Reconnaissance de caractères

Marin Costes
Jonathan Mboko
Manuel Mockus
Octave Mestoudjian

29 avril 2019

Table des matières

Chapitre 1

Introduction

Le cœur du projet est d'écrire un code permettant de reconnaître automatiquement des caractères manuscrits, le but étant de se familiariser avec des méthodes usuelles de reconnaissance de forme en les implémentant pour un cas particulier et en en comprenant la théorie. Nous utiliserons comme référence le travail de Lars Elden "Matrix Methods in Data Mining and Pattern Recognition [1]".

Les caractères à analyser seront ici des chiffres de 1 à 9 sous forme de vecteurs de taille $(28 \times 28, 1)$ à valeurs entières comprise entre 0 et 255 : chaque vecteur représente une image de taille 28×28 en nuance de gris ($0 =$ noir, blanc $= 255$). Nous disposons d'une base de données de 70000 chiffres manuscrits labellisés stockés sous le format décrit précédemment.

Nous avons choisi comme langage de programmation pour notre projet le Python.

Chapitre 2

Pré-traitement des données

La base de donnée étant un fichier matlab nous avons dans un premier temps dû la traduire dans un format utilisable sous python. Nous avons choisi comme format de stockage une liste de liste de vecteurs : le k^{eme} élément de cette liste est une liste des vecteurs correspondants au chiffre k. Nous avons ensuite divisé la base de donnée en quatre afin que chacun des membre du groupe puisse effectuer des tests indépendants.

Les algorithmes étudiés relèvent du domaine de l'apprentissage automatique : après une phase d'apprentissage où l'on donne à l'algorithme des images et leurs chiffres respectifs comme références, on a une phase de traitement où l'algorithme doit reconnaître le chiffre à partir de l'image seule. Pour avoir des résultats pertinents, il faut donc distinguer clairement les données utilisées pour l'apprentissage et celles utilisées pour le traitement. Le choix que nous avons fait est donc pour chaque base de donnée individuelle de diviser les données en 20% fixe de données utilisées exclusivement pour les tests et de prendre pour donnée d'apprentissage aléatoirement 50% des données restantes.

Chapitre 3

Première méthode : calcul des écarts à la moyenne

3.1 Théorie

La première méthode étudiée est une approche assez intuitive du problème. Dans un premier temps, à partir du jeu de données destiné à l'entraînement, pour chaque chiffre on crée une image moyenne de toutes les image du chiffre : le centre de gravité (centre de masse, isobarycentre) de tous les vecteurs de ce chiffre.
Une image inconnue est alors classée en cherchant l'image moyenne telle que l'écart de ressemblance entre l'image à tester et cette image soit minimum.

On définit l'écart de ressemblance entre deux images comme étant la distance dans une certaine norme choisie entre les deux vecteurs x et y de $\mathbf{R}^{784}$ correspondant aux images :

$$d(x, y) = \|x - y\|$$

Le choix de la norme et son influence sur les résultat est détaillé dans la partie pratique.

3.2 Pratique

Pour analyser l'efficacité de notre programme nous avons choisi d'observer deux types de données, le taux de réussite global de notre programme pour différentes normes, et le taux de réussite de notre programme pour chaque chiffre avec une norme fixe.
Nous avons comparé les taux de réussite de l'algorithme -pour chaque chiffre et en moyenne- pour les normes de Minkowski ($||v||_n = \sqrt[n]{\sum_k |v_k|^n}$) entre 1 et 10 et pour la norme infinie ($||v||_\infty = \max_k |v_k|$) (table 3.1).

TABLE 3.1 – Pourcentage de chiffres correctement identifiées par norme de Minkowsky

Norme	0	1	2	3	4	5	6	7	8	9	**Total**
1	84	99	41	66	70	35	80	77	35	75	**67**
2	89	97	74	82	85	72	87	83	74	77	**82**
3	90	90	82	81	84	81	88	81	84	77	**84**

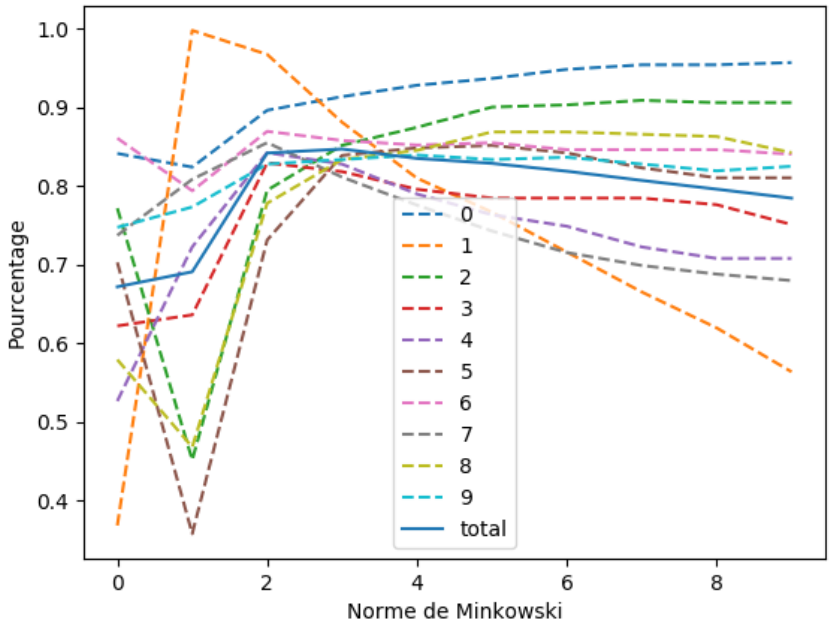

FIGURE 3.1 – Performance de l'algorithme pour chaque chiffre et en moyenne en fonction de la norme de Minkowsky (la valeur 0 en abscisse représente la norme infinie).

On remarque qu'en moyenne les normes 2 et 3 sont les plus performantes et aussi celles pour lequelles les résultats obtenus pour les différents chiffres sont les plus proches.
La distance associée à la norme infinie ne prenant en compte que l'écart maximal composante par composante entre deux vecteurs, appliquée à la comparaison d'image elle est inefficace car l'écart maximal n'est pas représentatif de l'ensemble de l'image. De plus, quand k augmente, le comportement de la norme k de Minkowski tend vers celui de la norme infinie, ce qui se vérifie dans les résultats obtenus.
Pour la norme 1, on observe de mauvais résultats moyens avec beaucoup de

disparités entre les différents chiffres(Figure 3.1). On a donc cherché à savoir lorsqu'un chiffre est mal reconnu avec quel chiffre il a été le plus confondu (Table 3.2).
On remarque pour la norme 1 que les 1 sont toujours correctement identifiés, mais que les autres chiffres sont souvent confondus avec des 1.

TABLE 3.2 – Représente pour les normes 1,2 et 3 et pour un chiffre i donné avec quel autre chiffre le chiffre i est le plus souvent confondu par l'algorithme

Norme	0	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	1	1	1	1	1
2	5	8	1	8	9	3	1	9	1	4
3	5	8	8	8	9	3	2	9	3	4

Chapitre 4

Deuxième méthode : utilisation de la décomposition en valeurs singulière (SVD)

4.0.1 Théorie

Etant donné les résultats décevants de notre première méthode, il est nécessaire d'utiliser une autre technique qui prenne en compte les variations entre les différentes images représentant un même chiffre, cette technique repose sur un outil mathématique très puissant : la décomposition en valeurs singulières (SVD).
Il s'agit ici dans un premier temps, de créer pour chaque chiffre (on parlera ici uniquement des trois pour alléger l'écriture) une matrice A de 784 lignes, comportant autant de colonnes que le nombre de 3 dont on dispose dans notre base d'entrainement. Chaque colonne de cette matrice représente une image de 3. On dispose alors d'un sous espace vectoriel de $\mathbf{R}^{784}$, engendré par les colonnes de A (on appelera cet espace vectoriel "l'espace des trois"), qui est nécessairement de faible dimension, sinon les espaces propres à chaque chiffre s'intersecteraient. Or le théorème de décomposition en valeurs singulières, nous permet de décomposer cette matrice ainsi :

$$A = \sum_{i=1}^{784} \sigma_i u_i v_i^T$$

Les u_i sont alors une base orthonormée de l'espace des trois. De plus lors d'une SVD, la base est ordonnée de manières à ce que les σ_i forment une suite décroissante, qui dans notre cas finit par s'annuler. Les premiers vecteurs u de la SVD sont alors représentatifs de l'espace vectoriel des 3, étant

donné que toutes les matrices de la somme sont de norme 1. On peut donc construire un espace des trois réduit ayant pour base (orthonormée) les premiers vecteurs u de la SVD. Ce sous-espace est de faible dimension par rapport à l'espace de dimension 784 dans le quel il vit et il est donc très peu probable qu'il intersecte le sous-espace relatif à un autre chiffre, ce qui garantit que les différents espaces réduits seront bien distincts dans le calcul à venir.
Maintenant qu'on dispose d'un espace vectoriel des 3 réduit (doté d'une base orthonormé), on peut facilement calculer la distance, pour la norme deux, du chiffre non identifié (c'est à dire un vecteur de $\mathbf{R}^{784}$) à cet espace (cela revient à un calcul des moindres carrés). Si on effectue ce calcul pour l'espace correspondant à chaque chiffre on peut alors labéliser l'image, en constatant de quel espace vectoriel elle est le plus proche.

4.0.2 Pratique

Pour cette méthode, nous avons effectué une classification (clustering) plus fine des résultats : si lors de l'analyse d'une image, on ne peut l'identifier de façon décisive à un chiffre, celle-ci est rejetée, ce qui permet d'améliorer le taux d'identification correct parmi les images non rejetées.
Plus précisément, une image z sera rejetée si le minimum $m1$ de l'ensemble des $\left\|(I - U_k U_k^T)z\right\|_2$ (les moindres carrés pour chaque chiffre) n'est pas "significativement" plus petit que le deuxième plus petit élément de cet ensemble $m2$ (deux bases de chiffres pourraient alors correspondre à l'image). Pour définir ce "significativement" nous avons introduit un seuil compris entre 0.5 et 1 : une image sera rejetée si après le calcul de ses moindres carrés $m1 > seuil * m2$. Plus le seuil est bas plus l'on est exigeant : avec un seuil de 1 aucune image n'est rejetée et avec un seuil de 0.5 l'évaluation d'une image est jugée concluante seulement si $m1 \leq 0.5m2$.

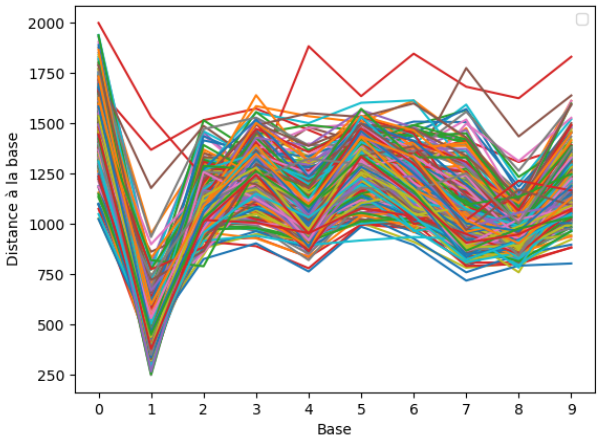

FIGURE 4.1 – Distance pour chaque image de 1 aux espaces vectoriels (de dimension 20) représentant les différents chiffres. On remarque que les images sont bien plus proches de l'espace des uns que des autres espaces et il y a donc très peu d'erreur.

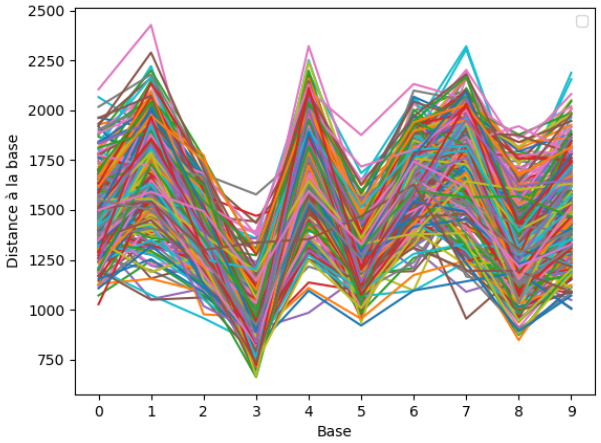

FIGURE 4.2 – Distance pour chaque image de 3 aux espaces vectoriels (de dimension 20) représentant les différents chiffres. On peut remarquer des pics secondaires pour les espaces des 5 et des 8 ce qui signifie que les 3 sont plus difficiles à identifier clairement que les 1 et seront généralement pris pour des 5 et des 8 par l'algorithme lorsqu'il se trompera.

4.0.3 Paramètres

Nous avons cherché à optimiser cette deuxième méthode en jouant sur deux paramètres : le nombre de vecteurs de base de U choisis k et le seuil.

Le premier jeu de test a été effectué pour des valeurs du seuil entre 0.9 et 1 (des valeurs inférieures du seuil donnant un taux de rejet trop élevées env. 20%) et une valeur de k (nombre de vecteurs de base) entre 1 et 7 (Figure 4.1).

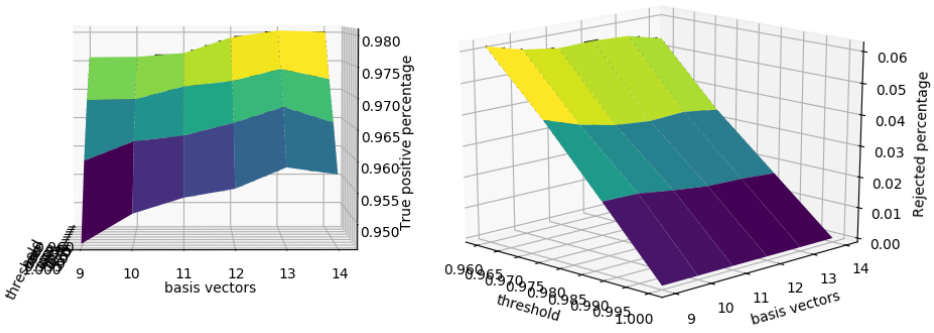

FIGURE 4.3 – A gauche, pourcentage d'images correctement identifiées et a droite, pourcentage d'images rejetées en fonction de nombre de vecteurs de base et du seuil de distance minimale

On remarque que quelque soit la base, l'efficacité de l'algorithme évolue approximativement de la même manière en fonction de la valeur du seuil. On peut donc par la suite se permettre d'étudier indépendamment l'influence du seuil et celle du nombre de bases.

Seuil

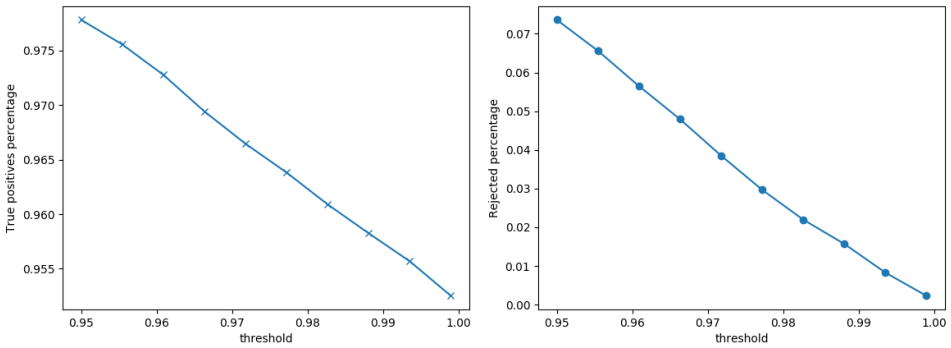

FIGURE 4.4 – Pourcentage de vrai positifs et de rejetés en fonction du niveau du seuil.

On remarque dans la figure 4.2 le pourcentage de vrais positifs et le pourcentages de rejetés sont proportionnelles et linéairement décroissantes en fonction du seuil. On n'a donc pas de choix optimal objectif pour le seuil, on choisira donc le seuil en fonction de la tolérance à l'erreur et le taux de rejet acceptable.

Nombre de vecteurs de base

On remarque que l'efficacité de l'algorithme est croissante en fonction du nombre k de vecteurs de bases pour ceux observés. Cependant après notre analyse théorique, on peut dire que si k devient trop grand, les espaces vectoriels qu'on associe à chaque chiffre (étant de dimension k) ont de fortes chances de s'intersecter (Figure 4.3). Il est donc intéressant de chercher la valeur de k optimale pour notre algorithme.

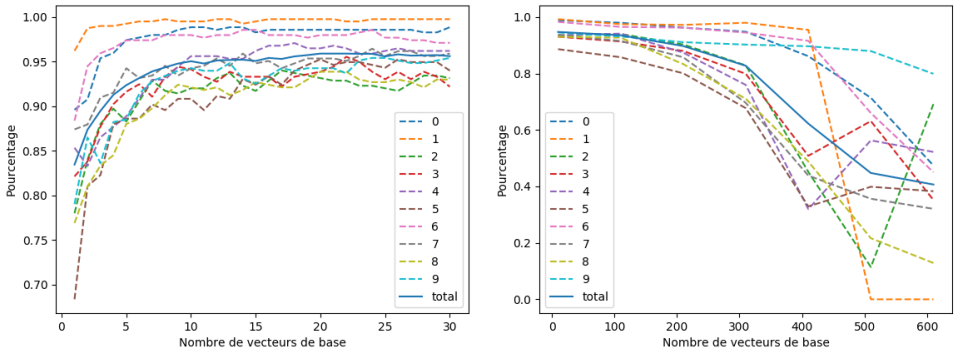

FIGURE 4.5 – Performance pour chaque chiffre et en moyenne en fonction du nombre de vecteurs de base. A gauche lorsque le nombre de vecteurs de base est encore faible l'algorithme devient plus performant en augmentant ce paramètre. A droite on assiste à l'effondrement du modèle lorsque le nombre de vecteurs de base est trop élevé et que les espaces relatifs à chacun des nombres s'intersectent.

En faisant varier k entre 1 et 30, on remarque que l'efficacité de l'algorithme se stabilise à partir de k=20.De plus le nombre de vecteurs de base k choisi n'influe pas significativement sur la complexité des calculs : en effet seul le calcul de $I - U_k U_k^T$ dépend de k et il est effectué une seule fois pour chaque chiffre. On fixe donc par la suite k=20.

Chapitre 5

Distance tangente

On souhaite désormais affranchir notre algorithme des erreurs liées aux petites variations usuelles que nos chiffres peuvent connaitre lorsque qu'un humain les écrits. En effet il est courant dans une base de données réelle que les chiffres soient légérement décalés, dillatés ou encore épaissis. Afin d'expliciter la méthode plus simplement, on va d'abord s'interesser uniquement à rendre notre algorithme insensible à la translation en X.
Si on considère une image I comme un point dans $\mathbf{R}^{784}$, l'ensemble des I translatés est une courbe continue dans $\mathbf{R}^{784}$. On peut décrire l'ensemble des points de cette courbe, comme l'image de $\mathbf{R}$ par la fonction s, où s(I,α) est l'élément de $\mathbf{R}^{784}$ correspondant à I décalé de α sur l'axe des X. Idéalement on aimerai définir la distance suivante, P étant un autre élément de $\mathbf{R}^{784}$: $d(I,P) = min_{\alpha,\beta\in\mathbf{R}}(||s(I,\alpha) - s(P,\beta)||)$. Cependant il serait inutile d'essayer d'implémenter cette distance, puisque son calcul est très complexe et qu'on ne s'interesse ici qu'à de petites translations. On va donc utiliser des approximations à l'ordre 1 des courbes des translatés, valables pour des petites valeures de $\alpha et \beta$. On effectue un développement de Taylor et on obtient :

$$s(I,\alpha) \approx s(I,0) + \frac{ds}{d\alpha}(I,0) * \alpha = I + \frac{ds}{d\alpha}(I,0) * \alpha$$

.
Ainsi on peut définir une nouvelle distance plus simple, à peu près équivalente au voisinnage de 0 : $d_2(I,P) = min_{\alpha,\beta\in\mathbf{R}}(||I+\frac{ds}{d\alpha}(I,0)*\alpha-P+\frac{ds}{d\beta}(P,0)*\beta)||)$.
Calculer

Chapitre 6

Conclusion : Test globaux

Pour tester la performance des algorithmes implémentées on a effectué une série de tests sur l'ensemble de la base de donnée : on a réservé 20% des images de la base de données comme ensemble de test, puis nous avons sélectionné aléatoirement la moitié des images restantes comme ensemble d'entraînement.

Pour le premier algorithme nous avons testé 50 fois, obtenant en moyenne des performances de 83% pour la norme 2 et 84% pour la norme 3 (Table 5.1) avec des résultats variées en fonction des chiffres analysés.

TABLE 6.1 – Performance Algorithme 1, pour les normes 2 et 3

Norme	0	1	2	3	4	5	6	7	8	9	**Total**
2	88.2	96.4	77.2	81.7	83.9	70.3	87.9	85.7	75.8	79.5	**83**
3	90.2	88.6	84.6	80.7	82.5	80.7	88.0	82.1	82.2	79.9	**84**

Le deuxième algorithme étant significativement (DE COMBIEN ?) plus coûteux quant au temps de calcul, on a effectuée 30 tests, obtenant une performance moyenne de 96% (Table 5.2). Les performances sont significativement plus consistantes pour toutes les chiffres par rapport à l'algorithme 1 (minimum 94%).

TABLE 6.2 – Performance Algorithme 2 avec 20 vecteurs base

Chiffre	0	1	2	3	4	5	6	7	8	9	**Total**
Performance	98.9	99.2	94.6	94.8	97.1	94.3	97.9	94.8	93.9	94.4	**96.0**

En ajoutant un seuil de ressemblance minimale, on est capable de mettre de coté les chiffres les plus difficiles à identifier (Tables 5.3 et 5.4). si on accepte un taux de rejet d'environ 5% on peut atteindre un pourcentage d'identification correcte d'environ 98%.

TABLE 6.3 – Performance et taux de rejet de l'Algorithme 2 avec 20 vecteurs base et un seuil de 0.99, pour les chiffres de 0 à 9 et en moyenne.

	0	1	2	3	4	5	6	7	8	9	**Total**
Performance	99.0	99.3	95.1	95.5	97.4	95.2	98.1	95.5	94.8	95	**96.6**
Rejetées	0.2	0.1	1.1	1.6	0.7	1.9	0.4	1.4	1.7	1.3	**1.0**

TABLE 6.4 – Performance et taux de rejet de l'Algorithme 2 avec 20 vecteurs base et un seuil de 0.95, pour les chiffres de 0 à 9 et en moyenne.

	0	1	2	3	4	5	6	7	8	9	**Total**
Performance	99.5	99.6	96.7	97.7	98.7	97.9	98.9	97.5	97.5	97.4	**98.2**
Rejetées	1.2	0.6	5.1	7.6	4.1	9.9	2.5	6.7	9.2	6.5	**5.3**

Bibliographie

[1] L. Elden, “Matrix methods in data mining and pattern recognition,” 2007.