeurs différentes, comme montre l'équation suivante :

$$P_i = \frac{\sum_{j=1}^{N} \mu_{ij}^m x_j}{\sum_{j=1}^{N} \mu_{ij}^m} \quad (77)$$

L'algorithme des c-moyennes floues est un algorithme stable et robuste. Au contraire de l'algorithme des c-moyennes classique, l'algorithme des c-moyennes floues n'est pas sensible au choix de l'initialisation des centres des classes ni au problème des minimas locaux [40]. Grâce à sa simplicité, l'algorithme des c-moyennes floues est très utilisé pour initialiser les prototypes des classes pour des algorithmes plus sophistiqués.

IV.1.b. Algorithme des c-moyennes floues

Le processus itératif de l'algorithme des c-moyennes floues est décrit comme suivant :

Algorithme des c-moyennes floues

---

1. *Fixer le nombre de classes C, le facteur de Fuzzyfication m, choisir une distance, initialiser la matrice U et choisir un critère d'arrêt ε.*
2. *Calculer les prototypes P_i à l'aide de l'équation* (77).
3. *Mettre à jour la matrice U à l'aide des équations* (75) *et* (76).
4. *Si $\Delta U > \varepsilon$ revenir à l'étape 2.*

IV.2. Algorithme de Gath et Geva

L'algorithme de Gath et Geva [41] est un cas particulier des c-moyennes floues caractérisé par l'utilisation d'une distance exponentielle basée sur l'estimation du maximum de vraisemblance. Cette distance est définit par:

$$d^2(x_j, P_i) = \frac{[det(F_i)]^{1\backslash 2}}{Q_i} exp\left[\frac{1}{2}(x_j - P_i)^T F_i^{-1}(x_j - P_i)\right] \quad (78)$$

Où F_i est la matrice de covariance floue associée à la classe i et Q_i est la probabilité à priori de sélectionner la i^{ème} classe. Ces matrices sont définies comme suit :

$$Q_i = \frac{1}{N}\sum_{j=1}^{N} \mu_{ij}^{m} \tag{79}$$

$$F_i = \frac{\sum_{j=1}^{N} \mu_{ij}^{m}(x_j - P_i)(x_j - P_i)^T}{\sum_{j=1}^{N} \mu_{ij}^{m}} \tag{80}$$

La distance définie ci-dessus engendre des classes sous forme hyper-ellipsoïdales. Mais ce genre de distances peut aussi causer des problèmes d'instabilité due aux minimas locaux. Pour éviter ces minimas, la phase d'initialisation doit être soigneusement élaborée. Pour cette raison, on utilise généralement l'algorithme de c-moyennes floues pour générer une estimation initiale de la partition floue.

IV.3. C-Moyennes possibilistes

La classification possibiliste a été proposée par Krishnapuram et Keller [42], elle se caractérise par une définition plus souple du concept de la partition floue. En effet, la définition d'une partition floue définie par l'équation (72) impose que la somme des degrés d'appartenances d'un élément à l'ensemble des classes soit égale à 1.

La figure (Figure 19) montre un exemple d'un problème de classification en deux classes où les deux points x_1 et x_2 ont les mêmes degrés d'appartenances égaux à 0.5. Pour x_1, la valeur parait normale car le point est situé exactement entre les deux classes. Alors que pour x_2, normalement il se situe très loin des deux classes, donc il doit avoir des degrés d'appartenance aux deux classes très faible au lieu d'une valeur de 0.5.

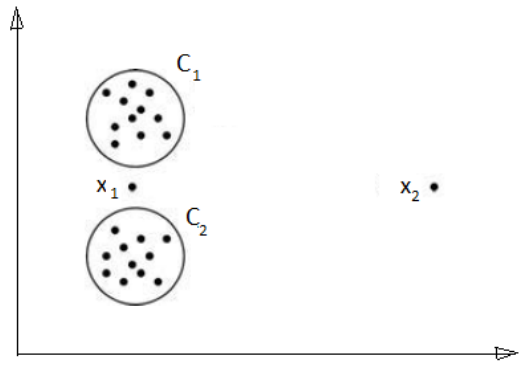


Figure 19 : Exemple où le degré d'appartenance d'un point (x_2) à une classe est inconvenable

Pour remédier à ce problème, Krishnapuram et Keller [42] ont proposé de relâcher cette contrainte en définissant une nouvelle partition floue comme suivant :

$$\begin{cases} \mu_{ij} \in [0,1] & \forall\, i,j \\ 0 < \sum_{j=1}^{N} \mu_{ij} < N & \forall i \\ \sum_{i=1}^{c} \mu_{ij} \leq 1 & \forall\, j \end{cases} \quad (81)$$

Avec la fonction objective suivante :

$$J_{P,U,D} = \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij}^{m} d^2(x_j, P_i) + \sum_{i=1}^{C} \eta_i \sum_{j=1}^{N} (1-\mu_{ij})^m \quad (82)$$

Où η_i est un paramètre positif qui définit l'importance du second terme ajouté à la fonction objectif (82). Il peut être défini de plusieurs façons :

$$\eta_i = K \frac{\sum_{j=1}^{N} \mu_{ij}^{m} d^2(x_j, P_i)}{\sum_{j=1}^{N} \mu_{ij}^{m}} \quad (83)$$

Où K est un paramètre positif souvent pris égal à 1.

La minimisation de la fonction objectif (82) conduit à une modification de la condition nécessaire utilisée pour la mise à jour des degrés d'appartenance :

$$\mu_{ij} = \frac{1}{1 + \left(\frac{d^2(x_j, P_i)}{\eta_i}\right)^{\frac{1}{m-1}}} \quad (84)$$

Une autre approche consiste à choisir des fonctions d'appartenance de type exponentiel comme montre l'équation suivante :

$$\mu_{ij} = exp\left[-\frac{1}{2}(x_j - P_i)^T F_i^{-1}(x_j - P_i)\right] \quad (85)$$

Où F_i représente la matrice de covariance floue associée à la classe i, définie en (80).

Cette approche permet d'éviter le problème de la détermination des paramètres η_i en s'appuyant sur l'évaluation des matrices de covariance F_i.

V. CLASSIFICATION FLOUE SUPERVISEE

En général, les techniques de la logique flous se basent essentiellement sur les connaissances à priori d'experts exprimées de manière explicite dans le système d'inférence flou d'un classifieur. En réalité, ces connaissances ne sont pas toujours disponibles. La conception d'un classifieur repose alors sur un apprentissage réalisé à partir de données d'apprentissage étiquetées selon leurs classes d'appartenance. Les connaissances sont alors extraites à partir de ces exemples lors d'un processus d'apprentissage.

Il existe de nombreuses approches pour ce problème, qui dépendent d'une part du type du classifieur et d'autre part, des données d'apprentissage utilisées. D'une manière générale, l'apprentissage consiste à générer des règles floues qui formalisent les connaissances embarquées dans les données étiquetées. A l'opposé à d'autres approches, les réseaux de neurones en particulier, le processus de génération des règles floues permet de construire des classifieurs transparents, d'où interprétables par un être humain.

Dans la suite de cette partie, nous décrivons les différentes méthodes pour extraire automatiquement des règles floues à partir de données d'apprentissage.

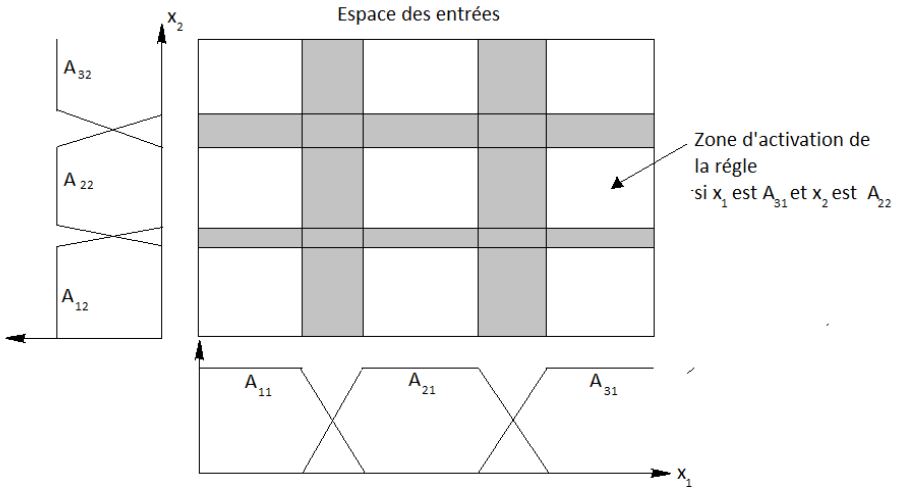


Figure 20 : Partition floue de l'espace des entrées

V.1. Principe

Les parties prémisse des règles floues dans un système d'inférences floues constituent une partition floue de l'espace des entrées. Ces prémisses définissent alors les domaines d'application ou d'activation des règles dans l'espace des entrées (Figure 20). La génération d'un système d'inférences floues détermine alors le partitionnement de l'espace des entrées.

Après un partitionnement flou adéquat, dans un système d'inférences floues, les règles les plus significatives correspondent aux régions de l'espace des entrées les plus denses. L'extraction des règles floues à partir des données d'apprentissage repose alors, principalement, sur la définition d'une partition floue de l'espace des entrées, guidée par la relation d'entrée-sortie décrite par les exemples d'apprentissage.

La figure (Figure 21) montre un exemple d'une partition floue simple et intuitive générée pour des données d'apprentissage appartenant à deux classes. Les règles floues extraites pour ces exemples sont comme suivant :

Si x_1 est A_{11} **et** x_2 est A_{22} **Alors** C_1

Si x_1 est A_{21} **et** x_2 est A_{12} **Alors** C_2

Dans des cas plus complexes, la génération des partitions floues et des règles d'inférences floues n'est pas d'une pareille simplicité. Ainsi, on trouve dans la littérature

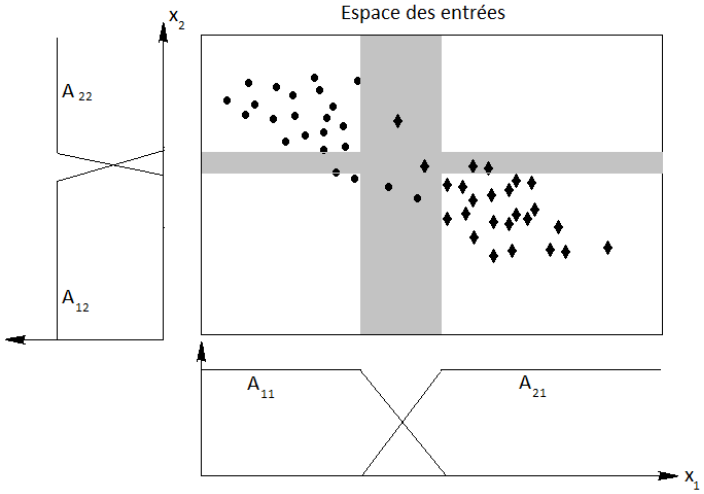


Figure 21: Exemple d'une partition floue fixe

plusieurs méthodes pour produire un système d'inférence flou le mieux adapté aux exemples d'entraînement fournis pour un problème de classification donné. Dans le suit de ce chapitre

V.2. Partition fixe simple

Cette méthode consiste à partitionner l'espace des entrées sans la prise en compte des données d'apprentissage. Les règles générées sont déduites du produit cartésien des sous-ensembles de l'espace des entrées sous la forme suivante :

Règle i : Si x_1 *est* A_{i1} ***et*** x_2 *est* A_{i2}*….* ***et*** x_n *est* A_{in} ***Alors*** C_k *avec un degré de certitude* f_{ik}

Où f_{ik} est le coefficient de certitude d'appartenance du vecteur x à la classe C_k. Ce coefficient caractérise le rapport entre le nombre de données d'apprentissage bien classées et celles qui sont mal classées. Il peut être évalué comme suit [43] :

$$f_{ik} = \frac{B_k - \sum_{t \neq k} \frac{B_t}{N-1}}{\sum_{t=1}^{N} B_t} \quad (86)$$

Avec

$$B_t = \sum_{x \in D_{it}} \prod_{j=1}^{n} \mu_{A_{ij}}(x_j) \ , t = 1, \dots, N \quad (87)$$

Où D_{it} est l'ensemble des données d'apprentissage appartenant à la classe C_t dans le domaine d'activation de la règle i, $B_k = \max_{i=1\dots N} B_i$ et N le nombre de classes.

Le problème avec cette approche réside dans le choix à priori de la partition initiale. En effet, d'une part, une partition fine pose des problèmes d'évaluation dans les domaines où il y a peu de données d'apprentissage. D'autre part, une partition grossière ne permet pas de réaliser une bonne discrimination entre les classes.

Pour remédier à ce problème, ils existent plusieurs méthodes qui utilisent les données d'apprentissage pour déterminer le partitionnement de l'espace d'entrées en sous-ensembles floues [44] [45]. Parmi ces méthodes, on trouve des méthodes statistiques et des méthodes se basant sur les réseaux de neurones.

V.3. Approche statistique

Dans cette approche, les fonctions d'appartenance utilisées sont des gaussiennes. Si l'espace des entrées est de dimension n et le nombre de classe est N, alors nous avons n×N fonctions d'appartenance. Pour chaque classe, on calcule la moyenne et la variance des composantes des données d'apprentissage comme le montre les équations suivantes :

$$m_{ki} = \frac{1}{N_k} \sum_{x \in C_k} x_i \qquad (88)$$

$$\sigma_{ki}^2 = \frac{1}{N_k} \sum_{x \in C_k} (x_i - m_{ki}) \qquad (89)$$

Où $N_k$est le nombre d'exemples dans la classe C_k et x_i est la i^{éme} composante du vecteur x. Une telle partition est représentée dans la figure (Figure 22) avec n=2 et N=3 ce qui donne six fonctions d'appartenances.

Pour un vecteur inconnu x, on calcule le degré d'appartenance de chacune de ses composantes aux différentes classes C_k par l'équation suivante :

$$\mu_{ki}(x) = e^{\frac{-|x_i - m_{ki}|}{\sigma_{ki}^2}} \qquad (90)$$

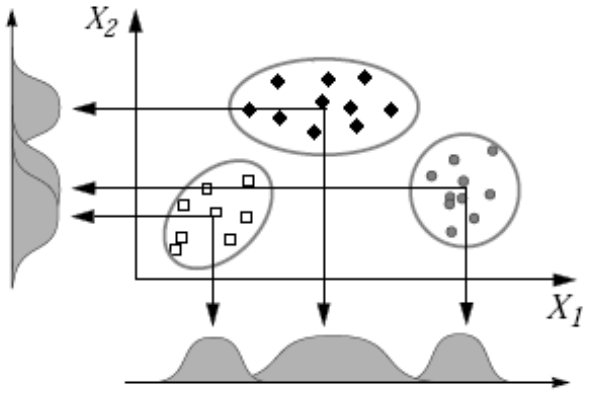


Figure 22: Exemple d'une partion floue par l'approche statistique

Cette équation montre que si toutes les composantes du vecteur x sont proches des moyennes m_{ki} alors $\mu_{ki}(x) \approx 1\ , i = 1, \dots, n$ et par suite l'inconnu x sera attribué à la classe C_k. Le degré d'appartenance d'un vecteur x à une classe C_k sera défini comme la moyenne des degrés d'appartenance $\mu_{ki}(x)$:

$$\mu_k(x) = \frac{1}{n}\sum_{i=1}^{n} \mu_{ki}(x) \qquad (91)$$

V.4. Logique floue et réseaux de neurones

Les réseaux de neurones et la logique floue sont deux techniques complémentaires. Les réseaux de neurones peuvent apprendre à partir d'un ensemble de données d'entraînement, mais la connaissance qu'ils représentent reste incompréhensible par l'être humain. Par contre, les systèmes flous sont facilement des systèmes transparents et maniables par un être humain car ils utilisent des termes linguistiques et des règles sous forme de " si-Alors ", mais à l'opposé des réseaux de neurones n'ont pas la capacité d'apprendre par l'exemple. Dans la littérature, plusieurs études et recherches ont été conduites pour fusionner les réseaux de neurones et la logique floue [46]. Ces recherches peuvent être classées en plusieurs catégories, parmi ces catégories on trouve : les systèmes neuro-flous et les systèmes flous pour les réseaux de neurones.

V.4.a. Systèmes neuro-flous

Dans ce type de système, les réseaux de neurones sont utilisés pour améliorer les performances du système flou. Les systèmes neuro-flous ont les caractéristiques des réseaux de neurones ainsi que celles des systèmes flous, ils peuvent apprendre comme les réseaux de neurones et peuvent déduire comme les systèmes flous.

Dans un réseau de neurone ordinaire, les nœuds (neurones) ont tous la même fonctionnalité et sont totalement connectés aux nœuds des couches adjacentes. Dans les systèmes neuro-flous, les nœuds ont différents rôles dans le système et ne sont pas totalement connectés. Les nœuds et les liens entre nœuds dans les systèmes neuro-flous correspondent à

des composantes spécifiques du système flou comme les variables d'entrées, de sorties et les règles floues.

Grâce à sa structure, un système neuro-flou intègre aisément les connaissances d'un expert avant la phase d'apprentissage, ce qui diminue considérablement le risque de converger vers des minimas locaux.

Un système neuro-flou se compose généralement de cinq couches [47] (Figure 23). Une couche d'entrée, une pour la fuzzification, une pour représenter les règles floues, une pour les variables de sortie floues et une pour la défuzzification.

La première couche ne fait aucun traitement. Elle ne fait que transmettre les variables d'entrée vers la couche suivante :

$$f_j^1(x) = x \tag{92}$$

Où $f_j^k(x)$ représente la fonction du nœud j de la couche k.

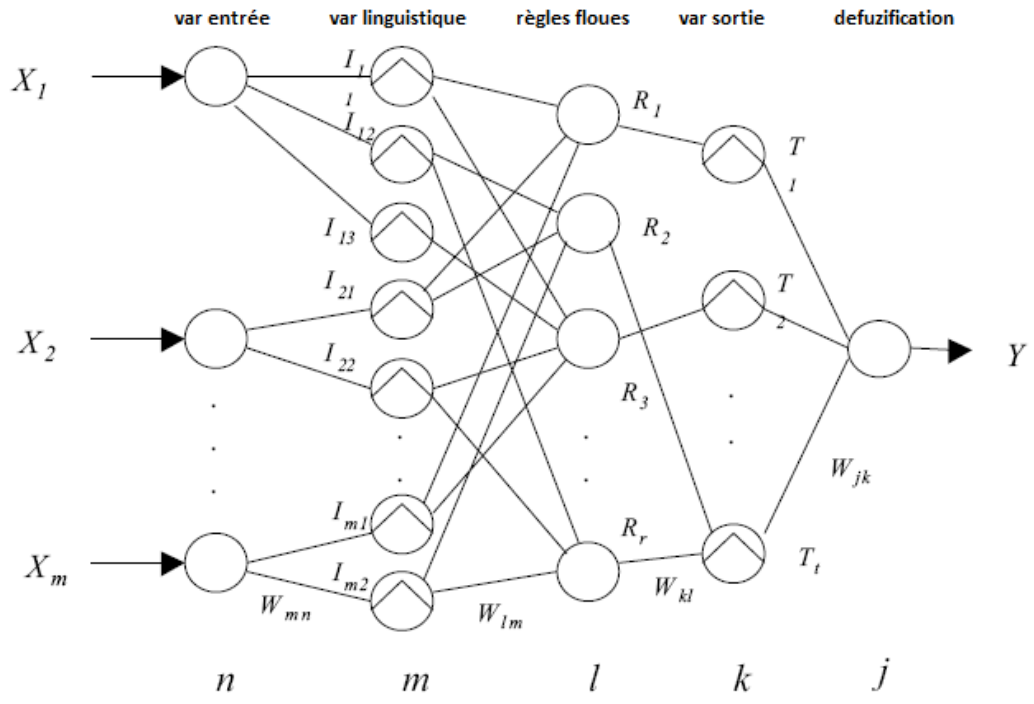


Figure 23 : Structure d'un système neuro-flou

Dans la seconde couche, chaque nœud représente une variable linguistique de la variable d'entrée. Ce nœud est caractérisé par un ensemble de paramètres définissants la fonction d'appartenance de la variable linguistique. Les nœuds de cette couche génèrent les degrés d'appartenance des variables d'entrées. Si par exemple un nœud représente un sous-ensemble flou A_j alors sa sortie est définie comme suit :

$$f_j^2(x) = \mu_{A_j}(x) \quad (93)$$

Les nœuds de la troisième couche correspondent aux propositions dans les règles floues définies pour le système. Par exemple dans la figure (Figure 23), la sortie du nœud R_1 est la valeur de la proposition floue suivant :

$$X_1 \text{ est } I_{11} \text{ et } X_2 \text{ est } I_{21} \text{ et } \dots X_m \text{ est } I_{m1}$$

Pour l'évaluation de cette proposition, on utilise l'un des opérateurs du tableau (Tableau 3). Les connexions entre la couche 2 et la couche 3 sont toutes de valeur constante égale à 1.

La quatrième couche représente les inférences floues dans le système. Les nœuds de cette couche correspondent aux variables floues de sortie. Par exemple dans la figure (Figure 23), le nœud T_1 a une seule entrée, la sortie du nœud R_1 et sa sortie est un sous-ensemble flou résultant de la règle floue suivante :

$$\text{Si } X_1 \text{ est } I_{11} \text{ et } X_2 \text{ est } I_{21} \text{ et } \dots X_m \text{ est } I_{m1} \text{ Alors Y est } T_1$$

Pour le nœud T_t, sa sortie est le sous-ensemble flou résultant des deux règles floues suivantes :

$$\text{Si } X_1 \text{ est } I_{12} \text{ et } X_2 \text{ est } I_{21} \text{ et } \dots X_m \text{ est } I_{m2} \text{ Alors Y est } T_t$$

$$\text{Si } X_1 \text{ est } I_{12} \text{ et } X_2 \text{ est } I_{22} \text{ et } \dots X_m \text{ est } I_{m2} \text{ Alors Y est } T_t$$

D'une façon générale, pour un nœud possédant q règles en entrée, sa sortie est :

$$f_j^4(x_1, x_2, \dots, x_q) = \max_{i=1\dots q}(w_{ji}x_i) \quad (94)$$

Où x_i est le sous-ensemble flou résultant de la règle i et w_{ji} un poids qui décrit l'importance de la règle issue du nœud i de la couche 3 pour le nœud j de la couche 4.

Les nœuds de la couche 5 s'occupent de l'agrégation et de la défuzzification. Pour réaliser cette tâche, deux approches peuvent être adoptées. Soit on utilise la méthode du centre de gravité [48] ou bien la méthode proposée par Kwak [49].

Soit $A_1, A_2, \dots, A_t$ les sous-ensembles flous sortants des nœuds de la couche 4. La sortie par la méthode du centre de gravité consiste à prendre l'abscisse du centre de gravité de l'union de $A_1, A_2, \dots et\ A_t$ en utilisant l'opérateur Max.

La méthode de Kwak consiste à appliquer la méthode du centre de gravité pour chacun des sous-ensembles flous $A_1, A_2, \dots, A_t$ puis calculer la valeur de sortie comme suit :

$$f_j^5(A_1, A_2, \dots, A_t) = \frac{\sum_{i=1}^{t} y_i . aire(A_i)}{\sum_{i=1}^{t} . aire(A_i)} \quad (95)$$

Où $aire(A_i)$ est l'aire en dessous de la courbe de la fonction d'appartenance de A_i et y_i l'abscisse du centre de gravité de l'aire de A_i.

La figure (Figure 24) illustre un simple exemple où la couche 4 est constituée de deux nœuds et la couche 5 d'un seul nœud. Alors le nœud de la couche 5 va recevoir deux sous ensemble flous $A_1\ et\ A_2$ (Figure 24 (a) et (b)). La méthode du centre de gravité est illustrée dans la figure (Figure 24 (c)) et la méthode de Kwak dans la figure (Figure 24 (d)).

On remarque que dans la méthode de Kwak, les surfaces qui se chevauchent, peuvent être évaluées plusieurs fois (Figure 24 (d)). Alors que dans la méthode du centre de gravité elle n'est évaluée qu'une seule fois (Figure 24 (c)). En outre, la méthode de Kwak présente l'avantage de la simplicité dans les calculs.

Les liaisons entre la couche 4 et 5 sont toutes des simples liaisons de poids égales à 1.

Pareil aux réseaux de neurones, après à la mise au point de l'architecture d'un réseau neuro-flou vient l'étape d'entraînement. Cet entraînement se fait pour les réseaux neuro-flous à l'aide de l'algorithme du rétro-propagation des erreurs. A la fin de l'entraînement, les poids entre la couche 3 et 4 sont ajustés ainsi que les paramètres des fonctions d'appartenance dans les nœuds des couches 2 et 4.

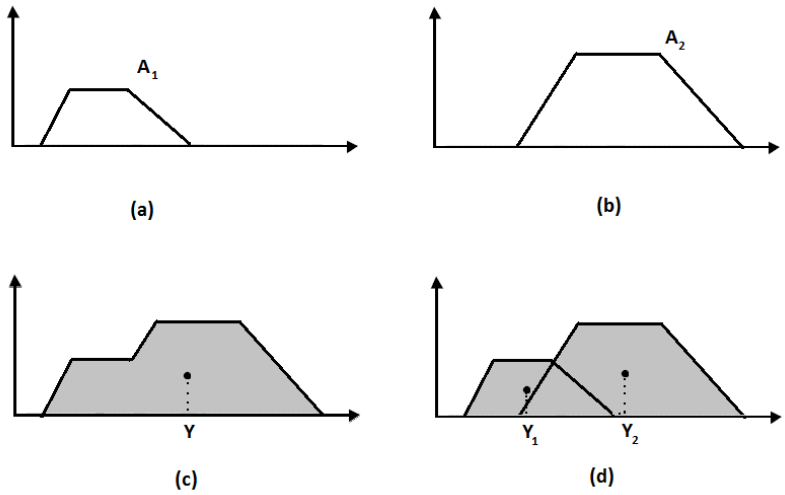


Figure 24 : défuzzification par (c) méthode de centre de gravité, (d) méthode de Kwak (a) et (b) sont les deux sous-ensembles flous sortant des nœuds 1 et 2 de la couche 4.

On peut trouver dans la littérature plusieurs variantes de ce type de système, nous citons par exemple, les réseaux de neurones flous Min-Max [50] et les systèmes ANFIS (Adaptive Neural netwok Fuzzy Inference System) [51].

V.4.b. Systèmes flous pour des réseaux de neurones

Dans un système neuro-flou, on a vu comment améliorer les performances d'un système flou à l'aide d'un réseau de neurones. Dans cette approche nous allons voir comment un système flou peut aider à construire un réseau de neurones.

En effet, Au lieu de construire et entrainer un unique réseau de neurone pour la totalité des données d'entraînement, on utilise d'abord un classifieur flou pour classer les données d'entraînement en plusieurs classes, puis on crée un réseau de neurones pour chaque classe (Figure 25). On peut alors présenter le système sous forme d'un système d'inférences floues comme suit :

$$\begin{aligned} &\textit{Si } x \textit{ est } C_1 \;\; \textit{Alors } Y = NN_1(x) \\ &\textit{Si } x \textit{ est } C_2 \;\; \textit{Alors } Y = NN_2(x) \\ &\qquad\vdots \\ &\textit{Si } x \textit{ est } C_n \;\; \textit{Alors } Y = NN_n(x) \end{aligned}$$

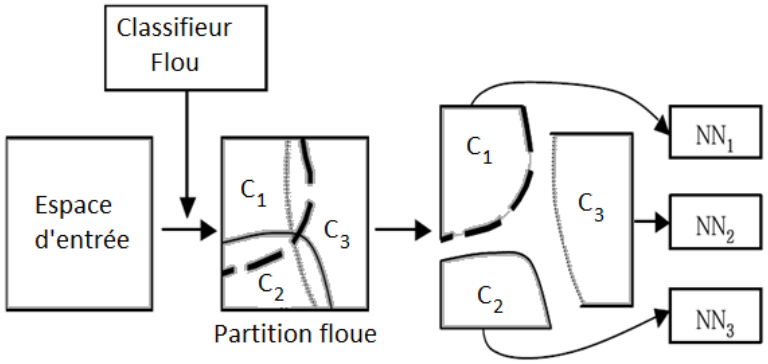


Figure 25 : Construction des réseaux de neurones à l'aide d'un système flou.

Ainsi, pour un vecteur caractéristique x, le classifieur flou génère un degré d'appartenance de ce vecteur à chacune des classes. La sortie du système est alors évaluée par l'équation suivante:

$$y = \frac{\sum_{i=1}^{n} \mu_{c_i}(x) NN_i(x)}{\sum_{i=1}^{n} \mu_{c_i}(x)} \qquad (96)$$

Où $\mu_{c_i}(x)$ est le degré d'appartenance du vecteur x à la classe C_i, calculé par le classifieur flou.

Cette technique est très utile lorsqu'un problème peut être subdivisé en plusieurs sous-problèmes. Par exemple, pour réaliser un OCR multi-fonts, on peut développer un classifieur flou pour les fonts puis un OCR spécialisé pour chaque font.

On peut constater qu'avec cette technique, on utilise notre connaissance à priori du problème afin de réduire la complexité et augmenter les performances du système. Ces connaissances peuvent être extraites à partir des exemples d'entraînement, puis intégrées dans le système sous forme de règles floues.

V.5. k-plus proches voisins flous

Le k-plus proches voisins flous [52] est la version floue du k-plus proches voisins classique. Cet algorithme se base essentiellement sur l'équation suivante :

$$\mu_i(x) = \frac{\sum_{j=1}^{k} \frac{\mu_{ij}}{d\left(X, P_j\right)^{2/(m-1)}}}{\sum_{j=1}^{k} \frac{1}{d\left(X, P_j\right)^{2/(m-1)}}} \quad (97)$$

- ✓ $\mu_i(x)$: Le degré d'appartenance du vecteur x à la classe i.
- ✓ P_j : Le j^éme élément parmi les k plus proches voisins.
- ✓ m : Le facteur de fuzzification.
- ✓ μ_{ij} : Le degré d'appartenance du prototype P_j à la classe C_i. Ce degré est généralement égale à 1 si le prototype P_j appartient à la classe C_i, 0 sinon.

Nous reviendrons en détail sur cet algorithme dans les chapitres 3 et 5.

V.6. Comparaison des performances

Dans un problème de détection des défauts dans des tuyaux d'égout souterrains et à l'aide de 500 images de tuyaux, Sinha et al. [53] ont publié des résultats comparatifs entre l'algorithme des K-PPV, K-PPVF, PMC et une variante des SNF (Neauro-Fuzzy projection network). Les résultats obtenus pour la détection de trois types de défauts (D1, D2, D3) sont listés dans le tableau ci-dessous (Tableau 5). Ce tableau montre que les SNF aboutissent à des résultats bien meilleurs que les PMC, les K-PPVF et les K-PPV.

Tableau 5: Taux de reconnaissance en (%) obtenus par différents classifieurs

	k-PPV	k-PPVF	PMC	SNF
D1	81	84.6	87.2	94.1
D2	76.8	79.3	81.6	88.2
D3	80.3	82.9	84.5	91.8

Dans une comparaison entre différents systèmes neuro-flous pour la classification des cancers de prostate dans des images ultrasoniques, Lorenz et al. [54] ont reporté un taux de reconnaissance de 86.1% pour les systèmes ANFIS, 78.8% pour les réseaux de Kohonen Flous, 78.4% pour les K-PPV et 71.4% pour les réseaux Bayésiens.

Les systèmes flous ont été utilisés pour la détection des ports ouverts dans un serveur réseau par Shafiq et al. [55]. Dans cet article, une comparaison entre les SIF, RNA et les ANFIS a été réalisée, les résultats obtenus par les trois systèmes respectivement sont: 95.4%, 79% et 96.8%.

Dans une étude comparative entre les systèmes neuro-flous et les SVM dans différents domaines d'application Nayak et al. [56] ont obtenu une moyenne des taux de reconnaissance de 94.75% pour les système neuro-flous et 97.75% pour les SVM. Le même auteur a cité aussi que les systèmes neuro-flous sont plus complexes que les SVM et leur temps de classification est considérablement plus élevé.

Dans une comparaison entre les réseaux de neurones, les systèmes d'inférences flous pour la classification des chromosomes, Badawi et al. [57] ont obtenu un taux de reconnaissance de 94.76% pour le RNA et 93.54% pour SIF.

Chin-Teng Lin et al. [58] ont proposé une combinaison des SNF et les SVM qu'ils ont baptisé SVFNNC. Ces algorithmes ont été appliqués sur la base de données IRIS DATA, Vehicl et Dna de Statlog Collection. Le tableau suivant (Tableau 6) illustre les résultats obtenus.

Tableau 6: Taux d'erreurs en (%) obtenus par différents classifieurs pour différentes bases de données [58]

	ANFIS	SVM	SVFNNC
IRIS	4.3	3.3	4
Vehicl	29.9	1	1.42
Dna	16.6	4.21	5.1

VI. CONCLUSION

Dans un premier lieu nous avons présenté la théorie des sous-ensembles flous et des systèmes d'inférences floues qui sont à l'origine de tout système flou. Après, nous avons présenté les variantes floues de quelques algorithmes classiques, comme l'algorithme des c-moyenne floues et l'algorithme des k-PPVF.

Les systèmes flous sont très ouverts et interprétables par un être humain, donc facilement paramétrables par un expert. D'un autre côté, les réseaux de neurones sont des systèmes très puissants en matière de classification, mais se comportent comme des boîtes noires, donc leur

maintenance et leur adaptation à un problème particulier reste une tâche fastidieuse. De ce fait, une grande partie de ce chapitre a été dédiée aux techniques de fusion de ces deux approches. La première technique consiste à utiliser les réseaux de neurones pour déterminer et modifier les paramètres du système flou en se basant sur des exemples d'apprentissage au lieu d'un expert. Une autre technique, qu'on peut considérer à l'opposé de la première, consiste à utiliser le système flou pour réaliser une classification grossière et ensuite utiliser les réseaux de neurones pour prendre la décision finale. A la fin de ce chapitre nous avons cité quelques résultats comparatifs de ces méthodes de classification dans divers domaines.

Notre but dans ce travail de thèse est de réaliser un système de reconnaissance des manuscrits Tifinagh. Jusqu'à présent, nous n'avons parlé que des systèmes de reconnaissance de formes d'une manière générale. Dans le chapitre suivant, nous allons présenter les principaux composants d'un système de reconnaissance de caractères manuscrits, les résultats obtenus dans ce domaine avec une comparaison entre les différentes méthodes utilisées. Dans la dernière partie de ce chapitre, nous allons présenter une partie de notre contribution à la reconnaissance des caractères manuscrits Tifinagh par l'algorithme des c-moyennes floues et l'algorithme des k-PPVF en se basant sur une combinaison linéaire de deux méthodes d'extraction des caractéristiques.

DEUXIEME PARTIE

CONTRIBUTIONS

CHAPITRE 3. RECONNAISSANCE AUTOMATIQUE DES CARACTERES MANUSCRITS TIFINAGH

I. INTRODUCTION

Pour longtemps, la lecture des manuscrits était une caractéristique de l'être humain sans aucun concurrent. Les manuscrits existaient depuis des millénaires comme moyen de communication entre Hommes et comme une extension de la mémoire humaine. Dans le but de découvrir les secrets du cerveau humain, plusieurs recherches ont été consacrées pour imiter son intelligence. Un axe de ces recherches est dédié à la reconnaissance automatique des caractères imprimés et manuscrits.

Au début, la reconnaissance automatique des caractères (imprimés ou manuscrits) avait pour but de produire des machines capables de lire pour les aveugles [59]. La première machine (optophone) pour aider les aveugles était réalisée par Tyurin puis par Fournier d'Albe en 1912 [60]. Le concept du système OCR (Optical Character Recognition) est apparu après, grâce à Tausheck en 1929 suivi par Handel en 1933 [61], comme un système destiné uniquement à la reconnaissance des caractères imprimés.

Avec l'apparition des ordinateurs, la reconnaissance automatique des caractères a commencé une nouvelle ère. En 1954, Jacob Rabinow a réalisé une machine capable de reconnaître l'alphabet (majuscule) avec une vitesse d'un caractère par minute [62] . Vers les années 60, IBM a lancé des recherches dans le domaine des OCR triomphés par la commercialisation de leur premier système OCR nommé IBM's 1418 [61]. Ce système comme d'autres de cette époque n'avait pas la capacité de reconnaître des fonts variés. Après, une

nouvelle génération des systèmes OCR capable de reconnaître des manuscrits est apparue. Le système OCR " IBM's 1287 " était le premier système commercialisé de cette génération.

De nos jours et avec les avancées technologiques, la machine possède la capacité d'identifier, de reconnaître et d'interpréter un texte imprimé ou manuscrit. La reconnaissance est la transformation des symboles graphiques en un symbole qu'on peut stocker dans un système informatique sous forme de code ASCII sur 8 bits ou Unicode sur 16 bits. Interpréter veut dire qu'une machine peut déterminer le sens d'un mot ou une suite de mots (phrase). Identifier c'est l'identification de l'auteur d'un manuscrit.

Dans le domaine de la reconnaissance automatique des caractères, on distingue entre deux types de reconnaissances : la reconnaissance en ligne et la reconnaissance hors-ligne. Dans la reconnaissance en ligne, les manuscrits sont numérisés en temps réel à l'aide de média spécialisé (souvent un stylet et une surface électronique) où les coordonnées des points du symbole sont collectés avec le mouvement du stylet sur la surface. Dans la reconnaissance hors-ligne, le manuscrit est présenté au système sous forme d'une image numérisée à laquelle on applique un ensemble d'opérations avant de procéder à la reconnaissance.

II. CARACTERES MANUSCRITS TIFINAGH

Le Tifinagh est utilisé dans les manuscrits Amazighe depuis 600 ans AJ en Afrique du nord, au Sahel, aux Îles Canaries et à Siwa en Egypte. Au fil du temps, ce caractère a subi des modifications et des transformations du Libyque au Tifinagh Saharien et Tifinagh Touareg jusqu'au Néotifinagh. Ce dernier, le plus jeune, a été proposé et développé vers la fin des années 60 par l'Académie berbère, sur la base du Tifinagh Touareg, pour représenter les parlers Amazighes du Maghreb. Après, Au Maroc et depuis la création de l'Institut Royal de la culture Amazighe (IRCAM) en 2002, la standardisation et la normalisation de cette langue a eu un nouveau souffle. Grâce aux efforts de cet institut, la langue Amazighe possède désormais une orthographe officielle, des règles de grammaire, des règles de segmentation des chaînes parlées, un manuel de conjugaison, un vocabulaire grammatical, un codage propre dans le standard Unicode et un clavier Amazighe [63].

L'IRCAM a développé un système Tifinaghe-IRCAM dans le but de normaliser la graphie Amazighe. Le Tableau 7 présente le répertoire officiel de l'alphabet Tifinagh-IRCAM avec leurs correspondants en arabe et en latins.

Au début, l'Amazighe dans les anciennes inscriptions s'écrivait dans les quatre sens. L'orientation adoptée dans l'écriture Amazighe moderne est horizontale de gauche à droite.

L'informatisation de la langue Amazighe a passé par plusieurs étapes : le codage des caractères Tifinagh par l'ASCII étendu, la création des polices de caractères Tifinagh puis vient le codage dans le standard Unicode et l'élaboration des normes pour la disposition du clavier Amazighe. Dans cette dernière étape, l'IRCAM a proposé deux types de claviers, l'un pour la saisie uniquement des 33 lettres de l'alphabet de base, l'agencement de ce clavier est illustré dans la figure (Figure 26) et l'autre pour la saisie de l'alphabet de base plus 22 lettres de l'alphabet Tifinagh étendu (Figure 27).

Le texte Amazighe peut alors être utilisé dans n'importe quelle application informatique. En particulier, on peut utiliser n'importe quel outil de traitement de texte avancé pour éditer et

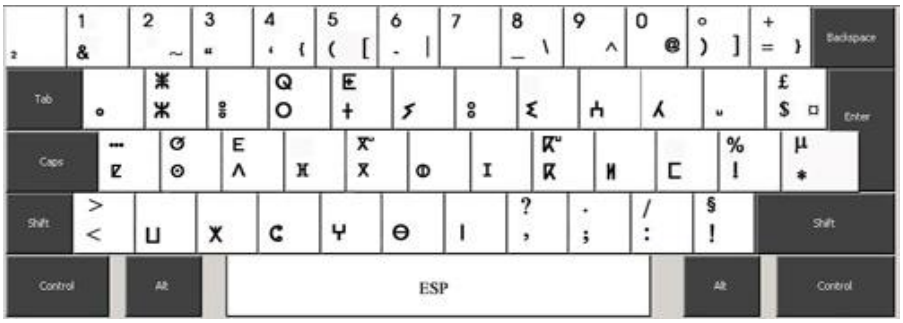


Figure 26 : Clavier Tifinagh de base

mettre en forme un texte Amazighe en Tifinagh et produire des documents de bonne qualité.

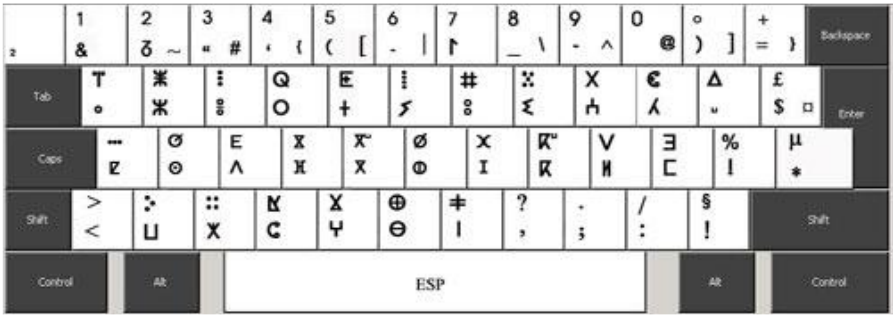


Figure 27 : Clavier Tifinagh étendue

Tableau 7 : Répertoire officiel de l'alphabet Tifinaghe-IRCAM avec leurs correspondants en arabe et en latins

	TIFINAGHE	Correspondance latine	Correspondance arabe	Exemples
ya	ⵄ	a	ا	ⵄⴰⵄⵄ
yab	ⴱ	b	ب	ⵄⴱⵄⵉⴰ
yag	ⴳ	g	گ	ⵜⵉⵅⵔⵔⵉ
yagʷ	ⴳⵯ	gʷ	گ،	ⵄⵣⵅⵅᵘⵄⵅ
yad	ⴷ	d	د	ⵄⵀⵙⴰ
yaḍ	ⴹ	ḍ	ض	ⵄⴹⵄⵟ
yey	ⵢ	e		ⵄⵀⵟⵟⵙⵢⵢⵙ
yaf	ⴼ	f	ف	ⵄⵀⵙⵙ
yak	ⴽ	k	ک	ⵜⵉⵙⵓⵔⵙⵜ
yakʷ	ⴽⵯ	kʷ	ک،	ⵄⵔⴰⴰⵄⵓⴽⵓⵀ
yah	ⵀ	h	ه	ⵄⵀⴰⴰⵙⵏ
yaḥ	ⵃ	ḥ	ح	ⵄⵃⵍⵄⵛ
yaɛ	ⵄ	ɛ	ع	ⵄⵄⵜⵙⵏ
yax	ⵅ	x	خ	ⵜⵉⵅⵅⵙⵉ
yaq	ⵇ	q	ق	ⵄⵇⵄⵙ
yi	ⵉ	i	ي	ⵉⵔⵉ
yaj	ⵊ	j	ج	ⵄⵢⵏⵊⵄ
yal	ⵍ	l	ل	ⵄⵔⵍⵄⵍ
yam	ⵎ	m	م	ⵄⵔⵄⵏ
yan	ⵏ	n	ن	ⵉⵙⵓⵃⵏ
yu	ⵓ	u	و	ⵙⵃⵔ
yar	ⵔ	r	ر	ⵙⵙⵄⵄ
yaṛ	ⵕ	ṛ	ڑ	ⵙⵜⵕⵕⵄ
yaɣ	ⵖ	ɣ	غ	ⵄⵢⵙⵙⵔ
yas	ⵙ	s	س	ⵉⵍⵙ
yaṣ	ⵚ	ṣ	ص	ⵜⵄⵔⵚⵕⵇⵉⵖⵜ
yac	ⵛ	c	ش	ⵄⵛⵍⵍⵄⵍ
yat	ⵜ	t	ت	ⵜⵄⵀⵙⵄⵙⵜ
yaṭ	ⵟ	ṭ	ط	ⵜⵉⵟⵟ
yaw	ⵡ	w	ۇ	ⵄⵡⵄⵍ
yay	ⵢ	y	ۋ	ⵜⵄⵖⵙⵄⵄ
yaz	ⵣ	z	ز	ⵄⵔⵄⵣⵉⵢ
yaẓ	ⵥ	ẓ	ژ	ⵄⵣⵄⵍⵉⵔ

La reconnaissance automatique des caractères Tifinagh (imprimés ou manuscrits) utilisés dans la langue Amazighe est l'un des sujets d'actualité qui a été abordé par un ensemble limité de travaux. Les paragraphes qui suivent, donnent un survol de quelques travaux réalisés dans ce domaine.

Un système de reconnaissance hybride composé des réseaux de neurones et les modèles de Markov cachés a été élaboré par El Kessab et al [64]. Le principe utilisé pour l'extraction des caractéristiques consiste à extraire cinq images de même dimension que l'image originale du caractère en question à l'aide de cinq opérations de dilatation vers le haut, le bas, la gauche, la droite et le centre. Le nombre des pixels de valeur 1 dans chaque image constituent alors les éléments du vecteur caractéristique. Ce vecteur est donné en entrée d'un perceptron multicouche modifié de telle sorte que ces sorties soient des probabilités d'appartenance du caractère aux différentes classes. Ces probabilités sont alors utilisées comme des probabilités d'émission d'un MMC qui donne la sortie finale du système. Ce système a été testé sur 18 classes de caractères Tifinagh manuscrits et imprimés. Le tableau (Tableau 8) illustre les résultats obtenus.

Tableau 8: Taux de reconnaissance en (%) obtenus par différents classifieurs [64]

	PMC	HMM	MCP+HMM
Caractères Manuscrits	84.91	92.22	92.30
Caractères imprimés	63.83	70.00	87.77

El Ayachi et al. [65] ont utilisé les réseaux de neurones pour la roconnaissance des caractères manuscrits Tifinagh. La transformation de Walsh a été adopté comme méthode d''extraction des caractèristiques. Ils ont obtenu un taux de reconnaissance de 93.52% avec un perceptron constitué d'une seule couche cachée. Dans un autre article [66] ils ont donné un tour d'horizon sur la reconnaissance automatique des scripts Tifinagh.

Amrouch et al. [67] ont utilisé les modèles de Markov cachés discrets dans la phase de classification. La méthode d'extraction utilisée consiste à utiliser la transformée de Hough de l'image avec un taux de déplacement de 30°. Les lignes obtenues sont alors inclinées de 0°, 30°, 60°, 90°, 120°, 150° et 180°. Ensuite l'image est scindée en 256 zones. Les observations utilisées par les MMCs sont Les directions dominantes dans chacune de ces zones. Dans cette approche, chaque classe de caractères est représentée par un MMC. Ainsi, un caractère inconnu sera affecté à la classe dont le MMC représentant réalise le meilleur score. Le taux de reconnaissance obtenu avec cette démarche est de 97.89%.

Abaynagh et al. [68] ont proposé une méthode d'extraction des caractéristique améliorée à base des moments de Legendre optimisés par le principe du maximum d'entropie. Cette

méthode est utilisée avec un réseau de neurones d'une seule couche cachée pour la reconnaissance des caractères manuscrits Tifinagh. Le taux de reconnaissance obtenu avec cette méthode d'extraction des caractéristiques était de 97.77% contre 94.68% obtenu par l'utilisation de la méthode des densités des pixels et 95.23% par la méthode de la distance euclidienne.

Aharrane et al. [69] ont proposé une méthode d'extraction des caractéristiques des caractères manuscrits Tifinagh basée sur la méthode de zoning. Le taux de reconnaissance obtenu à l'aide d'un perceptron multicouche est de 96.47%.

On peut citer aussi les traveaux de Fakir et al [70] qui ont utilisé la géométrie Riemanniennes pour l'extraction des caractéristiques, les SVM et les réseaux de neurones pour la classification. Une étude comparative entre les arbres de décision et les réseaux de neurones en utilisant les distances geodisique a été proposée par Bencharef et al [71]. La théorie des graphes a été utilisée par Boutaounte et al [72]. Gounane et al [73] ont publié une étude comparative entre l'algorithme des k-plus proches voisins flous et les cartes auto-organisatrices. Une approche syntaxique par les automates finies appliquée sur les caractères imprimés a été proposée par Es Saady et al [74]. Une base d'exemples des caractères manuscrits Tifinagh pour la recherche (AMHCD) a été construite par Es Saady et al [75]. Le même auteur a proposé une autre approche utilisant les lignes de base pour l'extraction des caractéristiques [76] .

Dans ce chapitre, nous présenterons une nouvelle approche pour la reconnaissance automatique des caractères manuscrits Tifinagh [77]. Cette approche s'articule essentiellement autour de trois axes ; une combinaison linéaire de deux méthodes d'extraction des caractéristiques, un classifieur flou et le modèle Bi-grams du langage Amazighe pour introduire le contexte afin d'améliorer le taux de reconnaissance.

Ainsi, notre système a besoin en entrée d'un mot entier au lieu d'un caractère isolé. La décision finale est alors prise en utilisant les degrés d'appartenance du caractère aux différentes classes possibles fournies par le classifieur flou et les informations concernant les caractères situés avant et après le caractère en question. Cette technique a montré son efficacité surtout pour les caractères très ambigus que même un être humain ne peut identifier en absence du contexte.

III. COMPOSANTES D'UN SYSTEME DE RECONNAISSANCE DE CARACTERES

Un système de reconnaissance de caractères est souvent constitué d'un ensemble de composantes : Numérisation, prétraitement, Segmentation, extraction des caractéristiques et la reconnaissance.

III.1. Prétraitement

III.1.a. Binarisation

Après la numérisation de l'image, elle est souvent transformée en une image au niveau de gris. Après cette transformation, l'intensité de chaque pixel varie entre 0 et 255. Dans plusieurs travaux, on trouve une tendance à utiliser des images en noir et blanc plutôt que d'utiliser des images en niveau de gris [78]. Passer d'une image au niveau de gris à une image en noir et blanc s'appelle la binarisation ou seuillage. Elle consiste à fixer un seuil. Les pixels possédants une intensité supérieure à ce seuil sont considérés comme des pixels blancs et le reste sont considérés comme des pixels noirs.

Il existe plusieurs algorithmes pour la détermination du seuil de binarisation. Ces algorithmes peuvent être classés en deux catégories : des méthodes globales et des méthodes localement adaptatives [78]. Dans les méthodes globales, un seul seuil est calculé pour l'image entière [79] . Dans la plupart des cas, les méthodes globales sont insatisfaisantes à cause de l'uniformité dans l'arrière-plan ou dans l'objet représenté dans l'image. Par conséquence, plusieurs seuils doivent être utilisés. Pour ce type, la binarisation est appelée seuillage adaptatif (adaptive thresholding). On calcule le seuil suivant les pixels adjacents pour chaque pixel [78]. Les méthodes globales présentent l'avantage de la rapidité dans l'exécution, mais donnent des faibles résultats pour des images de faible contraste avec un arrière-plan bruité et d'intensité variable.

Seuillage Globale

La méthode la plus intuitive dans ce type de binarisation consiste à utiliser l'histogramme des intensités de pixels de l'image. Un histogramme est une fonction qui lie chaque valeur du niveau de gris au nombre de pixels possédant cette valeur dans l'image. Une image contenant un caractère et un arrière-plan bien identifiable aura un histogramme formé de deux pics. La valeur minimale comprise entre ces deux pics sera prise comme une valeur pour le seuillage (Figure 28). La méthode d'Otsu est l'une des méthodes les plus utilisées pour ce type de binarisation [80] [81].

Seuillage adaptatif

Ce type de méthodes se base sur l'analyse des niveaux de gris dans des fenêtres locaux pour la détermination des seuils locaux de binarisation. Le problème majeur dans ces méthodes est de choisir la taille de ces fenêtres. Il doit être suffisamment large pour contenir suffisamment de pixels de l'arrière-plan afin de bien estimer leur moyenne, d'une part. D'autre part cette taille ne doit pas être très large pour ne pas contenir des portions d'arrière plans d'intensités différentes.

Une des méthodes les plus utilisées pour ce type de binarisation est la méthode de Niblack [80]. Elle se base sur le calcul de la moyenne locale et la déviation standard locale comme le montre l'équation suivante :

$$T(x,y) = m(x,y) + k * s(x,y) \tag{98}$$

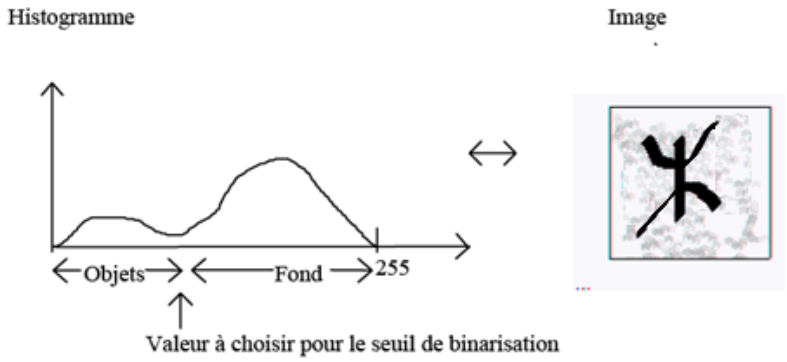


Figure 28 : Binarisation Globale

Où m(x, y) est la moyenne, s(x, y) est la déviation standard dans la fenêtre.

III.1.b. Détection et Correction d'inclinaison

En général, un document texte est sous forme d'un ensemble de ligne de texte. Normalement les lignes possèdent un angle d'inclinaison nulle. Mais après la numérisation, à l'aide d'un scanner ou un appareil photo ou autres, un angle d'inclinaison non nulle peut être introduite. D'où la nécessité d'appliquer une technique de correction d'inclinaison.

Ils existent plusieurs méthodes pour la détection d'inclinaison dans la littérature. L'une des méthodes la plus utilisée est la méthode des histogrammes de projection [82] [83] . Avec un angle d'inclinaison convenable, l'histogramme possède un pic pour chaque ligne (Figure 29), les valeurs minimales de cet histogramme sont les interlignes. Ils existent d'autres variantes de cette méthode qui minimisent le temps de calcule et augmentent la précision.

Une autre méthode très intéressante se base sur l'application de la transformée de Hough aux centres de gravités des composante connexes [84] [85]. Après la détection de l'angle d'inclinaison du document, on peut appliquer un algorithme pour la rotation de l'image.

III.2. Segmentation

La segmentation du texte d'un document se fait en trois étapes : la segmentation en lignes, en mots puis en caractères.

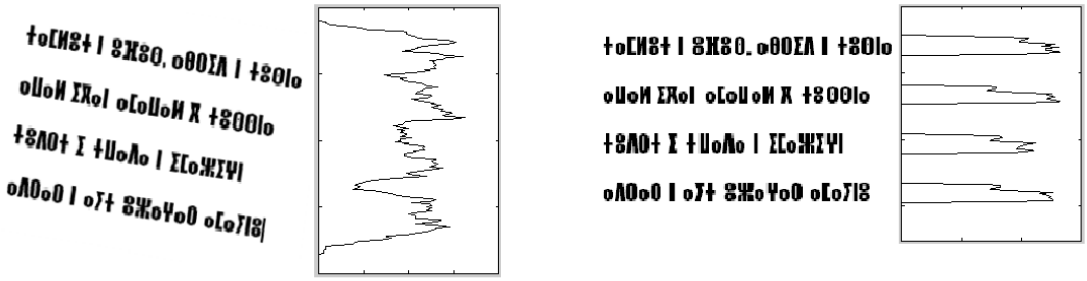


Figure 29 : Méthode de projection horizontale pour la détection d'inclinaison (a) image inclinée (b) après correction d'inclinaison

III.2.a. Segmentation en lignes

On désigne par la segmentation en ligne, la décomposition de l'image d'un document texte en un ensemble d'images de lignes. Dans la littérature on peut trouver plusieurs méthodes, la plus ancienne et d'ailleurs la plus simple, utilise l'histogramme de la projection horizontale (Figure 30).

Cette méthode nécessite au préalable la correction d'inclinaison, les interlignes correspondent aux minimas locaux de l'histogramme [86]. Le principal inconvénient est que dans certaines langues comme la langue Arabe, une ligne de texte est munie d'un ensemble de points, accents et diacritiques, dans ce cas ces symbole sont détectés comme des lignes. D'autres méthodes utilisent l'algorithme des k-plus proches voisins pour classer les caractères en lignes. D'autres utilisent la transformée de Hough [87] [88].

III.2.b. Segmentation en mots

Une méthode intuitive pour la segmentation d'une ligne de texte, en un ensemble de mots, repose sur le fait que l'espace entre les mots est souvent plus grand que l'espace entre les caractères. L'histogramme de la projection verticale peut être utilisé aussi comme le montre la figure (Figure 31).

III.2.c. Segmentation en caractères

La segmentation d'un mot en caractère est une phase de grande importance lorsqu'il s'agit de reconnaissance automatique des caractères isolés. Une mauvaise segmentation affecte

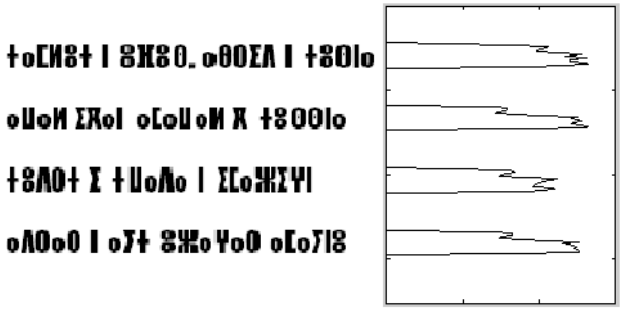


Figure 30 : Méthode de projection horizontale pour la segmentation en ligne

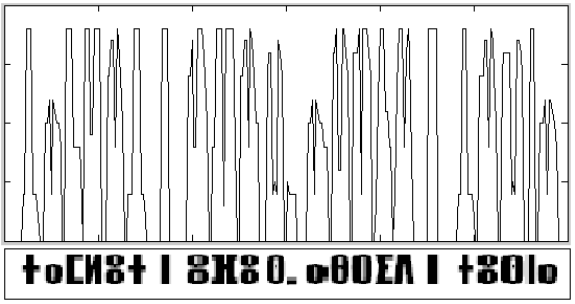


Figure 31 : Méthode de projection verticale pour la segmentation en mots

énormément les performances du système. Dans cette phase, on décompose l'image du mot en un ensemble d'images de caractères isolés. Pour cette fin, il faut d'abord tracer l'histogramme de la projection verticale du mot (Figure 32) et ensuite segmenter ce mot suivant les valeurs nulles dans cet histogramme.

III.3. Squelettisation

A l'origine, le modèle de l'écriture est un modèle linéaire sans épaisseur. Dans ce contexte, la squelettisation consiste à éliminer l'épaisseur du trait ou plutôt le mincir jusqu'à l'épaisseur minimale d'un pixel. L'avantage de la squelettisation est la conservation d'un ensemble des caractéristiques géométriques et topologiques du caractère. En même temps, elle réduit les bruits incorporés dans l'image du caractère et réduit aussi la taille de l'image du caractère [89] [90].

Dans la littérature on trouve plusieurs algorithmes de squelettisation. Ces algorithmes peuvent être classés en trois types : séquentiels, parallèles et non itératifs [90].

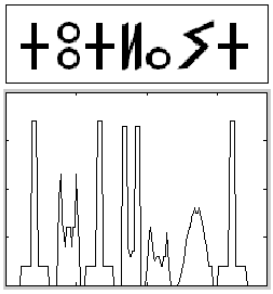


Figure 32 : Méthode de projection verticale pour la segmentation en caractères

1. Les algorithmes séquentiels : ils examinent les pixels du contour dans une image dans un ordre prédéterminé et marquent les points à supprimer. La conservation ou la suppression d'un pixel du contour dépend des pixels voisins dans les huit directions. Pour éviter la suppression séquentielle d'une branche entière, on marque les pixels à supprimer et à la fin de l'itération tous les pixels marqués seront supprimés. De cette manière, l'algorithme garantit qu'une seule couche des pixels sera supprimée par itération. Pour marquer un pixel pour la suppression, il doit être d'abord un pixel noir, non isolé et doit appartenir au contour du caractère avec au moins 4 pixels voisins blanc.
2. Les algorithmes parallèles : Dans ce type d'algorithme, une fenêtre 3×3 est utilisée pour examiner les pixels voisins du pixel en question. L'existence d'un pixel noir dans la $n^{éme}$ itération dépend de son état et à celles des 8 pixels voisins à la $(n-1)^{éme}$ itération. Cela permet aux pixels d'être traités simultanément.
3. Les algorithmes non itératifs : Dans les deux types précédents, on marque itérativement les pixels à supprimer. A l'opposé de ces deux méthodes, les méthodes non itératives se basent sur le parcours d'une ligne entre le contour externe et le contour interne du caractère. Ce type de méthodes est robuste face aux bruits, plus rapide et plus précis que les deux types précédents [91]. Malgré leurs avantages multiples, les méthodes de squelettisation non itératives souffre de l'incapacité de préserver plus de détails sur le squelette du caractère.

La figure (Figure 33) montre un exemple de l'image d'un texte manuscrit à laquelle nous avons appliqué une méthode de seuillage adaptatif puis l'algorithme de squelettisation de zhang-suen [92].

III.4. Normalisation

Dans le prétraitement, on trouve aussi la normalisation. Un caractère manuscrit peut varier en taille, en inclinaison, en rotation, en épaisseur des traits, etc. La normalisation consiste donc à donner une taille uniforme aux caractères, à corriger les inclinaisons, à donner aux caractères une épaisseur uniforme égale à l'unité, etc.

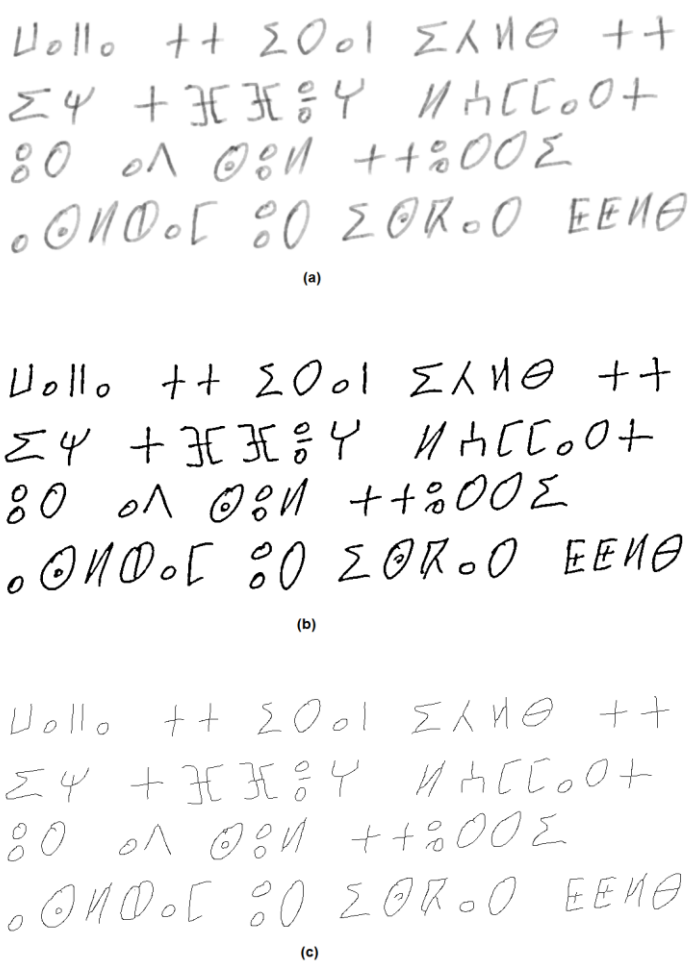


Figure 33: (a) Image originale (b) Image après seuillage adaptatif (c) après squelettisation par l'algorithme de Zhang-Suen

Plusieurs techniques de redimensionnement existent qu'on peut trouver dans plusieurs articles. Une technique standard consiste à modifier les coordonnées cartésiennes dans l'image d'origine par une multiplication par une constante [79] puis on utilise une des techniques d'interpolation pour remplir les pixels dans l'image redimensionnée [93].

III.5. Extraction des caractéristiques

L'extraction des caractéristiques joue un rôle décisif dans un système de reconnaissance automatique de caractères. Cette opération a pour but l'extraction des informations caractérisant l'image d'un caractère pour une meilleure classification. Dans la littérature, on trouve

différentes méthodes pour l'extraction de ces caractéristiques. Ces méthodes peuvent être classées en deux types [94] :

1. Des caractéristiques globales
2. Des caractéristiques structurelles

III.5.a. Caractéristiques globales

Les caractéristiques globales sont extraites de la totalité des pixels composant l'image d'un caractère. Elles ne sont pas sensibles aux bruits. Le problème avec ce type de caractéristique réside dans l'utilisation de la totalité des pixels d'où les vecteurs caractéristiques sont de grandes dimensions. Pour remédier à ce problème, des techniques basées sur la distribution des pixels sont adoptées. Parmi ces techniques on trouve :

Les moments :

Dans la littérature on trouve plusieurs types de moments. On peut calculer les moments bruts qui sont en fonction des coordonnées des pixels de l'image [95]. Une autre méthode plus élaborée basée sur les moments centraux. Ces moments sont calculés en utilisant les distances entre les pixels et le centre de gravité du caractère. Ces types de moment ont l'avantage d'être invariants aux translations et produisent un taux de reconnaissance plus élevé que celui obtenu en utilisant le type précédent.

Zoning :

Dans cette méthode, l'image du caractère est subdivisée en plusieurs zones. Les densités des pixels noirs dans chaque zone sont alors calculées et utilisées comme caractéristiques [96].

Croisement et distances :

Cette technique utilise le nombre de fois que le caractère se croise avec des vecteurs de différentes directions. Toujours dans la catégorie des caractéristiques globales, on trouve les transformations et les séries. Ces descripteurs sont invariants à la rotation, la translation et à la taille tout en réduisant la dimension du vecteur caractéristique. Parmi ces descripteurs on peut citer : la transformée de Fourier, les séries de Karhunen-Loeve, la transformée de Hough… etc

III.5.b. Caractéristiques structurelles ou topologiques

Dans cette catégorie, l'extraction des caractéristiques se base sur la structure et la topologie du caractère. Ces caractéristiques peuvent représenter l'aspect global et local du caractère. Elles représentent les segments, les points, les intersections des segments, les boucles ainsi que d'autres éléments composant l'allure d'un caractère. D'autres caractéristiques de ce type peuvent être extraites du contour d'un caractère [97] .

Une autre approche consiste à extraire la chaîne des codes plus d'autres informations concernant la localisation, l'orientation et la cavité du contour extérieur du caractère. Cette technique a été utilisée par Cai et Liu en utilisant les modèles du Markov cachés et a donné d'excellents résultats pour la reconnaissance automatique des chiffres [98].

Dans notre système, nous avons utilisé deux méthodes pour l'extraction des caractéristiques : La méthode des densités de pixels et la méthode de la distance du centre de gravité. Ces deux méthodes sont toutes de type zoning et sont simples et ne demandent pas une grande capacité de calcul.

Dans les deux méthodes, l'image du caractère est subdivisée en L×H zones. Chaque zone peut contenir une partie du caractère ou non. La méthode des densités de pixels consiste à extraire le nombre de pixels noirs de chaque zone. Le vecteur caractéristique est alors constitué de ces nombres normalisés par le nombre total des pixels dans une zone (Figure 34).

Dans la méthode des distances des centres de gravité, dans chaque zone on calcule la distance du centre de gravité de la portion du caractère qu'il contient au coin gauche en bas de

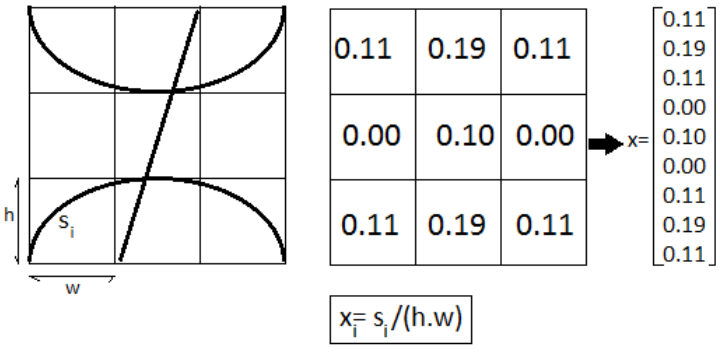


Figure 34 : Méthode des densités de pixels

la même zone. Le vecteur caractéristique est alors constitué de ces distances normalisées par la diagonale d'une de ces zones (Figure 35).

On peut facilement constater que les vecteurs caractéristiques obtenus par ces deux méthodes sont normalisés et leurs éléments sont tous compris entre 0 et 1 et sont invariants à la taille de l'image du caractère.

Le classifieur est alimenté par une combinaison linéaire des deux vecteurs caractéristiques comme le montre l'équation suivante :

$$v = \gamma v_g + (1-\gamma)v_d \quad 0 \leq \gamma \leq 1 \tag{99}$$

Le recours à l'utilisation de cette combinaison est expliqué par le besoin d'un vecteur caractéristique contenant les informations représentées par les deux vecteurs. Le vecteur caractéristique obtenu à l'aide de la première méthode véhicule des informations concernant le taux de pixel occupés par chaque fragment du caractère. Alors que la seconde méthode offre des informations concernant les positions de ces fragments dans leurs zones. Il faut mentionner qu'il n'existe aucune méthode précise pour le choix des paramètres L, H et γ. Ils sont déterminés expérimentalement comme nous allons voir dans la suite de ce chapitre.

Finalement, il faut noter que l'extraction des caractéristiques du contour ou du squelette d'un caractère présente des avantages et des inconvénients. Dans l'extraction des caractéristiques du contour, on risque des redondances d'information à cause de l'utilisation du contour extérieur et intérieur d'un caractère, alors que ce problème ne se pose pas lors de

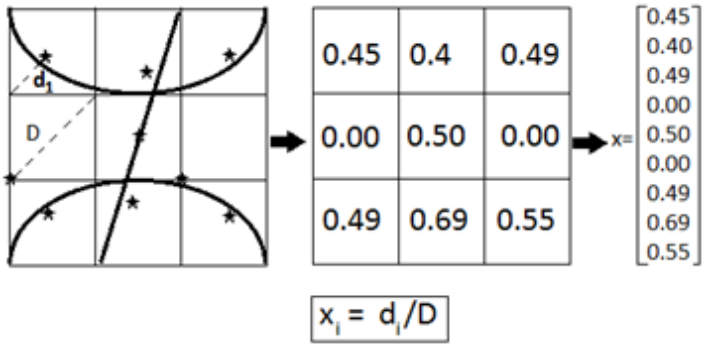


Figure 35 : Méthode des distances des centres de gravité

l'utilisation du squelette [99]. D'autre part, après la squelettisation, des informations pertinentes peuvent disparaitre.

Dans cette section, nous avons présenté un ensemble de techniques utilisées pour l'extraction des caractéristiques. D'autres techniques existent dans la littérature. Pour plus de détails, le lecteur est invité à lire une vue d'ensemble de ces techniques dans Suen et al. [100]. Dans la section suivante, nous allons présenter un ensemble de techniques de classification des caractères manuscrits.

III.6. Classification

Dans le chapitre 2, nous avons présenté les différents types de classifieurs utilisés dans le domaine de la reconnaissance de formes en général. Cette section sera consacrée à l'application de ces classifieurs en matière de reconnaissance des caractères manuscrits.

Après la segmentation, qui s'intéresse à l'extraction des images de caractères isolés, on procède à l'extraction des caractéristiques de ce caractère. Ces caractéristiques permettent alors au classifieur de prendre une décision d'appartenance de ce caractère à une classe particulière des caractères. Les performances de classification et le choix du classifieur sont très dépendants de la méthode d'extraction des caractéristiques utilisée. Un classifieur a toujours besoin d'un ensemble de données pour son entraînement avant de procéder à la classification.

Avec des vecteurs caractéristiques extraits en utilisant l'une des méthodes dites globales, on utilise souvent un classifieur statistique [99]. Dans ce type de classifieurs on peut citer les classifieurs utilisant l'appareillement de forme ou bien ceux utilisant la règle de Bayes…etc.

Lorsqu'on utilise des méthodes structurelles pour l'extraction des caractéristiques, l'utilisation des classifieurs syntaxiques est souhaitable. Lorsqu'un caractère est présenté à un classifieur de ce type, il sera comparé avec un ensemble de règles décrivant chacune des classes. D'autres classifieurs sous forme d'arbre de décision peuvent être utilisés. Cet arbre est sous forme d'un ensemble de sous-classes. Chaque sous-classe regroupe des caractères possédants une propriété particulière. Pour un caractère donné, l'arbre est parcouru du haut vers le bas en suivant le chemin qui s'accorde avec toutes les caractéristiques de ce caractère.

Plusieurs travaux ont abordé la reconnaissance des caractères manuscrits en utilisant les réseaux de neurones [101] [102]. Un réseau de neurone se compose d'un ensemble de couches de neurones interconnectés. Chaque neurone calcule une somme pondérée de ses entrées qui sera ensuite transformée à l'aide d'une fonction linéaire ou non linéaire nommée fonction d'activation. Pour qu'un réseau de neurones puisse classifier correctement ses entrées, une phase d'entraînement doit être effectuée au préalable. Ils existent plusieurs algorithmes pour l'entraînement de réseaux de neurones. Le rôle de ces algorithmes est la modification des poids d'interconnexion entres les neurones jusqu'à l'atteinte d'un critère préalablement défini.

D'autres techniques de classification consistent à utiliser des classifieurs simples pour les entrées facilement identifiables dans un premier lieu, puis utiliser des classifieurs plus complexes et performants pour les entrées ambigües [103] [104] [105] [106].

Finalement, nous présentons une comparaison des différents classifieurs qu'on peut rencontrer dans plusieurs travaux. Lee a comparé le k-PPV, les RNA et les RBF pour la reconnaissance automatique des chiffres [107]. Il a abouti au fait que les trois classifieurs ont des performances semblables où la RBF en tête avec un taux d'erreur de 4.77% suivi par le k-PPV avec un taux d'erreur de 5.14% puis les réseaux de neurones avec taux de 5.15%.

On peut alors conclure que les réseaux de neurones et l'algorithme des k-PPV présentent les meilleurs résultats pour la reconnaissance automatique des chiffres manuscrits. L'un peut être plus performant que l'autre suivant la méthode utilisée pour l'extraction des caractéristiques. Dans [108], on trouve que les RNA sont les plus performants alors que dans [109] la méthode des k-PPV est la meilleure.

Il faut noter qu'une telle comparaison n'est pas toujours faisable surtout lorsqu'il s'agit d'un ensemble de données d'entraînement différent pour chaque méthode. En outre, pour le choix d'un classifieur il faut aussi prendre en considération l'usage de mémoire et le temps de calcul. Dans ce contexte, la méthode des k-PPV présente l'inconvénient d'utiliser plus de mémoire que les RNA, Alors que ces derniers présentent l'inconvénient de consommer un temps considérable pendant la phase d'entraînement.

III.7. K-PPVF pour la reconnaissance automatique des caractères manuscrits Tifinagh

Comme nous l'avons déjà présenté dans le chapitre 2, l'algorithme des K-PPVF est une méthode de classification supervisée. Cette méthode est appliquée en plusieurs applications dans différents domaines; data mining, reconnaissance de formes, traitement d'image et autres [110] [111].

Avec l'algorithme de k-PPV, on détermine les k vecteurs les plus proches du vecteur inconnu x. Le vecteur x appartient à la classe possédant plus de représentant parmi les k plus proches de x. Dans la figure (Figure 36), on remarque que le vecteur x est classé avec l'ensemble représenté par des points alors que sa classe intuitive est celle représentée par des étoiles. Ce problème est dû à deux facteurs majeurs :

1. la présence des intrus (outliers) : Un intrus est un vecteur mal étiqueté dans l'ensemble des exemples présentés au classifieur, ou bien la méthode utilisée pour l'extraction des caractéristiques produit un vecteur semblable aux vecteurs caractéristiques d'une autre classe.
2. L'absence d'utilisation de la proximité des k exemples du vecteur x. La majorité des k-plus proches voisins appartenant à la même classe sont plus loin du vecteur x que les autres. Dans la figure (Figure 36), les deux vecteurs de type étoiles sont très proches du vecteur x, donc ces trois vecteurs sont très semblables, par conséquent, et d'une façon intuitive, ce vecteur doit être classé parmi les éléments de type étoiles.

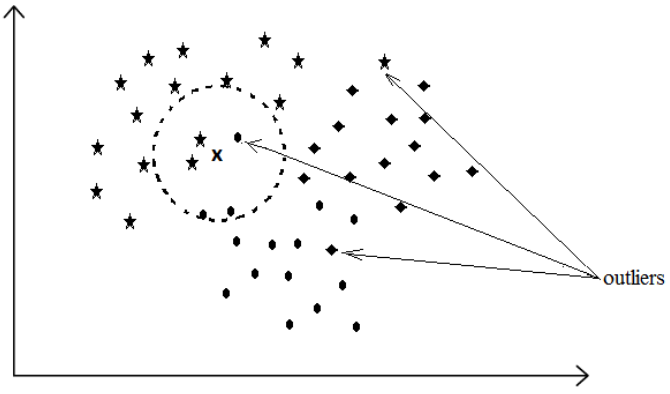


Figure 36: Anomalie produite par l'utilisation des k-PPV

Dans un premier lieu, pour éliminer ces intrus (outliers) et pour réduire le nombre d'exemples utilisés pour la classification, nous avons appliqué l'algorithme des c-moyennes floues sur les éléments de chaque classe afin de désigner un ensemble de prototypes pour représenter chaque classe (Figure 37). Cet ensemble de prototypes est le nouvel ensemble d'exemples utilisés pour la classification. Dans ce nouvel ensemble, on remarque l'absence des intrus ainsi que la réduction dans le nombre de ses exemples d'entrainement (Figure 38).

Pour remédier au second problème, nous avons utilisé la version floue de l'algorithme des K-PPV. Cet algorithme est caractérisé par la prise en considération des distances entre le vecteur inconnu x et les k plus proches voisins. Cette caractéristique est décrite dans l'équation (97) du chapitre 2.

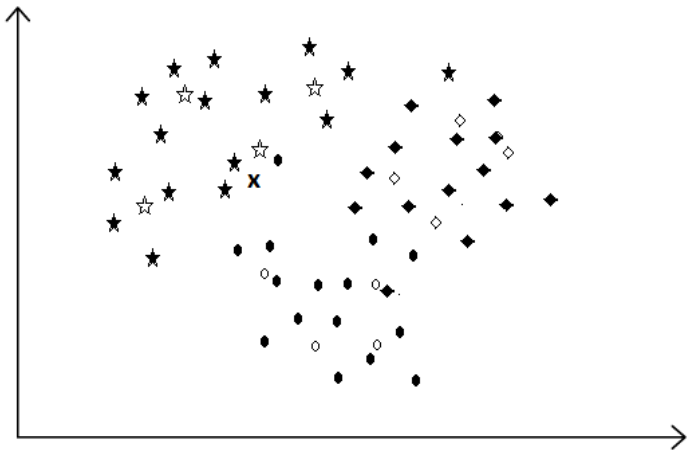


Figure 37: Détermination des prototypes par l'algorithme des c-moyennes flou

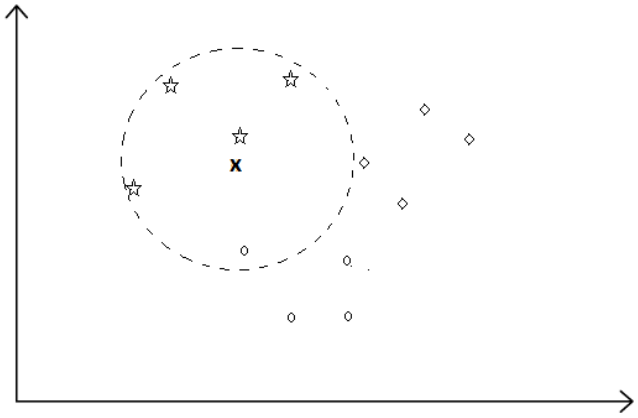


Figure 38 : Après détermination des prototypes par l'algorithme des c-moyennes floues

A l'aide de ces deux techniques, nous avons arrivé à extraire de la base de données des caractères manuscrits Tifinagh un ensemble réduit de prototypes contenant les informations discriminantes représentées dans la totalité des exemples tout en éliminant les mauvais exemples qui peuvent affecter la décision du classifieur.

Tout de même, il faut bien choisir le nombre des sous-classes utilisées dans l'algorithme des c-moyennes floues, sinon en risque d'avoir une perte d'informations importantes représentées par les exemples originaux.

Dans notre approche, nous avons utilisé une base de données de caractères Tfinagh manuscrits, dans cette base de données existe un ensemble de caractères difficilement identifiables, d'autres ressemblent à des caractères d'une autre classe que celle à laquelles ils appartiennent. Après l'utilisation de l'algorithme des c-moyennes floues, pour chaque classe on détermine un ensemble de prototypes représentant des sous-classes. Ces prototypes sont alors utilisés par l'algorithme des k-PPVF lors de la phase de reconnaissance.

Après la segmentation d'un mot manuscrit. Les vecteurs caractéristiques de chaque caractère isolé seront présentés au classifieur. Ce dernier propose un ensemble de candidats pour chacun de ces caractères muni d'un degré d'appartenance à chacune des classes existantes. On ne garde que les candidats munis d'un degré d'appartenance non nul.

Le tableau (Tableau 9) montre un exemple de reconnaissance des caractères dans le mot "ⵎⵓⵔⴰ". On remarque que pour chaque image de caractère, le classifieur donne les degrés d'appartenance de ce caractère à chacune des classes existantes. La majorité de ces degrés sont nuls. Normalement, avec ce type de classifieur, la déscision finale consiste à prendre comme résultats les candidats avec les degrés d'appartenance maximaux. Le résultat sera donc le mot "ⵎⵓⵕⴰ". Le classifieur utilisé a donc proposé le caractère 'ⵕ' avec un degré d'appartenance de 0.5962 à la place du caractère 'ⵔ' muni du degré d'appartenance 0.4124. Grâce à ces deux degrés d'appartenance, on peut bien remarquer que le classifieur exprime une sorte d'ambiguïté vu que les deux valeurs sont 'approximativement' les mêmes.

Tableau 9: Degrés d'appartenance des caractères du mot « ΣHoOl » aux différentes classes

	Σ	H	o	O	l
o	0.00	0.00	**0.9004**	0.00	0.00
Ж	0.00	0.00	0.00	0.00	0.00
Ж	0.00	0.00	0.00	0.00	0.00
O	0.00	0.00	0.00	0.00	0.00
Q	0.00	0.00	0.00	0.00	0.00
+	0.00	0.00	0.00	0.00	**0.4831**
Ε	0.00	0.00	0.00	0.00	0.00
Y	0.00	0.00	0.00	0.00	0.00
oo	0.00	0.00	**0.0995**	0.00	0.00
Σ	**0.7427**	0.00	0.00	0.00	0.00
Λ	0.00	0.00	0.00	0.00	0.00
Ч	0.00	0.00	0.00	0.00	0.00
Z	**0.2572**	0.00	0.00	0.00	0.00
O	0.00	0.00	0.00	**0.4124**	0.00
Ø	0.00	0.00	0.00	**0.5062**	0.00
Λ	0.00	0.00	0.00	0.00	0.00
E	0.00	0.00	0.00	0.00	0.00
H	0.00	**0.9186**	0.00	0.00	0.00
X	0.00	0.00	0.00	0.00	0.00
C	0.00	0.00	0.00	0.00	0.00
I	0.00	**0.0813**	0.00	0.00	0.00
K	0.00	0.00	0.00	0.00	0.00
И	0.00	0.00	0.00	0.00	0.00
Г	0.00	0.00	0.00	0.00	0.00
Ш	0.00	0.00	0.00	0.00	0.00
Ж	0.00	0.00	0.00	0.00	0.00
G	0.00	0.00	0.00	0.00	0.00
Ч	0.00	0.00	0.00	0.00	0.00
Θ	0.00	0.00	0.00	**0.0813**	0.00
l	0.00	0.00	0.00	0.00	**0.5168**
oo	0.00	0.00	0.00	0.00	0.00
⌴	0.00	0.00	0.00	0.00	0.00

IV. CONCLUSION

Dans ce chapitre, nous avons présenté les différentes composantes d'un système de reconnaissance de caractères imprimés ou manuscrits. L'acquisition de l'image, son

prétraitement et la segmentation en mots puis en caractère sont les premières techniques indispensables pour obtenir la forme la plus simple et représentative du caractère à reconnaître.

Vient ensuite les techniques d'extraction des caractéristiques. Cette phase est un point crucial et déterminant lors de la conception d'un système de reconnaissance des caractères manuscrits ou imprimés. Lors de cette étape, nous avons présenté une nouvelle méthode pour l'extraction des caractéristiques sous forme d'une combinaison linéaire de la méthode des DCG et la méthode des DP.

Ensuite, nous avons présenté le rôle de l'algorithme des c-moyennes floues dans la réduction du taux d'exemples et l'élimination des données redondantes et des intrus. Nous avons illustré dans un exemple l'avantage de l'utilisation du k-PPVF au lieu du k-PPV.

Enfin, nous avons vu que pour certaines images de caractère, l'algorithme des k-PPVF exprime une sorte d'ambiguïté par la génération d'un ensemble de degrés d'appartenance voisins.

Dans ce genre de cas d'ambiguïté, un être humain utilise ses connaissances du langage pour prédire la totalité du mot en question et en suite décide du caractère ambigu. En s'inspirant de ce comportement, et afin d'améliorer les performances de notre système, la mise au point d'un modèle de langage Amazighe s'avère d'une importance remarquable.

Ainsi, le chapitre suivant abordera le domaine du traitement du langage naturel. Lors de ce chapitre, nous allons décrire quelques techniques utilisées pour la détection et correction automatique des fautes d'orthographe. Ensuite, nous allons présenter la théorie des modèles de langage N-gramme que nous avons appliqué au langage Amazighe à l'aide d'un corpus que nous avons construit pour cette fin.

CHAPITRE 4. TRAITEMENT AUTOMATIQUE DU LANGAGE NATUREL

I. INTRODUCTION

Le but du traitement automatique du langage naturel (Natural Language Processing) est de réaliser des agents intelligents capables d'interagir avec un être humain via une langue naturelle. On parle donc d'un système capable d'écouter, de lire et de comprendre un langage humain. Il est aussi capable de retrouver des informations à partir des documents textuels ainsi qu'extraire des faits pertinents de ces documents et faire des résumés et des conclusions. Ces agents doivent aussi être capables de faire de la correction orthographique et grammaticale et de la traduction d'une langue à une autre.

La différence majeure entre les techniques de traitement des langages naturels et les autres systèmes de traitement de données, déjà vus dans les chapitres précédents, est qu'un système de traitement du langage naturel a besoin d'une bonne connaissance de la langue traitée. La connaissance d'une langue peut être décrite en 6 catégories :

1. Phonétique : L'étude des sons linguistiques.
2. Morphologie : L'étude des composantes significatives des mots.
3. Syntaxe : L'étude des relations structurelles entres les mots.
4. Sémantique : Pour produire une phrase (une suite de mots) ayant un sens.
5. Pragmatiques : utilisation des formules supplémentaires dans une phrase. par exemple les formules de politesse.
6. Discours : l'étude des unités linguistiques plus larges.

Dans ce chapitre, on s'intéresse essentiellement à l'étude de la morphologie, la syntaxe et à la sémantique dans une langue donnée, en particulier la langue Amazighe. Ces concepts seront traités en deux parties ; la première partie traite des modèles pour la détection et la

correction des erreurs d'orthographe. Tandis que la seconde partie présente les modèles n-grammes, en particulier le modèle Bi-grammes.

II. MODELES POUR LA CORRECTION D'ORTHOGRAPHE

La détection et correction des erreurs d'orthographe est utilisée dans plusieurs applications où le caractère isolé n'est pas parfaitement identifiable : reconnaissance automatique des caractères (manuscrit ou imprimés) et reconnaissance en ligne des manuscrits, reconnaissance automatique de la parole…etc.

Dans cette partie de ce chapitre nous allons présenter des techniques pour la détection et d'autres pour la correction d'erreurs d'orthographe dans un texte saisi par un être humain. Ces techniques peuvent être aussi utilisées pour un système OCR. Les systèmes OCR ont souvent un taux d'erreur élevé par rapport à un être humain et les erreurs commises ne sont pas aussi différentes.

Damerau [112] a trouvé que 80% des erreurs de frappe commises par un être humain sont dues aux erreurs suivantes : Insertion, suppression, substitution ou transposition. D'autre part, dans les systèmes OCR, les erreurs peuvent êtres groupées en 5 classes : la substitution, la substitution multiple, la suppression d'espace, l'insertion d'espace et l'échec de reconnaissance.

Un problème de correction d'orthographe peut être réparti en trois problèmes majeurs [113] :

1. Détection d'un mot inexistant dans le langage.
2. Correction d'un mot isolé inexistant dans le langage.
3. Détection et correction d'une erreur selon un contexte (phrase) : Utiliser le contexte pour détecter un mot mal placé dans une phrase même si ce dernier existe dans le langage.

II.1. Détection des erreurs d'orthographe

La détection des erreurs d'orthographe dans un texte qu'elle soit commise par un être humain ou par un système OCR, est souvent réalisée à l'aide d'un dictionnaire. Un mot non existant dans un dictionnaire est considéré comme une erreur.

II.2. Modèle du canal bruité (Noisy channel model)

Le problème de la correction des erreurs d'orthographe commises soit par un être humain lors de la saisie ou par un système OCR est modélisable comme un problème de correspondance d'une chaine de symbole vers une autre. Par exemple, pour une série de caractères représentant un mot incorrecte, il faut déterminer une autre série de caractères pour proposer une correction au mot erroné. Pour réaliser cette correspondance, on trouve dans la littérature plusieurs techniques. Parmi ces techniques, on trouve le modèle du canal bruité (Noisy channel) [114] [115].

Le but de cette technique est de modéliser un canal de communication bruité afin de voir comment ce canal transforme un mot correcte en un mot erroné à cause des bruits extérieurs et par suite pouvoir reconstruire le mot correcte (Figure 39).

II.2.a. Principe

Soit O le mot observé à la sortie du canal bruité. Notre but est de trouver parmi les mots du vocabulaire le mot $\widehat{w}$ le plus probable comme correction du mot observé O. Ce mot vérifie l'équation suivante :

$$\widehat{w} = \underset{w \in V}{\operatorname{argmax}}\, P(w|O) \qquad (100)$$

Le terme $P(w|O)$ ne peut pas être évalué, alors qu'on peut estimer le terme $P(O|w)$. En utilisant le théorème de Bayes, $P(w|O)$ peut être exprimé en fonction de $P(O|w)$, $P(w)$ et de $P(O)$ comme suit :

$$P(w|O) = \frac{P(O|w)P(w)}{P(O)} \qquad (101)$$

Figure 39: Canal bruité

L'équation (100) redevient :

$$\widehat{w} = \underset{w \in V}{argmax} \frac{P(O|w)P(w)}{P(O)} \quad (102)$$

Le terme $P(O)$ n'est pas une fonction de w, l'équation (102) est alors équivalente à l'équation suivante :

$$\widehat{w} = \underset{w \in V}{\text{argmax}}\, P(O|w)P(w) \quad (103)$$

Le mot *w* le plus probable d'être à l'origine de l'observation O est donc le mot qui maximise le produit de deux termes; *P(w)* et *P(O|w)*. Reste donc à savoir comment calculer ces deux probabilités.

II.2.b. Sélection des candidats

Dans la partie précédente, on a vu que la correction *w* du mot erroné observé, est le mot qui maximise le produit *P(O|w).P(w)* parmi tous les mots du vocabulaire du langage alors qu'en réalité, il faut choisir un ensemble de candidats susceptibles d'être la bonne correction de l'observation. Pour cette fin, Church et al. [114], ont proposé une simplification pour la sélection des candidats. Cette simplification consiste à supposer que le mot correct est différent du mot erroné d'une unique insertion, suppression, substitution ou transposition. Par exemple le tableau (Tableau 10) liste un ensemble de propositions du mot "ⵜⴰⵍⵉⵜ".

Tableau 10: Proposition des condidats pour le mot "ⵜⴰⵍⵉⵜ"

Mot erroné	Proposition	Edition
ⵜⴰⵍⵉⵜ	ⵜⴰⵍⵉⴰ	Insertion de 'ⵜ'
	ⵜⴰⵍⴰⵉⵜ	Transposition de 'ⴰⵉ'
	ⵜⴰⵍⵉⴰⵀⵜ	Suppression 'ⵀ'
	ⵙⴰⵍⵉⴰⵜ	Substitution ⵙ par ⵜ

En se basant sur cette simplification, l'équation (103) sera sous la forme suivante :

$$\widehat{w} = \underset{w \in C}{\operatorname{argmax}} P(O|w)P(w) \qquad (104)$$

Où C représente l'ensemble des corrections différentes d'une seule transformation du mot erroné. Le terme *P(w)* peut être estimé par le nombre d'occurrence du mot *w* dans un corpus divisé par le nombre total des mots dans le même corpus, comme le montre l'équation suivante :

$$P(w) = \frac{C(w)}{N} \qquad (105)$$

Un corpus peut ne pas contenir tous les mots d'un langage donné. Ce fait peut causer des *P(w)* nulles, par suite le produit *P(O|w).P(w)* sera nul. Pour éviter ce problème, des techniques de lissages, qui seront décrites en détail dans les parties suivantes, sont utilisées. Par exemple la technique du lissage de Laplace décrite par l'équation suivante:

$$P(w) = \frac{C(w) + 1}{N + v} \qquad (106)$$

Où v est la taille du vocabulaire utilisé.

II.2.c. Sélection de la correction

L'unique problème restant est l'estimation du terme *P(O|w)*. Ce terme représente la probabilité qu'un mot correcte *w* soit saisi par erreur (mal classé pour les systèmes OCR), il dépend essentiellement de l'état physique et psychique de l'écrivain et de l'outil de saisie utilisé. Dans le cas d'un système OCR, ce terme dépend du choix des paramètres du système, de l'algorithme d'apprentissage utilisé et des données d'entraînement de test et de validation. Ainsi, le terme *P(O|w)* ne peut pas être exactement estimé.

Pour dépasser ce problème, on se base uniquement sur des statistiques des différentes transformations possibles (insertion, suppression, substitution et transposition) [114]. Ces statistiques sont représentées sous forme de ce qu'on appelle des matrices de confusion, une matrice pour chaque type de transformation. Par exemple, pour un ensemble de 32 caractères, la matrice de confusion des substitutions est une matrice de dimension 32x32, et représente le

nombre de fois un caractère est saisie par erreur au lieu d'un autre caractère. La figure (Figure 40) est un exemple de matrice de confusion extrait du [114].

Ces quatre matrices sont définies comme suit :

1. *Supp[x,y]* : le nombre de fois où les caractères *xy* ont été saisis comme *x*.
2. *Ins[x,y]* : le nombre de fois où le caractère *x* a été saisi comme *xy*.
3. *Sub[x,y]* : le nombre de fois où le caractère *x* a été saisi comme *y*.
4. *Trans[x,y]* : le nombre de fois où les caractère *xy* ont été saisis comme *yx*.

En utilisant ces matrices et sachant que le mot observé *O* est le résultat d'une unique transformation du mot correcte *w*, le terme *P(O|w)* est estimé comme suit [114]:

$$P(O|w) = \begin{cases} \dfrac{supp[w_{p-1}, w_p]}{C(w_{p-1}w_p)} & en\ cas\ de\ suppression \\ \dfrac{ins[w_{p-1}, O_p]}{C(w_{p-1})} & en\ cas\ d'insertion \\ \dfrac{sub[O_p, w_p]}{C(w_p)} & en\ cas\ de\ substitution \\ \dfrac{trans[w_p, w_{p+1}]}{C(w_p w_{p+1})} & en\ cas\ de\ transposition \end{cases} \quad (107)$$

X \ Y	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	2	1	1	0	0	0	19	0	1	14	4	25	10	3	0	27	3	5	31	0	0	0	0	0
b	0	0	0	0	2	0	0	0	0	0	0	1	1	0	2	0	0	0	2	0	0	0	0	0	0	0
c	0	0	0	0	1	0	0	1	85	0	0	15	0	0	13	0	0	0	3	0	7	0	0	0	0	0
d	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	0
e	1	0	4	5	0	0	0	0	60	0	0	21	6	16	11	2	0	29	5	0	85	0	0	0	2	0
f	0	0	0	0	0	0	0	0	12	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
g	4	0	0	0	2	0	0	0	0	0	0	1	0	15	0	0	0	3	0	0	3	0	0	0	0	0
h	12	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
i	15	8	31	3	66	1	3	0	0	0	0	9	0	5	11	0	1	13	42	35	0	6	0	0	0	3
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	11	0	0	12	20	0	1	0	4	0	0	0	0	0	1	3	0	0	1	1	3	9	0	0	7	0
m	9	0	0	0	20	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	4	0	0	0	0	0
n	15	0	6	2	12	0	8	0	1	0	0	0	3	0	0	0	0	0	6	4	0	0	0	0	0	0
o	5	0	2	0	4	0	0	0	5	0	0	1	0	5	0	1	0	11	1	1	0	0	7	1	0	0
p	17	0	0	0	4	0	0	1	0	0	0	0	0	0	1	0	0	5	3	6	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	12	0	0	0	24	0	3	0	14	0	2	2	0	7	30	1	0	0	0	2	10	0	0	0	2	0
s	4	0	0	0	9	0	0	5	15	0	0	5	2	0	1	22	0	0	0	1	3	0	0	0	16	0
t	4	0	3	0	4	0	0	21	49	0	0	4	0	0	3	0	0	5	0	0	11	0	2	0	0	0
u	22	0	5	1	1	0	2	0	2	0	0	2	1	0	20	2	0	11	11	2	0	0	0	0	0	0
v	0	0	0	0	1	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w	0	0	0	0	0	0	0	4	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	8	0
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
y	0	1	2	0	0	0	1	0	0	0	0	3	0	0	0	2	0	1	10	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

III. MODELES DE LANGAGE N-GRAMMES

Le principe général des modèles n-grammes est la prédiction d'un mot (un caractère) à la fin d'une série de mots (de caractères). Ce principe est une tâche de grande importance dans la reconnaissance de la parole, la reconnaissance des caractères manuscrits et la détection et correction orthographique…etc. Dans ces applications, l'identification des mots (caractères) n'est pas du tout simple, car l'entrée est souvent bruitée ou ambigüe ou les deux à la fois. D'où, dans une phrase, l'utilisation des mots précédents le mot en question peut être d'une grande utilité pour la prédiction de ce dernier.

Ce problème de prédiction du mot suivant est directement lié au calcul de la probabilité de rencontrer une phrase (série de mot) dans un texte. Par exemple, la probabilité d'existence de la phrase suivante dans un texte est très faible :

"ⵍⵎⵖⵔⵉⴱ ⵜⵉⵍⴰⵍⵜ ⵉⵜⵜⵓⵙⵜⴰ ⵜⴰⵎⵣⵡⴰⵔⵓⵜ ⴰⵎⵎⴰⵙ ⵏ"

Alors qu'avec les mêmes mots, dans un ordre différent, on peut construire une phrase d'une plus grande probabilité :

"ⵉⵜⵜⵓⵙⵜⴰ ⵍⵎⵖⵔⵉⴱ ⴰⵎⵎⴰⵙ ⵏ ⵜⵉⵍⴰⵍⵜ ⵜⴰⵎⵣⵡⴰⵔⵓⵜ"

Un algorithme qui peut affecter une probabilité à une série de mots peut le faire aussi pour le mot suivant dans une suite incomplète de mots.

III.1. Corpus

L'approche statistique du traitement automatique du langage naturel se base essentiellement sur la qualité et la taille du corpus. Un corpus est une énorme collection des textes et des discoures de différents genres : magazines, journaux, commerce, science…etc. Par exemple le corpus Brown Corpus, réalisé à l'université de Brown, contient 1milion mots, 61.805 mots différents, extraits de 500 textes de différents genres. Un autre exemple pour l'Anglais est le corpus de Switchboard composé de 2.4 million de mots, 20.000 mots différents.

Dans ce travail de recherche, nous avons construit un corpus pour la langue Amazighe. Ce corpus est extrait en sa totalité des sites web de type journaux et magazine. Il se compose de

133178 mots, 13000 mots différents. Pour le calcul des probabilités, on va compter des mots dans ce corpus.

III.2.N-Gramme simple

La plus simple façon pour modéliser une série de mots est de supposer que chaque mot a la même probabilité de suivre n'importe quel autre mot, ce qui n'est pas le cas dans tous les langages humains. Un autre modèle plus complexe (modèle uni-gramme) consiste à supposer qu'un mot suivra un autre selon sa probabilité d'occurrence dans le corpus. Si par exemple dans un corpus de 1000 mots, le mot "ᴑᑕᑕᴑʘ" est cité 100 fois et le mot "+ᴑᑕᛯЦᴑOȣ+" une fois, alors la probabilité que le mot " ᴑᑕᑕᴑʘ " suivra n'importe quel autre mot est 0.1 et la probabilité que le mot "+ᴑᑕᛯЦᴑOȣ+" suivra n'importe quel mot est 0.001. Autrement dit, pour n'importe quelle suite de mots, la prédiction du mot suivant dépendra uniquement des fréquences d'occurrence des mots dans le corpus. Suivant ce modèle, pour la série de mots "Σ++ȣʘ+ᴑ ИᑕЧOΣΘ ᴑᑕᑕᴑʘ ı +ΣИᴑИ+ +ᴑᑕᛯЦᴑOȣ+" le mot "ᴑᑕᑕᴑʘ" a plus de chance pour suivre le mot "+ΣИᴑИ+" que le mot "+ᴑᑕᛯЦᴑOȣ+", alors que ce dernier est le plus souhaitable pour donner un sens à la phrase "+ΣИᴑИ+ +ᴑᑕᛯЦᴑOȣ+". On peut automatiquement constater la nécessité d'utiliser des probabilités conditionnelles aux lieux des simples fréquences d'occurrences des mots dans le corpus. On aura alors la probabilité que le mot "+ᴑᑕᛯЦᴑOȣ+" apparait après le mot "+ΣИᴑИ+" supérieure à la probabilité d'avoir le mot "ᴑᑕᑕᴑʘ" après le mot "+ΣИᴑИ+".

Dans tout le reste de ce chapitre, une phrase est souvent représentée par une série de mots comme suivant : $w_1\, w_2\, \ldots . w_n$ ou bien w_1^n. La probabilité de rencontrer cette phrase dans un texte est :

$$P(w_1\, w_2\, \ldots . w_n) = P(w_1^n) \qquad (108)$$

En utilisant la formule des probabilités composées, cette probabilité est reformulée comme suit :

$$P(w_1^n) = P(w_1)\prod_{k=2}^{n} P(w_k|w_1^{k-1}) \quad (109)$$

Le problème qui se pose avec cette dernière équation est que le calcul du terme $P(w_k|w_1^{k-1})$ n'est pas tout-à- fait évident.

Pour pallier ce problème, on utilise une approximation proposée par Markov. La probabilité d'un mot sachant la totalité des (n-1) mots qui le précèdent est approximativement la même que la probabilité de ce mots sachant juste le mot qui le précède directement. Autrement dit :

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1}) \quad (110)$$

Ce cas particulier est appelé un modèle bi-gramme. On peut généraliser les bi-grammes en trigrammes et aux N-grammes comme suivant :

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1}) \ \ avec\ N < n \quad (111)$$

En utilisant le model bi-gramme, l'équation (109) deviennent:

$$P(w_1^n) \approx P(w_1)\prod_{k=2}^{n} P(w_k|w_{k-1}) \quad (112)$$

Pour intégrer des informations concernant le début et la fin de phrase, le corpus est augmenté en ajoutant des délimiteurs de phrases; <s> pour le début et </s> pour la fin d'une phrase. Par exemple, pour calculer la probabilité d'occurrence de la phrase suivante *"Σ++8O+o ИCЧOΣΘ oCCoO I +ΣИoИ+ +oCЖШoO8+"* on procède comme suivant :

P(Σ++8O+o ИCЧOΣΘ oCCoO I +ΣИoИ+ +oCЖШoO8+)=

P((Σ++8O+o | <s>)×P(ИCЧOΣΘ | Σ++8O+o)×P(oCCoO | ИCЧOΣΘ)×

P(I | oCCoO)×P(+ΣИoИ+ | I)×P(+oCЖШoO8+ | +ΣИoИ+)×

P(<\s> | +oCЖШoO8+)

Le même exemple avec le modèle trigrammes :

P(Σ++ƧO+o ИLЧOΣΘ oLLoO I +ΣИoИ+ +oLЖUoOƧ+)=

P((Σ++ƧO+o | <s><s>)×P(ИLЧOΣΘ | <s>Σ++ƧO+o)×P(oLLoO | Σ++ƧO+o ИLЧOΣΘ)×

P(I | ИLЧOΣΘ oLLoO)×P(+ΣИoИ+ | oLLoO I)×P(+oLЖUoOƧ+ | I +ΣИoИ+)×

P(<\s> | +ΣИoИ+ +oLЖUoOƧ+)

Le calcul du terme $P(w_n|w_{n-1})$ se fait en se basant sur le dénombrement des N-grammes dans le corpus comme suit :

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} \quad (113)$$

Comme le nombre des bi-grammes commençant par w_{n-1} est exactement le nombre d'occurrences de ce mot dans le même corpus alors l'équation (113) devient :

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \quad (114)$$

Cette dernière équation peut être généralisée pour les N-grammes comme suit :

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})} \quad (115)$$

Les N-grammes ont deux défauts majeurs : l'augmentation de précision avec le nombre N et leurs forte dépendance du corpus utilisé pour le calcul des probabilités.

III.3. Lissage (Smoothing)

Les N-grammes sont entraînés à l'aide d'un corpus, ce qui présente un problème pertinent à cause de son nombre limité de mots ou plutôt de phrase. Ainsi, il peut exister des N-grammes bien correctes dans la même langue que celle du corpus mais n'existent pas dans ce dernier, ce qui engendre des probabilités nulles. Pour éviter ce problème, plusieurs techniques existent dans la littérature. Parmi ces techniques on trouve la technique de lissage (Smoothing).

III.3.a. Lissage de Laplace

Cette technique est à la base de plusieurs autres techniques. Elle consiste à ajouter un à tous les éléments du corpus et supposer qu'un événement inconnu peut se produire au moins une fois.

Pour les uni-grammes et avec un corpus de taille N nous avons :

$$P(w) = \frac{c(w)}{\sum_i c(w_i)} = \frac{c(w)}{N} \tag{116}$$

Où c(w) est le nombre d'occurrences du mot w dans le corpus.

Les plupart des méthodes de lissage repose sur la modification de la fonction c(w) afin d'éliminer les probabilités nulles. Cette modification peut être réalisée par l'ajout de 1 aux comptes puis la multiplication par un facteur de normalisation comme montre l'équation suivante :

$$c_i^* = (c_i + 1)\frac{N}{N + v} \tag{117}$$

Où v est la taille du vocabulaire utilisé dans le corpus.

Les probabilités sont alors calculées en utilisant c^* normalisés par N. On peut directement calculer ces nouvelles probabilités pour les uni-grammes à l'aide de l'équation suivante :

$$P^*(w) = \frac{c(w) + 1}{\sum_i c(w_i) + v} \tag{118}$$

De la même façon les probabilités pour les bi-grammes :

$$P^*(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n) + 1}{c(w_{n-1}) + v} \tag{119}$$

Pour bien illustrer cette technique, nous allons utiliser l'exemple de la phrase Amazighe suivante : *"ⵎⵔⴰⵔⴰⵏⵜ ⵜⵉⵎⵉⵥⴰⵔ ⵍⵍⵉ ⵎⴱⵏⵏⵉⵏ ⵅ ⴱⵔⴰⵜⵉ"*. Le tableau (Tableau 11) montre un extrait des dénombrements des bi-grammes qu'on peut composer de l'ensemble des mots de

cette phrase. On remarque que la majorité des valeurs sont nulles. Le tableau (Tableau 12) représente les probabilités associées aux bi-grammes du tableau précédent avant l'utilisation du lissage de Laplace. Là aussi on remarque que la majorité des bi-grammes ont une probabilité d'occurrence dans le corpus égale à zéro. Le tableau (Tableau 13) affiche les dénombrements des bi-grammes après l'application de la technique de lissage de Laplace, tous les dénombrements des bi-grammes ont été augmentés de 1. Ce changement des comptes a bien affecté les nouvelles probabilités (Tableau 14). Les probabilités nulles ne le sont plus mais en contrepartie les probabilités non nulles ont été diminuées.

En utilisant l'équation (117), on peut reconstruire les comptes après le lissage de Laplace (Tableau 15). On remarque que les comptes ont été remarquablement modifiés afin de redistribuer les probabilités sur les bi-grammes non existants dans le corpus. Par exemple le Bi-gramme "ⵅ ⵙⵔⵓ+ⵚ" a changé de 76 à 14.6287, en résultat, la probabilité P(ⵙⵔⵓ+ⵚ | ⵅ) a diminuée de 0.0201 à 0.0038. En contrepartie, le dénombrement du Bi-gramme "ⵙⵔⵓ+ⵚ ⵅ" a changé de 0 à 0.01483, en résultat, la probabilité P(ⵅ | ⵙⵔⵓ+ⵚ) a augmenté de 0 jusqu'à $1.2360\ 10^{-4}$.

Donc, On remarque que lissage de Laplace retranche une partie de la masse de probabilité des bi-grammes les plus fréquents, pour la redistribuer aux Bi-grammes moins fréquents ou inexistants dans le corpus.

Tableau 11 : Exemple de dénombrement des Bi-grammes

	ⵛⵔⵓⵔⵏ+	+ⵚⵛⵚⵊⵓⵔ	ⵉⵏⵚ	ⵛⵙⵏⵚⵏ	ⵅ	ⵙⵔⵓ+ⵚ
ⵛⵔⵓⵔⵏ+	0	1	0	0	0	0
+ⵚⵛⵚⵊⵓⵔ	0	0	3	0	0	0
ⵉⵏⵚ	0	0	1	1	29	0
ⵛⵙⵏⵚⵏ	0	0	0	0	1	0
ⵅ	0	39	0	0	0	76
ⵙⵔⵓ+ⵚ	0	0	0	0	1	0

Tableau 12 : Probabilités associées

	⊏ΘₒΘₒlᐩ	ᐩΣ⊏Σ丯ₒO	ИИΣ	⊏8IIΣI	ⵅ	8Θₒᐩ Σ
⊏ΘₒΘₒlᐩ	0	1.00	0	0	0	0
ᐩΣ⊏Σ丯ₒO	0	0	0.0236	0.0006	0	0
ИИΣ	0	0	0.0006	1.00	0.0180	0
⊏8IIΣI	0	0	0	0	0.0434	0
ⵅ	0	0.0103	0	0	0	0.0201
8Θₒᐩ Σ	0	0	0	0	0.0083	0

Tableau 13 : Dénombrement des bi-grammes après le lissage de Laplace

	⊏ΘₒΘₒlᐩ	ᐩΣ⊏Σ丯ₒO	ИИΣ	⊏8IIΣI	ⵅ	8Θₒᐩ Σ
⊏ΘₒΘₒlᐩ	1	2	1	1	1	1
ᐩΣ⊏Σ丯ₒO	1	1	4	1	1	1
ИИΣ	1	1	2	2	30	1
⊏8IIΣI	1	1	1	1	2	1
ⵅ	1	40	1	1	1	77
8Θₒᐩ Σ	1	1	1	1	2	1

Tableau 14 : Probabilités des bi-grammes après le lissage de Laplace

	⊏ΘₒΘₒlᐩ	ᐩΣ⊏Σ丯ₒO	ИИΣ	⊏8IIΣI	ⵅ	8Θₒᐩ Σ
⊏ΘₒΘₒlᐩ	$6.2258\ 10^{-5}$	$1.2451\ 10^{-4}$	$6.2258\ 10^{-5}$	$6.2258\ 10^{-5}$	$6.2258\ 10^{-5}$	$6.2258\ 10^{-5}$
ᐩΣ⊏Σ丯ₒO	$6.1774\ 10^{-5}$	$6.1774\ 10^{-5}$	$2.4709\ 10^{-4}$	$6.1774\ 10^{-5}$	$6.1774\ 10^{-5}$	$6.1774\ 10^{-5}$
ИИΣ	$5.6596\ 10^{-5}$	$5.6596\ 10^{-5}$	$1.1319\ 10^{-4}$	$1.1319\ 10^{-4}$	0.0016	$5.6596\ 10^{-5}$
⊏8IIΣI	$6.2173\ 10^{-5}$	$6.2173\ 10^{-5}$	$6.2173\ 10^{-5}$	$6.2173\ 10^{-5}$	$1.2434\ 10^{-4}$	$6.2173\ 10^{-5}$
ⵅ	$5.0433\ 10^{-5}$	0.0020	$5.0433\ 10^{-5}$	$5.0433\ 10^{-5}$	$5.0433\ 10^{-5}$	0.0038
8Θₒᐩ Σ	$6.1800\ 10^{-5}$	$6.1800\ 10^{-5}$	$6.1800\ 10^{-5}$	$6.1800\ 10^{-5}$	$1.2360\ 10^{-4}$	$6.1800\ 10^{-5}$

Tableau 15 : Reconstruction des dénombrements après le lissage de Laplace

	⊏⊙ₒ⊙ₒ|+	+Σ⊏Σ⋇ₒO	ИИΣ	⊏8||Σ|	⊼	8⊙ₒ+Σ
⊏⊙ₒ⊙ₒ|+	$6.22\ 10^{65}$	$1.24\ 10^{-4}$	$6.22\ 10^{-5}$	$6.22\ 10^{-5}$	$6.22\ 10^{-5}$	$6.22\ 10^{-5}$
+Σ⊏Σ⋇ₒO	0.0078	0.0078	0.0313	0.0078	0.0078	0.0078
ИИΣ	0.0910	0.0910	0.1820	0.1820	2.7302	0.0910
⊏8||Σ|	0.0014	0.0014	0.0014	0.0014	0.0028	0.0014
⊼	0.1899	7.5999	0.1899	0.1899	0.1899	14.6287
8⊙ₒ+Σ	0.0074	0.0074	0.0074	0.0074	0.01483	0.0074

En général, la méthode de lissage de Laplace aboutit à des résultats non satisfaisants par rapport à d'autres méthodes plus complexes comme la méthode de Good-Turing [116], d'une part. D'autre part cette méthode conduit à des probabilités très différentes de celles obtenues avant le lissage de Laplace.

III.3.b. Lissage Good-Turing

Cette technique a été d'abord introduite par Good en 1953 puis améliorée par Turing. Son principe est d'utiliser les N-grammes de plus grand effectif pour estimer les probabilités des N-grammes d'effectif nul ou très faible. La reconstruction des effectifs des N-grammes est décrite par l'équation suivante :

$$c^* = (c+1)\frac{N_{c+1}}{N_c} \qquad (120)$$

Où N_c représente le nombre des N-grammes d'effectif égale à c.

Par exemple, le nouvel effectif des bi-grammes absents dans le corpus est le nombre des bi-grammes vus une seule N_1 fois divisé par le nombre des bi-grammes absents N_0. Puisqu'on connait la taille du vocabulaire v, alors le nombre total des bi-grammes est v^2, d'où $N_0 = v^2 - N_1$.

Normalement, La ré-estimation des effectifs doit être appliquée juste pour les N-grammes d'un faible effectif. Après l'introduction d'un seuil k, la nouvelle équation pour l'estimation des effectifs est :

$$c^* = \frac{(c+1)\frac{N_{c+1}}{N_c} - c\frac{(k+1)N_{k+1}}{N_1}}{1 - \frac{(k+1)N_{k+1}}{N_1}}, si\ 1 \leq c \leq k \qquad (121)$$

Dans cette équation on remarque l'absence du cas où l'effectif des N-grammes est nul. Dans ce cas, on traite les N-grammes d'effectifs nuls comme ceux d'effectif égal à 1.

III.3.c. Lissage par repli (Backoff)

Le lissage par repli (Backoff) est une autre technique utilisée pour dépasser le problème des probabilités nulles. Son principe est relativement simple, si par exemple, un trigramme $w_{n-2}w_{n-1}w_n$ n'existe pas dans le corpus, alors on peut estimer la probabilité $P(w_n|w_{n-2}w_{n-1})$ en utilisant la probabilité du bi-gramme $w_{n-1}w_n$; $P(w_n|w_{n-1})$ et si ce bi-gramme n'existe pas à son tour dans le corpus, alors on utilise la probabilité de l'uni-gramme $P(w_n)$. Cette démarche est bien décrite dans l'équation suivante :

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \begin{cases} P(w_n|w_{n-2}w_{n-1}) & si\ C(w_{n-2}w_{n-1}w_n) > 0 \\ \alpha_1 P(w_n|w_{n-1}) & si \begin{cases} C(w_{n-2}w_{n-1}w_n) = 0 \\ et\ C(w_{n-2}w_{n-1}) > 0 \end{cases} \\ \alpha_2 P(w_n) & sinon \end{cases} \qquad (122)$$

III.4. Évaluation du model

III.4.a. L'entropie

L'entropie est une mesure d'information, elle permet de mesurer à quel point une grammaire est appropriée à un langage donné. Pour le calcul de l'entropie, il faut établir une variable aléatoire X représentant l'entité à prévoir (mot, caractère,…) possédant une fonction de probabilité p(x). L'entropie de cette variable est définie comme suit:

$$H(X) = -\sum_{x \in \chi} p(x) log_2 p(x) \qquad (123)$$

III.4.b. Perplexité

La perplexité est la probabilité de la totalité de la base de test obtenu par le modèle normalisé par la taille de la base de test :

$$PP(T) = \sqrt[N]{\frac{1}{P(w_1, w_2, .. w_N)}} \qquad (124)$$

Si le modèle du langage utilisé est le bi-gramme, l'équation (124) devient :

$$PP(T) = \sqrt[N]{\frac{1}{P(w_1)\prod_{i=2}^{N} P(w_i|\, w_{i-1})}} \qquad (125)$$

D'après cette équation, la maximisation des probabilités des mots revient à minimiser la perplexité. Un bon modèle de langage doit donc avoir une perplexité minimale.

IV. CONCLUSION

Dans ce chapitre nous avons présenté l'algorithme du canal bruité pour la correction des erreurs d'orthographe. La réalisation d'une telle tâche nécessite la disponibilité d'un certain nombre d'information sur l'utilisation d'une langue. Dans le cas du modèle de canal bruité, ces informations sont regroupées dans les quatre matrices de confusion.

Dans une seconde partie, nous avons présenté la théorie des N-grammes et leurs utilisations pour la modélisation d'un langage naturel. Une bonne modélisation d'une langue par les N-grammes nécessite la disponibilité d'un large et cohérant corpus. Nous avons donc construit un corpus pour la langue Amazighe que nous avons utilisé pour l'estimation des probabilités d'occurrences des différents N-grammes.

Dans le chapitre suivant, qui est essentiellement les résultats de nos contribution, nous allons voir comment utiliser les techniques vues dans les chapitres précédents et celles vues dans ce chapitre pour concevoir un système de reconnaissance automatique des manuscrits Tifinagh.

CHAPITRE 5. RESULTATS ET DISCUSSION

I. INTRODUCTION

Le but de ce travail est d'améliorer les performances de notre système de reconnaissance des manuscrit Tifinagh. Nous avons utilisé des techniques issues de deux domaines différents, le domaine de la reconnaissance de formes pour percevoir le monde extérieur et le domaine du traitement des langages naturels afin d'incorporer des connaissances linguistiques pour bien conduire à une bonne discision.

Les résultats obtenus dans notre approche, dépendent de plusieurs paramètres autres que la base des exemples et le corpus : il s'agit de la méthode d'extraction des caractéristiques, la taille de la grille utilisée, le nombre des plus proches voisins (k) et le facteur de fuzzification (m) pour l'algorithme des k-plus proches voisins ainsi que le nombre de sous classes et le facteur de fuzzification dans l'algorithme des c-moyennes floues.

Dans ce chapitre, nous allons présenter la structure générale de ce système, l'impact du choix de ses divers paramètres sur ses performances de généralisation et les résultats obtenu par un choix optimal de ces paramètres.

II. DESCRIPTION DU SYSTEME

Afin de tester les performances de notre approche, nous avons développé une application java (Figure 41). Pour l'entraînement, la validation et le test de l'algorithme des k-plus proches voisins flous, nous avons conçu une base d'exemples des manuscrits Tifinagh de 12812 caractères sous forme de mots dont 8847 caractères pour l'entraînement et la validation et 3956 caractères pour le test. Le corpus utilisé pour l'algorithme du modèle N-gramme est constitué de 133178 instances de 13971 mots Amazighe écrits en Tifinagh.

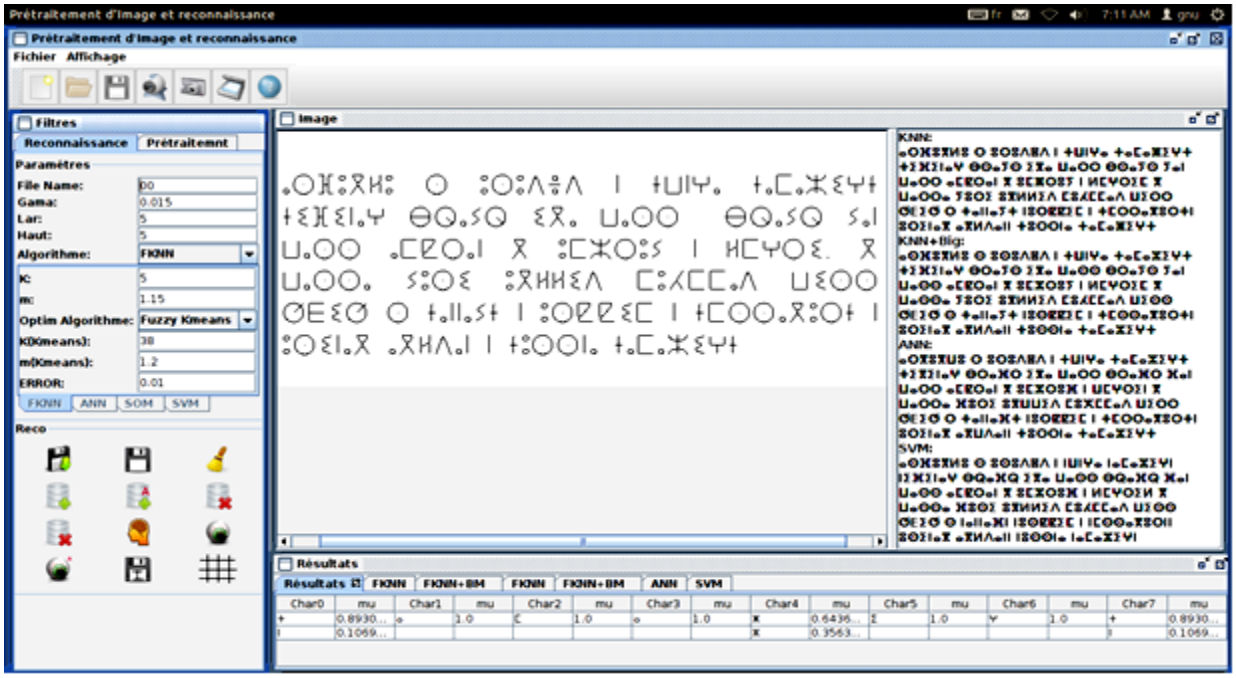


Figure 41: Application développée pour la reconnaissances des manuscrits Tifinagh

Le système de reconnaissance des manuscrits Tifinagh développé dans ce travail se compose d'un ensemble de blocks (Figure 42) : Prétraitement, segmentation, extraction des caractéristiques, classification et Post-classification.

Dans la phase du prétraitement, les opérations d'élimination des bruits, binarisation, correction d'inclinaison et squelettisation sont effectuées en utilisant un ensemble d'algorithmes classiques décrits dans le chapitre 3.

Jusqu'à nos jours, il n'existe aucune écriture cursive du Tifinagh. Par conséquence, la segmentation d'un mot écrit en Tifinagh demeure une tâche simple et triviale. Une simple méthode de segmentation est largement suffisante. La méthode des Histogrammes donne des résultats très satisfaisants dans ce cadre. Une image d'un mot est alors transformée en une suite d'images de caractères isolés.

Pour chacune des images générées dans la phase de segmentation, le classifieur flou propose un ensemble de candidats munis de leurs degrés d'appartenance aux différentes classes existantes (l'alphabet de la langue Amazighe utilisé par l'IRCAM à l'exception des caractères ‘ⵅ◌ⵯ’ et ‘ⵠ◌ⵯ’ qui seront traités comme ‘ⵅ’+‘◌ⵯ’ et ‘ⵠ’+‘◌ⵯ’). A partir De ces candidats, on génère les différents mots possibles, chacun avec un degré de vérité.

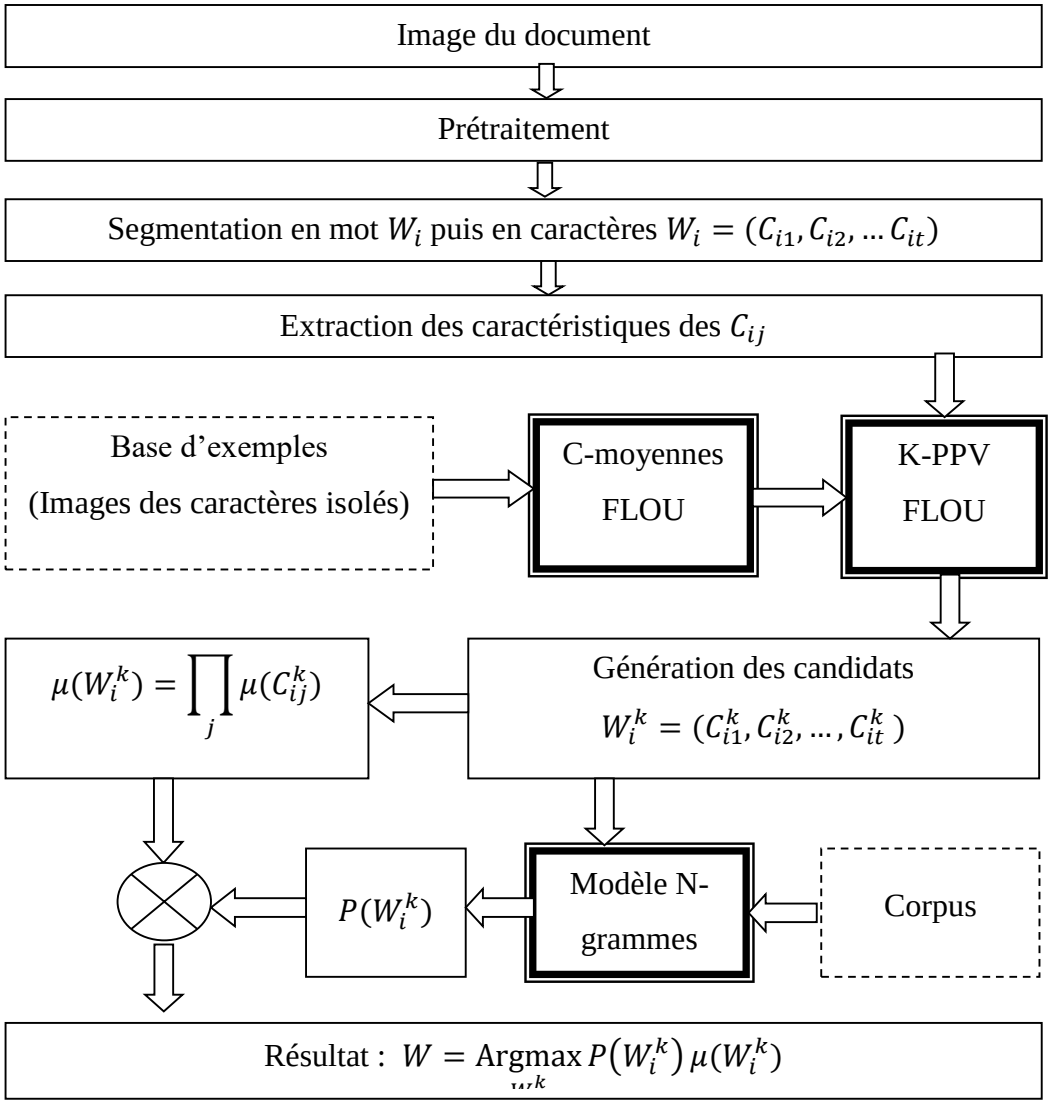


Figure 42 : Structure du système de reconnaissance

Pour chaque mot, le modèle des N-grammes calcule la probabilité d'occurrence de ce mot dans un dictionnaire préalablement défini. Il propose donc la série de caractères la plus probable.

Soit $P(w)$ la probabilité que le mot $w = \{c_1, c_2, \dots, c_n\}$ existe dans un dictionnaire donné. En utilisant les équations vues dans le chapitre 4, Nous avons :

$$P(w) = P(c_1) \times \prod_{i=2}^{n} P(c_i / c_1, c_2, \dots, c_{i-1}) \qquad (126)$$

Par la simplification de Markov nous avons :

$$P(w) \approx P(c_1) \times \prod_{i=2}^{n} P(c_i/c_{i-1}) \qquad (127)$$

Pour introduire le contexte du début et de la fin de mot. On augmente ce mot avec deux caractères spéciaux <s> et </s> respectivement au début et à la fin du mot. Par exemple, pour calculer P("ѻЦѻИ"), il faut d'abord augmenter ce mot par <s> et </s> puis nous obtiendrons :

P("ѻЦѻИ")=P(ѻ/<s>).P(Ц/ѻ).P(ѻ/Ц).P(И/ѻ).P(</s>/И)

Le but d'utilisation des N-grammes (plus précisément les Bi-grammes) dans notre approche est de donner un avis sur les différentes séquences de caractères proposés par le classifieur flou.

Enfin, la décision prise est le mot possédant la plus grande valeur du produit de son degré de vérité et de sa probabilité d'occurrence dans le dictionnaire.

$$\tilde{w} = argmax_{w\epsilon\{w_{i=1..m}\}}(P(w).\mu(w)) \qquad (128)$$

La figure (*Figure 43*) illustre cette démarche avec l'image du mot " ϮϬΛOϮ ". Pour chacun des deux caractères ; le premier 'Ϯ' et le dernier 'Ϯ' , le classifieur propose un seul candidat avec un degré d'appartenance égal à l'unité, alors que pour le carcatère 'Ϭ' il propose trois candidats {Ϭ, Ϭ, I} pour le caractère 'Λ' il propose {Ⱥ, ʌ, Λ} et deux pour le caractère O {O, Ø, Ц}. Les mots qu'on peut générer à partir de ces candidat sont {ϮϬⱭOϮ, ϮϬʌOϮ, ϮϬΛØϮ, ϮϬΛOϮ, ϮIΛOϮ, ϮϬΛOϮ … }. En totalité, nous avons 27 mots. Si on utilise uniquement les degrés d'appartenance calculés par l'algorithme des k-plus proches voisins flous, le résultat serait " ϮϬΛØϮ", mais puisque nous avons :

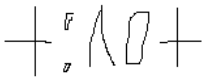
log(P(ϮϬΛØϮ))=-24.84 et log(μ(ϮϬΛØϮ))=log(1×0.59×0.71×0.60×1)=-1.36

log(P(ϮϬΛOϮ))=-18.97 et log(μ(ϮϬΛOϮ))=log(1×0.59×0.71×0.60×1)=-2.24

on a donc log(P(ϮϬΛØϮ))×μ(ϮϬΛØϮ)=-26.21 < log(P(ϮϬΛOϮ))×μ(ϮϬΛOϮ)=-21.21

Par conséquence, le mot reconnu par notre système est "ϮϬΛOϮ".

Image du Mot :



Proposition des candidats par le k PPV Flou :

Résultats

Résultats | ANN | WordsScore

Char0	mu	Char1	mu	Char2	mu	Char3	mu	Char4	mu
+	1.0	8	0.59269...	R	0.16238...	O	0.24968...	+	1.0
		8	0.11873...	⁄	0.12064...	Ø	0.60045...		
		I	0.28857...	Λ	0.71697...	U	0.14985...		

Génération des mots et calcul des probabilités :

Résultats

Résultats | ANN | WordsScore

word	Log(mu(word))	Log(P(word))	Log(mu(word)*P(word))
+IRU+	-4.958651660702216	-27.01671522148424	-31.975366882186457
+8ΛU+	-2.753862448976083	-21.03205999201319	-23.78592244098927
+8ΛO+	-3.851156104158372	-19.022114726602986	-22.873270830761356
+8ΛØ+	-2.973657715784205	-24.89292014249128	-27.866577858275488
+8⁄U+	-6.143855755354095	-22.003695492413023	-28.14755124776712
+8⁄O+	-4.025555909115103	-20.403566652361135	-24.429122561476238
+IΛO+	-2.96308499085193	-22.215993107297905	-25.179078098149834
+8⁄Ø+	-3.148057520740936	-22.644333114158425	-25.79239063489936
+IΛØ+	-2.085586602477763	-28.086798523186197	-30.172385125663958
+I⁄U+	-5.255784642047653	-26.583868234227836	-31.839652876275487
+8RU+	-4.238926023298018	-21.038848282641638	-25.277774305939655
+8RO+	-5.336219678480307	-18.87818679985281	-24.214406478333117
+8RØ+	-4.458721290106141	-22.910712730878153	-27.369434020984293
+8ΛU+	-4.36165919968672 36	-21.08084735999271	-25.442506559679433
+IRO+	-4.448148565173865	-24.939964082591835	-29.388112647765702
+8ΛO+	-2.2433593534477314	-18.973327358623465	-21.216686712071194
+IRØ+	-3.5706501767996977	-28.97249001361718	-32.54314019041688
+8ΛØ+	-1.3658609650735642	-24.84413277451176	-26.209993739585325
+8⁄O+	-5.633352659825744	-20.684630055218136	-26.31798271504388
+8⁄U+	-4.536059004643454	-21.722632089556022	-26.258691094199477
+IΛU+	-3.473588086380281	-24.27472574068763	-27.74831382706791
+8⁄Ø+	-4.755854271451577	-22.925396517015425	-27.681250788467004
+8RU+	-5.846722774008659	-20.95493793874521	-26.80166071275387
+I⁄O+	-4.745281546519302	-25.26480279703295	-30.01008434355225
+8RO+	-3.7284229277696666	-18.962097143749236	-22.690520071518904
+I⁄Ø+	-3.8677831581451345	-27.50556925883024	-31.373352416975372
+8RØ+	-2.8509245393954994	-22.99462307477458	-25.84554761417008

Figure 43: Génération des candidats par le k-PPVF et le modèle Bi-gramme

Algorithme

1. ***Entrainement***

Pour chaque classe de caractère

Engendrer un ensemble réduit de prototypes à l'aide du c-moyenne floues

Fin pour

2. ***Classification***

Pour chaque image de mot w

Pour chaque image de caractère c dans w

Engendrer les candidats et leurs degrés d'appartenance à l'aide de l'algorithme des k-PPVF

Fin pour

Générer les m combinaisons de caractères possibles $w_{i=1..m}$.

Pour chaque mot $w_{i=1..m}$

Calculer $\mu(w_i)$

Calculer $P(w_i)$

Fin pour

Le mot sélectionné $\tilde{w}$ *est obtenu par l'équation* (128)

Fin Pour

Remarque :

Il faut noter que les probabilités obtenues par le modèle bi-grammes sont des valeurs très faibles et diminuent de plus en plus avec des corpus très larges. De ce fait et pour éviter l'élimination d'une valeur très faible à cause de plusieurs multiplication par un système numérique, on utilise le logarithme afin de transformer une multiplication en une sommation.

II.1. Extraction des caractéristiques

Dans cette phase, on divise l'image du caractère en une grille de taille W×H. Puisque la majorité des caractères Tifinagh ont des formes symétriques horizontalement ou verticalement alors un choix d'une valeur impaire pour le H et le W est plus approprié. Pour le choix des valeurs de ces deux paramètres. D'un côté, si ces valeurs sont très grandes alors on perd de généralité ; deux images différentes du même caractère seront alors vues comme deux caractères différents. D'un autre côté, si ces valeurs sont très petites on perd de précision ; deux images de deux caractères différents seront vues comme celles d'un même caractère.

La figure (Figure 44) montre les taux de reconnaissance obtenus par la méthode des K-PPVF combinée avec le modèle bi-grammes pour différentes valeurs des paramètres W et H. On peut remarquer que pour les taux les plus bas sont obtenus par le choix de WxH=3x3 et WxH=9x9. Alors que le meilleur taux est obtenu par le choix de WxH=5x5.

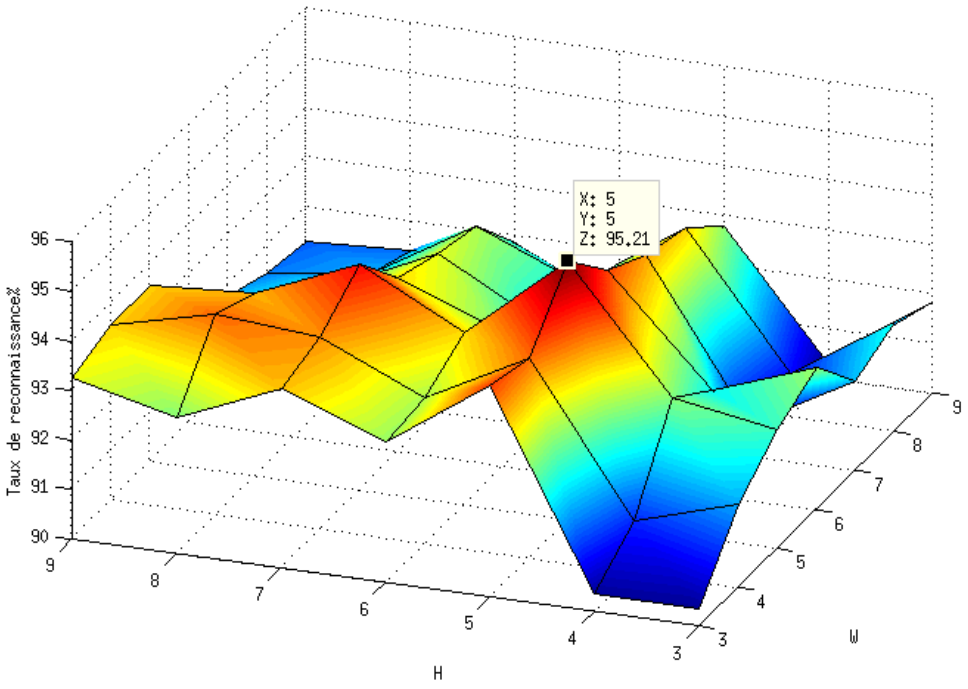


Figure 44 : Taux de reconnaissance en fonction des paramètres W et H. par l'algorithme des k-PPVF & Bi-grammes

Le tableau (Tableau 16) montre le taux de reconnaissance obtenu pour différents choix des valeurs de ces paramètres, pour l'algorithme des K-PPVF ainsi que la combinaison de ce dernier avec le modèle bi-grammes. On constate que la valeur optimale pour ces valeurs est 5×5. En plus, on peut remarquer que le modèle bi-grammes améliore, dans tous les cas, les performances de l'algorithme des k-plus proches voisins.

Tableau 16 : Taux de reconnaissance pour différents choix de la taille de la grille

	Taux de reconnaissance en %					
Taille de la grille	***3×3***	***3×5***	***5×5***	***5×7***	***7×5***	***9×9***
K-PPVF	86.10	91.70	93.29	92.33	92.13	89.18
K-PPVF & Bi-grammes	90.34	94.27	95.20	94.55	93.77	91.29

Nous avons utilisé deux méthodes d'extraction des caractéristiques, la méthode des distances des centres de gravités et la méthode des densités des pixels. La première méthode donne des informations sur les positions des portions du caractère dans chaque zone de la grille. Ainsi, deux zones contenant différent nombre de pixel noir mais avec un même centre de gravité auront la même caractéristique (Figure 45).

La deuxième méthode s'intéresse aux taux de pixels noirs dans une zone mais pas aux positions de ces pixels. Dans ce sens, si deux zones possèdent le même taux de pixels alors elles seront caractérisées par la même valeur pour n'importe quelle position ou orientation du trait dans ces zones (Figure 46).

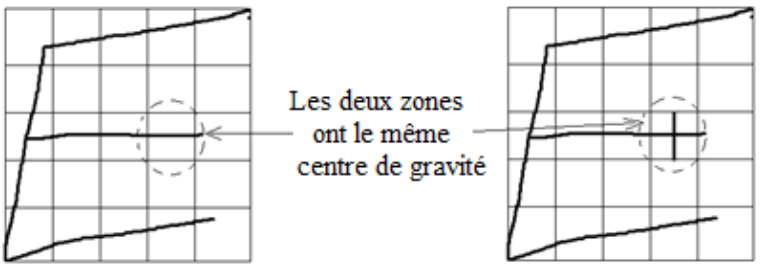


Figure 45 : Deux images de deux caractères différents mais avec un même vecteur caractéristique obtenu par la méthode des distances des centres de gravité.

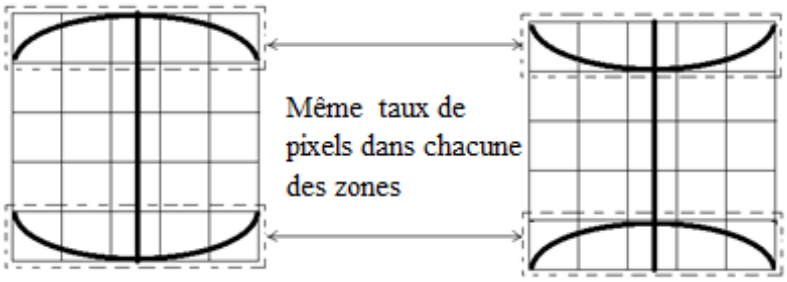


Figure 46 : Deux images de deux caractères différents mais avec exactement le même vecteur caractéristique obtenu par la méthode des densités de pixels.

Pour remédier à ce problème, et afin d'exploiter les avantages de chacune des deux méthodes, le vecteur caractéristique v utilisé dans notre approche est une combinaison linaire des deux vecteurs caractéristiques obtenus par la méthode des distances des centres de gravité v_g et celui obtenu par la méthode des densités des pixels v_d.

La figure (Figure 47) représente la courbe du taux de reconnaissance obtenu par les deux méthodes en fonction du paramètre γ dans l'équation (99). On peut constater que le meilleur taux est obtenu pour $\gamma = 0.01$.

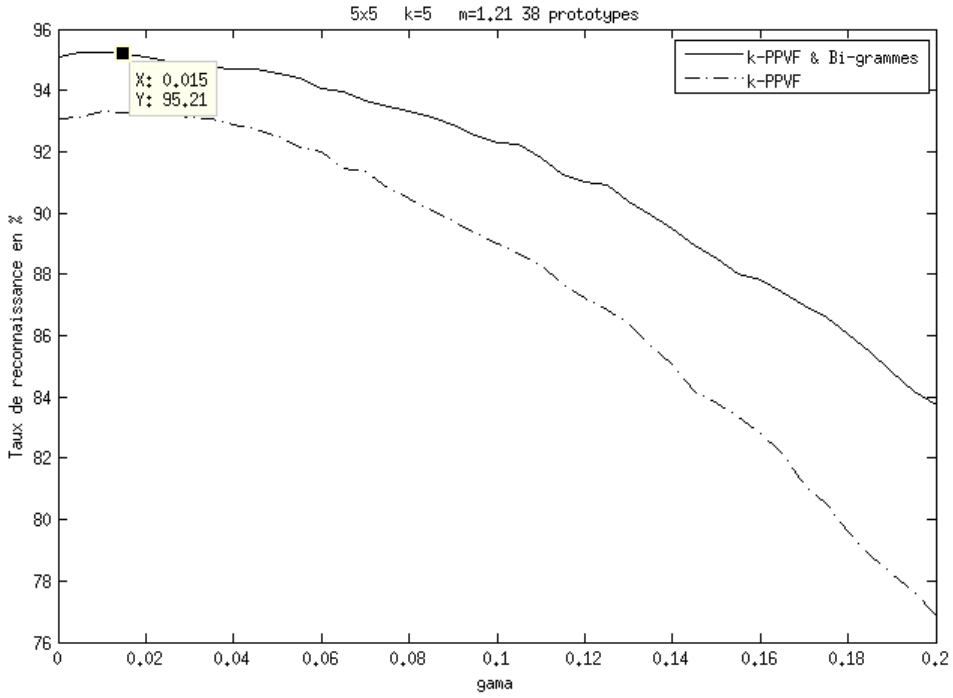


Figure 47 : Taux de reconnaissance en fonction de γ, k-PPVF & bigrammes en trait continu, k-PPVF en trait discontinu

On peut aussi remarquer qu'avec la méthode de densité des pixels ($\gamma = 0$) on obtient des résultats meilleurs qu'avec l'utilisation de la méthode des distances des centres de gravité($\gamma = 1$). Il faut noter que cette comparaison dépend de la taille de la grille utilisée. On peut aboutir à d'autres cas où la méthode des distances des centres de gravité réalise des résultats meilleurs que la méthode des densités des pixels.

On remarque que pour toutes les valeurs du paramètre γ , la combinaison de l'algorithme des k-plus proches voisins avec le modèle bi-grammes réalise des performances meilleures.

Finalement, il faut noter que le choix de ces paramètres dépend aussi du classifieur utilisé. Avec les mêmes exemples d'entraînement et les mêmes exemples de test, les valeurs de W, H et γ qui maximise le taux reconnaissance d'un classifieur ne sont pas nécessairement les mêmes pour un autre. Nous avons déjà vu que le choix de H=5 et W=5 est le plus adéquat à l'algorithme des k-plus proches voisins flous dans notre cas. Nous avons alors utilisé les mêmes données d'entraînement et de test pour l'algorithme des machines à vecteurs supports et pour un perceptron multicouche constitué d'une seule couche cachée de 220 neurones et, nous avons obtenu les résultats illustrés dans la figure (Figure 48) et la figure (Figure 49).

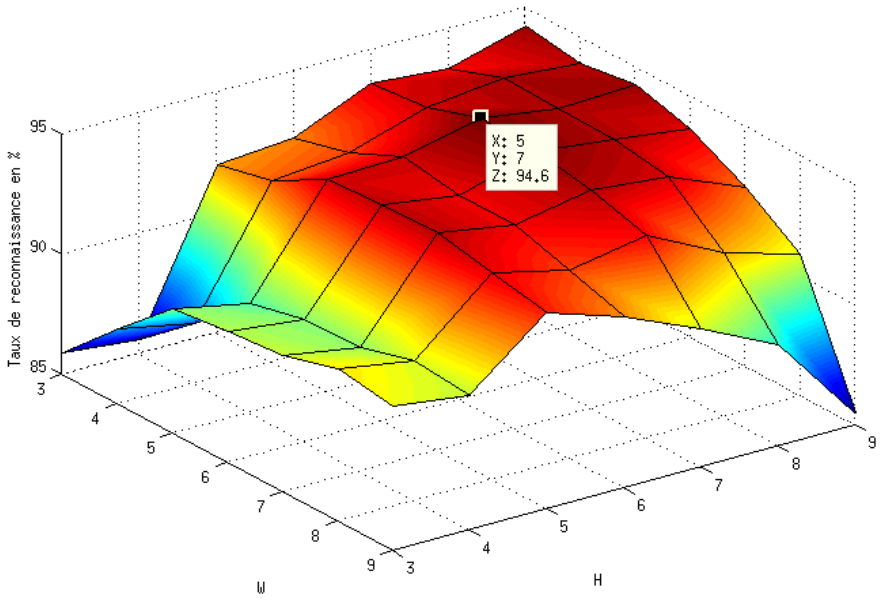


Figure 48 : Taux de reconnaissance en fonction de la taille de la grille HxW, pour l'algorithme des machines à vecteurs supports

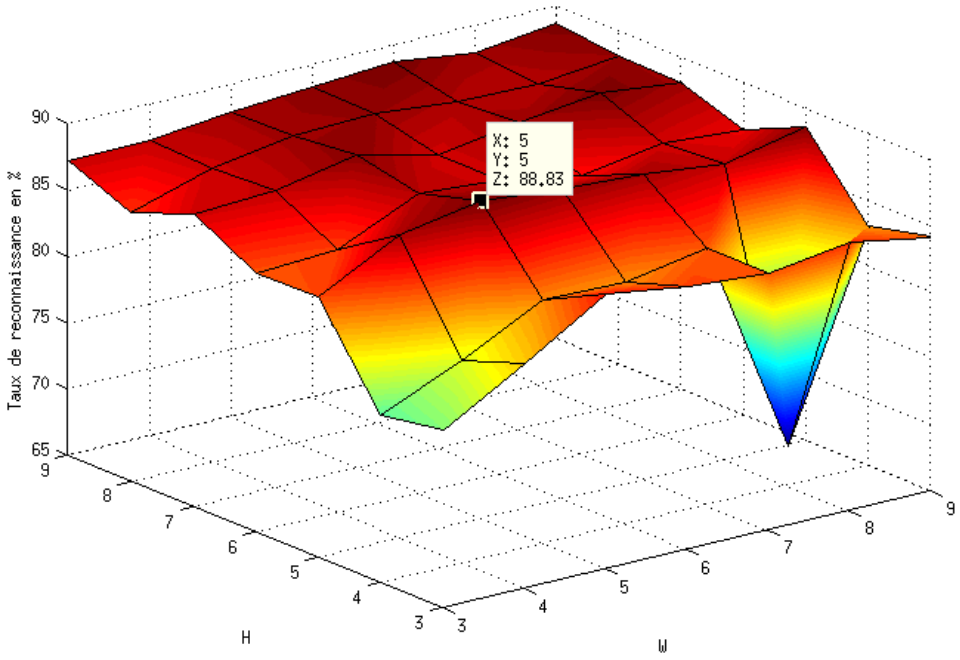


Figure 49 : Taux de reconnaissance en fonction de la taille de la grille HxW, pour un perceptron multicouche d'une seule couche caché de 220 neurones

On remarque que pour un perceptron multicouche, le meilleur résultat est obtenu (88,83%) pour W=5 et H=5, alors que pour les machines à vecteurs supports les valeurs adéquates sont W=5 et H=7 qui ont abouti à un taux de reconnaissance de 94,6%.

II.2. Les paramètres k et m

L'algorithme des K-PPVF se base essentiellement sur l'équation (97) vue dans le chapitre 2. Dans cette équation on trouve deux paramètres qui affectent notablement les performances de classification de l'algorithme : le nombre des plus proches voisins k et le facteur de fuzzyfication m.

Des grandes valeurs du paramètre k produisent un grands nombre de proposition pour chacun des caractères constituant le mot en question avec des faibles degrés d'appartenances. D'où on obtient un très grand nombre de combinaisons possibles de ces propositions. Par suite le modèle bi-grammes s'implique fortement dans la prise de la décision au détriment de l'algorithme des k-plus proches voisins flous. D'autre part, des faibles valeurs de k restreignent le nombre des combinaisons possibles, d'où l'intervention du modèle bi-grammes dans la prise de discision est moins importante.

Les faits précédemment cités sont illustrés dans la figure (Figure 50), qui représente le taux de reconnaissance obtenu par les deux méthodes pour différentes valeurs de k. Pour k=1, le taux de reconnaissance est le même pour les deux méthodes, ceci est expliqué par le fait que l'algorithme des k-PPVF génère une proposition pour chaque caractère, ce qui donne une seule combinaison possible, la probabilité de cette séquence de caractères est calculée par le modèle bi-gramme et puisque c'est la seule et unique proposition, alors cette proposition est le seul résultat possible. En augmentant la valeur de k, on remarque que le modèle bi-grammes aide notablement le k-PPVF à prendre les bonnes décisions. A un certain stade (k>11), on remarque que l'intervention du modèle bi-grammes n'est plus souhaitable, car elle ne fait que détériorer les performances des k-PPVF.

Le choix du facteur de fuzzification m affecte la sévérité de classification de l'algorithme des k-PPVF. Si m=1 alors on effectue un classement dur, autrement dit, un élément peut appartenir ou non à une classe donnée, l'algorithme se comporte alors exactement comme sa version classique (k plus proches voisins). Si m est plus grand que 1, la classification devient plus lisse, par suite un élément peut appartenir à plusieurs classes avec des degrés d'appartenance différents.

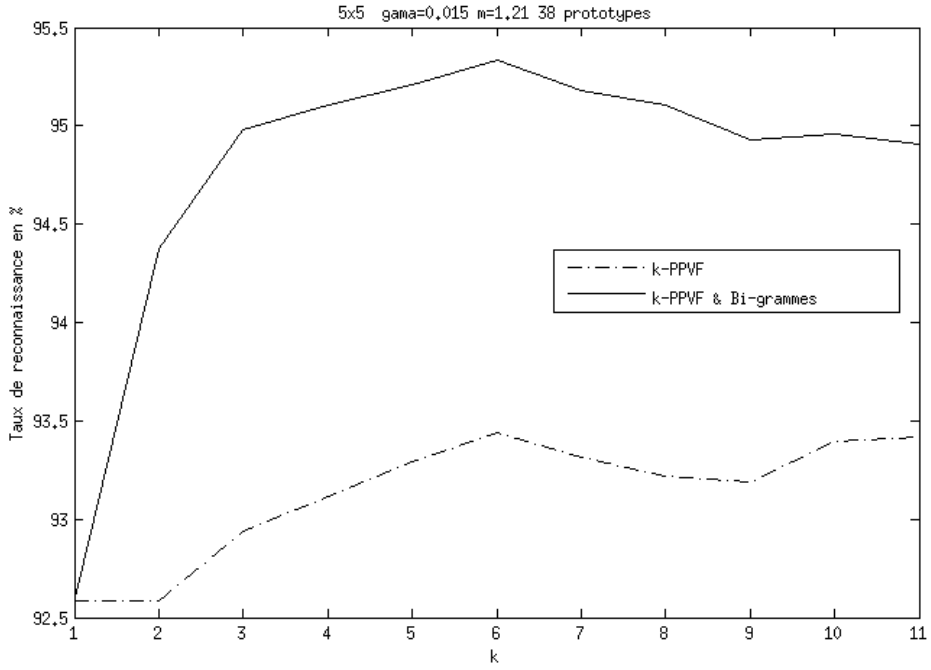


Figure 50 : Taux de reconnaissance en fonction de k, k-PPVF & bi-grammes en trait continu, k-PPVF en trait discontinu

La figure (Figure 51) illustre la courbe du taux de reconnaissance obtenu par les deux méthodes pour différentes valeurs du paramètre m. Le taux de reconnaissance augmente suivant m jusqu'à une valeur maximal 95,23% pour m=1.21 puis diminue. Là aussi, on remarque l'apport du modèle bi-grammes à l'amélioration du taux de reconnaissance.

Finalement, il faut noter que ces courbes sont obtenues par l'utilisation des c-moyennes floues pour la réduction du nombre d'exemples. On peut aussi utiliser la méthode des c-moyens classique mais les résultats seront inférieurs aux précédents comme le montrent les figures (Figure 52) et (Figure 53).

II.3. Nombre des sous classes

Dans notre système, nous avons utilisé l'algorithme des c-moyennes flou afin de réduire le nombre d'exemple d'entraînement et d'éliminer les intrus. Nous passons alors d'un ensemble d'exemples d'environs 270 exemples par classe à un autre plus réduit (38 exemples par classe) et plus dispersé dans l'espace des entrées tout en gardant les performances de reconnaissance intactes voir même plus élevées parfois. Le tableau (Tableau 17) montre les résultats obtenus par notre système par l'utilisation de l'algorithme des c-moyens classique et des c-moyennes floues.

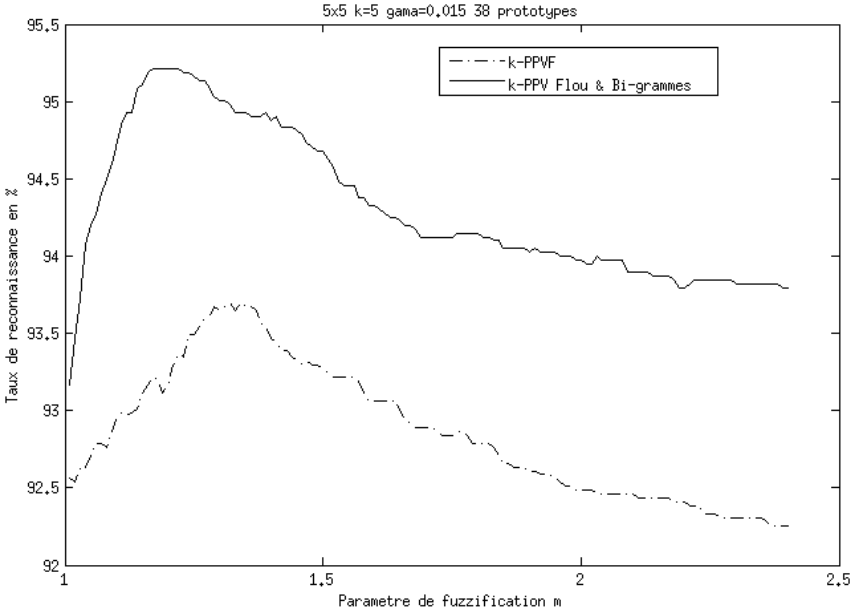


Figure 51 : Taux de reconnaissance en fonction de m, k-PPVF & bi-grammes en trait continu, K-PPVF en trait discontinu

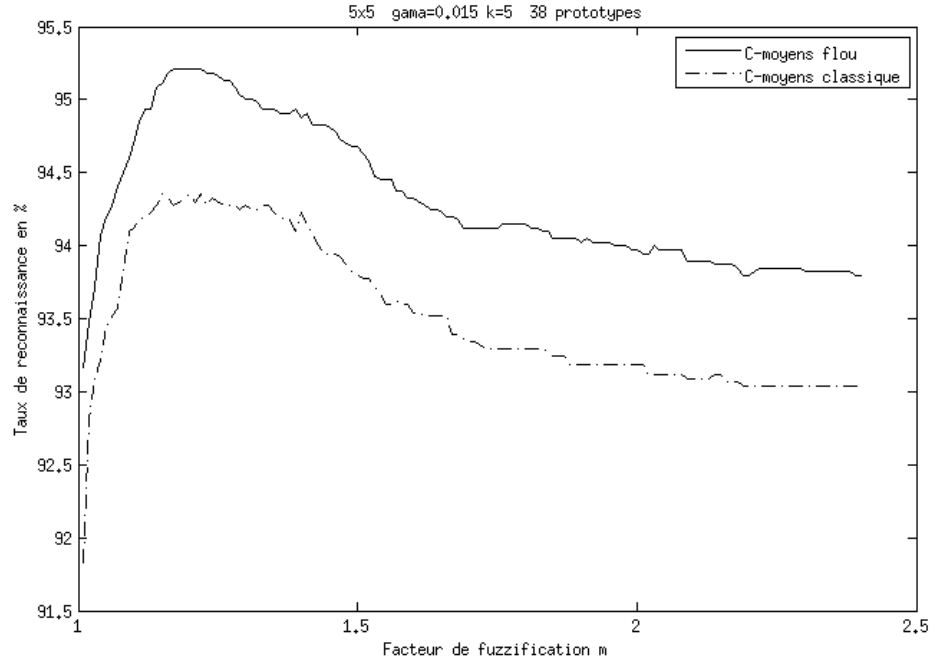


Figure 52 : Comparaison des Taux de reconnaissance en fonction de m pour le k-PPVF & bi-grammes selon la méthode d'extraction des prototypes utilisée

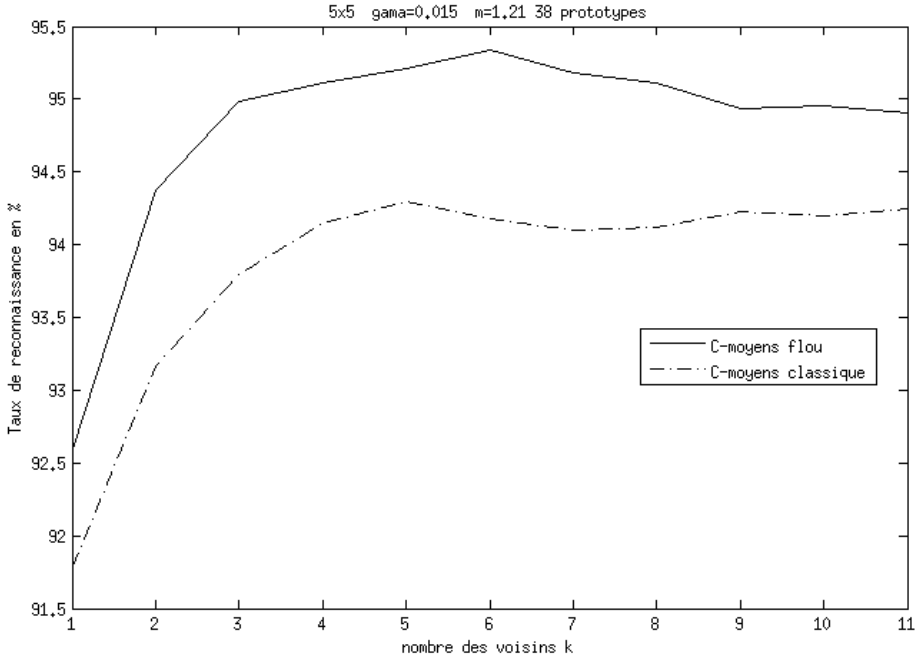


Figure 53 : Comparaison des taux de reconnaissance en fonction de k pour le k-PPVF & bi-grammes selon la méthode d'extraction des prototypes utilisé

Tableau 17: Taux de reconnaissance suivant la méthode d'extraction des prototypes utilisée

Méthode D'extraction des prototypes	*Nombre des prototypes par classe*	*Taux de reconnaissance*		*Temps d'exécution*	
		k-PPVF	*k-PPVF & Bi-grammes*	*k-PPVF*	*k-PPVF & Bi-grammes*
C-moyens classique	*10*	*89,86%*	*92,43%*	*2,71s*	*2,78s*
	38	*92,73%*	*94,30%*	*3,598s*	*3,58s*
C-moyennes floues	*10*	*90,66 %*	*92,89%*	*2,018s*	*2,04s*
	38	*93,29%*	*95,23%*	*3,456s*	*3,61s*
Aucun	*~ 270*	*93,19%*	*95,107%*	*24,472s*	*24,987s*

Il faut noter que l'utilisation des modèles Bi-grammes améliore le taux de reconnaissance dans tous les cas de figure. On remarque aussi dans les deux dernières lignes de ce tableau que le taux de reconnaissance réalisé par l'algorithme des k-PPVF combiné avec le modèle Bi-gramme après une réduction de prototypes par la méthode des c-moyennes floues (95,23%), est approximativement le même sans aucune réduction de prototypes (95,10%). Bien que ces deux résultats soient semblables, on remarque une grande différence dans le temps d'exécution des deux méthodes (24,98s pour l'utilisation de la totalité des exemples comme prototypes contre 3,61s pour 38 prototypes par classe).

Cette différence s'explique par le fait que l'algorithme des k-PPVF ne possède pas une phase d'entraînement. A chaque présentation d'un vecteur inconnu, l'algorithme compare ce dernier à tous les autres prototypes (270 prototypes par classe). Par contre, dans notre approche, on passe par une phase qu'on peut appeler une phase d'entraînement où on calcule un nombre de représentants pour chaque classe (38 prototype) par l'algorithme des c-moyennes flou. Ainsi, au lieu de comparer un vecteur inconnu aux 270 prototypes par classe on n'utilise que 38 par classe.

La figure (Figure 54) illustre le taux de reconnaissance des deux méthodes en fonction du nombre des prototypes par classe. On remarque qu'avec un nombre réduit (5 prototypes par classe) de ces prototypes on obtient déjà un taux de reconnaissance acceptable (91,2%). Ce taux augmente jusqu'à 95,23% pour un nombre de prototypes proche de 40. Au-delà des 40 prototypes par classe, on remarque que le taux de reconnaissance reste stationnaire voir même décroissant.

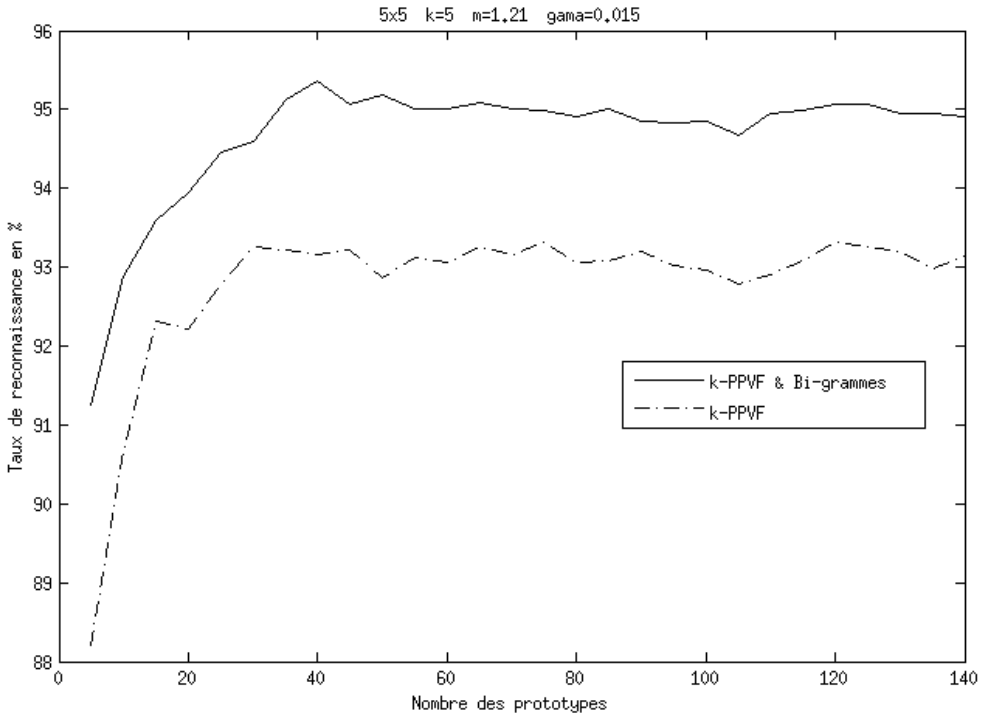


Figure 54 : Taux de reconnaissance en foction du nombre des prototypes utilisé.

III. COMPARAISON DES RESULTATS

Pour mettre en valeur notre approche, nous avons testé différents types de classifieurs en utilisant le même ensemble d'exemple d'entraînement et le même ensemble d'exemple de test avec la même méthode d'extraction des caractéristiques.

Le tableau (Tableau 18) illustre les différents paramètres utilisés pour la comparaison. Le tableau (Tableau 19) montre les taux de reconnaissance pour chaque méthode.

Tableau 18: Différents paramètres utilisés

	Paramètre	Valeur
Bases de données	Entrainement	8847 caractères
	Test	3965 caractères
	Corpus	133178 Mots
Extraction des caractéristiques	Taille de la grille	5×5
	Gama	0.015
C-moyennes floues	Nombre de sous classes	38
	Facteur de fuzzyfication m	1.2
k-PPVF	Nombre des voisins k	5
	Facteur de fuzzyfication m	1.15
PMC	Nombre de neurones dans la CC	220
	Fonction d'activation	Sigmoïde
	Erreur	0.01
	Nombre d'itérations maximal	50
SVM	Erreur	0.01
	Nombre d'itérations maximal	5

On remarque que le meilleur taux de reconnaissance est obtenu par la méthode des K-PPVF combinée avec le modèle de langage Bi-grammes, suivi par la SVM et les K-PPVF, puis vient en dernier lieu le PMC.

Tableau 19: Taux de reconnaissances obtenus

Algorithme	Taux de Reconnaissance	Temps d'exécution
k-PPVF	93,29%	3,598s
k-PPVF & Modèle Bi-grammes	95,23%	3,571s
PMC	88,83 %	1,007s
SVM	93,82 %	5,164s

Pour les temps d'exécutions cités dans le tableau (Tableau 19), il s'agit du temps écoulé pour extraire les caractéristiques de la totalité des exemples de test, leurs classifications ainsi que le calcul du taux de reconnaissance. Ces valeurs dépendent de plusieurs critères : le matériel utiliser, le système d'exploitation, le langage de programmation… etc. Le tableau (Tableau 20) liste les différentes caractéristiques de l'environnement d'exécution de notre application.

Tableau 20: Caractéristique de l'environnement de travail

Caractéristique	Valeur
Microprocesseur	Intel Core I5-2520M 2,50GH (4CPUs)
RAM	4Go
System d'exploitation	Ubuntu 12.4.04 32bit
Langage de programmation	Java (JRE 7)

La figure (Figure 55) montre un exemple d’une image d’un texte écrit en Tifinagh. La figure (Figure 56) montre la même image après son prétraitement et squelettisation.

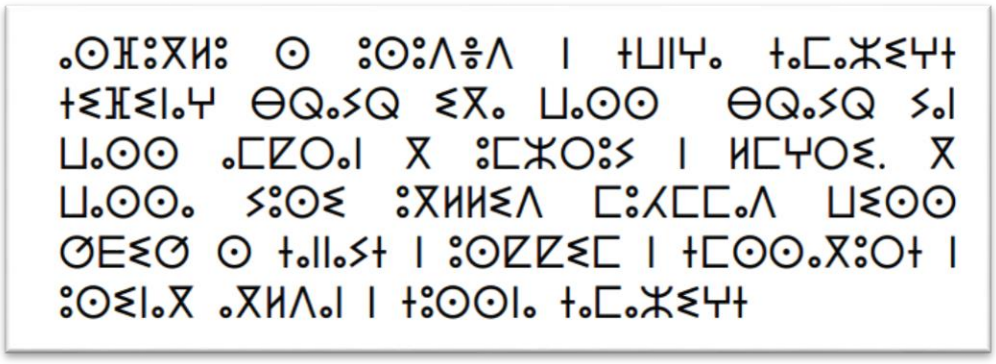


Figure 55: Exemple d'une image d'un texte en Tifinagh

Le texte obtenu par les différentes méthodes est le suivant (les caractères en gras

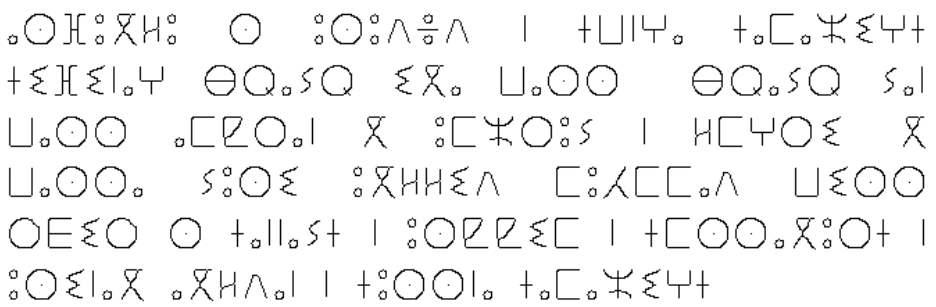


Figure 56: Limage dans la figure 55 après squelitisation

soulignés sont mal classés) :

1. K Plus proche voisins Flous:

ⴰⵙⵅⵙⵅⵏⵙ ⵙ ⵙⵙⵙⵏⵙⵏ ⵏ ⵜⵡⵏⵡⴰ ⵜⴰⵎⴰⵣⵅⵙⵜ
ⵜⵅⵅⵅⵏⴰⵡ ⵀⵇⴰⵔⵇ ⵅⵅⴰ ⵡⴰⵙⵙ ⵀⵇⴰⵔⵇ ⵔⴰⵏ
ⵡⴰⵙⵙ ⴰⵎⵣⵅⴰⵏ ⵅ ⵙⵎⵣⵙⵔ ⵏ ⵍⵎⵡⵅⵙⵙ ⵅ
ⵡⴰⵙⵙⴰ ⵔⵙⵙⵅ ⵙⵅⵍⵍⵅⵏⵏⴰⵏ ⵡⵅⵙⵙ
ⵙⵉⵅⵙ ⵙⵅⵍⵍⴰⵔⵜ ⵏ ⵜⵎⵙⵙⵅⵎ ⵏ ⵜⵎⵙⵙⴰⵅⵙⵔⵜ ⵏ
ⵙⵙⵅⵏⴰⵅ ⴰⵅⵍⵏⴰⵏ ⵜⵙⵙⵙⵏⴰ ⵜⴰⵎⴰⵣⵅⵔⵜ

2. K Plus proche voisins Flous & Modèle Bi-grammes:

ⴰⵓⵀⵙⵅⵏⵙ ⵓ ⵙⵓⵙⵍⵙⵍ ⵏ ⵜⵍⵏⵢⴰ ⵜⴰⵛⴰⵣⵚⵢⵜ
ⵜⵚⵀⵚⵏⴰⵢ ⵝⵇⴰⵥⵇ ⵚⵅⴰ ⵍⴰⵓⵓ ⵝⵇⴰⵥⵇ ⵥⴰⵏ
ⵍⴰⵓⵓ ⴰⵛⵣⵔⴰⵏ ⵅ ⵙⵛⵣⵔⴰⵙⵥ ⵏ ⵉⵛⵕⵔⴰⵚ ⵅ
ⵍⴰⵓⵓⴰ ⵥⵙⵓⵚ ⵙⵅⵉⵉⵚⵍ ⵇⵙⵍⵛⵛⴰⵍ ⵍⵚⵓⵓ
ⵔⵚⵚⵔ ⵓ ⵜⴰⵏⵏⴰⵥⵜ ⵏⵙⵓⵣⵣⵚⵛ ⵏ ⵜⵇⵓⵓⴰⵅⵙⵔⵜⵏ
ⵙⵓⵚⵏⴰⵅ ⴰⵅⵉⵍⴰⵏⵏ ⵜⵙⵓⵓⵏⴰ ⵜⴰⵛⴰⵣⵚⵢⵜ

3. Réseaux de neurones Artificiel :

ⴰⵓⵅⵙⵅⵏⵙ ⵓ ⵙⵓⵙⵍⵙⵍ ⵏ ⵜⵍⵏⵢⴰ ⵜⴰⵛⴰⵣⵚⵢⵜ
ⵜⵚⵅⵚⵏⴰⵢ ⵝⵓⴰⵀⵓ ⵚⵅⴰ ⵍⴰⵓⵓ ⵝⵓⴰⵀⵓ ⵀⴰⵏ
ⵍⴰⵓⵓ ⴰⵛⵣⵔⴰⵏ ⵅ ⵙⵛⵣⵔⴰⵙⵀ ⵏ ⵉⵛⵢⵔⵚ ⵅ
ⵍⴰⵓⵓⴰ ⵀⵙⵓⵚ ⵙⵅⵉⵉⵚⵍ ⵇⵙⵍⵛⵛⴰⵍ ⵍⵚⵓⵓ
ⵓⵚⵚⵓ ⵓ ⵜⴰⵏⵏⴰⵀⵜ ⵏⵙⵓⵣⵣⵚⵛ ⵏ ⵜⵇⵓⵓⴰⵅⵙⵔⵜⵏ
ⵙⵓⵚⵏⴰⵅ ⴰⵅⵉⵍⴰⵏⵏ ⵜⵙⵓⵓⵏⴰ ⵜⴰⵛⴰⵣⵚⵢⵜ

4. Machine à Vecteurs Support:

ⴰⵓⵀⵙⵅⵏⵙ ⵓ ⵙⵓⵙⵍⵙⵍ ⵏ ⵜⵍⵏⵢⴰ ⵜⴰⵛⴰⵣⵚⵢⵜ
ⵜⵚⵀⵚⵏⴰⵢ ⵝⵇⴰⵥⵇ ⵚⵅⴰ ⵍⴰⵓⵓ ⵝⵇⴰⵥⵇ ⵥⴰⵏ
ⵍⴰⵓⵓ ⴰⵛⵣⵔⴰⵏ ⵅ ⵙⵛⵣⵔⴰⵙⵥ ⵏ ⵉⵛⵢⵔⵚ ⵅ
ⵍⴰⵓⵓⴰ ⵥⵙⵓⵚ ⵙⵅⵉⵉⵚⵍ ⵇⵙⵍⵛⵛⴰⵍ ⵍⵚⵓⵓ
ⵓⵚⵚⵓ ⵓ ⵜⴰⵏⵏⴰⵥⵜ ⵏⵙⵓⵣⵣⵚⵛ ⵏ ⵜⵇⵓⵓⴰⵅⵙⵔⵜⵏ
ⵙⵓⵚⵏⴰⵅ ⴰⵅⵉⵍⴰⵏⵏ ⵜⵙⵓⵓⵏⴰ ⵜⴰⵛⴰⵣⵚⵢⵜ

On remarque que notre approche a bien classé la totalité des caractères, suivi par l'algorithme machine à vecteurs supports (deux caractères mal classés). Il faut noter qu'après l'opération de squelettisation, le mot "ØEΣØ" est devenu "OEΣO". Ainsi, en réalité, l'algorithme des machines à vecteurs supports a bien classé 'aveuglement' la totalité des caractères sans tenir compte de l'existence du mot en question dans la langue Amazighe alors que notre approche s'est bien rendu compte de l'inexistence de ce mot dans la langue Amazighe et a proposé exactement le mot original. Cet exemple illustre bien la robustesse de notre approche face aux bruits et aux différentes transformations morphologiques qu'un caractère peut subir.

Un autre exemple est présenté dans la figure (Figure 57). Dans cet exemple, le mot est très ambigu, ce qui explique la longue liste des mots proposés. En se basant uniquement sur les degrés d'appartenance calculés par l'algorithme des k-PPVF le résultat sera "+ΣΕΙ米ₒO". Mais

Image du Mot :

Proposition des candidats par le k-PPVF :

Résultats

Résultats | WordsScore

Char0	mu	Char1	mu	Char2	mu	Char3	mu	Char4	mu	Char5	mu	Char6	mu
+	1.0	Σ	0.757...	Ε	0.783...	Σ	0.14...	米	0.03...	ₒ	1.0	O	0.87...
		I	0.242...	Ⅽ	0.216...	I	0.85...	米	0.96...			ʘ	0.12...

Génération des mots et calcul des probabilités :

Résultats

Résultats | WordsScore

word	Log(mu(word))	Log(P(word))	Log(mu(word)*P(word))
+ΣΕΣ米ₒO	-2.6001787092338486	-28.086738154143617	-30.686916863377466
+ΣΕΣ米ₒʘ	-7.838700776457205	-28.51899454349732	-36.35769531995453
+IΕΣ米ₒO	-5.6981865172861506	-33.08077905335543	-38.77896557064158
+ΣΕI米ₒO	-2.8085881477754002	-33.30416085203651	-36.11274899981191
+ΣΕΣ米ₒO	-5.880840739101002	-28.690901209957204	-34.57174194905821
+IΕΣ米ₒO	-3.7403264799299465	-33.25268571981532	-36.993012199745266
+IΕΣ米ₒʘ	-8.978848547153303	-33.68494210916902	-42.66379065632233
+ΣΕI米ₒO	-6.089250177642553	-33.65604376244923	-39.745293940091784
+ΣΕI米ₒʘ	-0.8507281104191964	-33.47606751849639	-34.32679562891559
+IΕI米ₒʘ	-3.9487359184714985	-38.47010841770821	-42.41884433617971
+IΕΣ米ₒʘ	-7.0209885097971	-33.856848775628904	-40.877837285426004
+ΣΕI米ₒO	-4.131390140286349	-33.82795042890911	-37.95934056919546
+IΕI米ₒO	-1.9908758811152947	-38.642015084168094	-40.63289096528339
+IΕI米ₒʘ	-7.229397948338652	-38.82199132812093	-46.05138927645958
+IΕI米ₒO	-5.271537910982448	-38.99389799458081	-44.26543590556326
+ΣⅭΣ米ₒʘ	-5.8441243885735705	-25.582908342179476	-31.427032730753048
+ΣⅭΣ米ₒO	-9.124786418440724	-26.18707139799306	-35.311857816433786
+ΣⅭΣ米ₒO	-3.8862643512173665	-25.754815008639362	-29.64107935985673
+IⅭΣ米ₒʘ	-6.984272159269669	-31.03470479922129	-38.01897695849096
+ΣⅭI米ₒO	-4.094673789758918	-29.94194466184162	-34.03661845160054
+ΣⅭΣ米ₒO	-7.16692638108452	-26.35897806445294	-33.52590444553746
+IⅭΣ米ₒO	-5.026412121913466	-31.206611465681178	-36.233023587594644
+IⅭΣ米ₒO	-10.264934189136822	-31.63886785503488	-41.903802044171705
+ΣⅭI米ₒO	-7.375335819626072	-30.293827572254344	-37.66916339188042
+ΣⅭI米ₒO	-2.1368137524027144	-30.113851328301507	-32.25066508070422
+IⅭI米ₒʘ	-5.234821560455018	-35.39374111888344	-40.62856267933846
+IⅭΣ米ₒʘ	-8.307074151780618	-31.810774521494764	-40.11784867327538
+ΣⅭI米ₒO	-5.417475782269868	-30.465734238714226	-35.883210020984095
+IⅭI米ₒO	-3.276961523098813	-35.56564778534332	-38.842609308442135
+IⅭI米ₒO	-8.51548359032217	-35.74562402929616	-44.26110761961833
+IⅭI米ₒO	-6.557623552965966	-35.91753069575604	-42.475154248722006
+ΣΕΣ米ₒʘ	-4.558038746590053	-27.91483148768373	-32.472870234273785

Figure 57: Génération des candidats par le k-PPVF et le modèle Bi-gramme

avec la prise en compte des probabilités fournis par le modèle Bi-grammes des mots proposés, le résultat obtenu est " ⵜⵙⵛⵙⵣⵔⵕ ".

Afin d'avoir une idée générale sur les performances de notre approche, nous avons calculé les taux de reconnaissances par caractère. Les figures (Figure 58), (Figure 59), (Figure 60) et (Figure 61) représentent respectivement les matrices de confusion de l'algorithme des K-PPVF, du K-PPVF combiné avec le modèle Bi-gramme, le PMC et les SVM.

Dans ces matrices, l'intersection de la ligne i et la colonne j représente le nombre des images du caractère c_i reconnu en tant que c_j. la ligne en bas de la matrice représente le taux de reconnaissance de chaque caractère. Par exemple, dans la deuxième matrice (Figure 58), le caractère " ⵣ " est classé en tant que " ⵣ " pour 46 exemples et en tant que " ⵥ " pour 24 exemples le taux de reconnaissance de ce caractère est 60.52% (en bas de la colonne ⵣ). De même pour les trois autres matrices 81.57% (K-PPVF & Bi-grammes), 52.63% (Réseaux de neurones) et 52.63% (SVM).

IV. CONCLUSION

Dans ce chapitre, nous avons expliqué en détail toutes les composantes de notre système de reconnaissance des caractères manuscrits Tifinagh. L'apport de notre approche réside dans plusieurs parties de ce système. Lors de l'extraction des caractéristiques nous avons utilisé une combinaison linéaire de deux méthodes, à savoir, la méthode des densités de pixels et la méthode des distances des centres de gravité. Lors de la classification, nous avons introduit l'algorithme des c-moyennes floues pour extraire des prototypes qui vont servir comme exemples d'apprentissage de l'algorithme des k-plus proches voisins flous qui est au cœur de notre système. Finalement, afin d'introduire les informations linguistiques, nous avons introduit le modèle bi-grammes qui a amélioré significativement les performances du système.

Une comparaison avec d'autre algorithme de classification, comme les réseaux de neurones et les machines à vecteurs de supports a montré la robustesse de notre approche.

Figure 58: Matrice de confusion du k-PPVF

	°	Ж	Ж	Ö	Q	+	Ё	Г	ß	Σ	Н	Л	Z	О	Ø	Λ	Е	Н	Х	Ф	I	R	И	Г	Ц	Ж	Ç	Ч	Θ	Ι	~
°	667	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
Ж	0	62	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
Ж	0	10	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
ё	0	0	0	13	0	0	0	8	1	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0
О	0	0	0	0	169	4	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Q	1	0	0	4	54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
+	0	0	0	0	0	361	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	2	0	0
Ё	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Г	1	0	0	0	1	0	80	0	3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
ß	0	0	0	3	0	0	0	220	0	1	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0
Σ	2	0	0	1	1	0	0	1	483	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
Н	0	1	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Л	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Z	1	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
О	2	0	0	0	14	3	0	0	0	0	0	0	196	1	0	0	0	0	0	0	0	0	1	0	0	0	0	5	1	0	0
Ø	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Л	0	0	0	0	0	0	0	0	0	0	0	0	1	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Е	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Н	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Х	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	93	0	0	3	0	0	0	2	0	0	0	0	0	0
Ф	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	4	0	0	0
I	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0
Й	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	51	0	0	0	0	0	0	0	0	0	0
И	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	141	0	0	0	0	0	0	0	0	1
Г	1	0	0	0	0	0	0	1	2	0	0	0	0	0	0	3	0	0	0	0	0	0	200	1	0	0	0	0	0	0	0
Ц	1	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	103	0	0	0	0	1	0	0
Ж	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0
Ç	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0
Ч	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	72	0	0	0	0
Θ	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	70	0	0	0
Ι	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	274	0	7
~	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TP%	97.65	81.57	90.32	76.47	89.41	88.52	99.44	96.42	94.11	93.22	97.37	92.3	100.0	100.0	94.68	93.33	96.0	86.48	96.87	95.87	75.0	83.33	89.47	97.24	98.03	96.26	68.75	90.47	98.63	88.6	98.56	87.5

Figure 59: Matrice de confusion du k-PPVF & Bi-grammes

Figure 60: Matrice de confusion des PMC

Figure 61: Matrice de confusion de la SVM

CONCLUSION GENERALE ET PERSPECTIVES

Dans ce travail de thèse qui s'inscrit dans le cadre de la reconnaissance de formes par la logique flou, nous avons touché aux différents aspects de la reconnaissance de formes, en particulier la reconnaissance automatique des manuscrit Tifinagh : Le prétraitement d'une image numérique, la segmentation, l'extraction des caractéristiques, classification, validation et teste.

Au début, Nous avons présenté une étude bibliographique détaillée sur les composantes d'un système de reconnaissance de formes. Nous avons ensuite présenté un tour d'horizon sur les techniques floues utilisées pour la reconnaissance de formes. Finalement nous avons présenté notre système de reconnaissance des manuscrits Tifinagh ainsi que les résultats obtenus.

Comme dans toutes les langues, pour lire un texte, un être humain utilise souvent ses connaissances antérieures de la langue. Cette idée était l'essence de notre approche. La logique floue est une logique plus "humaine", la langue est aussi une caractéristique humaine. Notre contribution peut être présentée en deux étapes :

1. L'inconvénient majeur de l'algorithme des k-PPVF est l'absence d'une étape d'entraînement. De ce fait, à chaque présentation d'un vecteur en entrée l'algorithme fait appel à tous les exemples de la base d'entraînement pour donner une sortie. Nous avons donc introduit une phase de réduction d'exemples à l'aide de l'algorithme des c-moyennes floues, qu'on peut considérer comme phase d'entraînement.
2. Dans les cas des caractères ambigus un classifieur quelconque (même un être humain) ne peut faire un classement correct basé uniquement sur la base d'apprentissage. Un être humain fait toujours appel à ses connaissances de la langue ainsi qu'à sa connaissance des formes des différents caractères. Notre deuxième contribution consiste à l'utilisation des modèles N-grammes pour guider le classifieur à choisir la suite des caractères convenables parmi une liste pondérée de propositions engendrées par le k-PPVF.

Lors de nos recherches nous n'avons utilisé que des outils libres comme Eclipse, Java, JAI (Java Advanced Imagin) pour le traitement des images numériques, Encog pour l'implémentation de différent type de réseau de neurones. En résultat, nous avons pu réaliser

une application java tout-en-un pour le traitement de la langue Amazighe, qui peut servir comme une plateforme Libre pour les jeunes chercheurs dans le domaine. Nous avons aussi utilisé le produit incontournable de Google ‘Google App Engine ‘ pour la réalisation d’une application web (http://tutlaytino.appspot.com) pour la détection et la correction des erreurs d’orthographe dans un texte Amazighe.

Le prétraitement des manuscrits est une opération de grande importance dans le domaine de la reconnaissance de formes manuscrite. Il s’avère alors évident d’aller dans ce sens pour la mise en œuvre des nouvelles méthodes à base de logique floue pour le traitement de l’image numérique.

Le traitement automatique de la langue Amazighe est encore un champ fertile. Les méthodes à base de logique floue aussi ont leur mot à dire dans le domaine de la reconnaissance de formes, en particulier le nouveau concept de logique flou type-2 (Type-2 fuzzy logic) qu’on va essayer de l’appliquer comme des perspectives de notre travail.

BIBLIOGRAPHIE

[1] M. Cheriet, N. Karma, C.-L. Liu et C. Y. Suen, Character Recognition Systems: A guid for students and practitioners, New Jersey: Wiley, 2007.

[2] R. Polikar, «Pattern Recognition,» *Wiley Encyclopidai of Biomedical Engineering,* pp. 1-22, 2006.

[3] S. Theodoridis et K. Koutroumbas, Pattern Recognition (Fourth Edition), San Diego, California: Academic Press, 2009.

[4] A. Meyer-Bäse , Pattern Recognition in Medical Imaging, Elsevier, 2004.

[5] C. Kurian, «A Review on Technological Development of Automatic Speech Recognition,» *International Journal of Soft Computing and Engineering,* vol. 4, n° 4, pp. 80-86, 2014.

[6] R. P. Lippmann, «Speech recognition by machines and humans,» *Speech Communication,* vol. 22, pp. 1-15, 1997.

[7] R. Duda, P. Hart et D. Stork, Pattern Classification, 2nd ed., New York: Wiley, 2000.

[8] A. Ilin et T. Raiko, «Practical Approaches to Principal Component Analysis in the Presence of Missing Values,» *Journal of Machine Learning Research,* vol. 11, pp. 1957-2000, 2010.

[9] A. Jain et D. Zongker, «Feature selection: evaluation, application, and small performance,» *IEEE Trans. Pattern Analysis Machine Intellegence,* vol. 19, n° 2, p. 153–158, 1997.

[10] X. H. Daniela, J. H. Christopher et G. S. Roger, «Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages,» *International Journal of Computer Science Issues,* vol. 4, n° 1, pp. 16-23, 2009.

[11] F. Pernkopfa, «Bayesian network classifiers versus selective k-NN classifier,» *Pattern Recognition,* vol. 38, pp. 1-10, 2005.

[12] A. R. Webb, Statistical Pattern Recognition, 2nd Edition, New York: Wiley, 2002.

[13] C. Bishop, Neural Networks for Pattern Recognition, Oxford: Oxford University, 1995.

[14] A. Ghaffari, H. Abdollahi, M. R. Khoshayand, I. Soltani Bozchalooi, A. Dadgar et M. Rafiee-Tehrani, «Performance comparison of neural network training algorithms in modeling of bimodal drug delivery,» *International Journal of Pharmaceutics*, vol. 327, p. 126–138, 2006.

[15] M. A. Arbib, The Handbook of Brain Theory and Neural Networks, MIT Press, 2003, pp. 871-877.

[16] P. Banerjee et S. S. Dhal, «Comparative Performance Analysis of ANN Implemented LMS withANN for Channel Estimation in AWGN Scinareo,» *International Journal of Innovative Technology and Exploring Engineering,* vol. 1, n° 3, pp. 1-4, 2012.

[17] T. Kohonen, «The Self-Organizing Map,» *Proceeding of the IEEE,* vol. 78, n° 9, pp. 1464-1480, 1990.

[18] H. Yin, «The Self-Organizing Maps: Background, Theories, Extensions and Applications,» *Studies in Computational Intelligence (SCI),* vol. 115, p. 715–762, 2008.

[19] V. Vapnik, The nature of statistical learning theory, Springer, 1995.

[20] L. Xu, A. Krzyzak et C. Y. Suen, «Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition,» *IEEE Transaction on systems, man, and cybernetics,* vol. 22, n° 3, pp. 418-434, May/June 1992.

[21] C.-L. Liu et H. Fujisawa, «Classification and learning for character recognition: comparison of methods and remaining problems,» chez *International Workshop on Neural Networks and Learning in Document Analysis and Recognition*, 2005.

[22] J. Kittler, M. Hatef et R. P. Duin, «On Combining Classifiers,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, n° 3, pp. 226-239, March 1998.

[23] A. F. R. Rahman et M. C. Fairhurst, «Multiple Classifier Decision Combination Strategies for Character Recognition: A Review,» *International Journal on Document Analysis and Recognition,* vol. 5, n° 4, pp. 166-194, 2003.

[24] L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, New Jersey: Wiley Interscience, 2004.

[25] Y. Freund et R. E. Schapire, «A decision-theoretic generalization of on-line learning and an application to boosting,» *Journal of Computer and System Sciences,* vol. 55, n° 1, p. 119–139, 1997.

[26] Y. Freund et S. E. Robert , «Experiments with a newboosting algorithm,» chez *13th International Conference on Machine Learning*, Bari, Italy, 1996.

[27] J. Demsar, «Statistical Comparisons of Classifiers over Multiple Data Sets,» *Journal of Machine Learning Research,* vol. 7, p. 1–30, 2007.

[28] R. Caruana et A. Niculescu-Mizil, «An empirical comparison of supervised learning algorithms,» chez *ICML '06 Proceedings of the 23rd international conference on Machine learning*, 2006.

[29] C.-L. Liu, K. Nakashima, H. Sako et H. Fujisawa, «Handwritten digit recognition: benchmarking of state-of-the-art techniques,» *Pattern Recognition,* vol. 36, p. 2271–2285, 2003.

[30] V. Korde et N. Mahender, «Text Cassification and Classifiers: a survey,» *International Journal of Artificial Intelligence & Applications,* vol. 3, n° 2, pp. 85-99, 2012.

[31] S. Russell et P. Norvig, Artificial Intelligence: A Modern Approach second edition, New Jersey: Prentice Hall, 2003.

[32] J. V. de Olivera et W. Padrycz, Advances in Fuzzy Clustering and its Applications, West Sussex PO19 8SQ: John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, 2007.

[33] L. A. Zadeh, «Fuzzy Sets,» *Inform. Control*, vol. 8, pp. 338-353, 1965.

[34] J. C. Dunn, «A fuzzy relative oof the ISODATA process and its use in detecting compact, well separated clusters,» *Journal of Cybernetics,* vol. 3, pp. 32-57, 1974.

[35] E. H. Ruspini, «Numerical methodes for fuzzy clustering,» *Information Sciences,* vol. 2, n° 3, p. 319–350, 1970.

[36] I. Berget, B. H. Mevik et T. Naæs, «New Modifications and Applications of Fuzzy c-means Methodology,» *Computational Statistics and Data Analysis,* vol. 52, pp. 2403-2418, 2008.

[37] C.-T. Chang, J. Z. C. Lai et M.-D. Jeng, «A Fuzzy K-means Clustering Algorithm Using Cluster Center Displacement,» *Journal of Information Sciences and Engeneering*, vol. 27, pp. 995-1009, 2011.

[38] W. Tuker, «Couterexamples to the convergence for fuzzy ISODATA algorithms,» *The analysis of fuzzy Information*, p. FL: CRC, 1987.

[39] J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms, New York: Plenum Press, 1981.

[40] F. Klawonn, «Understanding the membership degrees in fuzzy clustering,» chez *In Proc. of the 29th Annual Conference of the German Classification Society*, 2006.

[41] I. Gath et A. B. Geva, «Unsupervised Optimal Fuzzy Clustering,» *IEEE Trans. Pattern Anal. Machine Intel.*, vol. 11, p. 773281, 1989.

[42] J. M. Krishnapuram et R. Keller, «A possibilistic approach to clustring,» *IEEE Transaction on fuzzy Systems*, vol. 1, n°2, pp. 98-110, 1993.

[43] H. Ishibuchi, K. Nozaki et H. Tanaka, «Efficient Fuzzy Partition of Pattern Space for Classification Problems,» *Fuzzy Sets and Systems*, vol. 59, pp. 295-304, 1993.

[44] R. J. Hammell, «Learning Fuzzy Rules From Data,» chez *The RTO SCI Symposium on " The Application of Information Technology (Computer Science) to Mission Systems"*, Monterey, California, 1998.

[45] M. G. C. A. Cimino, B. Lazzerini et F. Marcelloni, «A novel approach to fuzzy clustering based on a dissimilarity relation extracted from data using a TS system,» *Pattern Recognition*, vol. 39, pp. 2077-2091, May 2006.

[46] R. Fullér, Neural Fuzzy Systems, Abo: Abo Akademi University, 1995.

[47] K.-M. Lee, D.-H. Kwakb et . H. Leekwang, «Tuning of Fuzzy Models by Fuzzy Neural Networks,» *Fuzzy sets and Systems*, vol. 76, pp. 47-61, 1995.

[48] R. Saneifard et T. Allahviranloo, «Defuzzification Method for Ranking Fuzzy Numbers Based on Center of Gravity,» *Iranian Journal of Fuzzy Systems*, vol. 9, n° 6, pp. 57-67, 2012.

[49] H. L. Kwang, First Course on Fuzzy Theory and applications, Verlag Berlin Heidelberg: Springer, 2005.

[50] P. k. Simpson, «Fuzzy min-max neural networks-Part 1: Classification,» *IEEE Transaction on Fuzzy Systems,* vol. 3, pp. 776-786, 1992.

[51] J.-S. R. Jang, «ANFIS: Adaptive-Network-Based Fuzzy Inference System,» *IEEE Transactions on System, Man and Cybernetics,* vol. 23, pp. 665-685, 1993.

[52] J. M. Keller, M. R. Gray et J. A. Givens, «A fuzzy K-nearest neighbor algorithm,» *Systems, Man and Cybernetics, IEEE Transactions,* vol. 15, n° 4, pp. 580 - 585, Aug 1985.

[53] S. K. Sinha et P. W. Fieguth, «Neuro-fuzzy network for the classification of buried pipe defects,» *Automation in Construction,* vol. 15, pp. 73-83, 2006.

[54] A. Lorenz, M. Blüm, H. Ermert et T. Senge, «Comparison of Different Neuro-Fuzzy Classification Systems for the Detection of Prostate Cancer in Ultrasonic Images,» chez *IEEE Ultrasonic Symposium Proceeding*, 1994.

[55] Z. . M. Shafiq, F. Muddassar et S. A. Khayam, «A Comparative Study of Fuzzy Inference Systems, Neural Networks and Adaptive Neuro Fuzzy Inference Systems for Portscan Detection,» chez *EvoWorkshops 2008*, 2008.

[56] M. Nayak et J. R. Tripathy, «Pattern Classification Using Neuro Fuzzy and Support Vector Machine (SVM) - A Comparative Study,» *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 2, n° 5, pp. 2301-2306, May 2013.

[57] A. M. Badawi, K. G. Hasan, E.-E. A. Aly et R. A. Messiha, «Chromosomes classification based on neural networks, fuzzy rule based, and template matching classifiers,» chez *IEEE 46th Midwest Symposium on Circuits and Systems*, Cairo, 2003.

[58] C.-T. Lin, C.-M. Yeh et C.-F. Hsu, «Fuzzy Neural Network Classification Design using Support Vector Machine,» chez *ISCAS '04. Proceedings of the 2004 International Symposium on Circuits and Systems*, 2004.

[59] R. Plamondon et S. N. Srihari, «On-line an off-line Handwriting Recognition: A Comprehensive Survey,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, pp. 63-84, 2000.

[60] J. Mantas, «An overvieew of OCR Research an Development,» *Pattern Recognition,* vol. 19, pp. 425-430, 1986.

[61] S. Mori, C. Y. Suen et K. Yamamoto, «Historical Overview of OCR research and Development,» *Proceeding of the IEEE,* vol. 80, pp. 1029-1058, 1992.

[62] S. N. Srihari et S. W. Lam, «Character Recognition,» *Technical Report,* pp. CEDAR-TR-95-1, 1995.

[63] L. Zenkouar, «Normes des Technologies de l'Information pour l'Ancrage de l'écriture Amazighe,» *Etudes et Document Berbères,* vol. 27, pp. 159-172, 2008.

[64] B. El Kessab, C. Daoui, K. Moro, B. Bouikhalene et M. Fakir, «Recognition of Handwritten Tifinagh Characters Using a Multilayer Neural Networks and Hidden Markov Model,» *Global Journal of Computer Science and Technology,* vol. 11, n° 15, pp. 13-19, 2011.

[65] R. EL Ayachi, M. Fakir et B. Bouikhalen, «Recognition of Tifinaghe Characters Using a Multilayer Neural Network,» *International Journal Of Image Processing (IJIP),* vol. 5, n° 2, pp. 109-118, 2011.

[66] R. El Ayachi, K. Moro, M. Fakir et B. Bouikhalene, «On the recognition of Tifinagh Scripts,» *Journal of Theoretical and Applied Information Technology,* vol. 20, n° 2, pp. 61-66, 2010.

[67] M. Amrouch, M. Elyassa, A. Rachidi et D. Mammass, «Handwritten Amazighe Character Recognition Based on Hidden Markov Models,» *ICGST-GVIP Journal,* vol. 10, n° 5, pp. 11-18, 2010.

[68] M. Abaynarh, H. Elfadili et L. Zenkouar, «Enhanced Feature Extraction of Handwritten Characters and Recognition using Artificial Neural Networks,» *Journal of Theoretical and Applied Information Technology,* vol. 72, n° 3, pp. 355-365, 2015.

[69] N. Aharrane, K. Elmoutaouakil et K. Satouri, «Recognition of handwritten Amazigh characters based on zoning methods and MLP,» *Wseas Transactions on Computers,* vol. 14, pp. 178-185, 2015.

[70] M. Fakir, O. Bencharef, B. Bouikhalen et B. Minaoui, «Tifinagh Character Recognition Using Riemannian Metric, SVM & Neural Networks,» *International Journal of Advances in Science and Technology,* vol. 2, n° 6, pp. 1-9, 2011.

[71] O. Bencharef, M. Fakir, B. Minaoui et B. Bouikhalene, «Tifinagh Character Recognition Using Geodesic Distances, Decision Trees & Neural Networks,» *International Journal of Advanced Computer Science and Applications, Special Issue on Artificial Intelligence,* vol. Special Issue on Artificial Intelligence, pp. 51-55, 2011.

[72] M. Boutaounte, A. Elbalaoui, A. Merbouha et Y. Ouadid, «Tifinagh Character Recognition by Graphs Model Representation,» *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 4, n° 3, pp. 79-83, Mars 2014.

[73] S. Gounane, M. Fakir et B. Bouikhalen, «Recognition of Tifinagh Characters Using Self Organizing Map And Fuzzy K-Nearest Neighbor,» *Global Journal of Computer Science and Technology,* vol. 11, n° 15, pp. 29-33, 2011.

[74] Y. Es Saady, A. Rachidi, M. El Yassa et D. Mammass, «Prented Amazighe Character Recognition by Syntactic Approch using Finite Automata,» *ICGST-GVIP Journal,* vol. 10, n° 2, 2010.

[75] Y. Es Saady, A. Rachidi, M. El Yassa et D. Mammass, «AMHCD: A Database for Amazighe Handwritten Character Recognition Research,» *International Journal of Computer Applications,* vol. 27, n° 04, 2011.

[76] Y. Es Saady, A. Rachidi, M. El Yassa et D. Mammass, «Amazighe Handwritten Character Recognition based on Horizontal and Vertical Centerline of Character,» *International Journal of advanced Science and technology,* vol. 33, pp. 33-50, 2011.

[77] S. Gounane, M. Fakir et B. Bouikhalene, «Handwritten Tifinagh Text Recognition Using Fuzzy K-NN and Bi-gram Language Model,» *International Journal of Advanced Computer Science and Applications (IJACSA),* vol. Special Issue on Selected Papers

from: Third international symposium on Automatic Amazigh processing (SITACAM' 13), pp. 29-32, 2013.

[78] O. D. Trier et A. K. Jain, «Evaluation of Binarisation Methods for Document Images,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 17, pp. 1191-1201, 1995.

[79] W. K. Pratt, Digital Image Processing 2nd eddition, New York: Wiley, 1991.

[80] N. Otsu, «Atresholding Selection Methode From Gray-level Histogram,» *IEEE transactions on Systems, Man and Cybernetics,* vol. 9, pp. 62-66, 1979.

[81] I. Guyon, «A Scaling Law for the Validatio-set Trainin-set Size ratio,» *AT&T Bell Laboratories,* pp. 1-11, 1997.

[82] A. Bagdanov et J. Kanai, «Projection Profile Based Skew Estimation Algorithm for JBIG Compressed Images,» *The 4th ICDAR,* pp. PP. 401-405, 1997.

[83] T. Pavlidis et J. Zhou, «Page Segmentation and Classification,» *Computer Vision Graphics Images Process,* vol. 54, pp. 484-496, 1992.

[84] S. J. Perantonis, B. Gatos et N. Papamarkos, «Block Decomposition and Segmentation for Fast Hough Transform Evaluation,» *Pattern Recognition,* vol. 32, n° 5, pp. 811-824, 1999.

[85] S. Bergler, S. Khoury, B. C. Y. Suen et B. Waked, «Skew Detection, Page Segmentation and Script Classificatio of Printed Document Images,» *IEEE International Conference on Systems, Man, and Cybernetics,* pp. 4470-4475, 1998.

[86] A. Nicolaou et B. Gato, «Handwritten text Line Segmentation by Shedding Text into its Lines,» *Proceeding ICDAR,* pp. 626-630, 2009.

[87] G. Louloudis, B. Gatos, I. Pratikakis et C. Halatis, «Text Line and Word Segmentation of Handwritten Documents,» *Pattern recognition,* vol. 42, n° 12, pp. 3169-3183, 2009.

[88] S. L. Likforman, A. Zahour et B. Taconet, «Text Line Segmentation of Historical Documents: A Survey,» *IJDAR,* vol. 9, n° 2, pp. 123-138, 2007.

[89] J. K. Lakshmi et M. Punithavalli, «A Survey on skeletons in digital image processing,» chez *Digital Image Processing, 2009 International Conference,* Bangkok, 2009.

[90] L. Lam, S.-W. Lee et C. Y. Suen, «Thinning methodologies-a comprehensive survey,» *Pattern Analysis and Machine Intelligence, IEEE Transactions,* vol. 14, n° 9, pp. 869 - 885, 1992.

[91] N. P. Khanyile, J. R. Tapamo et E. Dube, «A Comparative Study of Fingerprint Thinning Algorithm,» chez *10th Annual Information Security for South Africa Conference* , Rosebank, South Africa, 2011.

[92] T. Y. Zhang et C. Y. Suen, «A Fast Parallel Algorithme Fot Thinning Digital Patterns,» *Image Processing and Computer Vision,* vol. 27, n° 3, pp. 236-239, March 1984.

[93] R. Crane, A Simplified Approch to Image Processing: Classical and Modern techniques in C, New Jersey: Prentice Hall, 1997.

[94] P. Purkait, «Optical Handwritten Character/Numeral Recognition,» chez *9th North-East workshop on Computational Information Processing*, ISI Kolkata, 2010.

[95] N. D. Tucker et F. C. Evans, «A Tow-step Strategy for Character Recognition Using Geometrical Moments,» *Proceedings of 2nd International conference on Pattern Recognition,* pp. 223-225, 1974.

[96] A. B. S. Hussain, G. T. Tousaint et R. W. Dnaldson, «Results Obtained using a Simple Character Recognition Procedure on Munson's Handprinted Data,» *IEEE transaction on Computing,* vol. 21, pp. 201-205, 1972.

[97] O. D. Trier, A. K. Jain et T. Taxt, «Feature Extraction Methods for Character Recognition- A Survey,» *Pattern Recognition,* vol. 29, n° 4, p. 641–662, 1996.

[98] J. Cai et Z. Q. Liu, «Integration of Structural and Statistical Information for Unconstrained Handwritten Numeral Recognition,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 21, pp. 263-270, 1999.

[99] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai et L. Lam, «Computer Recognition of Unconstrained Handwritten Numerals,» *Proceedings of the IEEE,* vol. 80, pp. 1162-1180, 1992.

[100] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet et L. Lam, «Building a New Generation of Handwriting Recognition Systems,» *Pattern Recognition Letters,* vol. 14, pp. 305-315, 1993.

[101] S. B. Cho, «Neural Network classifiers For recognizing Totally Unconstrained Handwritten Numerals,» *IEEE Transaction on Neural Networks,* vol. 8, pp. 43-53, 1997.

[102] S. W. Lee, «Off-line Recognition of Totally Unconstrained Handwritten Numerals using Multilayer Cluster Neural Network,» *IEEE transaction on Pattern Analysis and Machine Intelligence,* vol. 18, pp. 648-652, 1996.

[103] B. Duerr, W. Haettich, H. Tropf et G. Winkler, «A Combination of Statistical and Syntactical Pattern Recognition Applied to Classification of Un constrained Handwritten Numerals,» *Pattern Recognition,* vol. 12, pp. 189-199, 1980.

[104] A. Pervez et C. Y. Suen, «Computer Recognition of Totally Unconstrained Handwritten Zip Code,» *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 1, pp. 421-431, 1987.

[105] P. Gader, B. Forester, M. Ganzberger, A. Gillies, B. Mitchell, M. Whalen et T. Yocum, «Recognition of Handwritten Digits using Template and Model Matching,» *Pattern Recognition,* vol. 24, pp. 421-431, 1991.

[106] B. T. Mitchell et A. M. Gillies, «A Model based Computer Vision System for recognizing Handwritten Zip Codes,» *Machine Vision and Applications,* vol. 2, pp. 231-243, 1989.

[107] Y. Lee, «Hnadwritten Digit Recognition using a K Nearest Neighbor, Radial Basis Function and Backpropagation Neural networks,» *Neural Computation,* vol. 3, pp. 440-449, 1991.

[108] S. W. Jeong, S. H. Kim et W. H. Cho, «Performence Comparison of Statistical and Neural Network Classifires in handwritten Digit Recognition,» *Advances in Handwriting Recognition,* pp. 406-415, 1998.

[109] D. S. Lee et S. N. Srihari, «Handprinted Digit Recognition: A Comparison of Algorithms,» *Proceeding of the Third International Workshop on frontiers in Handwriting Recognition, buffalo, New York, May 25-27,* pp. 153-162, 1993.

[110] J. Sagau, «Logique floue en classification,» *Techniques de l'ingénieur,* p. H3638, 1996.

[111] M. Hammadlu, «A Fuzzy Model Based Recognition of Handwritten Hindi Numerals using Bacterial Foraging,» chez *6th IEEE/ACIS ICIS International Conference on Computer and Information Science*, Melbourne, Qld., 2007.

[112] F. J. Damenrau, «A technique for computer detection and correction of spelling errors,» *Communication of the association for Computing Machinery*, vol. 7, n° 3, pp. 171-176, 1964.

[113] k. Kukich, «techniques for automatically correcting words in text,» *ACM Computing Surveys*, vol. 24, n° 4, pp. 377-439, 1992.

[114] M. D. Kemighan, K. W. Church et W. A. Gale, «A Spelling Correction Program Based on a Noisy Channel Model,» chez *COLING '90 Proceedings of the 13th conference on Computational linguistics - Volume 2* , Stroudsburg, PA, USA , 1990.

[115] D. Jurafsky et J. H. Martin, Speetch an language processing, New Jersey: Prentice Hall, 2000.

[116] S. F. Chen et J. Goodman, «An empirical study of smoothing techniques for language modeling,» *Computer Speech and Language*, vol. 13, pp. 359-394, 1999.