RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE

FACULTÉ D'ÉLECTRONIQUE ET D'INFORMATIQUE
DÉPARTEMENT INFORMATIQUE

Projet de fin d'étude pour l'obtention du diplôme Licence

Domaine Mathématiques et Informatique

Spécialité

Ingénierie des Systèmes d'Information et des Logiciels

Thème

Combinaison de classifieurs pour une meilleure reconnaissance du manuscrit arabe

---

Proposé par :
Mme. Z. TAMEN

Présenté par :
— **Mohammed BOUKHALFA**
— **Houssem Eddine ALIOUCHE**

Soutenu devant le jury composé de :
— **Mme. N. BENSAOU Présidente**
— **Mme. C. IGHILAZA Membre**

Binôme Nº 119/2016

DÉDICACES

Tout comme les mathématiciens qui se basent parfois sur des postulats pour arriver à résoudre certains problèmes mathématiques complexes, nous aussi avons besoin d'expérience et de sagesse hérités de nos grands-pères pour déchiffrer le mystère de la vie.

Je dédie donc ce modeste travail en premier lieu à mes très chers grands parents BOUKHALFA Abd Elhakm et HADJOUDJ Laide que j'admire énormément, je leur souhaite une longue vie pleine de bonheur.

Je le dédie aussi à toute ma famille, plus particulièrement à celle qui grâce à ses prières, le bon Dieu m'a épargné toute difficulté à affronter tout au long de mon chemin dans la vie, à celle qui m'a toujours noyé par son amour et sa tendresse, à ma flamme d'espoir, ma très chère mère, que Dieu me la garde. sans oublier mon adorable jeune sœur Khadidja qui a toujours été présente dans ma vie, m'a soutenu et encouragé.

Je tiens aussi à dédier ce travail à Mr. Lakhdar RADI et toute sa famille que j'apprécie tant.

Pour finir, je dédie ce travail à mes meilleurs amis « les frères BENDEBICHE » si ce n'est pour dire mes très chers frères que je respecte énormément et en qui j'ai pu trouver l'entente et l'amitié dont j'avais besoin.

"Mohammed BOUKHALFA"

REMERCIEMENTS

La nature nous a appris que rien n'est isolé mais plutôt tout est en relation et influence réciproque, rien ne peut se construire seul.

Ce qui est sûr et évident est que ce travail est le fruit d'efforts de personnes multiples que je ne sais comment remercier car les mots ne sont pas toujours suffisants pour refléter ce que l'esprit comporte de gratitude et de reconnaissance.

De ce fait, j'adresse mes sincères remerciements à tous ceux qui ont contribué de près ou de loin à l'élaboration de ce mémoire.

En premier lieu, je remercie ma promotrice Mme Z. TAMEN qui m'a accompagné tout au long de ce semestre, je la remercie pour le temps qu'elle m'a consacré, les conseils qu'elle m'a donnés et ses précieuses directives et la qualité de son soutien. Je tiens aussi mes chaleureux remerciements à Mlle. BOUACHI Farida à qui je suis profondément reconnaissant pour son soutien, sa grande disponibilité et ses précieux conseils.

Ces remerciements ne seront pas complets sans mentionner mon binôme Mr. ALIOUCHE Houssem Eddine qui était très sérieux dans sa recherche et superbe par ses idées, je suis vraiment fier de lui et j'ai eu un grand plaisir de travailler avec lui.

"Mohammed BOUKHALFA"

En préambule à ce mémoire de fin d'étude, J'adresse mes remerciements les plus sincères aux personnes qui m'ont aidé à la réalisation de ce projet.

En premier lieu, mes vifs remerciements à ma promotrice Mme. Z. TAMEN de m'avoir accompagné pendant toute la durée de réalisation de ce travail, sa disponibilité et les conseils qu'il m'a prodigué tout au long de ce semestre.

Mes remerciements vont aussi à tous mes professeurs et enseignants qui m'ont accompagné durant mes trois années universitaires et m'ont fait partagé toutes leurs compétences.

Mes pensées à mes très chers parents pour leur soutien, leur confiance et leur support inestimable.

Sans oublier de témoigner ma reconnaissance à mon très cher binôme Mr. BOUKHALFA Mohammed avec qui j'ai trouvé l'entente et l'amitié dont j'avais besoin.

"Houssem Eddine ALLIOUCHE"

ملخص

يعتبر الخط اليدوي من أبسط وسائل الاتصال و الأكثر إستخداما. وعلى الرغم من تطور التكنولوجيا، إلا أنه مازال واحد من بين الوسائل الأساسية للتواصل. مما دفع بنا إلى تطوير نظام قادر على التعرف على الكتابة اليدوية العربية و تحويلها إلى نص رقمي.

هذا النظام يعتمد على الجمع بين مصنفات الشبكات العصبية الإصطناعية NN و أليات الدعم الشعاعي SVM

تم إعتماد أربعة آليات لدمج المصنفات تتمثل في التصويت بالأغلبية ، بوردا ،فضاء معرفة السلوك BKS و دامبستير شافير DS و هذا من أجل تحسين دقة التعرف على الكتابة اليدوية .

تم تطوير هذا النظام في بيئة المتلاب يتم التعامل معه من خلال واجه تطبيق ويب

— الكلمات المفتاحية :

الخط اليدوي، التعرف على الكتابه اليدوية، الجمع بين المصنفات، الشبكات العصبية الإصطناعية، أليات الدعم الشعاعي، اليات دمج المصنفات، التصويت بالأغلبية ، بوردا، فضاء معرفة السلوك، دامبستير شافير.

RÉSUMÉ

L'écriture manuscrite demeure aujourd'hui l'un des moyens de communication les plus simples et les plus expressifs. Malgré l'avènement des nouvelles technologies, elle reste un moyen de communication incontournable, d'où l'intérêt de mettre en place un système de reconnaissance de l'écriture manuscrite. La solution proposée consister à faire la combinaison de classifieurs à savoir les réseaux de neurones (NN) et les machines à vecteurs de support (SVM).

Notre travail consiste en l'élaboration d'une application web permettant la reconnaissance du manuscrit arabe à l'aide de quatre méthodes de combinaison de classifieurs à savoir : vote majoritaire, Borda, Behavior Knowledge Space (BKS) et Dempster Shafer (DS) et cela dans le but d'améliorer la reconnaissance de l'écriture manuscrite.

— **Mots-clés :**

écriture manuscrite, reconnaissance de l'écriture manuscrite, combinaison de classifieurs, réseaux de neurones, machines à vecteurs de support, méthodes de combinaison de classifieurs, vote majoritaire, Borda, Behavior Knowledge Space, Dempster Shafer.

ABSTRACT

The handwriting is still one of the simplest and most expressive means of communication. Despite the advent of new technologies, it remains an essential mean of communication, this is what created the need to set up a handwriting recognition system. The solution proposed consists in combining classifiers which are neural networks (NN) and support vector machines (SVM).

Our work consists on developing a web application for Arabic handwritten recognition and this by using four methods of combining classifiers which are: majority voting, Borda, Behavior Knowledge Space (BKS) and Dampster Shaffer (DS) and this in order to improve the handwriting recognition.

— **Keywords :**

handwriting, handwriting recognition, combining classifiers, neural networks, support vector machines, methods of combining classifier, majority voting, Borda, Behavior Knowledge Space, Dempster Shafer.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES SIGLES ET ABRÉVIATIONS

BKS	Behavior Knowledge Space
CSS	Cascading Style Sheets
DP	Decision Profile
DS	Dempster Shafer
DT	Decision Template
FT	Features
FTP	File Transfert Protocol
FU	Focal Unity
HTML5	Hypertext Markup Language 5
HTTP	HyperText Transfer Protocol
JS	JavaScript
LIBSVM	Library for Support Vector Machines
LM	Legendre Moments
NN	Neural Networks
PHP	Hypertext Preprocessor
RBF	Radial Basis Function
SVC	Support Vector Classification
SVM	Support Vector Machine
SVR	Support Vector Regression
XML	Extensible Markup Language

INTRODUCTION GÉNÉRALE

L'écriture manuscrite demeure jusqu'à aujourd'hui l'un des moyens de communication les plus simples et les plus expressifs, permettant d'exprimer l'identité et la culture d'un individu.

Bien qu'elle soit confrontée aux nouvelles méthodes d'écriture à savoir l'écriture au clavier, l'écriture manuscrite a su s'imposer et reste un moyen de communication incontournable.

Il serait donc intéressant de l'exploiter en utilisant les moyens informatiques et d'en récolter l'information qu'elle véhicule, on parle alors de reconnaissance du manuscrit.

La reconnaissance de l'écriture manuscrite demeure un sujet de recherche vivace et vaste. Pour le Latin et même le Chinois, elle a connu ces dernières années de grands progrès, et les succès des travaux de recherches ont donné lieu à de nombreuses applications industrielles, notamment dans le domaine de la lecture automatique de formulaires, de chèques ou d'adresses postales. Pour la langue arabe, beaucoup de travail reste à faire dans le domaine.

Le travail envisagé dans ce document consiste en une contribution dans le domaine de la reconnaissance du manuscrit arabe concernant surtout l'étape de classification.

Dans l'étape d'extraction des caractéristiques, nous allons utiliser des primitives structurelles et statistiques locales ainsi qu'une méthode globale basée sur les moments de Legendre. Ces deux types de caractéristiques seront entrainés avec les réseaux de neurones multicouches et les machines à vecteurs de support SVM.

Dans l'étape de classification, nous pouvons nous contenter d'envisager d'optimiser les classifieurs utilisés. Dans notre cas, en plus de l'optimisation des quatre classifieurs, réalisés à partir des deux types de caractéristiques et des deux types de classifieurs, il s'agira d'implémenter des méthodes de combinaison au niveau décisionnel des classifieurs.Quatre méthodes de combinaisons sont à envisager : le vote majoritaire, la méthode Borda, la méthode basée sur l'étude des environnements comportementaux : BKS et la méthode basée sur la théorie des évidences de Dempster-Shafer.

Une étude comparative des classifieurs pris individuellement ainsi que des méthodes de combinaisons sera entreprise.

Notre document s'articule comme suit :

— Une Introduction au domaine de la reconnaissance du manuscrit est décrite dans le premier chapitre.
— Le chapitre 2 est dédié à l'étape d'extraction de caractéristiques.

— Les méthodes de combinaison de classifieurs sont décrites en général dans le troisième chapitre.
— Les classifieurs utilisés sont détaillés dans le quatrième chapitre.

— La partie implémentation, test et analyse des résultats est assurée dans le cinquième chapitre.
— Nous terminerons notre document par une conclusion et la proposition de quelques perspectives au travail entrepris.

CHAPITRE 1 INTRODUCTION À LA RCONNAISSANCE DU MANUSCRIT

1.1 Introduction

Contrairement au Latin, la reconnaissance de l'écriture arabe manuscrite reste encore aujourd'hui au niveau de la recherche et d l'expérimentation, l'écriture arabe manuscrite est très riches en termes d'ambiguïtés graphiques, c'est ce qui rend la reconnaissance encore plus complexe.

Dans ce chapitre, nous allons introduire le domaine de la reconnaissance de l'écriture manuscrite et nous expliquerons en détails les caractéristiques de la langue arabe.

1.2 Reconnaissance de l'écriture manuscrite

La reconnaissance d'un manuscrit est la capacité d'un ordinateur à extraire de l'information à partir d'une image. En d'autres termes, c'est la possibilité de convertir une écriture manuscrite vers une écriture numérique. Le domaine de la reconnaissance de l'écriture manuscrite se divise en deux parties :

1.2.1 Reconnaissance hors ligne

Ce type de reconnaissance ne s'effectue pas au moment de l'écriture des mots mais à la fin, contrairement à la reconnaissance en ligne, on ne peut savoir comment ont été tracés les différents motifs, afin d'extraire l'information, on a donc recours à la reconnaissance de formes. Ce type de reconnaissance est privilégié pour la lecture de chèques bancaires ou le tri postal à titre d'exemple.

1.2.2 Reconnaissance en ligne

Elle se fait au même moment de l'écriture, ce type de reconnaissance concerne les applications dont la saisie du texte est sans clavier, c'est le cas notamment des stylos numériques ou des stylets sur agendas électroniques ou sur les tablettes tactiles. Contrairement à la reconnaissance hors ligne, elle est beaucoup plus robuste mais nécessite un matériel beaucoup plus robuste et coûteux.

1.3 Le manuscrit arabe

1.3.1 Présentation de la langue arabe

La langue arabe est une langue sémitique reconnue comme étant langue officielle dans vingt-huit états au sein du monde arabe.

Elle est originaire la péninsule arabique, elle s'est répandu en Afrique du nord ainsi qu'en Europe durant la colonisation de l'Empire Arabe au Moyen Age et celle de l'Islam. La langue arabe

est considérée comme étant la langue liturgique de l'Islam et du Coran. On distingue plusieurs variantes de la langue arabe à savoir : l'arabe littéral, l'arabe moderne, classique . . . etc.

1.3.2 Alphabet arabe

L'alphabet arabe est constitué de vingt-huit lettres fondamentales, contrairement à l'alphabet latin, il n'existe pas de notion de minuscule ou de majuscule, l'écriture est donc monocamérale.[3]

En revanche la plupart des lettres s'attachent entre elles et leur graphie change selon leur position :

— **Position initiale :** liée à la lettre suivante mais pas à la précédente.
— **Position médiane :** liée à la lettre suivante et précédente.
— **Position finale :** liée à la précédente mais pas à la suivante.
— **Position isolée :** la lettre n'a de liaison ni avec la précédente ni la suivante.

Le tableau suivant illustre la graphie de chacune des lettres de l'alphabet arabe selon sa position :

Nom de la lettre		Position Isolée	Position Initiale	Position Médiane	Position Finale	Nom de la lettre		Position Isolée	Position Initiale	Position Médiane	Position Finale
Français	Arabe					Français	Arabe				
Alif	ألف	ا		ـا		Dhad	ضاد	ض	ضـ	ـضـ	ـض
Ba'	باء	ب	بـ	ـبـ	ـب	Tta'	طاء	ط	طـ	ـطـ	ـط
Ta'	تاء	ت	تـ	ـتـ	ـت	Dha'	ظاء	ظ	ظـ	ـظـ	ـظ
Tha'	ثاء	ث	ثـ	ـثـ	ـث	A'in	عين	ع	عـ	ـعـ	ـع
Jeem	جيم	ج	جـ	ـجـ	ـج	Ghain	غين	غ	غـ	ـغـ	ـغ
H'a'	حاء	ح	حـ	ـحـ	ـح	Fa'	فاء	ف	فـ	ـفـ	ـف
Kha'	خاء	خ	خـ	ـخـ	ـخ	Qaf	قاف	ق	قـ	ـقـ	ـق
Dal	دال	ذ		ـذ		Kaf	كاف	ك	كـ	ـكـ	ـك
Thal	ذال	د		ـد		Lam	لام	ل	لـ	ـلـ	ـل
Rai	راي	ر		ـر		Meem	ميم	م	مـ	ـمـ	ـم
Zai	زاي	ز		ـز		Noon	نون	ن	نـ	ـنـ	ـن
Seen	سين	س	سـ	ـسـ	ـس	Ha'	هاء	ه	هـ	ـهـ	ـه
Sheen	شين	ش	شـ	ـشـ	ـش	Waw	واو	و		ـو	
Sad	صاد	ص	صـ	ـصـ	ـص	Ya'	ياء	ي	يـ	ـيـ	ـي

Figure 1.1 Alphabet arabe

1.3.3 Caractéristiques de l'alphabet arabe

- On distingue 22 lettres ayant quatre formes d'écriture, les six restantes ne peuvent être rattachées à leur successeur, elles ont donc deux formes d'écriture différentes. Ces six lettres sont : و - ذ - د - ر - ز - أ .
- L'écriture arabe s'écrit de droite-à-gauche contrairement aux différentes écritures occidentales qui elles s'écrivent de gauche-à-droite.
- Une lettre arabe peut contenir différentes formes : un trait horizontal (cas de la lettre ك), un trait vertical (cas de la lettre ظ) ou un zigzag (cas de la lettre ء).
- Les caractères de l'alphabet arabe n'ont pas de tailles fixes que ça soit en hauteur ou en largeur, les formes des caractères diffèrent aussi au sein d'un même caractère.
- Vingt-cinq caractères de l'alphabet arabe représentent les consonnes, seulement trois caractères représentent les voyelles : ي - و - ا.
- Un bon nombre de signes dit diacritiques contribuent à la vocalisation.
- Certains peuvent se trouver au-dessus du caractère arabe à savoir :
  - **Dhamma :** ُ - L'équivalent du son « ou ».
  - **Fatha :** َ - L'équivalent du son « a ».
  - **Soukoun :** ْ - Absence de de voyelle.
- D'autres peuvent se trouver au-dessous du caractère arabe à savoir :
  - **Kasrah :** ِ - L'équivalent du son « i ».
- Les signes diacritiques cités en dessus ne sont pas systématiquement utilisés dans l'écriture arabe, ils sont juste utilisés afin d'éviter des erreurs de prononciation.
- On distingue alors deux types de textes :
  - **Avec signes de voyelles :** Exemples : Le saint Coran et divers livres d'apprentissage pour enfants.
  - **Sans signes de voyelles :** Exemples : Livres, journaux, articles, publications, ... etc.
- D'autres signes diacritiques existent à savoir :
  - **Hamza :** son orthographe est assez spéciale, elle dépend de certaines règles grammaticales.
  - **Chedda :** représente l'accentuation d'une lettre, l'équivalent d'une double consonne, la Chedda doit toujours être accompagnée soit d'une Dhamma, d'une Kasrah ou d'une Fatha.
  - **Madda :** l'équivalent de deux Alifs combinés.
- Les signes diacritiques ne peuvent se trouver au-dessus et au-dessous d'un même caractère arabe en même temps.
- Un autre digne diacritique peut se former à partir d'un double Dhamma, d'un double Fatha ou d'un double Kasrah, il s'agit du Tanwin.
- Le signe diacritique Hamza peut se comporter comme :
  - Seul soit au milieu ou à la fin d'un mot.
  - Sur le Alif, Waw ou Ya' : ئ - ؤ - إ - أ

  - Avec la caractère ك pour le différentier du ل.
- Quinze caractères arabes possèdent au moins un point soit en-dessus soit en-dessous mais pas les deux en même temps.
- Trois points au maximum peuvent se trouver au-dessus d'un caractère arabe et deux au-dessous.
  - **Points au-dessus d'un caractère**
    - **Un point :** ن ـ ف ـ غ ـ ظ ـ ض ـ ز ـ خ.
    - **Deux points** ق ـ ت.
    - **Trois points** ش ـ ث.
  - **Points au-dessous d'un caractère**
    - **Un point** ج ـ ب.
    - **Deux points** ي.
- On peut regrouper les caractères arabes qui possèdent le même corps, ils diffèrent soit en termes de nombre de points ou d'emplacement. Voici quelques exemples :
  - { ذ ـ د} = د
  - { خ - ج - ح } = ح
  - { ز - ر } = ر [3]

1.4 Définitions

L'écriture manuscrite que nous cherchons à reconnaître se trouve dans ce qu'on appelle par image numérique. Il serait donc intéressant de la définir du moment qu'elle représente le support sur lequel se trouve l'information que nous voulons extraire.

1.4.1 Image numérique

Une image numérique est une fonction à support discret et borné et à valeurs discrètes, le support est multidimensionnel, généralement en deux dimensions 2D.

Une image numérique à deux dimensions est représentée par une matrice *(M x N)* de M lignes et N colonnes, l'intersection d'une ligne et d'une colonne est appelé pixel.

1.4.2 Pixel

Un pixel, est définit comme le plus petit élément d'une image numérique, il est désigné par le couple (i,j) tel que i est l'indice de colonne $i \in \{0, N-1\}$ et j est l'indice de ligne $j \in \{0, M-1\}$.

M et N sont respectivement les largeur et longueur d'une image numérique quelconque. Par convention, le pixel origine *(0,0)* est en général en haut à gauche.

Le pixel est considéré comme l'unité de base qui permet la définition d'une image numérique matricielle, sa valeur correspond au type de codage de l'image. En général, il est associé à une

couleur décomposée en trois composantes primaires, à savoir : le rouge, le vert et le bleu, chacune de ces composantes varie dans l'intervalle *[0, 255]*.

Du moment que nous utilisons des images numériques binaires, le pixel aura deux valeurs possibles : la valeur 0 qui est associée au noir et la valeur 1 qui est associée au blanc.

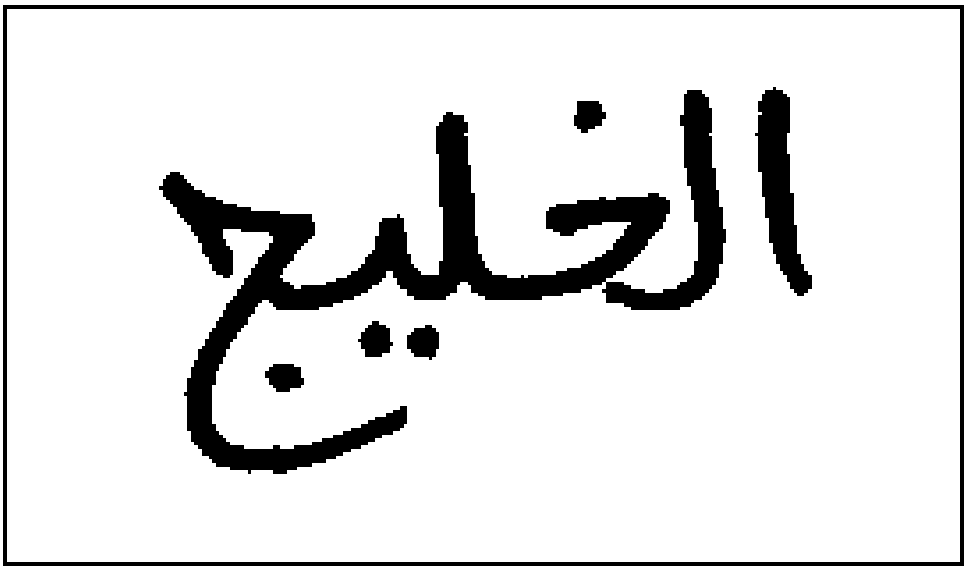


Figure 1.2 Exemple d'une image numérique binaire brute

Remarque : Cette image est un exemple extrait de notre base de données, elle porte une information écrite en langue arabe.

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1	0	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1
1	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	1	0	1	0	1
1	1	1	1	0	1	1	1	1	0	1	0	1	1	1	0	1	0	1	0	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 1.3 Exemple d'une représentation matricielle d'une image numérique binaire

1.4.3 Résolution d'une image

Elle est définie par un nombre de pixels par unité de longueur, plus ce nombre est grand,

plus la quantité d'information qui décrit cette structure est importante. La résolution d'une image numérique définie le degré de détail de l'image, plus elle est élevée, meilleure est la restitution.

Cependant, augmenter la résolution d'une image conduit à une taille très importante du fichier qui contiendra l'image et un temps de visualisation et d'impression plus long.

1.5 Traitement d'image

Le traitement d'image est un domaine de l'informatique et des mathématiques appliquées qui étudie les images dans le but d'améliorer leur qualité ou d'en extraire de l'information.

Le traitement d'image est un domaine qui dérive du traitement du signal dédié aux images ou autres données dérivées de l'image comme la vidéo par exemple.

Le traitement d'une image se fait en deux phases essentielles :

1.5.1 Traitement de bas niveau

Ces traitements opèrent sur des données de nature numérique et doivent donc simplifier l'image sans pour autant la dégrader. Le traitement de bas niveau se construit autour de méthodes d'analyse de l'image afin d'extraire des caractéristiques de l'image analysée sans les interpréter (contour, texture, segmentation, fenêtrage ...).

1.5.2 Traitement de haut niveau

Intègre l'ensemble des méthodes permettant d'interpréter les caractéristiques issues du traitement bas niveau (prise de décision, classification, intelligence artificielle ...).

1.6 Conclusion

La reconnaissance de l'écriture arabe est rattachée au vaste domaine de la reconnaissance de formes. Son but est de prendre une décision quant au contenu sémantique du message transmis à partir de sa représentation physique. Nous avons à travers ce chapitre introduit ce domaine en citant les caractéristiques de la langue arabe, nous avons également défini l'image numérique binaire, cette dernière véhicule le texte arabe manuscrit que nous devrons traiter dans les prochains chapitres pour en extraire le texte sous format numérique.

CHAPITRE 2 L'EXTRACTION DE CARACTÉRISTIQUES

2.1 Introduction

La reconnaissance du manuscrit arabe, consiste à déchiffrer des mots écrits en langue arabe dans une image.

L'image en elle-même ne peut pas être traitée directement pour être classifiée à l'aide d'un classifieur comme un réseau de neurones ou une machine à vecteur de support. Nous utilisons en entrée une sorte de résumé de toutes les caractéristiques de l'image qu'on appelle descripteur d'image.

L'objectif de ce chapitre est donc d'expliquer les méthodes que nous avons adoptées pour extraire ces descripteurs à partir de nos images.

2.2 Extraction de l'information par la méthode LM

Dans un espace vectoriel, tout élément peut être décrit comme une combinaison linéaire des vecteurs constituant la base de l'espace. En traitement de signal, toute fonction peut être décrite comme la combinaison linéaire de fonctions harmoniques comme le sinus et le cosinus. Dans ce cas, les coefficients trouvés sont les coefficients de la transformée de Fourier. Dans le cas où la base est constituée de polynômes, les coefficients trouvés sont appelés moments. Si la base est orthogonale, nous avons des informations contenues dans les coefficients sont discriminantes. Les moments de Legendre, sont les coefficients obtenus sur la base polynomiale utilisant les polynômes de Legendre comme fonctions noyaux. Ils furent introduits par Teague.[1][4]

Les polynômes de Legendre ont la propriété d'être des polynômes orthogonaux, cela dit, il n'y a pas de redondance ou de chevauchement d'information entre des moments d'ordres différents. Grace à cela, chaque moment est unique et porteur d'information indépendante dans une image.[1][4]

Le polynôme de Legendre d'ordre p, $P_p(x)$ est défini par :

$$P_p(x) = \sum_{k=0}^{p} \{ \frac{(-1)^{\frac{p-k}{2}} (p+k)! x^k}{2^p k! (\frac{p-k}{2})! (\frac{p+k}{2})} \} \tag{2.1}$$

Les moments de Legendre sont calculés suivant la formule suivante :

$$L_{pq} = \lambda_{pq} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} P_p(x_i) P_q(y_j) I(x,y) \tag{2.2}$$

Tel que :

— M **et** N **:** représentent la taille d'une image discrète.

— $I(x, y)$ **:** représente l'intensité d'une image discrète.
— λ_{pq} **:** est égal à : $\frac{(2p+1)(2q+1)}{(M*N)}$
— x_i **et** y_j **:** représentent les coordonnées normalisées d'un pixel dans l'intervalle [-1,1] avec $x_i = \frac{2x-(M-1)}{(M-1)}$ et $y_j = \frac{2y-(N-1)}{(N-1)}$

Concernant notre application, les moments de Legendre ont été calculés pour toutes les images jusqu'à l'ordre 10, donnant ainsi des vecteurs caractéristiques de 100 éléments. Cet ordre 10 a été déduit d'expérimentations ultérieures faites par des binômes sous la direction Mme Tamen, le montrant comme un ordre optimal réalisant le meilleur compromis entre taux de reconnaissance et temps de calcul.Concernant notre application, les moments de Legendre ont été calculés pour toutes les images jusqu'à l'ordre 10, donnant ainsi des vecteurs caractéristiques de 100 éléments. Cet ordre 10 a été déduit d'expérimentations ultérieures faites par des binômes sous la direction Mme Tamen, le montrant comme un ordre optimal réalisant le meilleur compromis entre taux de reconnaissance et temps de calcul.

2.3 Extraction des caractéristiques par la méthode FT

L'objectif de cette méthode est d'élaborer un vecteur contenant des informations qui sont en fait des caractéristiques de l'image à traiter. Ces caractéristiques sont obtenues en suivant plusieurs étapes que nous allons décrire dans ce qui suit.[1]

La méthode FT repose sur deux phases essentielles :

2.3.1 Phase de prétraitements

Elle a pour objectif d'obtenir une version d'image nettoyée qui elle va être utilisée dans les prochaines étapes. Cette étape consiste à éliminer toutes les parties de l'image inutiles à traiter, cela aide à réduire le temps de calcul et à minimiser l'espace mémoire utilisé.[1]

Durant cette phase, l'image subit plusieurs traitements que nous présentons dans ce qui suit :

2.3.1.1 Élaboration de la boite englobante de l'image

Cette étape consiste à minimiser la dimension de l'image en ne gardant que la partie de l'image où est représentée l'écriture. Le but est d'imiter l'être humain dans son processus de lecture vu qu'il ne concentre sa vision que sur la partie écrite de l'image.

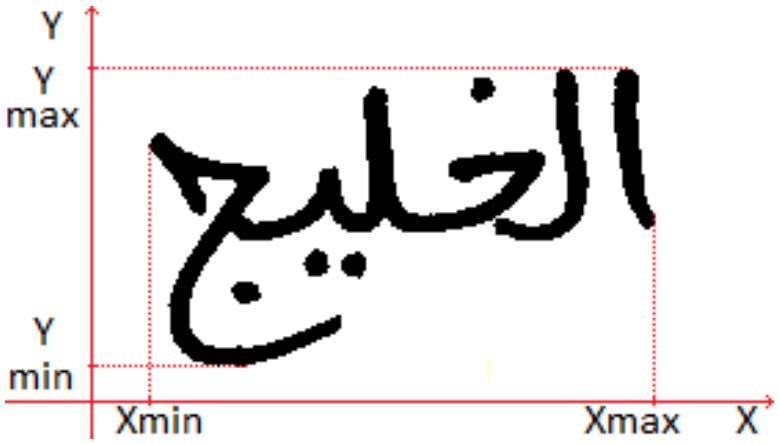


Figure 2.1 Méthode de réalisation de la boite englobante

Figure 2.2 Exemple d'une image réduite à sa boite englobante

2.3.1.2 Le fenêtrage de l'image

Cette étape consiste à découper l'image en fenêtres avec une largeur précise. Le découpage se fait soit avec chevauchement ou sans chevauchement.

Dans notre cas, nous choisissons de découper l'image en 4 fenêtres de largeurs égales avec un chevauchement égal à 3 pixels.

Figure 2.3 Exemple d'une image après découpage en 4 fenêtres avec un chevauchement égal à 3 pixels

Figure 2.4 Exemple d'une image après découpage en 4 fenêtres avec un chevauchement égal à 10 pixels

Figure 2.5 Exemple d'une image après découpage en 4 fenêtres sans chevauchement

Remarques : En découpant l'image en 4 fenêtres avec un chevauchement égal à trois pixels, la valeur de la largeur de l'image avant le fenêtrage doit être de préférence un multiple de 4, sinon il y a risque de perte d'information. Pour remédier à ce problème, on augmente la longueur ainsi

que la largeur de l'image jusqu'à obtenir une valeur multiple de 4, en prenant en considération le chevauchement.

La largeur de la fenêtre devient égale à :

$$\frac{LargeurDeL'image + Chevauchement * (NombreDeFenêtres - 1)}{NombreDeFenêtres} \tag{2.3}$$

Après découpage de l'image en fenêtres, toutes les étapes qui vont suivre se feront sur chaque fenêtre séparément.

Chaque fenêtre subit une nouvelle fois le traitement de la boite englobante.

2.3.1.3 Tracé du contour de l'image

Cette étape a pour but de réduire l'information à traiter car dans le cas de l'écriture, le fait de tracer le contour a beaucoup plus d'avantages que de laisser l'image à l'état brut car elle permettra de traiter l'image de manière plus efficace. Cette étape est très importante puisqu'elle servira à l'extraction des caractéristiques dans la phase qui suivra.

Le tracé du contour de l'image consiste à parcourir l'image pixel par pixel, pour chaque ligne et chaque colonne de l'image. S'il existe une transition du noir vers le blanc ou du blanc vers le noir, la valeur du pixel noir est mise à 2. A la fin du parcourt de l'image, les pixels dont la valeur est égale à 1 reçoivent la valeur 0, ils constitueront le fond de la forme, quant aux pixels dont la valeur est égale à 2, ils vont recevoir la valeur 1, qui eux vont constituer le contour de l'image.

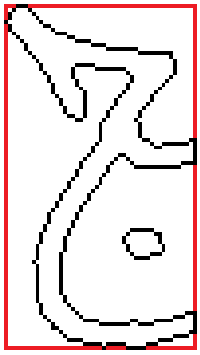


Figure 2.6 Exemple d'une fenêtre après détection de contour

2.3.2 Phase d'extraction[1]

2.3.2.1 Détermination du centre de gravité de l'image

Cette étape consiste à déterminer le centre de gravité qui permet de découper l'image dans tous les sens en des parties ayant des énergies égales. Ce point représente le centre de l'information écrite.

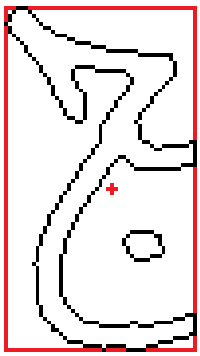


Figure 2.7 Exemple d'un centre de gravité d'une fenêtre

2.3.2.2 Utilisation de quatre directions de Freeman

Après avoir fait le traitement concernant le contour de l'image ainsi que la détermination du centre de gravité, quatre directions seront tracées à partir du centre de gravité, ces quatre directions sont appelées directions de Freeman, elles sont au total huit directions comme le montre la figure qui suit :

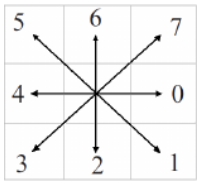


Figure 2.8 Les huit directions de Freeman

Dans notre cas nous utiliserons seulement quatre d'entre elles à savoir : les directions 1, 3, 5 et 7 d'après la figure ci-dessus. Ces quatre directions constituent en tout deux diagonales, dans notre cas chaque diagonale aura une épaisseur de trois pixels. Elles seront tracées dans notre image à partir du centre de gravité déterminé dans l'étape précédente comme le montre la figure suivante :

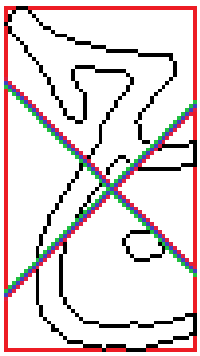


Figure 2.9 Exemple illustrant les deux diagonales d'épaisseur de trois pixels à partir du centre de gravité de l'image

Le but de cette étape est de tirer certaines informations à partir de l'image. Ces informations serviront par la suite à remplir notre vecteur caractéristique.

Les informations tirées à partir de cette étape sont : le nombre de points d'intersection du contour de l'image avec les deux diagonales, les distances des points du contours les plus éloignés par rapport au centre de gravité, la distance utilisée est la distance euclidienne.

Grace à cela, on arrive donc à construire la première partie du vecteur de caractéristiques qui est constituée de huit éléments du vecteur de caractéristiques comme le montre le schéma suivant :

Nb1	D1	Nb3	D3	Nb5	D5	Nb7	D7

Figure 2.10 Construction de la première partie du vecteur caractéristique – huit éléments

Tel que :

— **NBi :** Représente le nombre de points d'intersection du contour avec la direction de chaque diagonale.

— **Di :** Représente la distance normalisée du contour par rapport au centre de gravité.

2.3.2.3 Calcul de la densité de l'écriture

La densité de l'écriture est calculée en divisant la surface de l'écriture sur la surface de l'image. Dans notre cas et en suivant notre méthodologie, on a décidé de découper chaque fenêtre en six blocs, on aura donc à calculer la densité de l'écriture pour chaque bloc ainsi que pour la fenêtre complète.

Ce qui nous amène à ajouter au vecteur de caractéristiques sept nouveaux éléments comme le montre le schéma suivant :

Dr1	Dr2	Dr3	Dr4	Dr5	Dr6	DR

Figure 2.11 Construction de la deuxième partie du vecteur caractéristique – sept éléments

Tel que :

— **Dri :** Représente la densité relative du bloc i.
— **DR :** Représente la densité relative de la fenêtre entière

2.3.2.4 Calcul des transitions Noir/Blanc et Blanc/Noir

Le calcul des transitions Noir/Blanc e Blanc/Noir consiste à parcourir la fenêtre ou bien le bloc horizontalement puis verticalement tout en calculant le nombre de transitions Noir/Blanc et Blanc/Noir. Le résultat obtenu est divisé par la largeur dans le cas du parcours de la fenêtre de manière horizontale, ou alors divisé par la longueur de l'image dans le cas du parcours de l'image de manière verticale.

Du moment que nous avons découpé notre fenêtre en six blocs, pour chaque bloc, on aura deux valeurs de transitions une horizontale et une autre verticale, ce qui veut dire qu'on aura douze transitions dont les valeurs serviront à compléter le vecteur caractéristique, ajoutant deux autres valeurs de transitions concernant l'image entière comme le montre le schéma suivant :

TrH1	TrV1	TrH2	TrV2	TrH3	TrV3	TrH4	TrV4	TrH5	TrV5	TrH6	TrV6	TRH	TRV

Figure 2.12 Construction de la troisième partie du vecteur de caractéristiques – quatorze éléments

Tel que :

— **TrHi :** Représente la transition horizontale du bloc i.
— **TrVi :** Représente la transition verticale du bloc i.
— **TRH :** Représente la transition horizontale de la fenêtre entière.
— **TRV :** Représente la transition verticale de la fenêtre entière.

Grace au descripteur FT, nous avons pu extraire vingt-neuf éléments (29) par fenêtre, ce qui implique que le descripteur FT de chaque image sera constitué de cent seize (116) éléments.

2.4 Conclusion

Nous avons expliqué à travers ce chapitre deux différentes manières d'extraire les caractéristiques d'une image binaire à savoir la méthode LM et FT, grâce à cela, nous sommes en possession des informations nécessaires au traitement de l'image, ces dernières seront les entrées des classifieurs que nous aborderons dans le chapitre suivant.

CHAPITRE 3 LA COMBINAISON DE CLASSIFIEURS

3.1 Introduction

Nous allons à travers les pages de ce chapitre présenter la combinaison de classificateurs comme la solution utilisée dans le but d'augmenter la précision de la classification. Et contrairement à d'autres approches de la combinaison, celle que nous détaillons dans ce chapitre opère au niveau décisionnel de chaque classificateur où on utilise les sorties des classificateurs pour la combinaison. Cette approche est populaire car elle ne nécessite pas la connaissance de la structure interne des classificateurs et leurs caractéristiques.

3.2 Classification

3.2.1 Définition

Un classificateur est un système de reconnaissance dont la sortie est un vecteur dont les valeurs sont généralement comprises entre 0 et 1, et de dimension égale au nombre de classes. La classe, représente l'étiquète d'une image et elle est repérée par l'indice de la valeur maximale de ce vecteur. Il existe un moyen de catégoriser les combinaisons de classifieurs qui consiste à les grouper par rapport aux sorties des classifieurs utilisés dans la combinaison.

3.2.2 Types de sorties des classificateurs combinés

Les trois types de sorties des classifieurs les plus utilisés sont : [5]

3.2.2.1 Type I (niveau abstrait)

Il représente le niveau le plus bas du moment que le classifieur fournit le moins d'informations dans ce niveau. La sortie du classifieur est tout simplement une étiquette de classe unique ou un ensemble non ordonné de classes candidates.

A ce niveau, le classifieur généralement représenté par un vecteur où il y a uniquement deux valeurs distinctes dont l'une pour les classes acceptées et l'autre pour les classes rejetées.

On appelle classifieur Crisp le classifieur j qui accepte seulement une classe donc il peut prendre les sorties (les scores) S_i^j où :

$$S_i^j \in \{0,1\}, \sum_{i=1}^{C} S_i^j = 1 \tag{3.1}$$

3.2.2.2 Type II (niveau de rang)

La sortie du classifieur dans le niveau de rang est une séquence ordonnée de classes candidates, appelée la liste des N meilleures classes. La classe candidate qui se trouve à la première position est considérée comme la classe plus probable, tandis que la classe positionnée à la fin de la liste est la plus improbable. Notez qu'il n'y a pas de valeurs de confiance, attachées aux étiquettes de classe au niveau du classement. Seules leurs positions dans la liste des N meilleures indiquent leurs probabilités relatives. Le vecteur de sortie contient généralement des nombres naturels de 1 à N.

3.2.2.3 Type III (niveau de mesure)

En plus de la liste des N meilleures classes candidates ordonnées dans le niveau de rang (Type II), la sortie du classifieur dans le niveau de mesure possède des valeurs de confiance qui sont affectées à chaque entrée de la liste des N meilleures classes candidates. Ces scores, peuvent être des nombres réels arbitraires, en fonction de l'architecture du classifieur utilisé.

Généralement les valeurs de confiance appartiennent à l'intervalle [0,1]. Il existe deux cas différenciés par la somme des valeurs des confiances, si elle est égale1 le classifieur est appelé Fuzzy sinon il appelé Possibilistic.

— **Classificateur Flou (Fuzzy) :**

$$S_i^j \in [0,1], \sum_{i=1}^{C} S_i^j = 1 \tag{3.2}$$

— **Classificateur Possibiliste (Possibilistic) :**

$$S_i^j \in [0,1], \sum_{i=1}^{C} S_i^j > O \tag{3.3}$$

Nous pouvons conclure que le niveau de mesure contient le plus d'informations parmi les trois niveaux de sortie cités.

Le résultat donné par un classificateur est représenté par un vecteur avec une dimension égale au nombre de classes. Donc, le problème de combinaison peut être représenté par une matrice de dimension $(C * L)$ appelée Profile de décision Decision Profile (DP)

Où :

— **C :** représente le nombre de classes.
— **L :** représente le nombre de classifieurs.

Le but du classifieur de combinaison est de créer un classifieur qui opère sur les mêmes types de classifieurs de base et aussi sur les mêmes classes.

Si on note S_i^j le score d'une classe étiquetée par i donné par un classifieur j alors, la règle de combinaison est une fonction notée : f et le score final de la classe i est noté :

$$f(\{S_i^j\}, j = 1, \ldots, L) \tag{3.4}$$

3.2.3 Complexité des combinaisons des classifieurs

Les types de complexité permettent de savoir si tous les scores des classifieurs participent à la dérivation de chaque score final combiné et aussi de savoir si une seule fonction traite toutes les classes ou alors il existe une fonction pour chaque classe.[6]

On divise les algorithmes en 4 différents types :

3.2.3.1 Faible complexité

$$S_i = f(\{S_i^j\}, j = 1, \ldots, L) \tag{3.5}$$

Il existe une seule fonction qui traite toutes les classes et prend comme entrées les scores d'une classe particulière.

3.2.3.2 Moyenne complexité type I

$$S_i = f_i(\{S_i^j\}, j = 1, \ldots, L) \tag{3.6}$$

Pour chaque classe « i », il existe une fonction qui prend comme entrées seulement les scores de cette classe.

3.2.3.3 Moyenne complexité type II

$$S_i = f(\{S_i^j\}, j = 1, \ldots, L, \{S_k^j\}, j = 1, \ldots, L, k = 1, \ldots, C, k \neq i) \tag{3.7}$$

Contrairement à la fonction citée précédemment, cette fonction ne prend pas uniquement le score d'une classe précise, mais toutes les sorties des classifieurs.

La combinaison des scores de chaque classe est calculée en utilisant la même fonction mais le score de la classe i est considéré comme paramètre spécial.

3.2.3.4 Haute complexité

$$S_i = f_i(\{S_i^j\}, j = 1, \ldots, L, \{S_k^j\}, j = 1, \ldots, L, k = 1, \ldots, C, k \neq i) \tag{3.8}$$

Les fonctions qui calculent les scores finaux sont différentes pour chaque classe et prennent les sorties des classificateurs comme paramètres.

3.3 Les méthodes de combinaison

3.3.1 Vote majoritaire :

Du moment que les classificateurs utilisés sont de type Crisp, la fonction de combinaison utilise donc la classe la plus probable pour chaque classificateur et la classe finale la plus sûre est celle supportée par la majorité des classificateurs.[6]

Si on représente le score de la classe notée i dans le classificateur noté « j » par : S_i^j, la classe finale la plus sûre est alors :

$$Classe = C_{\max_{i=1}^{C} \sum_{j=1}^{L} S_i^j} \tag{3.9}$$

Où :

- **C :** représente le nombre de classes.
- **L :** représente le nombre de classifieurs.

3.3.1.1 Exemple

10 étudiants votent pour élire le délégué de la classe, Les étudiants peuvent choisir un seul candidat parmi les 3 candidats.

Le tableau ci-dessous représente les résultats obtenus :

$\frac{Etudiants}{Candidats}$	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	Somme
C_1	0	0	1	0	0	1	1	0	0	0	$\sum_{j=1}^{10} E_j = 3$
C_2	0	1	0	1	0	0	0	0	0	0	$\sum_{j=1}^{10} E_j = 2$
C_3	1	0	0	0	1	0	0	1	1	1	$\sum_{j=1}^{10} E_j = 5$

Tableau 3.1 Exemple vote majoritaire

$$Classe = C_{\max_{i=1}^{3} \sum_{j=1}^{10} S_i^j} = C_3 \tag{3.10}$$

D'après les résultats, le $3^{ème}$ candidat C_3 est élu délégué de la classe.

Début
$MAX \leftarrow 0$;
for $i \leftarrow 1$ A C **do**
  $S \leftarrow 0$;
  for $j \leftarrow 1$ A L **do**
    $S \leftarrow S + T[i, j]$;
  end for
  if S > MAX **then** $MAX \leftarrow S$; $CLASSE \leftarrow i$;
  end if
end for
ECRIRE ("La classe majorante est : ", CLASSE) ;
Fin

Algorithm 1 Exemple vote majoritaire

Tel que :

- **L :** Le nombre de classificateurs.
- **C :** Le nombre de classes.
- **CLASSE :** La classe majorante.

3.3.2 Méthode Borda :

C'est une méthode simple basée sur un ordonnancement des n-meilleurs classes pour chaque classificateur.

La méthode Borda additionne le classement des n-meilleurs classes, la classe la plus sûre est celle qui a le meilleur classement final.[6]

En termes mathématiques, si on note r_i^j le classement de la classe i dans le classificateur j, alors :

$$r_i = \sum_{j=1}^{L} r_i^j \tag{3.11}$$

Où :

- **L :** représente le nombre des classifieurs.
- r_i **:** représente le classement final de n-meilleurs classes.

3.3.2.1 Exemple

8 femmes classent 3 produits de nettoyage selon leur efficacité, les résultats du test appa-

raissent dans le tableau qui suit : [7]

$\frac{Femmes}{Produits}$	F1	F2	F3	F4	F5	F6	F7	F8	Somme
P_1	2	2	1	3	2	1	1	3	$\sum_{i=1}^{8} F_i = 15$
P_2	3	1	2	1	3	3	2	2	$\sum_{i=1}^{8} F_i = 15$
P_3	1	3	3	2	1	2	3	1	$\sum_{i=1}^{8} F_i = 15$

Tableau 3.2 Exemple 2 Méthode Borda

$$r_i = \sum_{j=1}^{8} r_i^j \tag{3.12}$$

Le produit ayant le plus de voix est : le minimum de r_i, il s'agit donc du produit P_1

```
DEBUT
K ← 1 ;
for i ← 1 A C do
    for j ← 1 A L do
        S ← S + T[i, j] ;
    end for
    V[K] ← S ;
    K ← K + 1 ;
end for
Min ← V[1] ;
CLASSE ← 1 ;
for k ← 2 A C do
    if V[k] < Min then
        Min ← V[k] ;
        CLASSE ← k ;
    end if
end for
ECRIRE (« la meilleure classe est : » CLASSE) ;
FIN
```

Algorithm 2 Exemple 1 Méthode Borda

Tel que :

- **L :** représente le nombre de classificateur.
- **C :** représente le nombre de classes.
- **V :** représente le vecteur de classement.

La méthode borda est très simple à calculer et ne nécessite aucun apprentissage. Pareil au vote majoritaire, on peut associer un poids à chaque classifieur. Le vecteur de sortie est donné par la formule suivante :

$$r_i = \sum_{j=1}^{L} w_j r_i^j \tag{3.13}$$

3.3.2.2 Exemple 2 (Borda avec poids)

Imaginons que quatre villes soient sollicitées pour déterminer la ville où sera construit l'hôpital les concernant.

Imaginons d'autre part que la ville A regroupe 42 % des votants, la ville B 26 % des votants, la ville C 15 % des votants et la ville D 17 % des votants.

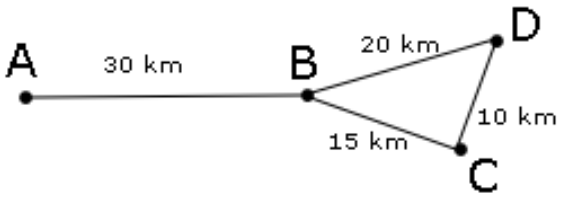


Figure 3.1 Exemple 2 méthode Borda

Considérons que chaque habitant souhaite que l'hôpital soit le plus proche possible de sa ville.

On obtient donc le classement suivant :

$\frac{Votants}{Villes}$	$H_1(42\%)$	$H_2(26\%)$	$H_3(15\%)$	$H_4(17\%$	Somme
$V_1(A)$	1	4	4	4	$\sum_{i=1}^{8} H_i * w_i = 274$
$V_2(B)$	2	1	3	3	$\sum_{i=1}^{8} H_i * w_i = 266$
$V_3(C)$	3	2	1	2	$\sum_{i=1}^{8} H_i * w_i = 227$
$V_4(D)$	4	3	2	1	$\sum_{i=1}^{8} H_i * w_i = 293$

Tableau 3.3 Exemple 2 méthode Borda

$$r_i = \sum_{j=1}^{L} w_j r_i^j \tag{3.14}$$

La ville gagnante est la ville C ayant le minimum de r_i.

Tel que :

— **L :** représente le nombre de classificateurs.

— **C :** représente le nombre de classes.

— **V :** représente le vecteur de classement

```
DEBUT
K ← 1 ;
for i ← 1 A C do
    for j ← 1 A L do
        S ← S + T[i, j] ∗ P[j] ;
    end for
    V[K] ← S ;
    K ← k + 1 ;
end for
Min ← V[1] ;
CLASSE ← 1 ;
for k ← 2 A C do
    if V[k] < Min then
        Min ← V[k] ;
        CLASSE ← k ;
    end if
end for
ECRIRE (« la meilleure classe est : » CLASSE) ;
FIN
```

Algorithm 3 Exemple 2 Méthode Borda avec poids

— **P :** représente le vecteur de poids.

3.3.3 Méthode du comportement d'espace des connaissances BKS

Cette méthode consiste à essayer pour chaque combinaison des résultats des classifieurs d'estimer la classe la plus probable basé sur les données d'apprentissage.[8]

Voici un exemple pour comprendre le principe :

Prenons par exemple deux personnes A et B qui doivent répondre à *400* questions dont la réponse peut être soit *vrai*, soit *faux*. Il y a *200* questions dont la réponse est *vrai (vrai-v)* et *200* autres dont la réponse est *faux (faux-f)*. Le comportement des deux personnes est décrit dans le tableau qui suit :

$\frac{A}{B}$	Vrai / Vrai	Vrai /Faux	Faux / Vrai	Faux / Faux
Vrai-v/ Faux-f	90/10	60/20	40/10	10/160
Estimation	Vrai-v	Vrai-v	Vrai-v	Faux-f

Tableau 3.4 Exemple 1 méthode BKS

Les deux personnes ont répondu par vrai à 100 questions dont 90 étaient *vrai-v* et 10 étaient *faux-f*, il y a 60 *Vrai-v* et 20 *faux-f* où la personne A a répondu par vrai et la personne B par *faux*...

En se basant sur les résultats obtenus, on peut estimer si une question donnée est *vrai-v* ou *faux-f*. Si les deux personnes répondent par *faux/faux* alors on peut dire que la question a plus de chance d'être *faux-f* sinon elle a une grande probabilité d'être *vrai-v*.

Soit $D_j(X)$ un classifieur qui a le vecteur X comme entrées,$j \in 1,..,L$. L Le nombre de classifieurs et $\{1,\ldots,C\}$ est l'ensemble des classes, C représente le nombre de classes. $D_j(X) = i_j, i \in \{1,\ldots,C\}$ signifie que le classifieur j assigne l'entrée X à la classe i. Donc i_j veut dire que la décision du classifieur j est i.

L'intersection des décisions de tous les classifieurs est une unité de la table de BKS, cette unité est appelée *Unité Focale* Focal Unity (FU), chaque unité contient trois types de données :

— $T_{D_1,\ldots,D_L}$: représente le nombre total des données d'apprentissage.
— $R_{D_1,\ldots,D_L}$: représente la meilleure classe représentée.
— $n_{D_1,\ldots,D_L}(i)$: représente le nombre d'échantillons pour chaque classe i.

Les nombres T et R peuvent être calculés par la formule :

$$T_{D_1,\ldots,D_L} = \sum_{i=1}^{C} n_{D_1,\ldots,D_L}(i) \tag{3.15}$$

$$R_(D_1,\ldots,D_L = \max_{i=1}^{C} n_{D_1,\ldots,D_L}(i) \tag{3.16}$$

Pour chaque unité, la seule classe qui doit être représentée est la meilleure classe.

3.3.3.1 La règle de décision

La méthode B.K.S opère en deux étapes :

— Premièrement, la modélisation des connaissances qui utilise l'ensemble d'échantillons d'apprentissage construit toutes les combinaisons des classes possibles où chaque combinaison est une unité puis calculer les trois nombres n, T et R.
— La deuxième étape consiste à combiner tous les résultats donnés par chaque classifieur individuel sur une entrée inconnue X, cette combinaison est l'Unité Focale, le résultat final est donné par la règle suivante :

$$E(X) = \begin{cases} R_{D_1,\ldots,D_L} & \text{si : } T_{D_1,\ldots,D_L} > 0 \text{ et } \frac{n_{D_1,\ldots,D_L}(RD_1,\ldots,D_L)}{TD_1,\ldots,D_L} > \lambda \\ 0 & \text{sinon} \end{cases} \tag{3.17}$$

Tel que :

— λ **:** représente le seuil $0 < \lambda < 1$ qui contrôle le degré de décision.

On peut représenter la méthode par un tableau (Un tableau de consultation) où chaque combinaison possible (unité) est une colonne (index). Le tableau est rempli en utilisant l'ensemble des données d'apprentissage.

3.3.3.2 Exemple :

Nombre de classes : *C=3*, nombre de classifieurs : *L=2*, nombre d'échantillons d'apprentissage : *N=100*.

Les unités	1,1	1,2	1,3	2,1	2,2	2,3	3,1	3,2	3,3
$n_{D_1,D_2}(i)$	10/3/3	3/0/6	5/4/5	0/0/0	1/16/6	4/4/4	7/2/4	0/2/5	0/0/6
E	1	3	1,3	0	2	1,2,3	1	3	3

Tableau 3.5 Exemple 2 méthode BKS

3.3.4 DS

3.3.4.1 Bref historique

C'est une théorie appelée : théorie de l'évidence, méthode de croyance introduite par Dempster (1968) et Shafer (1976) développée durant les années (1980, 1990).[9][9]

3.3.4.2 Notions et principes

Soit r un ensemble d'hypothèses $(H_1, .., H_(n))$ et 2^Ω l'ensemble des parties de Ω. On définit la fonction : $m : 2^\Omega \mapsto [0,1]/A \mapsto m(A)$. $m(A)$: est appelée la fonction de masse qui consiste à calculer les degrés de croyance ou de plausibilité d'une partie A de 2^Ω.[9]

3.3.4.3 Propriétés de « m »

- $\sum_{A_i \in 2^\Omega}(m(A_i)) = 1$: chaque $A_i \in 2^\Omega$ a un degré de croyance ou de plausibilité exactement comme probabilité et car $m(A)$ définie aussi une distribution de probabilités.
- $m(\emptyset) = 0$

3.3.4.4 Fonctions en relation avec « m »

- **Bel(A) :** Croyance (Belief en anglais) que la vérité est dans A donc nécessairement les éléments B_i qu'on prend sont inclus en A (autrement dit deviennent comme 2^Ω) on a donc :

$$Bel(A) = \sum_{B_i \subset A}(m(B_i)) \tag{3.18}$$

- **Pl(A) :** la plausibilité que la vérité est dans A, dans ce cas : $B_i \cap A \neq \emptyset$. La formule est donnée par :

$$Pl(A) = \sum_{B_i \cap A \neq \emptyset}(m(B_i)) \tag{3.19}$$

Remarque : $Pl(A)$ représente la borne supérieure de de la croyance, c'est-à-dire que : $Pl(A) > Bel(A)$.

D'une certaine façon, c'est le degré de croyance après la prise en compte de la nouvelle information.

3.3.4.5 Classification

Soit X un vecteur d'entrées non-étiqueté et soit L le nombre de classifieurs noté par$D^{j=1...L}$. Le vecteur de sortie donné par le classifieur j qui a comme entrée le vecteur X est alors donné par la formule :

$$D^j(X) = (S_1^j, \ldots, S_C^j) \tag{3.20}$$

Tel que :

— **C :** représente le nombre de classes.

— **k :** représente la décision du classifieur j si : $S_k^j = \max_{i=1}^{C}(S_i^j)$. Alors la classe k est celle qui correspond à l'index du maximum des composantes du vecteur de sortie.

Pour chaque classifieur D^j et chaque classe i, on calcule la valeur $e_i(D^j(X))$ qui représente une valeur d'évidence pour que la classe de sortie du classifieur D^j soit la classe i ce qui permet d'augmenter la qualité de la sortie. En effet, la théorie de Dampster-Shaffer introduit ces valeurs d'évidence qui nous permettent de choisir la classe la plus évidente.

3.3.4.6 Decision Template (DT)

Afin de calculer la valeur d'évidence, on introduit la notion de Décision-Template qui vont être déduites à partir de l'ensemble des données d'apprentissage Z. Ce dernier contient N vecteurs d'entrée étiquetés : $Z = \{z_1, \ldots, z_k, \ldots, z_N\}$.

La décision Template $DT_i(Z)$ d'une classe i est la matrice $L * C$ tel que :

$$DT_i(Z) = [dt_i(p,q)(Z)] \tag{3.21}$$

Avec :

$$dt_i(p,q)(Z) = \frac{\sum_{k=1}^{N} ind(z_k, i) S_p^q}{\sum_{k=1}^{N} int(z_k, i)}, p = 1, ..., C, q = 1, ..., L \tag{3.22}$$

La fonction $ind(z_j, i)$ est un indicateur qui confirme que le vecteur z_j est étiqueté par la classe i où elle renvoie *1*, cela permet de garder le résultat $S_p^q(z_k)$. Par contre, si ce vecteur z_j est étiqueté par une autre classe différente de i, la fonction renvoie *0*, lerésultat $S_p^q(z_k)$ est donc ignoré.[5]

Remarques :

— Dans la DT de la classe i, on a besoin uniquement des vecteurs d'apprentissage qui ont été étiquetés par i.

— Le résultat gardé dans la cellule $dt_i(p,q)$, est la moyenne de tous les résultats des données

d'apprentissage étiquetés par i et donnés par le classifieur q à la classe p.

— Plus généralement, la DT d'une classe i est la moyenne des Profils de Décision DP [1] donnée par chaque vecteur d'apprentissage z_j est étiqueté par la classe i.

— La DT d'une classe i, est le résultat de référence, et tout vecteur d'entrée X non-étiqueté est de classe i si son DP est similaire à la DT_i

3.3.4.7 Mesure de similarité

Afin de calculer la similarité entre la Décision Profile d'un vecteur d'entrée $DP(X)$ et la DT_i on a besoin de la fonction $Sim(DP(X), D_i)$. On peut définir la fonction Sim de différentes manières, vu que l'idée générale est de comparer la matrice $DP(X)$ avec les C matrices de Décision Template $(DT_1 \dots DT_C)$.[9]

3.3.4.8 DS appliquée à l'aide des DT

Cette technique dépend totalement du Modèle de décision (DT). Les classificateurs sont de type possibilistique. Au lieu de calculer la similarité entre le DT_i et les sorties des classificateurs l'algorithme va plus loin.[5][6]

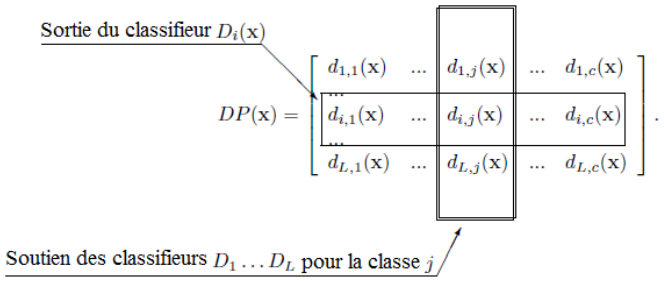
1. Soit DT_i^j la j ème ligne de DT de la classe i. On calcule la proximité entre DT_i^j et $D_j(x)$ pour la classe $i = 1, \dots, C$ et chaque classificateur $j = 1, \dots, L$.

$$\Phi_{i,j} = \frac{(1 + ||DT_i^j - D_j(x)||^2)^{-1}}{\sum_{k=1}^{C}(1 + ||DT_i^j - D_j(x)||^2)^{-1}} \tag{3.23}$$

2. Avec $\Omega_{i,j}$ on calcule pour chaque classe $i = 1, \dots, C$, et chaque classificateur $j = 1, ..., L$, le degré de croyance.

$$b_i(D_j(x)) = \frac{\Phi i, j(x)\Pi_{k \neq i}(1 - \Phi_{k,j}(x))}{1 - \Phi_{i,j}(x)[1 - \Pi_{k \neq i}(1 - \Phi_{k,j}(x))]} \tag{3.24}$$

1. On appelle « Decision Profile » (DP), l'organisation des classifieurs comme une matrice dont chaque ligne représente un classifieurs (de 1 à L) et chaque colonne représente une classe[5].



3. Le vecteur de sortie final a comme composantes :

$$S_i^E(x) = k\Pi_{j=1}^{L} b_i(D_j(x)), i = 1, \ldots, c \tag{3.25}$$

Où :

— **k :** représente une constante de normalisation.

— **E :** représente le vecteur final de la combinaison.

3.4 Conclusion

Dans ce chapitre, nous avons mis le point sur les différentes méthodes de combinaison qui ont pour objectif majeur d'améliorer la reconnaissance du manuscrit arabe. Le chapitre suivant sera consacré aux différents classifieurs qui vont constituer la plate-forme de cette combinaison dans le but d'augmenter la précision de la classification.

CHAPITRE 4 LES CLASSIFIEURS UTILISÉS

4.1 Introduction

Après extraction des vecteurs caractéristiques à partir d'images brutes, nous allons nous intéresser dans ce chapitre aux différents classifieurs en utilisant les réseaux de neurones Neural Networks (NN) et les machines à vecteurs de support ou séparateurs à vaste marge SVM. Chacun d'entre eux (Réseaux de neurones et machines à vecteurs de support) fait son apprentissage grâce à deux différents descripteurs d'image : les moments de Legendre LM et le descripteur FT. Le travail réalisé s'appuie sur une base de données de 3597 images de 10 classes. L'apprentissage se fait sur 2397 images soit environ 70 % de la base de données, le test est de 30 %.

4.2 Les réseaux de neurones

Les réseaux de neurones sont des technologies artificielles qui représentent un modèle de calcul dont la conception est très schématiquement inspirée du fonctionnement des neurones biologiques. Pendant que vous êtes en train de lire ces quelques mots, vous utilisez un réseau de neurones biologiques très complexe. Vous possédez un ensemble de 10^{11} neurones très interconnectés pour faciliter votre lecture ou réflexion par exemple.[1]

Un réseau de neurones artificiel permet de réaliser certaines tâches, telle que la mémorisation associative, l'apprentissage, l'accomplissement de deux taches en parallèle, ... etc.

Le neurone artificiel est loin de posséder toutes les capacités du neurone biologique [10]

Historiquement, le développement des réseaux neurones a commencé vers la fin du dix-neuvième siècle et au début du vingtième siècle. Il consistait au début en un travail dans le domaine de la physique, de la psychologie et de la neurophysiologie, élaboré par quelques scientifiques comme Hermann Von Helmholtz, Ernst Mach, Ivan Pavlov et bien d'autres ... etc.

Les réseaux de neurones modernes sont apparus en 1940 grâce au travail fait par Warren McCulloch et Walter Pitts, qui montrent qu'un réseau de neurones artificiel peut principalement calculer toute fonction arithmétique ou logique, leur travail a été l'origine du réseau de neurones artificiel.

En 1950 Frank Rosenbatt et ses collègues ont construit la première application pratique des réseaux neurones un 'preceptron network'. Ils ont démontré son pouvoir de reconnaissance des motifs (pattern recognition). Entre temps, Bernard Widrow et Ted Hoff ont introduit un algorithme d'apprentissage utilisé pour apprendre un réseau de neurones linéaire, certaines limitations ont fait que le développement a cessé.

La clé qui a relancé le développement des réseaux de neurones en 1980 est "backpropagation algorithm" pour l'apprentissage des multicouches "perceptron network".

Les réseaux de neurones ont pris une place importante et permanente comme étant un outil efficace et essentiel, utilisé dans certaines situations appropriées.

Les réseaux de neurones ont été utilisés pour la reconnaissance des écritures individuelles et ont été utilisés par un institut italien afin de tester la pureté de l'huile d'olive. Google utilise les réseaux de neurones pour tagger (identification automatique), Microsoft les a développés pour la traduction vocale de l'anglais vers le chinois.[11]

4.3 Architectures

Les neurones artificiels sont des simples fonctions mathématiques qui permettent de former une fonction très utile, il n'existe pas qu'une seule notation de neurones acceptée universellement, nous prenons les modèles les plus simples.[11]

4.3.1 Modèle d'un simple neurone

Un neurone simple avec une seule entrée est représenté comme

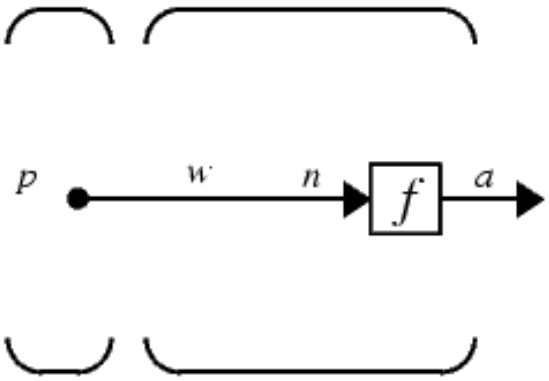


Figure 4.1 Schéma d'un neurone sans biais

Un scalaire d'entrée p transmet via une connexion où ce dernier est multiplié par un poids w. Le produit pw est le seul argument de la fonction de transfert f qui produit le résultat de sortie notée a. On note : $a = f(pw)$.

Le plus souvent, on note le paramètre de la fonction par n alors $n = pw$ et $a = f(n)$.

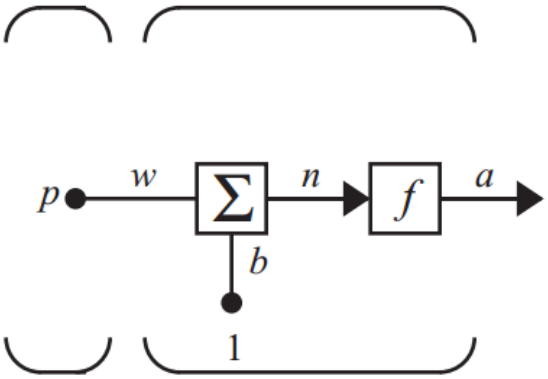


Figure 4.2 Schéma d'un neurone avec biais

Le neurone avec biais a un scalaire b, le biais est simplement additionné au produit pw. On note $a = f(pw + b)$.

Les paramètres w et b sont ajustables par les règles d'apprentissage et le résultat final dépend de la fonction de transfert choisie.

Différentes fonctions (linéaires ou non linéaires) peuvent être utilisées comme fonctions de transfert, selon le problème, certaines fonctions paraissent plus utiles que d'autres, nous résumons certaines fonctions souvent utilisées dans le tableau qui suit :

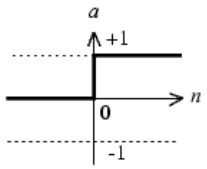
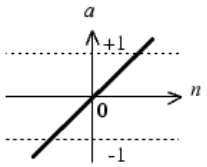
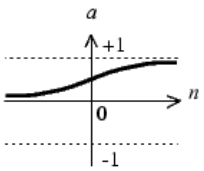
Nom	***Entrée/sortie***	***Courbe***
La fonction de Heaviside	$a = 0, n < 0$ $a = 1, n \geq 0$	
Linéaire	$a = n$	
Sigmoïde	$a = \frac{1}{1+e^{-n}}$	

Figure 4.3 Schéma d'un neurone multi-entrées

4.3.2 Modèle d'un neurone multi-entrées

En pratique, un neurone a plus d'une seule entrée, il peut avoir un vecteur d'entrées de dimension R (R entrées notées $p_1, p_2, ..., p_R$) pour chaque entrée, on associe un poids $(w_1, w_2, ..., w_R)$.

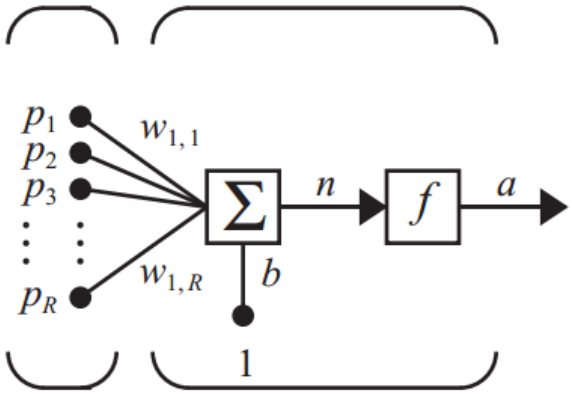


Figure 4.4 Schéma d'un neurone multi-entrées

Le paramètre n de la fonction de transfert est donné comme suit :

$n = w_1p_1 + w_2p_2 + ... + w_Rp_R + b.$

4.3.3 Couches de neurones

Même avec plusieurs entrées, un seul neurone est insuffisant pour résoudre les problèmes rencontrés en pratique. Un ensemble de plusieurs neurones qui opèrent en parallèles est appelé une couche. Une couche de neurones contient un nombre S de neurones, ce nombre est indépendant du nombre d'entrées R.[11]

Tous les neurones de la couche ont le même vecteur d'entrée mais chaque neurone peut avoir un vecteur de poids différent. Comme la sortie d'un seul neurone est un scalaire, alors la sortie de la couche est un vecteur de dimension S égale au nombre de neurones de la couche.[11]

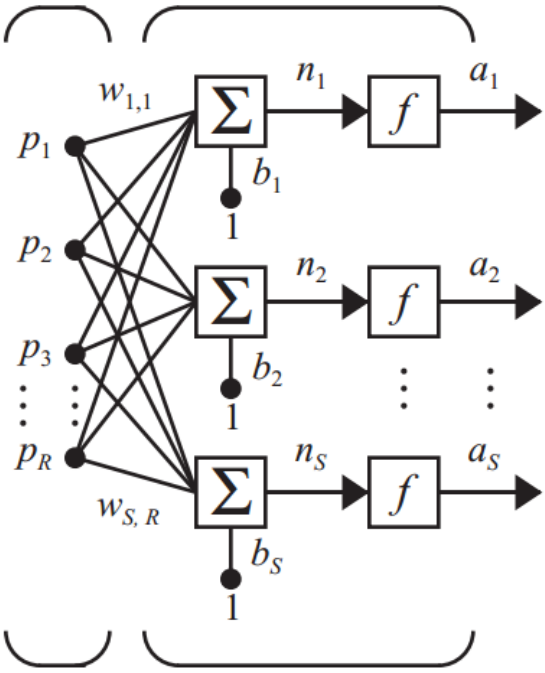


Figure 4.5 Schéma d'une couche de neurones

4.3.4 Réseaux multi-couches

Afin d'avoir un réseau de neurones plus fort, on peut utiliser plus d'une couche, les différentes manières dont les couches sont reliées offrent différentes architectures des réseaux de neurones (perception, Hamming network, Hopfield Network, ...). [11]

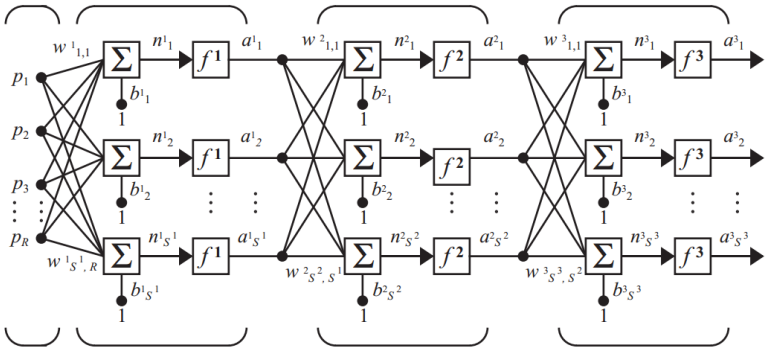


Figure 4.6 Schéma d'un réseau multi-couches

Les réseaux de neurones peuvent être classés en deux catégories :[11]

4.3.4.1 Réseaux de neurones feed-forward

Les réseaux de neurones feed-forward (généralement représentés par les perceptrons) se composent d'une ou plusieurs couches. Ces couches sont reliées entre elles, en série telle que les sorties de chaque couche sont les entrées de la couche suivante jusqu'à la dernière couche où ses sorties sont les résultats du réseau, cette couche est appelée couche de sortie, les autres couches s'appellent les couches cachées.

Chaque couche a sa matrice de poids et son vecteur de biais.

4.3.4.2 Réseaux de neurones feedback

Dans les réseaux de neurones feedback, les sorties de ce réseau sont connectées aux entrées.

Comme nous l'avons déjà évoqué, les réseaux multicouches sont plus forts que les réseaux formés d'une seule couche, un réseau constitué de deux couches nous offre le choix d'utiliser deux fonctions de transfert différentes, ce qui ne peut pas être fait dans un réseau formé d'une seule couche.

4.4 Les Algorithmes d'apprentissage

Déterminer la matrice des poids la plus adaptée pour le réseau se fait en utilisant différents algorithmes (appelés les règles d'apprentissage "learning rules en anglais"). Ces algorithmes ajustent les valeurs des poids afin d'obtenir les résultats les plus corrects. Ces algorithmes d'apprentissage sont classés en trois catégories :[11]

4.4.1 Apprentissage supervisé

Dans ce mode d'apprentissage, on utilise un ensemble d'apprentissage $P_1, T_1, \ldots, P_i, T_i, \ldots, P_n, T_n$ où P_n est l'entrée du réseau correspondant à la cible (la sortie correcte) T_n Target en anglais. L'algorithme ajuste les matrices des poids et biais afin de mettre la sortie plus proche de la cible. Parmi ces algorithmes on a : Hebb, Widrow-Hoff et Backpropagation pour l'apprentissage d'un réseau multicouche.

4.4.2 Apprentissage non supervisé

Dans ce mode il n'y a pas d'ensemble de cibles (Targets) mais seulement l'ensemble des entrées, les algorithmes de ce type caractérisent les entrées en un nombre fini de classes. Ce type d'apprentissage peut être utile dans certaines applications.

4.5 Les machines à vecteurs de support SVM

4.5.1 Définition des machines à vecteurs de support

Les machines à vecteur de support SVM sont un ensemble de techniques d'apprentissage pour la classification, la régression et d'autres tâches d'apprentissage. Les SVM appartiennent à une famille de généralisation des classifieurs linéaires. En d'autres termes, la SVM est une classification et régression d'outil de prédiction qui utilise la théorie de l'apprentissage de machine pour maximiser la précision prédictive automatiquement en évitant le sur-ajustement (over-fit) de données. Les SVM peuvent être définies comme des systèmes qui utilisent l'hypothèse de l'espace fonctionnel linéaire de dimension très élevée, formées avec un algorithme d'apprentissage dérivé de la théorie d'apprentissage statistique.[12][13][14]

4.5.2 Historique

L'algorithme du vecteur de support est une généralisation non linéaire de l'algorithme appelé Portrait Généralisé développé en Russie dans les années soixante. Dans les années soixante-dix, Vapnik et Chervonenkis ont développé la théorie de Vapnik et Chervonenkis (VC) avec l'exploration des fondations théoriques des SVM. Dans sa forme actuelle, La Machine à vecteurs de support a été développée chez AT&T Bell Laboratoires par Vapnik et ses collègues. En raison de ce contexte industriel, la recherche SV a jusqu'à ce jour eu une orientation vers les applications du monde réel. Le travail initial était basé sur La reconnaissance optique de caractères (OCR). Dans un court laps de temps, les SV classificateurs sont devenus compétitifs avec les meilleurs systèmes disponibles comme l'OCR et la Reconnaissance de formes). Un tutoriel complet sur les SV classificateurs a été publié par Burges en 1998. Mais aussi dans la régression et les séries chronologiques de prédiction, d'excellentes performances ont été rapidement obtenues. L'apprentissage du SV a évolué dans le domaine de la recherche.[13][14]

4.5.3 Principe général

Les réseaux de neurones peuvent poser quelques problèmes à noter :[13][14]

— Le fait d'avoir de nombreux locaux minimaux, on ne connaît pas à priori le nombre de neurones nécessaire à une tâche, cela amène à nous demander si l'optimalité de ce réseau de neurones est atteinte.

— Une autre chose à noter est que même si les solutions de réseaux de neurones utilisés tendent à converger, cela peut ne pas aboutir à une solution unique.

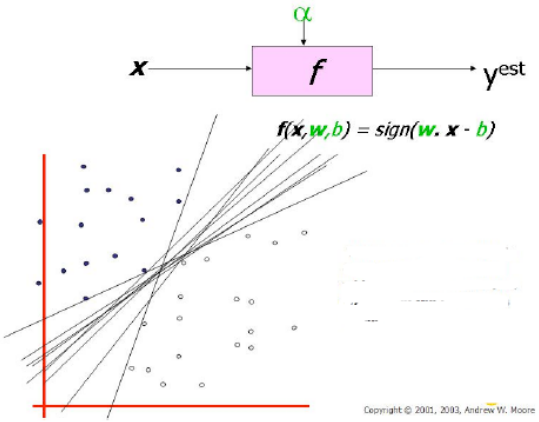


Figure 4.7 Schéma montrant les différents classificateurs linéaires qui peuvent être adaptés pour classer les données

La figure ci-dessus montre les différents classificateurs linéaires qui peuvent servir à classer les données, mais lequel est considéré comme le meilleur hyperplan. La nécessité d'une SVM se pose [2]. Il y a beaucoup de classificateurs linéaires appelés aussi hyperplans qui séparent les données. Toutefois, seul un d'entre eux réalise une séparation maximale. La raison pour laquelle nous en avons besoin est que si nous utilisons un hyperplan pour classer, il pourrait finir en étant plus proche d'un ensemble de données par rapport aux autres, or nous ne voulons pas que cela se produise, c'est pour cela que nous considérons que le concept de classificateur de marge maximale est une solution évidente. L'illustration suivante donne un exemple de classificateur de la marge maximale qui apporte une solution au problème mentionné ci-dessus.[13][14]

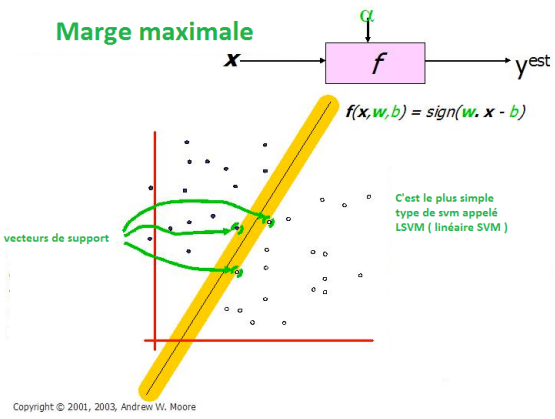


Figure 4.8 Illustration d'un SVM linéaire[2]

L'expression de la marge maximale est :

$$margin = argmind(x) = armin \frac{|x.w + B|}{\sqrt{\sum_{i=1}^{d} w_i^2}}, x \in D \tag{4.1}$$

La figure ci-dessus illustre une classification linéaire maximale. Dans ce contexte, il est un exemple d'un simple classifieur linéaire SVM. Une autre question intéressante à se poser est pourquoi la marge maximale ? Il y a quelques bonnes explications qui incluent une meilleure performance. Une autre raison est que même si nous avons fait une petite erreur dans l'emplacement de la limite cela nous donne moins de chances de provoquer une erreur de classification. L'autre avantage serait d'éviter les locaux minimaux et d'avoir une meilleure classification.[13][14]

Maintenant, nous essayons d'exprimer le SVM mathématiquement, dans cette partie nous essayons de présenter un SVM linéaire. Pour les calculs mathématiques que nous avons :

1. $Si : Y_i = +1; wx_i + b \geq 1$
2. $Si : Y_i = -1; wx_i + b \leq 1$
3. Donc, pour tout i : $y_i(w_i + b) \geq 1$

Tel que :

— **x :** représente un point vecteur.

— **w :** représente le poids et est également un vecteur

Donc, pour séparer les données (1) doit toujours être supérieure à zéro. Parmi tous les hyperplans possibles, SVM sélectionne celui où la distance de l'hyperplan est aussi grande que possible. Si les données de l'apprentissage sont bonnes, chaque vecteur de test est situé dans un rayon « r » du vecteur d'apprentissage.

Maintenant, si l'hyperplan choisi se trouve plus loin possible des données. Cet hyperplan maximise la marge et bis-secte les lignes entre les points les plus proches. Ainsi, nous avons les équations (1), (2) et (3)

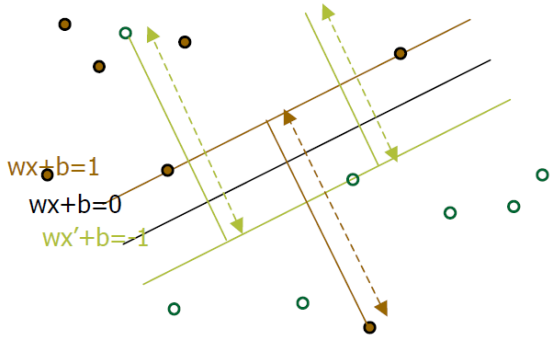


Figure 4.9 Représentation des hyperplans

La distance du point d'un hyperplan le plus proche de l'origine peut être trouvée en maximisant le x avec celui se situant sur l'hyperplan. Le même procédé est appliqué pour les autres points du côté, ainsi la résolution et la soustraction de deux distances, nous donnent la distance résumée de l'hyperplan qui sépare les points les plus proches.

La marge maximale notée M est calculée par : $M = \frac{2}{||w||}$

Maintenant, la maximisation de la marge est identique au minimum, d'où un problème d'optimisation quadratique que nous devons résoudre pour w et b. Pour résoudre cela, nous devons optimiser la fonction quadratique avec des contraintes linéaires. La solution consiste en la construction d'un problème dual. Nous devons trouver w et b tels que :

$\Phi(w) = \frac{1}{2}|w'||w|$ est minimisée, et pour tout $\{(x_i, y_i)\} : y_i(w * x_i + b) \geq 1$

Après résolution, nous obtenons :

$w = \sum a_i * x_i$ et $b = y_k - w * x_k$ pour tous x_k tel que $ak \neq 0$

La fonction de classement aura la forme suivante :

$$f(x) =_s uma_i y_i x_i * x + b \tag{4.2}$$

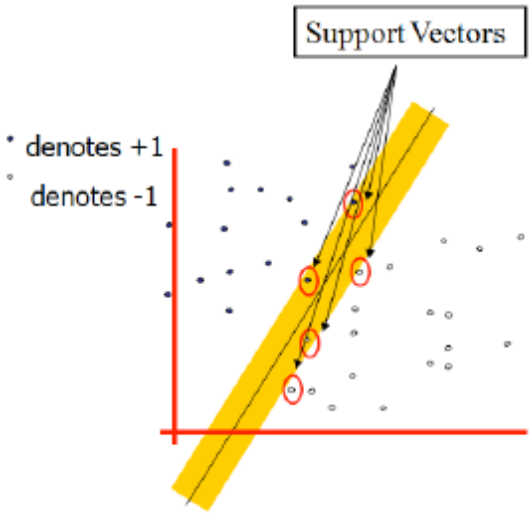


Figure 4.10 Représentation des vecteurs de support [2]

4.6 LIBSVM

LIBSVM est une bibliothèque pour les machines à vecteurs de support, Elle a été développée depuis l'an 2000 par un groupe de chercheurs appartenant au Conseil national des sciences de Taïwan. Le but est d'aider les utilisateurs à appliquer facilement les SVM à leurs applications. La

LIBSVM est très populaire dans l'apprentissage de la machine et de nombreux autres domaines.

Depuis l'an 2000 un groupe des chercheurs ont mis au point le paquet LIBSVM comme une bibliothèque pour les machines à vecteurs de support.

L'adresse Web du paquet est :http://www.csie.ntu.edu.tw/ cjlin/libsvm

LIBSVM est actuellement l'un des logiciels de SVM les plus utilisés

Domaine	Travaux faits
Vision par ordinateur	LIBPMK (Grauman et Darrell) Maltparser (Nivre et al.)
Traitement automatique du langage naturel	Maltparser (Nivre et al., 2007)
Neuroimagerie	PyMVPA (Hanke et al., 2009)
Bio-informatique	BDVal (Dor_ et al., 2010)

Tableau 4.1 Travaux faits dans certains domaines qui ont utilisé LIBSVM avec succès

LIBSVM prend en charge les tâches d'apprentissage suivantes :

— Support Vector Classification (SVC) bi-classes et multi classes
— Support Vector Regression (SVR)
— SVM avec une seule classe (One-Class).

Une utilisation typique de la LIBSVM implique deux étapes

— Un apprentissage d'un ensemble de données pour l'obtention d'un modèle.
— L'utilisation du modèle pour la prédiction de l'information d'un ensemble de données de test.

4.6.1 Formulation de SVM

4.6.1.1 C- SVC

Étant donné un vecteur d'apprentissage $x_i \in R^n, i = 1, \ldots, l$, en deux classes, et un vecteur d'indicateur, $y \in R^l$ tel que$y_i \in 1, -1$, Boser et al., 1992 et Cortes et Vapnik, 1995 résolurent le problème d'optimisation suivant :

$$min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{4.3}$$

Sujet à : $y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i$, **(1)** tel que $\xi_i \geq 0, i = 1, ..., l$

Où : ϕ est une transformation non linéaire de l'espace d'entrée x en un espace de dimensions supérieures noté $\phi(x)$ et C est le paramètre de régulation. En raison du fait que la dimension de la variable du vecteur w peut être élevée, on résout généralement le problème dual suivant :

$$min_{\alpha}\frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{4.4}$$

Sujet à : $y^T\alpha = 0$ **(2)** tel que $0 \leq \alpha_i \leq C, i = 1, \ldots, l$,

Où : $e = [1, ..., 1]$ est un vecteur unitaire, Q est une matrice positive lxl semi définie, $Q_{ij} = y_i y_j K(x_i, x_j)$, et $K(x_i, y_j) = \phi(x_i)^T\phi(x_j)$ qui est la fonction du noyau.

Après avoir résolu le problème **(2)**, et en utilisant la relation **(1)** et **(2)**, l'optimal w satisfait : $w = \sum_{i=1}^{l} y_i\alpha_i\phi(x_i)$ **(3)**

Et la fonction de décision est :

$$sgn(w^T\phi(x_i) + b) = sgn(\sum_{i=1}^{l} y_i\alpha_i K(x_i, x_j) + b) \tag{4.5}$$

On stocke $y_i\alpha_i \forall i, b$, les noms d'étiquette, les vecteurs de support, et d'autres informations telles que les paramètres du noyau dans le modèle pour la prédiction.

4.6.1.2 ν- SVC

En 2000, Schölkopf et al. ont introduit un nouveau paramètre $\nu \in]0, 1]$.

Il est prouvé que ν est une borne supérieure dans la fraction d'apprentissage d'erreurs et une borne inférieure dans la fraction des vecteurs de support.

Étant donné un vecteur d'apprentissage $x_i \in R^n, i = 1, \ldots, l$, en deux classes, et un vecteur d'indicateur, $y \in R^l$ tel que $y_i \in \{1, -1\}$, le problème d'optimisation primal est :

$$min_{w,b,\eta,\rho}\frac{1}{2}w^T w - \nu\rho + \frac{1}{l}\sum_{i=1}^{l}\eta_i \tag{4.6}$$

Sujet à : $y_i(w^T\phi(x_i) + b) \geq \rho - \eta_i$, **(4)** tel que : $\eta_(i) \geq 0$, $i = 1, \ldots, l, \rho \geq 0$

Le problème dual est :

$$min_{\alpha}\frac{1}{2}\alpha^T Q\alpha \tag{4.7}$$

Sujet à : $0 \leq \alpha_i \leq \frac{1}{l}, i = 1, .., l$,

$e^T\alpha \geq \nu, y^T\alpha = 0$

Où : $Q_{ij} = y_i y_j K(x_i, x_j)$.

En 2001, Chang et Lin ont montré que ce problème **(5)** est résoluble si et seulement si :

$$\nu \leq \frac{2min(\#y_i = +1, \#y_j = -1)}{l} \leq 1 \tag{4.8}$$

Donc la plage utilisable de ν est plus petite que]0,1]

La fonction de décision est :

$$sgn(\sum_{i=1}^{l} y_i \alpha_i K(x_i, x_j) + b) \tag{4.9}$$

Il est démontré que $e^T\alpha \geq \nu$ peut être remplacé par $e^T\alpha = \nu$ d'après Crisp et Burges (2000), et Chang et Lin, (2001).

Dans la LIBSVM, le problème **(5)** est résolu à une version réduite car numériquement, α_i peut être trop petit en raison de la contrainte $\alpha_i \leq \frac{1}{l}$

$$min_\alpha \frac{1}{2}\overline{\alpha}^T Q \overline{\alpha} \tag{4.10}$$

Sujet à :$0 \leq \overline{\alpha}_i \leq \frac{1}{l}, i = 1, .., l$ **(5)** $e^T\overline{\alpha} = \nu l, y^T\overline{\alpha} = 0$

Si α est optimal pour le problème dual **(5)** et ρ optimal pour le primal problème **(4)**, Chang et Lin ont montré que $\frac{\alpha}{\rho}$ est une solution optimale de C-SVM avec $= \frac{1}{\rho l}$. Ainsi, en LIBSVM, la sortie (output) est $(\frac{\alpha}{\rho}, \frac{b}{\rho})$ dans le modèle.

4.6.2 Les noyaux

L'algorithme de la marge maximale est la recherche de l'hyperplan proposé par Vapnik en 1963 qui a construit un classifieur linéaire. Toutefois, en 1992, Bernhard E. Boser, Isabelle M. Guyon et Vladimir N. Vapnik ont suggéré un moyen de créer des classifieurs non linéaires en appliquant le Kernel Trick (initialement proposé par Aizenman et al. pour la marge maximale de l'hyperplan. De plus, $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ est appelée la fonction du noyau. Bien que de nouveaux noyaux soient proposés par les chercheurs, on peut trouver dans les livres sur les SVM quatre noyaux de base qui sont :

- Le noyau linéaire : $K(x_i, x_j) = {x_i}^T x_j$.
- Le noyau polynomial : $K(x_i, x_j) = ((\gamma x_i)^T x_j + r)^d, \gamma \geq 0$.
- Le noyau Radial Basis Function (RBF) : $K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma \geq 0$.
- Le noyau sigmoïde : $K(x_i, x_j) = \tanh(\gamma {x_i}^T x_j + r)\gamma$ γ, r et d sont des paramètres de noyau.

4.7 Conclusion

Après avoir introduit les réseaux de neurones et les machines à vecteurs de support, nous allons dans le chapitre qui suit, nous intéresser à comment implémenter la combinaison des classifieurs afin d'augmenter la précision de la reconnaissance du manuscrit arabe.

CHAPITRE 5 IMPLÉMENTATION, TESTS ET RÉSULTATS

5.1 Introduction

Il existe un bon nombre d'applications ayant recours au domaine de la reconnaissance de l'écriture manuscrite telles que la lecture automatique de bons de commande, le traitement automatique des chèques, la vérification de signatures ou encore le tri automatique du courrier.

Cela montre que la reconnaissance du manuscrit a pris ces dernières année un nouvel essor et fait l'objet d'applications de plus en plus nombreuses.

Dans le cadre de notre projet, nous avons choisi de mettre en œuvre un système de reconnaissance manuscrite hors-ligne en effectuant la combinaison des classifieurs déjà traités dans chapitres précédents. Notre mise en œuvre repose sur l'élaboration d'une application permettant la reconnaissance du manuscrit.

Dans ce chapitre, nous allons présenter les différentes étapes d'implémentation de ce système.

5.2 Environement de travail

5.2.1 Environement matériel

- `Processeur: Intel(R) Core(TM) i3-3220 CPU @ 3.30GHz (4 CPUs), ~3.3GHz.`
- `Mémoire vive : 6144 Mo.`
- `Disque dur  500 Go.`
- `Système d'exploitation : Windows7 Edition Intégrale 64-bit.`

5.2.2 Environement logiciel

5.2.2.1 Logiciels utilisés

1. **Matlab**
   Matlab est un logiciel commercial de calcul interactif, très performant pour effectuer de puissants calculs numériques selon plusieurs méthodes comme la différenciation et l'intégration ou bien l'analyse de Fourier. Cet environnement de développement permet notamment de concevoir des interfaces utilisateur et de déployer des applications au langage MATLAB.
   Version utilisée : 7.9.0.529 (R2009b)
2. **XAMPP**
   XAMPP est un ensemble de logiciels permettant la mise en place facile d'un serveur web, d'un serveur File Transfert Protocol (FTP) et d'un serveur de messagerie électronique. Il est réputé pour être facile d'utilisation, rapide d'installation et fonctionne sur les systèmes d'exploitation les plus répondus.

Version utilisée : 5.6.11

3. **Brackets**
   Brackets est un éditeur de code destiné au développement et au design de pages web aux formats Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) ou JavaScript. Développé par les équipes Adobe Systems, ce logiciel est disponible sous licence OpenSource.
   Version utilisée : 1.6 édition 1.6.0-16680
4. **Navigateurs web**

   — **Opera**
   Opera est un navigateur Web gratuit et multiplate-forme développé par la société norvégienne Opera Software, qui propose plusieurs logiciels relatifs à Internet. Opera est un navigateur facile à configurer possédant des fonctionnalités avancées.
   Version utilisée : 37.0.2178.43

   — **Mozilla Firefox**
   Mozilla Firefox est un navigateur web libre et gratuit, développé et distribué par la Mozilla Foundation. Firefox est un navigateur libre et gratuit proposant de nombreuses fonctionnalités, options de personnalisation.
   Version utilisée : 46.0.1

   — **Google Chrome**
   Google Chrome est un navigateur web de nouvelle génération créé par Google et basé sur le projet open source Chromium. Chrome est un navigateur innovant et compatible avec les derniers standards du web, comme le Hypertext Markup Language 5 (HTML5) et le Flash
   Version utilisée : 50.0.2661.102

5.2.2.2 Langages de programmation utilisés

1. **Matlab**
   Matlab est un langage de calcul technique haut niveau et un environnement interactif pour le développement d'algorithmes et l'analyse de données. Il est émulé par un environnement de développement qui porte le même nom.
2. **Hypertext Preprocessor (PHP)**
   PHP est un langage de programmation impératif orienté objet et libre, il sert essentiellement à produire des pages web dynamiques via un serveur HyperText Transfer Protocol (HTTP).
3. **JavaScript (JS)**
   JavaScript est un langage de programmation de scripts orienté objet. Majoritairement utilisé sur Internet, conjointement avec les pages Web HTML, le JavaScript s'inclut directement dans les pages Web et permet de les dynamiser.

5.3 Implémentation

5.3.1 Implémentation d'un réseau de neurones

Afin de créer un réseau de neurones performant adapté à notre problème de classification, on a choisi d'utiliser les réseaux multi-couches en utilisant l'algorithme `Back Propagation` pour l'apprentissage.

5.3.1.1 Création d'un réseau de neurones

La création d'un réseau de neurones multi-couches pour un apprentissage par l'algorithme de `Back Propagation` se fait sous Matlab en utilisant le script suivant : [15]

```
net = newff(P,T,[S1,S2,...,S(n-1)],{TF1,TF2,...,TFn},BTF);
```

Où :

- **P :** matrice d'entrées.
- **T :** matrice des cibles.
- **Si :** le nombre de neurones dans la couche cachée.
- **TFi :** la fonction de transfert dans la couche i.
- **BTF :** la fonction d'apprentissage de la Back Propagation.

Cette fonction peut avoir plus de paramètres, nous avons laissé le reste des paramètres par défaut.

5.3.1.2 La répartition d'entrées

La répartition de notre matrice d'entrées se fait comme suit :

- `net.divideParam.trainRation =` ratio d'apprentissage (default=0.6).
- `net.divideParam.testRation =` ratio de test (default=0.2).
- `net.divideParam.valRation =` ratio de validation (default=0.2).

5.3.1.3 Les paramètres d'itérations d'apprentissage

- `net.trainParam.max-fail =` nombre d'itérations sans amélioration pour arrêter l'apprentissage.
- `net.trainParam.epochs =` nombre d'itérations d'apprentissage.
- `net.performFcn =` la fonction de performance.

5.3.1.4 Lancement de l'apprentissage

- `[net,tr] = train (net, P, T)`

  tr : le taux d'apprentissage.

5.3.1.5 La simulation des entrées

La simulation a pour but d'essayer de reconnaître l'image (une seule image ou plusieurs).

La simulation des entrées avec le réseau *'net'* se fait avec la fonction : `y=sim(net,E);`

— **E :** vecteur d'entrées (ou matrice d'entrées).
— **Y :** vecteur de sorties (ou matrice de sorties), il correspond à la sortie du réseau de neurone *'net'* où l'indice de la valeur maximale représente la classe.

5.3.1.6 Affichage des résultats

Pour afficher les résultats de la performance du réseau entre l'ensemble des sorties de réseau et l'ensemble des cibles on trouve la fonction : `postreg = (y, C) ;`

C : la matrice des cibles correspondant aux sorties y.

5.3.1.7 Affichage des résultats

1. **Avec le descripteur LM**

Indice	**Paramètre**	**Résultat de `postreg`**	**Résultat calculé**
Net-LM-1	`fct=trainlm` `[20 20]`	95.67 %	92.00 %
Net-LM-2	`fct=trainbfg` `[20 20] 'logsig'`	97.92 %	91.42 %
Net-LM-3	`fct=trainlm` `[35]`	97.78 %	90.75 %
Net-LM-4	`fct=trainoss` `[35]`	91.98 %	93.17 %
Net-LM-5	`fct=trainrp` `[35] 'logsig'`	93.08 %	93.54 %

Tableau 5.1 Résultats de la performance du réseau de neurones en utilisant le descripteur LM

2. **Avec le descripteur FT**

Indice	Paramètre	Résultat de postreg	Résultat calculé
Net-FT-1	`fct=trainlm [20 20]`	99.23 %	97.08 %
Net-FT-2	`fct=trainlm [35]`	99.03 %	96.17 %
Net-FT-3	`fct=trainoss [35]`	97.67 %	95.42 %
Net-FT-4	`fct=trainrp [35] 'logsig'`	97.47 %	95.42 %

Tableau 5.2 Résultats de la performance du réseau de neurones en utilisant le descripteur FT

5.3.2 Implémentation d'une SVM

Afin d'implémenter deux modèles SVM, nous avons décidé d'utiliser la bibliothèque LIBSVM (voir, sect. 4.6) dont la version est *3.21*. Cette bibliothèque facilite la création de certains types de SVM, la compilation de LIBSVM produit deux programme `svmtrain` et `svmpredict` [1]

5.3.2.1 Les types de SVM supportés par LIBSVM

La bibliothèque LIBSVM supporte les types de SVM (voir, sect. 4.6.1) qui suivent :

— *C-SVC :* Classification multi-classes
— *ν-SVC :* Classification multi-classes
— *One-class SVM*
— *ϵ-SVR :* régression
— *ν-SVR :* régression [16]

Comme notre problème concerne la classification multi-classes, nous pouvons utiliser uniquement le type C-SVC ou le type nu-SVC qui sont adaptés à notre problème, dans notre cas, nous avons travaillé avec le type C-SVC.

5.3.2.2 Les types de noyaux supportés par LIBSVM

Parmi les types de fonctions de noyaux sous LIBSVM (voir, sect. 4.6.2), nous citons :

— **Le noyau linéaire :** $u' * v$.
— **Le noyau polynomial :**$(gamma * u' * v + coef0)^degree$.
— **Le noyau Radial Basis Function :** $exp(-gamma * |u - v|^2)$.
— **Le noyau sigmoïde :** $tanh(gamma * u' * v + coef0)$.[16]

Dans notre cas, nous avons utilisé uniquement la fonction Radial Basis.

1. Pour le téléchargement de LIBSVM et les détails de compilation voir http ://www.csie.ntu.edu.tw/ cjlin/libsvm

5.3.2.3 Apprentissage des SVM

À l'aide de LIBSVM, la création et l'apprentissage d'un model SVM se fait avec la fonction `svmtrain()` comme suit :

```
modèle = svmtrain(vecteur d'étiquettes des classes, matrice d'entées ['libsvmoptions']);
```

Où :

- **Vecteur d'étiquettes des classes :** Est un vecteur de dimension m d'étiquettes des classe données dans la matrice d'apprentissage (le type doit être double).
- **Matrice d'entrées :** Est une matrice de dimension $m * n$ constituée de m éléments d'apprentissage. (le type doit être double).
- **libsvm options :** Est une chaine de caractères qui comporte les options d'apprentissage tel que le type de SVM, la fonction de noyaux, ... etc. Ces options sont exprimées sous un format spécial.

L'apprentissage d'un modèle SVM de type RBF se fait en modifiant les valeurs des deux paramètres *c 'cost'* et *g 'gamma'*, pour cela, nous avons utilisé un algorithme pour automatiser la sélection de ces deux paramètres, cet algorithme permet en fait de donner les meilleures valeurs pour c et g, nous l'avons trouvé dans [2]

5.3.2.4 La simulation des entrées

Pour classifier des entrées inconnues avec un modèle SVM, on utilise la fonction `svmpredict ()` comme suit :

```
[étiquette, précision, probabilité d'estimation] = svmpredict(vecteur d'étiquettes
 test, vecteur d'entrée, modèle ['libsvm options']);
```

Où :

- **Vecteur d'étiquettes de test :** Utilisé dans les cas d'un test de valeurs., pour les valeurs inconnues, on utilise simplement une valeur aléatoire (le type doit être double).
- **Vecteur d'entrée :** Ce vecteur correspond au descripteur d'image à utiliser (LM ou FT dans notre cas).
- Modèle : Correspond à la sortie de la fonction svmtrain.
- **libsvm options :** Même description concernant l'apprentissage des SVM (voir, sect. 3.1.2.3).

5.3.2.5 Résultats obtenus

1. **Avec le descripteur LM**

2. Cet algoruthme se trouve à l'adresse : http ://www.csie.ntu.edu.tw/ cjlin/libsvm/faq.html#f803.
Remarque : Cet algorithme comporte une erreur, il se trouve qu'une variable n'a pas été initialisée, nous l'avons corrigé en initialisant la variable bestc à zéro. Cela s'est fait en ajoutant la ligne `bestc=0;` à ce code.

Indice	Paramètre	Résultat de l'algorithme	Résultat calculé
SVM-LM	`-c 8 -g 0.125`	95.619 %	95.00 %

Tableau 5.3 Résultats de la performance de la SVM en utilisant le descripteur LM

2. **Avec le descripteur FT**

Indice	Paramètre	Résultat de l'algorithme	Résultat calculé
SVM-FT	`-c 8 -g 0.0625`	97.2466 %	96.58 %

Tableau 5.4 Résultats de la performance de la SVM en utilisant le descripteur FT

5.3.3 La Combinaison de classifieurs

Nous avons présenté dans le troisième chapitre les différentes méthodes de combinaison de classifieurs. Dans cette partie, nous allons expliquer la manière dont nous avons implémenté ces méthodes, nous avons décidé de mettre en œuvre quatre méthodes de combinaison qu'on a réparti en deux catégories à savoir :

5.3.3.1 Méthodes qui ne passent pas par la phase d'apprentissage

Les algorithmes de ces méthodes sont très simples à implémenter, le principe de ces méthodes a déjà été expliqué dans le troisième chapitre (voir, sect. 3). Nous avons été obligés d'écrire certains algorithmes pour normaliser les sorties des classifieur.

1. **Méthode Vote Majoritaire**

   Cette méthode utilise des classificateurs de type Crisp (voir sect. 3.3.1) dont la classe acceptée est représentée par 1 et les autres classes pas des 0, or les sorties de nos classifieurs sont de type Possibiliste, pour cela, nous avons implémenté l'algorithme suivant pour convertir ces classifieurs Possibilistes en des classifieurs de type Crisp :

---

function [(D)]=crisp(D)
  $[L, C] = size(D)$;
  $Dm = zeros(L, C)$;
  for $i = 1$: L **do**
    $[\ , classe] = max(D(i, :))$;
    $Dm(i, classe) = 1$;
  end for
  $D = Dm$;
end function

Algorithm 4 Conversion d'un classifieur Possibiliste en un classifieur Crisp

---

Où :

— **D :** est le DP de nos classifieurs (voir sect. 3.3.4.8).

Résultat obtenu :

La combinaison des quatre classifieurs donne le résultat illustré dans le tableau suivant :

Classifieurs combinés	Résultat
Net-LM-5 (93.08%)	97.00 %
Net-FT-1 (97.08%)	
SVM-LM (95.00%)	
SVM-FT (96.58%)	

Tableau 5.5 Résultat de la combinaison des classifieurs en utilisant la méthode Vote Majoritaire

Analyse du résultat obtenu :

Le résultat de la combinaison est jugé insuffisant, ce résultat peut être amélioré et cela en prenant en compte le cas de confusion, c'est-à-dire le cas où il y égalité entre certaines valeurs de vote.

2. **Méthode Borda**

La méthode Borda consiste à ordonner n-meilleurs classes pour chaque classificateur (voir sect. 3.3.2), le classement se fait de manière croissante, dans notre cas, nous avons ordonné toutes les sorties des classifieurs en se basant sur la meilleure classe dont la distance (la valeur absolue de soustraction) est égale à 1 qui est la plus petite. Notre classifieur essaie toujours de modifier les valeurs des sorties comme un vecteur cible donné, dans notre cas par un vecteur Crisp, la meilleur classe aura 1 les autres auront des 0. Donc les meilleures sorties seront très proche 1 (soit inférieur ou supérieur mais très proche de 1) le reste sera proche de 0.

La fonction de classement des sorties d'un classifieur est présentée comme suit :

Résultat obtenu :

La combinaison des quatre classifieurs donne le résultat illustré dans le tableau suivant :

Classifieurs combinés	Résultat
Net-LM-5 (93.08%)	97,6667 %
Net-FT-1 (97.08%)	
SVM-LM (95.00%)	
SVM-FT (96.58%)	

Tableau 5.6 Résultat de la combinaison des classifieurs en utilisant la méthode Borda

Analyse du résultat obtenu :

```
function [(D)]=classement(D)
    [L, C] = size(D) ;
    Dm = D ;
    for i = 1 : L do
        for j = 1 : C do
            min = count − min(D(i, :)′, D(i, j)) ;
            Dm(i, j) = min ;
        end for
    end for
    D = Dm ;
end function
function M(i)n=count-min(vec,val)
    min = 1 ;
    for i = 1 : size(vec) do
        if (val > vec(i)) then
            min = min + 1 ;
        end if
    end for
end function
```

Algorithm 5 Fonctions de classement Méthode Borda

Le résultat de la combinaison est jugé acceptable, nous avons pu obtenir une amélioration considérable du meilleur classifieur *(Net-FT-1)* qui est égale à *0,5857 %*.

5.3.3.2 Méthodes qui passent par la phase d'apprentissage

Ces méthodes sont plus compliquées à implémenter et nécessitent des données d'apprentissage, dans notre cas, nous avons utilisé la même base de données d'apprentissage des classifieurs pour l'apprentissage de ces méthodes, nous avons juste créé une matrice qu'on a personnalisé et nommé matrice *Z*, cela afin de faciliter l'apprentissage, cette matrice a de plus pour chaque colonne d'une sortie de classifieur d'apprentissage des information concernant le classifieur utilisé pour avoir ces résultats l'étiquette de la classe et le code d'échantillon (numéro d'échantillon).

Le nombre de colonnes de la matrice Z est $N * L$ où N représente le nombre d'échantillons et L représente le nombre de classifieurs. Comme nous possédons *2397* échantillons et *4* classifieurs, alors le nombre de colonnes est égal à *9588* et le nombre de lignes représente la taille de sortie du classifieur qui est égal au nombre de classes (*10* dans notre cas) plus *3* cases à savoir : *étiquette, classifieur et code échantillons,* nous travaillons donc avec une matrice d'apprentissage de dimension $13 * 9588$.

1. **Méthode BKS**

   Cette méthode (voir, sect. 3.3.3) se base sur un tableau indexé par toutes les combinaisons des classes possibles données par la décision des classifieurs utilisés,(dans notre cas,

nous avons quatre classifieurs.) Nous avons C^L combinaisons possibles, la quantité C^L représente le nombre de colonnes des tables d'index (comme nous avons *10* classe et *4* classifieurs alors notre table d'index comporte *10000* colonnes).

Nous avons développé un algorithme pour la création de ces unités.

Après cette étape de création d'unités (combinaisons possibles ou index), on doit pour chaque combinaison des résultats des classifieurs estimer la classe la plus probable on se basant sur les données d'apprentissage, pour réaliser cela, nous avons créé un autre tableau avec le même nombre de colonnes que le tableau d'index, ce tableau doit contenir dans chaque colonne notée j à la ligne i le nombre d'échantillons de la classe étiquetée i dont la combinaison se trouve dans la colonne notée j du tableau d'index. Nous avons rempli ce tableau grâce aux données d'apprentissage qui se trouvent dans la matrice Z.

Pour estimer la classe d'une image inconnue on prend la combinaison du résultat des classifieurs en cherchant la position (numéro de colonne) dans le tableau d'unités (Unité Focale), dans cette position, dans le tableau d'apprentissage, on vérifie si le vecteur n'est pas nul alors l'indice de la valeur maximale représente la classe la plus probable sinon l'image est rejetée.

---

function [(c),bel] = BKS(D,unts,ResApp)

$FU = focalunite(D)$; ▷ recherche l'unité focale

$pos = chercherPos(unts, FU)$; ▷ rechercher la position de l'unité focale dans le tableau d'unités

if (sum(ResApp(:,pos))>0) **then** ▷ si le vecteur n'est pas nul

$[bel, c] = max(ResApp(:, pos))$; ▷ retourne l'indice comme la classe la plus probable est la valeur comme la croyance de l'estimation.

end if

end function

Algorithm 6 Méthode d'estimation de la classe la plus probable

---

Résultat obtenu :

Nous avons réalisé la combinaison des quatre classifieurs, nous avons obtenu le résultat illustré dans le tableau ci-dessous :

Classifieurs combinés	**Résultat**
Net-LM-5 (93.08%)	91,6667 %
Net-FT-1 (97.08%)	
SVM-LM (95.00%)	
SVM-FT (96.58%)	

Tableau 5.7 Résultat de la combinaison des classifieurs en utilisant la méthode BKS

Analyse du résultat obtenu

La combinaison donne un résultat de *91.6667 %*

Le résultat obtenu n'est pas satisfaisant, après l'avoir analysé, nous avons remarqué que le tableau d'apprentissage contient *9945* vecteurs nuls ! donc *99.45 %* des combinaisons seront rejetées. Cela est dû au fait que notre base de données est très limitée.

Solution proposée

Nous proposons d'augmenter la taille de la base de données d'apprentissage, nous avons importé une nouvelle base de données plus volumineuse qui comporte *6048* éléments pour l'apprentissage et *1512* éléments pour le test.

Après avoir refait toutes les étapes d'apprentissage en utilisant les quatre classifieurs, nous avons obtenus les résultats suivants :

Type de classifieur	**Descripteur utilisé**	**Résultat de réapprentissage**
NN	LM	91.93 %
NN	FT	91.40 %
SVM	LM	97.35 %
SVM	FT	97.49 %

Tableau 5.8 Résultats de réapprentissage des quatre classifieurs

La combinaison donne un résultat de *93.9153 %*.

Ce résultat reste encore insatisfaisant, nous pouvons conclure que le problème ne se pose pas au niveau de l'augmentation de la taille de la base de données d'apprentissage car *234241* combinaisons parmi *234256* sont nulles alors, autrement dit, *99,9934 %* des éléments ont été rejetés.

Si ce problème n'est pas dû à la taille de la base de données, alors nous ne pouvons pas rejeter ce nombre très important de combinaisons mais nous devons par contre essayer de les reconnaitre par différentes autres manières, soit en se basant sur les résultats du meilleur classifieur ou alors en passant par d'autre méthodes de combinaison.

Solution 1 : En basant sur le meilleur classifieur

Pour pouvoir reconnaitre les éléments déjà rejeté, nous nous somme basée sur le résultat du meilleur classifieur à savoir *NN-FT*, grâce à cette méthode nous avons pu obtenir un résultat de combinaison égal à *97,25 %*

Solution 1 : En basant sur d'autres méthodes de combinaison

Nous avons utilisé trois méthodes de combinaison comme le montre le tableau suivant

Méthode utilisée	**Résultat de la combinaison**
Méthode Vote majoritaire	97.0000 %
Méthode Borda	97.5833 %
Méthode DS	98.0833 %

Tableau 5.9 Résultats de combinaison en utilisant d'autres méthodes de combinaison

2. **Méthode DS**

Cette méthode se base sur DT (voir, sect. 3.3.4.6) les DT représentent la moyenne des DP donnés par les échantillons des données d'apprentissage qui sont représentées dans la matrice Z, le traitement des entrées inconnues se fait via les calculs des fonctions (ϕ et beleif,) déjà présentées (voir en détails, sect. 3.3.4).

---

function [(c),bel]=DS(D,DT)

$[L, C] = size(D)$; ▷ On calcule la proximité

$p = phi(DT, D)$; ▷ le degré de croyance

$b = belief(p)$; ▷ Le vecteur de sortie final

$res = resultat(b, 100)$;

$[bel, c] = max(res)$;

end function

Algorithm 7 Méthode d'estimation de la classe la plus probable

---

Résultats obtenus :

Avec cette méthode, nous avons essayé de combiner les différents classifieurs obtenus auparavant, les résultats en détail sont présentés dans le tableau ci-dessous :

Classifieurs combinés	Résultat	Classifieurs combinés	Résultat
Net-LM-5 (93.08%) Net-FT-1 (97.08%) SVM-LM (95.00%) SVM-FT (96.58%)	98.0833 %	Net-LM-3 (90,75%) Net-FT-1 (97.08%) SVM-LM (95.00%) SVM-FT (96.58%)	98.0000 %
Net-LM-1 (92.00%) Net-FT-1 (97.08%) SVM-LM (95.00%) SVM-FT (96.58%)	98.0833	Net-LM-4 (93.17%) Net-FT-1 (97.08%) SVM-LM (95.00%) SVM-FT (96.58%)	98.0833% %
Net-LM-2 (91,42%) Net-FT-1 (97.08%) SVM-LM (95.00%) SVM-FT (96.58%)	97.9167 %	Net-LM-5 (93,94%) Net-FT-2 (96,17%) SVM-LM (95.00%) SVM-FT (96.58%)	97.9167%

Tableau 5.10 Résultat de la combinaison des classifieurs en utilisant la méthode DS

Analyse des résultats obtenus Le meilleur résultat obtenu est égal à *98,0833 %* qui est aussi le meilleur résultat parmi toutes les méthodes de combinaison de classifieurs implémentées.

5.3.4 Présentation de l'application 'Khattii'

'Khattii' est une application web dédiée à la reconnaissance du manuscrit arabe, elle est disponible en langue arabe et en langue française. Elle a été conçue en utilisant les deux langages de programmation principaux à savoir Matlab et PHP pour les traitements qui se font en arrière plan : les scripts Matlab ont permis le traitement de l'image et les codes PHP ont servi dans ce contexte à les exécuter.

L'interface graphique de l'application a été réalisée en HTML5, CSS et JavaScript et aussi en incluant les framework *JQuery* et *BootStrap*.

L'échange de données entre les différents langages s'est fait en utilisant les structures de données Extensible Markup Language (XML)

Le principal service offert par *Khattii* est la reconnaissance de l'écriture arabe manuscrite qui se fait soit en insérant des images numériques ou en écrivant le mot désiré avec la souris.

Ce service peut être utilisé suivant deux différentes méthodes :

5.3.4.1 Méthode simple

Ne requiert aucune connaissance du domaine de reconnaissance du manuscrit auprès de l'utilisateur, il lui suffit juste d'insérer l'écriture manuscrite pour en extraire le texte numérique.

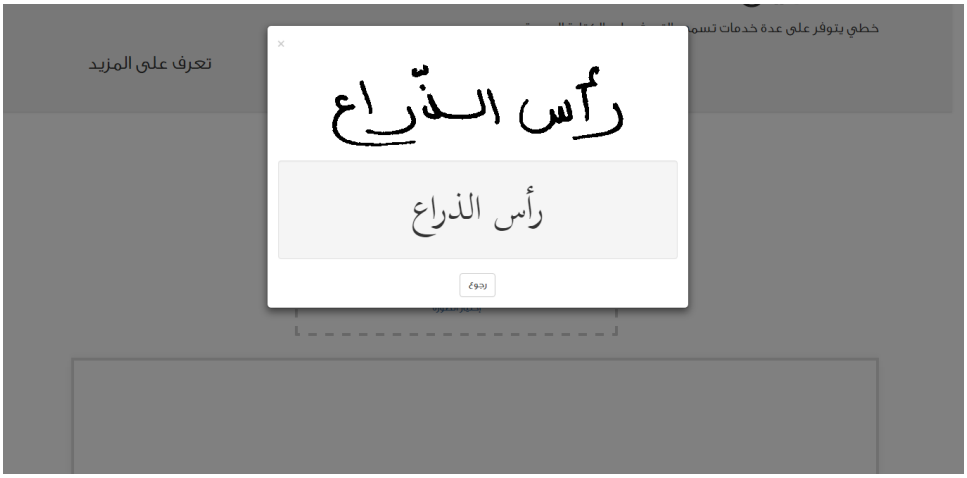


Figure 5.1 Reconnaissance via la méthode simple de l'application 'Khattii' -version arabe-

figure 2

5.3.5 Méthode avancée

En utilisant cette méthode, l'utilisateur a la possibilité d'obtenir des résultats de reconnaissance plus diversifiés, il y a en effet quatre classifieurs utilisée et la combinaison se fait suivant quatre méthodes à savoir : Vote majoritaire, Borda, BKS et DS, cela permet de comparer les résultats de chaque méthode.

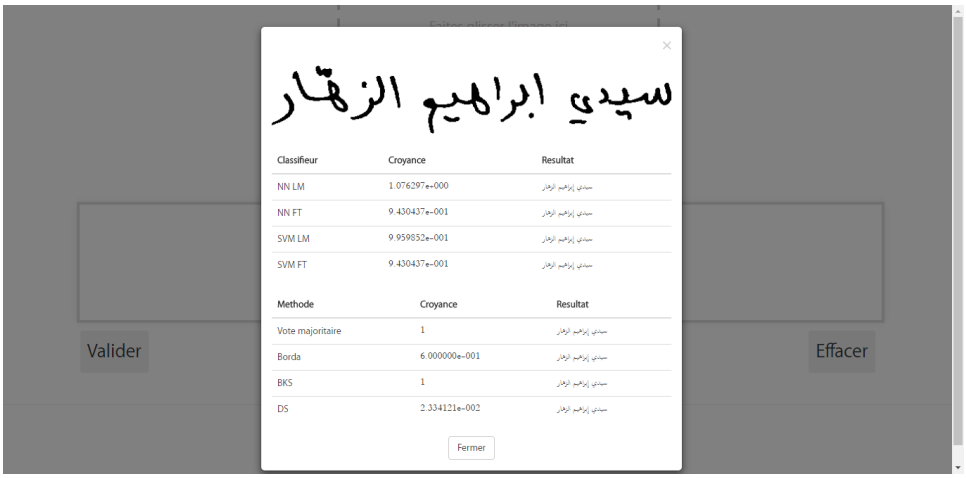


Figure 5.2 Reconnaissance via la méthode avancée de l'application 'Khattii' -version française-

5.3.6 Principales interfaces de l'application 'Khattii'

Figure 5.3 Interface d'accueil de l'application 'Khattii' -version arabe-

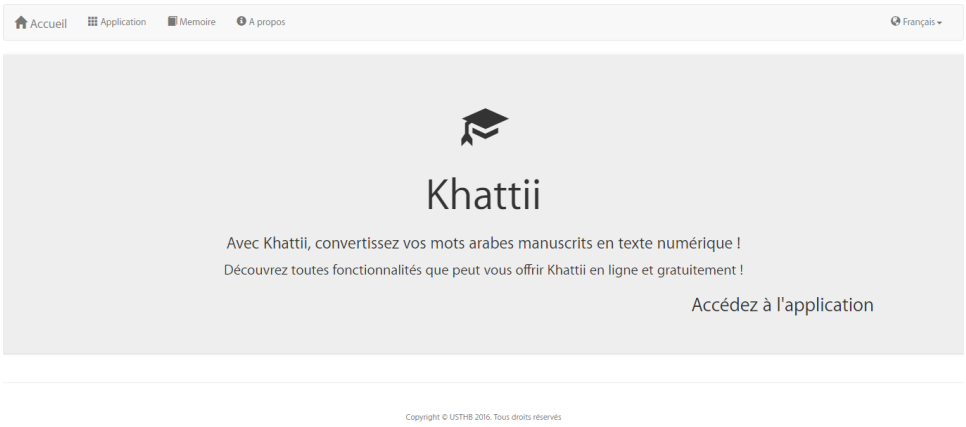


Figure 5.4 Interface d'accueil de l'application 'Khattii' -version française-

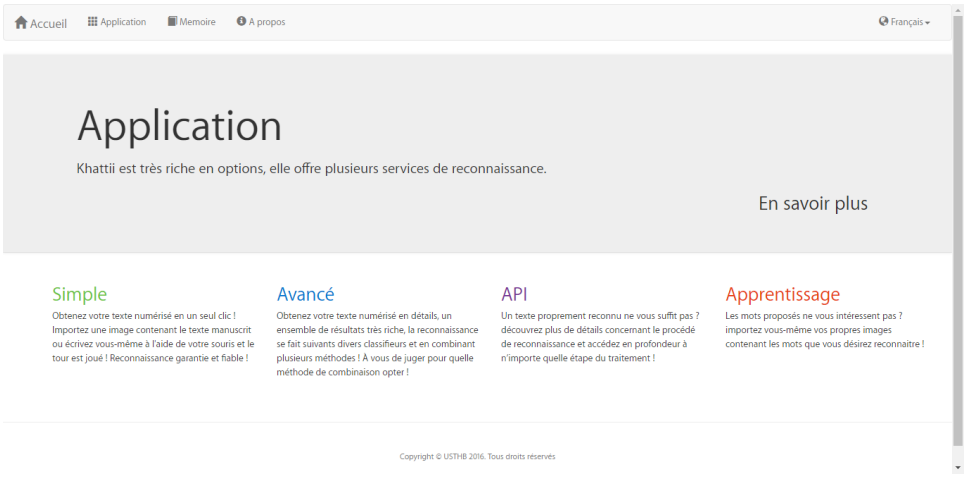


Figure 5.5 Tous les services offerts par l'application 'Khattii' -version française-

5.4 Conclusion

Nous avons à travers les pages de ce chapitre expliqué en détails la procédure de l'implémentation de notre système en passant par la présentation des divers outils utilisées à la réalisation.

L'implémentation de notre système de reconnaissance nous a permis de constater l'importance de la combinaison de classifieurs, en effet elle permet d'améliorer la reconnaissance.

CONCLUSION GÉNÉRALE

Aux termes de ce projet, nous avons été aptes à mettre en œuvre un système de reconnaissance d'écriture manuscrite arabe, « Une application web dénommée 'Khattii' ».

Au cours de cette année d'études nous sommes entrés dans différents mondes aussi attirants et profonds les uns que les autres.

Le premier monde était celui du traitement d'images où nous avons appris ce qu'est l'image pour l'ordinateur, comment la traiter pour tirer les informations les plus pertinentes en vue de la reconnaissance des formes à l'intérieur de ces images. Nous avons compris la différence entre caractéristiques de type structurel et statistique où nous entrons à l'intérieur de la forme pour calculer des contours, estimer des distances par rapport au centre de gravite etc. et les caractéristiques globales où on prend l'image en entier pour la décomposer dans une base polynomiale pour calculer les moments de Legendre.

Le deuxième monde était celui du 'Machine Learning' où nous avons appris comment faire apprendre à la machine à partir de données la prise de décisions de classification sur des entrées inconnues. Les machines que nous avons maitrisées sont les réseaux de neurones multicouches NN et les machines à vecteurs de support : SVM. Bien que ces deux machines assurent l'apprentissage supervisé, elles ont des techniques de programmation et des stratégies de décision différentes.

Le troisième monde et qui n'est pas des moindres est celui de la combinaison des classifieurs. Une multitude de méthodes de combinaisons à différents niveaux existent dans la littérature. Nous avons touché aux méthodes au niveau décisionnel et nous avons implémenté quatre de ces méthodes à savoir : Vote majoritaire, Borda, Behavior Knowledge Space (BKS) et Dempster Shafer (DS). Bien que les deux premières méthodes soient du type intuitif et sont simples à implémenter les deux dernières nous ont demandé un approfondissement encore dans l'apprentissage pour en tirer d'autres informations propres à être utilisées dans la combinaison. Ces notions sont par exemple, les notions de DT (Decision Templates), DP (Decision Profiles) etc.

Le dernier monde que nous avons exploré est celui de la programmation web pour mettre au point une application que l'on peut atteindre à travers le net et qui de ce fait peut marcher même sur les nouveaux dispositifs actuels à savoir les smartphones.

Avant de parler de combinaison des classifieurs, il a fallu donc mettre au point les classifieurs eux-mêmes pour en tirer le meilleur sachant que cette étape nécessite le temps le plus long pour assurer une bonne paramétrisation de l'apprentissage. Les méthodes de combinaison ont permis des résultats satisfaisants en majorité. Comme attendu, la méthode vote majoritaire a donné des résultats comparables à ceux du meilleur classifieur puisque nous remarquons que les quatre classifieurs pris en compte sont de bonne qualité. Ils dépassent tous les 90% de taux de reconnaissance. Donc, il était attendu que quand ils s'entendent sur une classification, qu'elle soit la bonne.

La méthode Borda améliore le résultat des classifieurs pris individuellement. Ce résultat était aussi attendu puisque dans cette méthode les classifieurs ne sont pas tranchants dans leurs décisions offrants des classements pour les différentes classes. Cette contribution de tous les classifieurs avec le système de notation a permis d'améliorer les résultats.

Notre surprise a été dans les résultats obtenus par la BKS. Bien que cette dernière se base aussi sur l'apprentissage qui devrait améliorer les résultats des classifieurs pris individuellement, elle a donné des résultats assez médiocres. Au départ, nous avons pensé à un problème dû à la taille de la base de données. Nous l'avons testé sur une base de données deux fois plus grande le résultat est resté dans le même ordre, ce qui nous amène à penser que le problème est plutôt dû au fait, qu'à chaque fois, on peut trouver dans le test des situations non prévues dans l'apprentissage. Une solution possible serait l'utilisation combinée de cette méthode avec une validation croisée du type 'K-Fold Cross-Validation' qui permettrait d'alterner à chaque fois les ensembles de test et d'apprentissage assurant ainsi la prise en compte de toutes les situations possibles.

La méthode la plus réussie est la Dempster-Shafer DS qui a donné le meilleur résultat parmi toutes les méthodes implémentées.

Avec ça, nous pouvons dire que nous avons donc pu concrétiser les objectifs de ce projet. Toutefois, comme dans tout travail humain, plusieurs améliorations peuvent être faites. Une d'entre elles serait de combiner la BKS avec la cross-validation.

RÉFÉRENCES

[1] M. FAKIR et B. MINAOUI R. EL AYACHI, M. OUJAOURA. Code braille et la reconnaissance d'un document écrit en tifinagh.

[2] Andrew W. Moore.

[3] Leila CHERGUI. Combinaison de classifieurs pour la reconnaissance de mots arabes manuscrits.

[4] R. Mukundan Chee-Way Chong, P. Raveendran. Translation and scale invariants of legendre moments, 2003.

[5] Decision templates for multiple classifier fusion : An experimental comparison, 2001.

[6] Venu Govindaraju1 Sergey Tulyakov, Stefan Jaeger and David Doermann. Review of classifier combination methods.

[7] Exemple méthode borda. `https://fr.wikipedia.org/wiki/M%C3%A9thode_Borda`.

[8] Thierry Paquet Yousri Kessentini, Thomas Burger. Combination of multiple classifiers using local accuracy estimates, 1997.

[9] Kevin Woods. A dempster–shafer theory based combination of handwriting recognition systems with multiple rejection strategies, 2014.

[10] Gérard DREYFUS. Reconnaissance de forme par réseau de neurone (1. fonctionnement général d'un réseau de neurone), 2002.

[11] Martin T. Hagan. Neural network design.

[12] Hassiba Nemmour and Youcef Chibani. Handwritten arabic word recognition based on ridgelet transform and support vector machines.

[13] Tristan Fletcher. Support vector machines explained.

[14] Jitendra Kumar Gita Sinha. Arabic numeral recognition using svm classifier, 2013.

[15] Howard B. Demuth Mark Hudson, Martin T. Hagan. Matlab neural network toolbox tm.

[16] MATLAB/OCTAVE interface of LIBSVM. Fichier 'lisez-moi' le plus important de libsvm. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.