Reconnaissance de chiffres
Compte-rendu de TP - IN54

Adrien Berthet et Camille Mougin

UTBM - Automne 2014

Introduction

L'objectif de ce TP est de réaliser un système complet de reconnaissance de chiffres manuscrits. Chaque chiffre sera d'abord isolé du reste, pour ensuite être analysé et classé par deux systèmes de classifications différentes. Ces classifieurs permettront d'obtenir une probabilité quant à la nature exacte du chiffre, et ces probabilités seront donc assemblées pour obtenir un résultat final.
Pour obtenir une probabilité, le système effectuera d'abord une séquence d'apprentissage, qui lui permettra d'avoir une base de données de comparaison.
Le schéma suivant donne un déroulement simple des différentes étapes permettant d'atteindre le résultat final.

1 Pré-traitements : localisation et extraction des chiffres manuscrits

Dans l'exemple étudié, la façon la plus simple de localiser les chiffres consiste d'abord à rechercher, à partir de l'histogramme de projections horizontales de pixels noirs, les plages correspondant à un nombre de pixels noirs non nul. Le début et la fin de chaque plage détectée sur l'histogramme des projections horizontales définissent une ligne de chiffres dans l'image. En appliquant le même principe (mais à partir d'un histogramme de projections verticales) sur chaque ligne détectée, on peut alors déterminer l'emplacement de chaque chiffre de l'image.

1.1 Recherche des lignes

Le principe est très simple, puisqu'il suffit dans un premier temps de parcourir chaque ligne pour vérifier qu'elle contient un pixel noir ou non. Pour accélérer le processus et parcourir rapidement l'image, un histogramme du nombre de pixels noirs est récupéré. Ceci est possible facilement car l'image est déjà binaire. On peut alors remarquer de façon évidente les différentes lignes. Les valeurs de celles-ci (début et fin) sont alors sauvegardées pour être réutilisées lors de la recherche des colonnes.

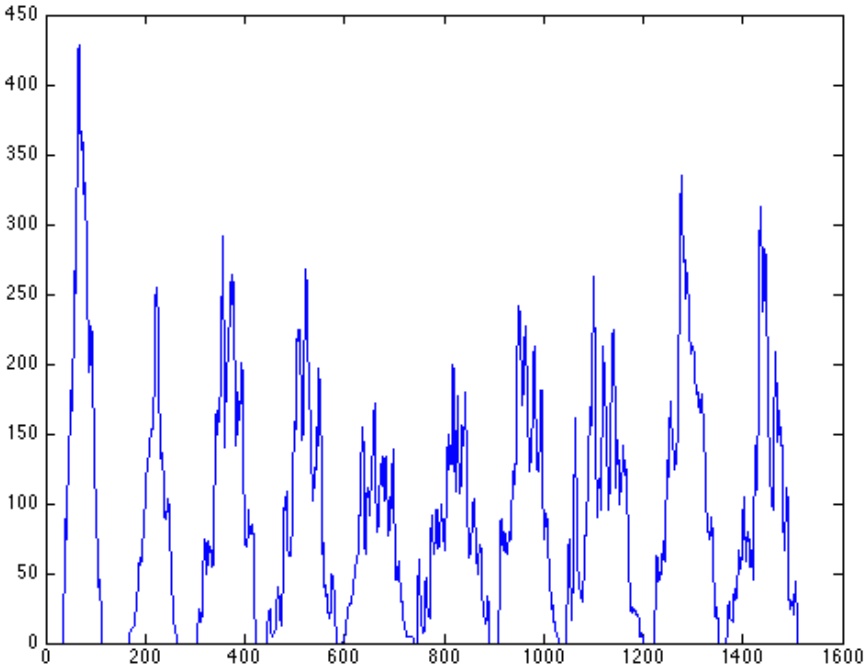

Figure 1 – Histogramme du niveau de noir

1.2 Recherche des colonnes pour chaque ligne

Le même principe utilisé précédemment est recommencé, avec cette fois un histogramme du niveau de noir sur chaque colonne, pour une ligne unique (soit une ligne de chiffre). On obtient alors plusieurs histogrammes semblables au précédent, qui permettent d'obtenir un tableau avec les colonnes englobant chaque chiffre pour chaque ligne.

1.3 Détermination du rectangle englobant

On pourrait penser qu'il n'y a plus d'étapes de découpe et qu'il faut uniquement assembler les coordonnées colonnes avec celles des lignes, mais il est nécessaire de repasser pour chaque chiffre dans le découpage des lignes, car ils possèdent un profil vertical beaucoup trop grand pour certains (puisqu'il s'agit du plus haut et plus bas chiffre pour chaque ligne). Ainsi, en effectuant une découpe plus précise, on obtient au final le résultat suivant.

1.4 Avantages et inconvénients d'une telle méthode

Cette méthode de localisation par projection est avantageuse puisque dans le cas ci présent, elle fournit un résultat parfait quant à la découpe des chiffres. Seulement, celle-ci peut perdre énormément en pertinence si les lignes de chiffres n'étaient pas aussi droites.
Imaginons deux lignes consécutives, écrites de façon à ce qu'elles montent toutes les deux vers le haut. On se retrouve avec le premier chiffre de la première ligne qui se retrouve à la même hauteur que le dernier chiffre de la deuxième ligne. Ainsi, la méthode de l'histogramme ne convient pas du tout car on ne pourrait faire la différence

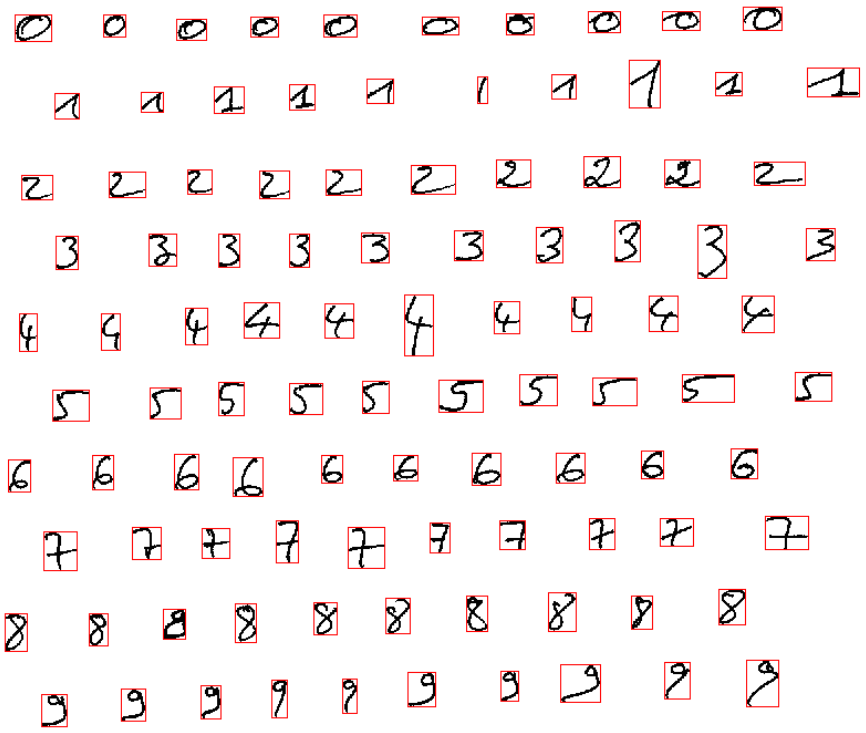

Figure 2 – Résultat des découpes

entre les deux lignes.
On peut également remarquer que cette méthode devient vite lourde pour les images de grande taille et comprenant un nombre important de caractères (en plus de chiffres), et il serait donc pertinent de trouver une méthode plus optimisée.

2 Classifieur 1 : profils et classifieur par distance euclidienne minimum

Listes des fonctions mentionnées dans cette section : *extractprofile*, *computepdistances*, *learningclassifier1*, *decisionclassifier1*.

2.1 Principe et implémentation

Ce premier classifieur vise à déterminer un chiffre en fonction de ses profils gauche et droit. La première étape consiste à extraire ces profils du chiffre, puis de décider de la nature de la classe de ces profils, grâce à la distance euclidienne minimum.
Pour pouvoir effectuer un classement des profils, il est nécessaire de passer par une phase d'apprentissage, afin d'obtenir une base de profils de référence. Il est alors possible de comparer les profils de référence et ceux récupérés sur le chiffre souhaitant être lu.

2.1.1 Fonctions utilisées

extractprofile(base, d) : profile

Entrées : **base** est l'image du rectangle contenant le chiffre à lire, grâce au prétraitement effectué. **d** est le nombre de composantes d'un profil
Sortie : **profile** est le vecteur contenant les deux profils (gauche et droit), de dimension $d \times 2$

Pour un chiffre donné, on découpe dans celui-ci d lignes équidistantes les unes des autres, sur lesquelles les profils vont être lus. Pour chaque ligne, son profil gauche ainsi que son droit sont alors relevés. Le profil est alors normalisé, pour qu'il n'y ait aucune incohérence entre chaque chiffre et la base de référence.

computepdistances(centerslearning, vectordistance) : pbelonging

Entrées : **conterslearning** contient les centres d'une classe, obtenus lors de la phase d'apprentissage. **vectordistance** contient les centres de la classe à tester.
Sortie : **pbelonging** contient les probabilités d'appartenance de la classe à tester dans la classe d'apprentissage

Il s'agit ici de calculer la probabilité pour une classe précise, suivant la formule suivante :

$$p(\omega_i/x) = \frac{exp(-d(x,\omega_i)}{\sum_{j=0}^{9} exp(-d(x,\omega_j))}$$

2.1.2 Phase d'apprentissage

learningclassifier1(rectangleslearning, learningimage, d) : vectordistancelearning

Entrées : **rectangleslearning** correspond aux rectangles déterminés précédemment permettant d'isoler chaque chiffre sur l'image d'apprentissage. **learningimage** est l'image d'apprentissage, et **d** le nombre de composantes pour un profil

Sortie : **vectordistancelearning** contient l'ensemble des centres des profils pour chaque chiffre

Grâce à *extractprofile*, un profil est déterminé pour chaque chiffre. Les vecteurs déterminant les profils récupérés, la moyenne des centres pour chaque classe est calculé, à l'aide de *mean*. Ces résultats sont sauvegardés pour faire office de base de référence.

2.1.3 Phase de décision

decisionclassifier1(rectangles, image, vectordistancelearning, d) : pbelonging

Entrées : **rectangles** correspond aux rectangles déterminés précédemment permettant d'isoler chaque chiffre sur l'image à lire. **image** est l'image à lire. **vectordistancelearning** est la base de référence déterminée dans la phase d'apprentissage. **d** est le nombre de composantes pour un profil

Sortie : **pbelonging** contient les probabilités d'appartenance de chaque chiffre par rapport à l'ensemble des classes d'apprentissage

Le fonctionnement est semblable à la phase d'apprentissage. Il y a d'abord une extraction du profil pour chaque chiffre, puis la probabilité d'appartenance à chaque classe en fonction de ses centres est calculée grâce à *computepdistances*.

2.2 Analyse et conclusion

2.2.1 Evaluation des performances

La figure **??** indique la probabilité, pour chaque chiffre lu, d'appartenir à telle classe de chiffre. De gauche à droite, les classes vont de 0 à 9. Une probabilité élevée indique une plus forte chance pour un chiffre de se trouver dans cette classe.

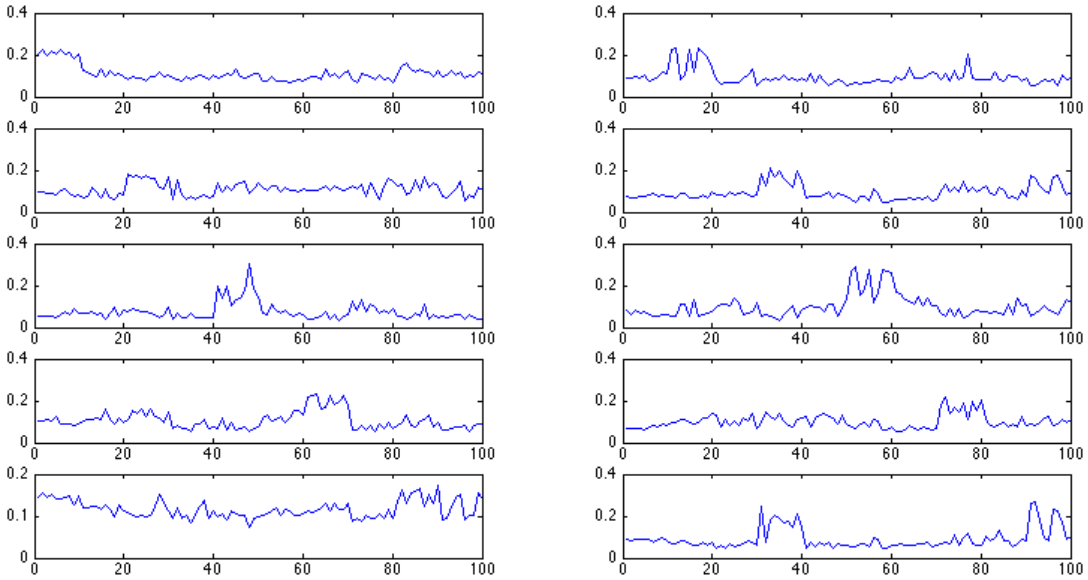

FIGURE 3 – Probabilité de classe pour chaque chiffre

On peut remarquer que le classifieur repère des tendances dans les chiffres analysés. Si l'on prend par exemple la classe 0, un pic se forme là où il y a bien des 0 sur l'échantillon à analyser (figure **??**, gauche). Cependant, sur certains chiffres, comme la classe 2 (figure **??**, droite), il est difficile d'obtenir un résultat convenable partout. Ici, l'analyse des chiffres 8 (entre 80 et 90) amène à penser qu'il peut s'agir de la classe 2, car les probabilités sont aussi hautes que lorsqu'il s'agissait de la classe 2 elle-même.

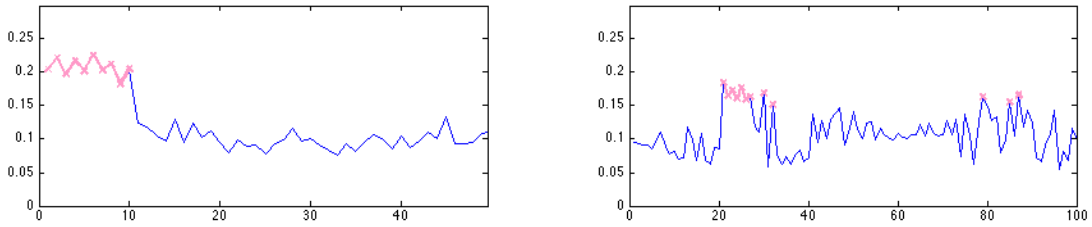

FIGURE 4 – Probabilité pour les classes 0 et 2

Cette méthode aide donc à obtenir un premier classement, mais n'est pas tout le temps précise. Elle nécessite l'apport d'un classifieur supplémentaire.

2.2.2 Variation de la valeur d

Le paramètre qu'il est possible de varier ici est d, qui correspond à la taille du vecteur contenant les profils gauches et droits d'un chiffre. Ainsi, pour une valeur plus élevée de d, environ 10 ou plus, ce premier classifieur permet d'obtenir des résultats à 80% satisfaisants. Par contre, des valeurs moins importantes que 5 ne permettent pas d'avoir un classifieur correct, puisque le taux de confiance se situe en-dessous de 60%.

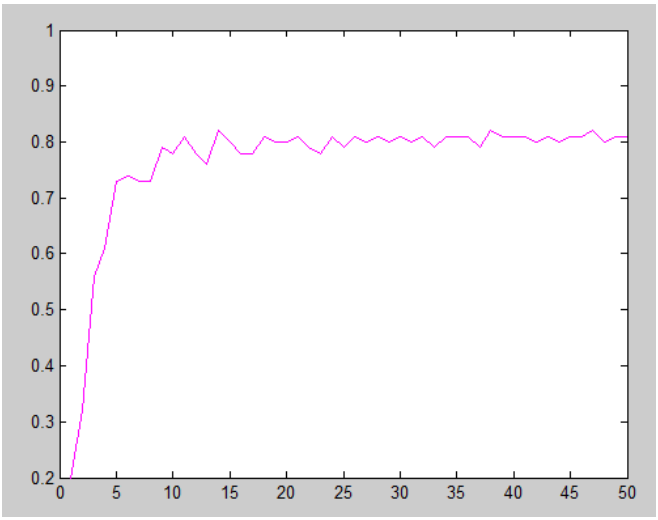

FIGURE 5 – Influence de d

Il est donc raisonnable de posséder un nombre de mesure par profil au delà d'un seuil, à savoir ici 10 ou plus. La figure **??** fournit des résultats plus contrastés que

ceux relevés dans la figure **??**, pour $d = 15$. Pour certains chiffres, il y a même une certitude de quasiment 100% qu'il s'agit de ceux-ci.

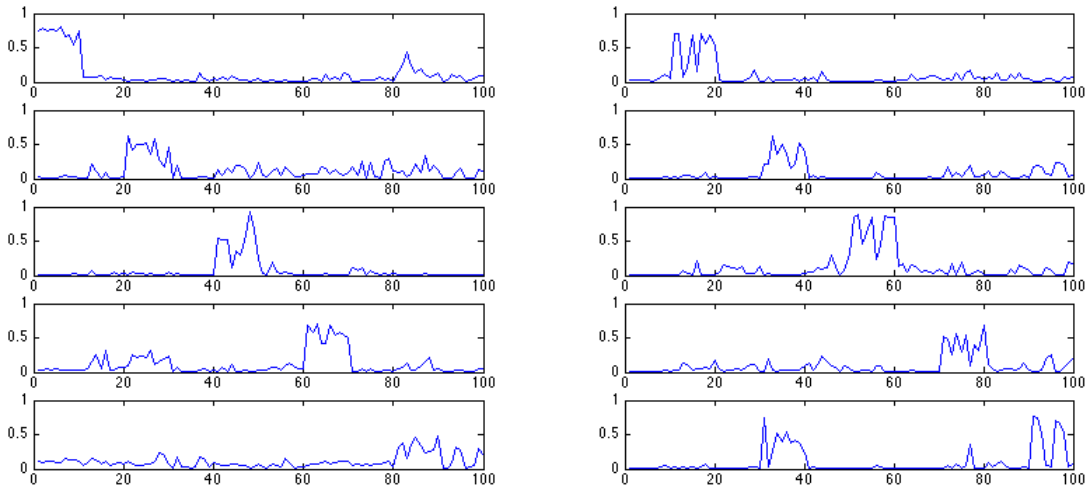

FIGURE 6 – Probabilité de classe pour $d = 15$

3 Classifieur 2 : Densités et K plus proches voisins

Liste des fonctions mentionnées dans cette section : *getdensity*, *computepdensities*, *learningclassifier2*, *decisionclassifier2*

3.1 Principe et implémentation

Pour compléter les données obtenues, on utilise donc une seconde méthode de classification. L'identification du chiffre est à présent réalisée à partir de la répartition des pixels noirs au sein de zones décomposant le rectangle de base. La première étape consiste donc à diviser en $m \times n$ zones le rectangle encadrant le chiffre à identifier. Il suffit ensuite de calculer la densité de pixels noirs dans chacune de ces zones.

Comme précédemment, le classifieur nécessite de passer par une phase d'apprentissage afin d'obtenir une base de densités de référence. Il est ensuite possible d'identifier le chiffre en comparant les densités obtenues avec les densités de référence. Les probabilités d'appartenance du chiffre à chacune des classes est finalement calculée en fonction du nombre de représentants de chaque classe parmis ses k plus proches voisins.

3.1.1 Fonctions utilisées

getdensity(rectangle, m, n) : density

Entrées : base, coordonnées du rectangle contenant le chiffre à lire, obtenues grâce au pré-traitement.
m, nombre de zones qui découpent le rectange sur la hauteur.
n, nombre de zones qui découpent le rectangle sur la largeur.
Sortie : density, vecteur contenant les densités normalisées relatives à chaque zone, de dimension *mn*

Calcule la densité de pixels noirs dans chaque zone de l'image. Les zones sont obtenues par division du rectangle contenant le chiffre en m parties sur la hauteur et n parties sur la largeur. Les densités sont calculées de la manière suivante :

$$density_i j = \frac{nbpixelsnoirs_i j}{hauteurrectangle \times largeurrectangle}$$

computepdensities(vectordensity, vectordensitylearning, nbrectangleslearning, k) : pbelonging

Entrées : vectordensity, vecteur de densité d'un objet à identifier.
vectordensitylearning, résultat de la phase d'apprentissage.
nbrectangleslearning, nombre d'objets contenus dans la base d'apprentissage.
k, paramètre k.
Sortie : pbelonging, vecteur de probabilités d'appartenances de dimension 10, soit le nombre de classes d'appartenance.

Calcule la différence entre le vecteur de densités de l'objet à identifier et les vecteur de densités de chaque élément contenu dans la base d'apprentissage respectivement. On obtient un vecteur de distances de dimension *taille de la base d'apprentissage*. On se place ensuite dans un espace 1d abstrait où apparaît l'objet à identifier : chaque élément de la base d'apprentissage est disposé dans cet espace en fonction de sa distance à l'objet calculée précédemment. On sélectionne enfin les k plus proches voisins de l'objet à identifier et détermine les probabilités d'appartenance de la manière suivante :

$$pbelonging1(\omega_i/x) = \frac{k_i}{k}$$

avec x l'objet testé pour la classe d'appartenance ω_i et k_i le nombre de voisins appartenant à ω_i parmi les k plus proches

3.1.2 Phase d'apprentissage

learningclassifier2(rectangleslearning, learningimage, m, n) : vectordensitylearning

Entrées : rectangleslearning, coordonnées des rectangles contenant chacun des chiffres connus de la base d'apprentissage.
learningimage, image d'apprentissage.
m, n, nombre de zones à découper dans un rectangle respectivement en hauteur en en largeur.

Sortie : vectordensitylearning, matrice regroupant les vecteurs de densités propres à chaque objet de la base d'apprentissage, de dimension $taillebaseapprentissage \times mn$

La phase d'apprentissage a pour objectif de calculer les vecteurs de densités de tous les objets de la base d'apprentissage afin de constituer une base de référence pour identifier les éléments à traiter. On obtient en sortie les vecteurs de densités de l'ensemble des composants de la base d'aprentissage qui sera utilisé lors de la phase de décision.

3.1.3 Phase de décision

decisionclassifier2(rectangles, image, vectordensitylearning, m, n, k) : pbelonging

Entrées : rectangles, coordonnées des rectangles contenant chacun des chiffres à identifier.
image image à traiter.
m, n, nombre de zones à découper dans un rectangle respectivement en hauteur en en largeur.

Sortie : pbelonging, matrice de probabilités d'appartenance aux dix classes pour chaque objet à identifier, de dimension $s \times 10$, avec s le nombre d'objets à identifier et 10 le nombre de classes d'appartenance.

La phase de décision est divisée en deux étapes. D'abord, le calcul de densités est effectué sur chacun des objets à identifier, à l'image de ce qui a été réalisé en phase d'apprentissage. Une fois les vecteurs de densités obtenus, les probabilités d'appartenance de chaque objet sont calculées à l'aide de la fonction *computepdensities*.

3.2 Analyse et conclusion

3.2.1 Résultats obtenus

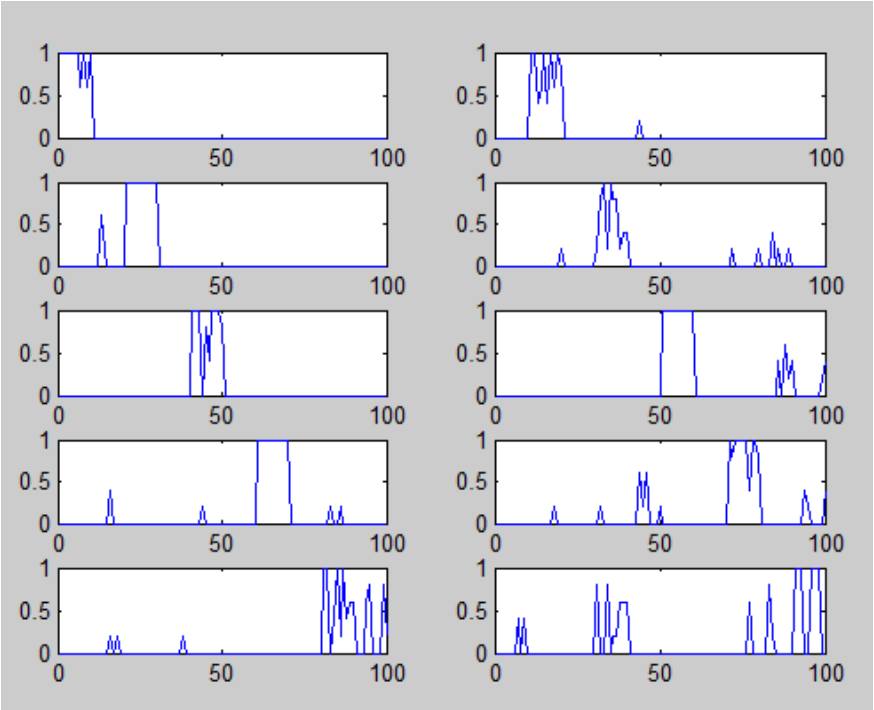

Figure 7 – Probabilité d'appartenance à une classe pour chaque objet

On observe ici les probabilités d'appartenance de chacune des dix classes en fonction des objets x traités. Les paramètres utilisés sont les suivants : $k = 5$, $m = 5$ et $n = 5$. On se rapproche ici d'une fonction binaire de probabilité 1 pour les objets appartenant à la classe et 0 pour les autres, en particulier pour les classes 0, 2, 5, et 6 où les taux de reconnaissance sont particulièrement bons. On remarque également des confusions entre paires de classes : ah ouais lesquelles ?

3.2.2 Influence du paramètre k

L'influence de la valeur de k sur le taux de reconnaissance du classifieur 2 est très important. Comme le montre la figure ci-dessous, le meilleur taux de reconnaissance est atteint pour $k = 1$ et régresse rapidement quand k augmente. Ce résultat paraît surprenant : k définit le nombre de voisins à considérer pour le calcul de probabilité d'appartenance et on pourrait penser que considérer un nombre relativement important de voisins aurait un effet "stabilisant" sur le taux de reconnaissance. La taille réduite de la base d'apprentissage explique peut-être ce comportement.
On retient donc un taux maximal pour $k = 1$.

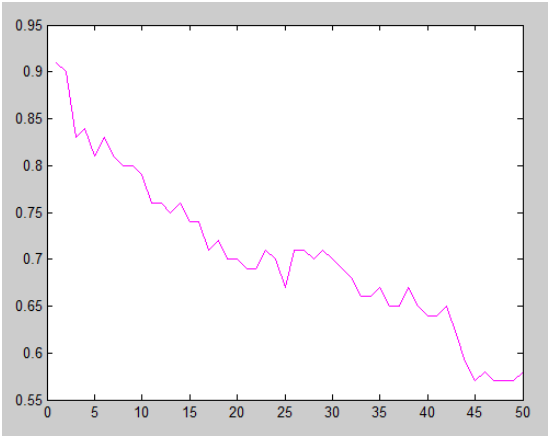

FIGURE 8 – Variation du taux de reconnaissance en fonction du paramètre k

3.2.3 Influence des paramètres m et n

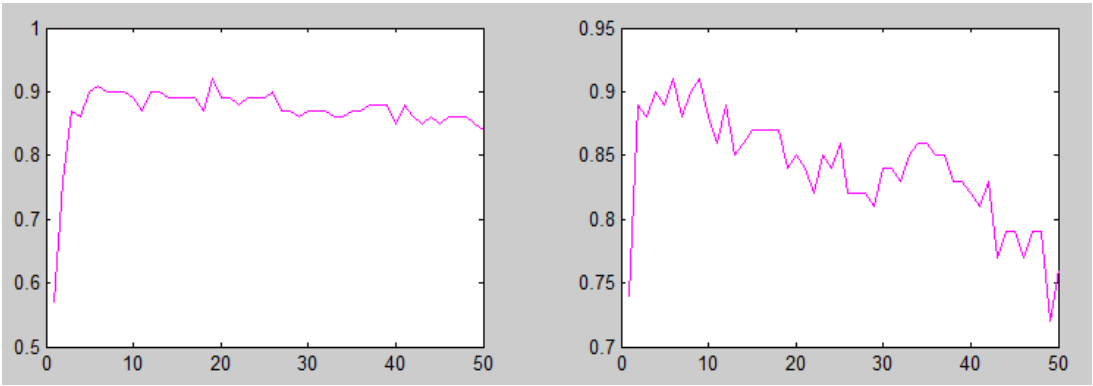

FIGURE 9 – Variation du taux de reconnaissance en fonction des paramètres m (à gauche) et n (à droite)

D'après la figure ci-dessus, les influences de m et de n sur les résultats du classifieur 2 sont limitées. Les taux en fonction de m restent en effet quasiment stable pour $5 < m < 50$, tandis qu'on observe une légère régression dans les taux en fonction de n pour $n > 5$. On repère un taux maximal pour respectivement $m = 19$ et $n = 7$.

3.2.4 Calcul des résultats avec optimisation des paramètres

Les taux de reconnaissance du classifieur 2 montrent l'amélioration des résultats faisant suite à l'optimisation des paramètres :

$A(k = 5, m = 5, n = 5)$: **taux de reconnaissance** $= 0.83$
$B(k = 5, m = 19, n = 7)$: **taux de reconnaissance** $= 0.83$
$C(k = 1, m = 5, n = 5)$: **taux de reconnaissance** $= 0.90$
$D(k = 1, m = 19, n = 7)$: **taux de reconnaissance** $= 0.90$

Résultat A : On utilise les paramètres de départs.
Résultat B : On optimise m et n sans changer k. Le taux de reconnaissance ne change pas.
Résultat C : On optimise k sans changer m et n. Le taux de reconnaissance augmente passe de 0,83 à 0,90.
Résultat D : On optimise k, m et n. Le taux reste inchangé à 0,90.

Les paramètres m et n ne semblent pas influer sur le taux de reconnaissance. Pour confirmer ce résultat, on cherche à représenter à nouveau la variation du taux de reconnaissance en fonction de m et n, avec cette fois m et n égaux et avec $k = 1$.

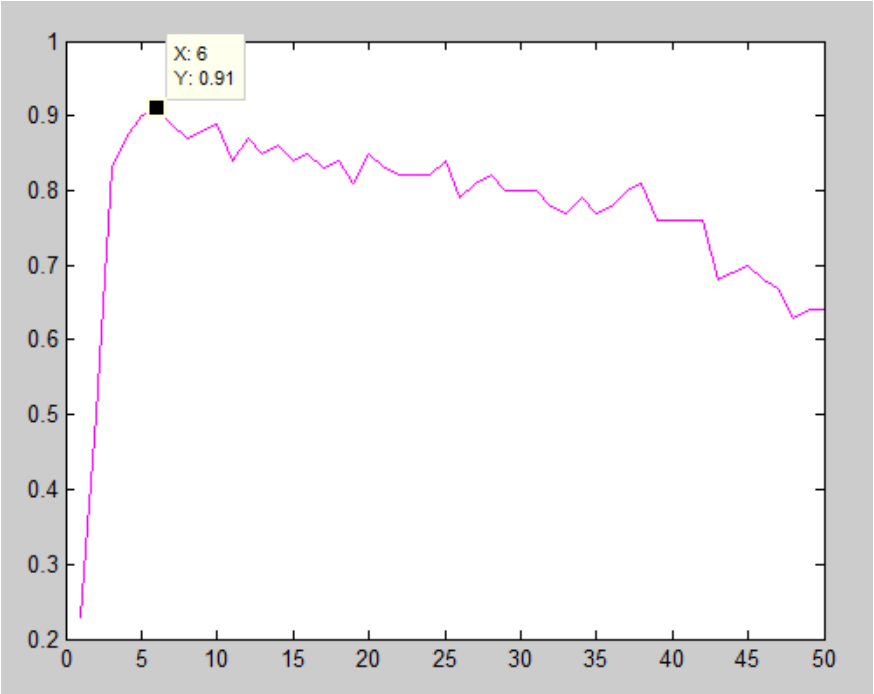

FIGURE 10 – Variation du taux de reconnaissance en fonction de $m = n$ avec $d = 14$ et $k = 1$ optimaux

On obtient un nouveau maximum en $m = n = 6$.
$E(k = 1, m = 6, n = 6)$ **: taux de reconnaissance** = 0.91

Résultat E : On change finalement les paramètres selon ce résultat et on obtient une légère progression du taux à 0,91.

4 Combinaison de classifieurs

4.1 Combinaison par somme et produit de probabilités

Même si le second classifieur affiche un taux de reconnaissance plus élevé que le premier, on peut imaginer que les résultats obtenus puissent être complémentaires. C'est pourquoi, afin d'améliorer le taux de reconnaissance global du système, il est possible de combiner les probabilités obtenues par chaque classifieur afin d'obtenir un nouveau vecteur de probabilité d'appartenance. On espère le résultat ainsi obtenu plus précis que ceux donnés par leur utilisation indépendante,.
Deux méthodes de combinaison sont utilisées : la somme et le produit.
Soient pbelonging1 et pbelonging2 les probabilités d'appartenance respectivement obtenus pour les classifieurs 1 et 2. Les nouvelles probabilités, obtenues par combinaison en utilisant la somme et le produit, sont données de la façon suivante :

$$\begin{aligned} pbelongingsum(x/\omega_i) &= pbelonging1(x/\omega_i) + pbelonging2(x/\omega_i) \\ pbelongingprod(x/\omega_i) &= pbelonging1(x/\omega_i) pbelonging2(x/\omega_i) \end{aligned}$$

avec x l'objet testé pour la classe d'appartenance ω_i

4.2 Critique des résultats obtenus

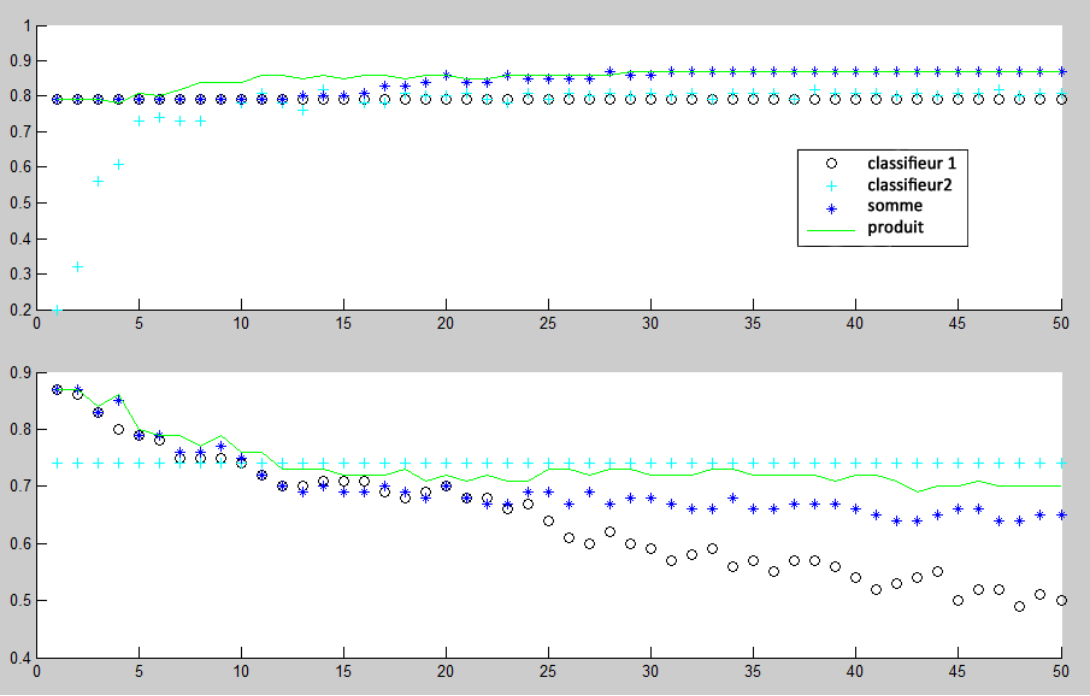

Figure 11 – Variation des quatre taux de reconnaissance obtenus en fonction des paramètres d (en haut) et k (en bas)

Globalement on observe de meilleurs résultats pour les combinaisons de classifieurs, représentés dans sur la figure ci-dessus en vert et en bleu foncé. Cependant, si l'on utilise les paramètres optimaux définis précédemment, les résultats sont biaisés : étant

donné le paramètre k fixé à 1, les probabilités résultant du second classifieur sont binaire, ce qui annule les résultats du premier classifieur dans le cas du produit mais aussi de la somme.

Conclusion

Les probabilités obtenues en fin de processus sont très correctes : 91% de reconnaissance en optimisant les paramètres et combinant les classifieurs.
Cependant, de meilleurs résultats pourraient être obtenus en agrandissant la base d'apprentissage : en effet, nous avons vu que la méthode des k plus proches voisins n'est pas utilisée à hauteur de son potentiel étant donné que les meilleurs résultats sont obtenus en considérant seulement le voisin le plus proche. Les probabilités obtenues sont donc très contrastées ce qui limite l'action de la combinaison des classifieurs.
D'autre part, il semble que les chiffres de la base d'apprentissage ainsi que ceux à analyser aient été rédigés par une même personne, ce qui limite la différence de forme entre les chiffres. Pour obtenir des résultats significatifs, il faudrait idéalement varier les styles d'écriture dans la base d'apprentissage, ce qui réduirait le taux de reconnaissance mais améliorerait les performances réelles du système.