

Reconnaissance de caractères

...

17 mars 2019

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.

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 du 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 les 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 set d'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 entre 1 et 10 et pour la norme infinie (table 3.1).

TABLE 3.1 – Pourcentage de chiffres correctement identifiés 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

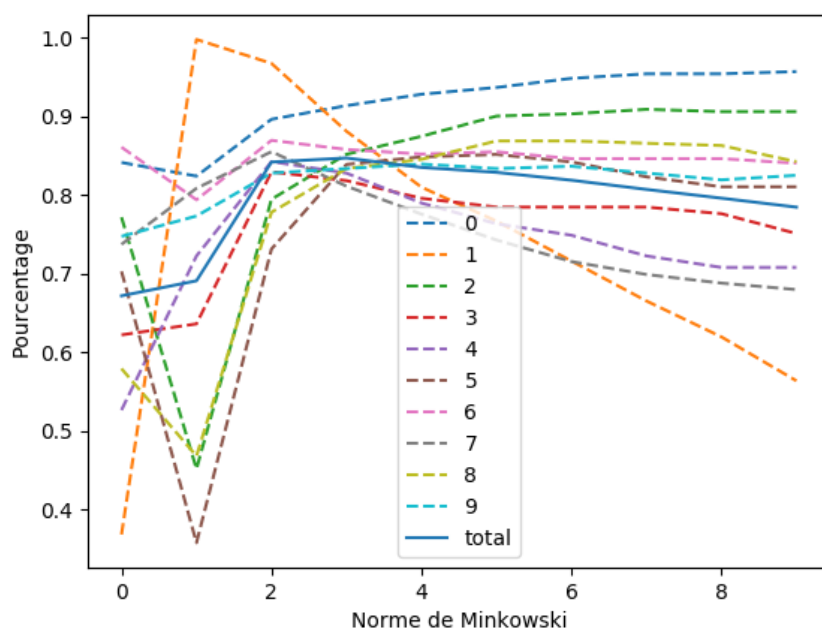


FIGURE 3.1 – Performance de l'algorithme pour chaque chiffre et en moyenne en fonction de la norme de Minkowsky. (la valeur 0 en abscisse représente la norme infinie)

On remarque qu'en moyenne les normes 2 et 3 sont les plus performantes avec une variance faible.

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 (cf. tableau 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 – Chiffres les plus souvent confondus par norme de Minkowsky

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 matrices 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'entraînement. 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 appellera 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 matrice singulière, 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 orthogonale de l'espace des trois. De plus lors d'une SVD, la base est ordonnée de manière à ce que les σ_i forment une suite décroissante, qui finit par s'annuler dans notre cas (pour garantir une faible dimension). Les premiers vecteurs u de la SVD sont alors représentatifs

de l'espace vectoriel des 3, étant donné que toutes les matrices dans la somme sont de norme 1.

Maintenant qu'on dispose d'une base réduite de l'espace vectoriel des 3, on peut facilement calculer la distance pour la norme deux entre un chiffre non identifié (c'est à dire un vecteur de \mathbf{R}^{784}) et l'espace des 3 (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

Seuillage

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 $\|(I - U_k U_k^T)z\|_2$ (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 $m1 > \text{seuil} * m2$. 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 $m1 \leq 0.5m2$.

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 set de test a été effectuée 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.

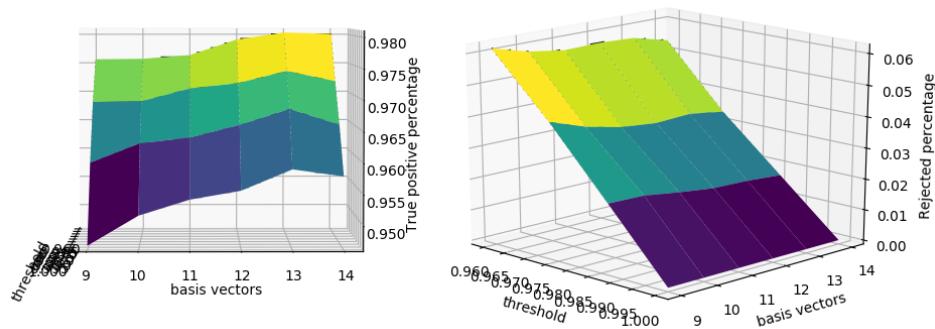


FIGURE 4.1 – A Modifier...

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

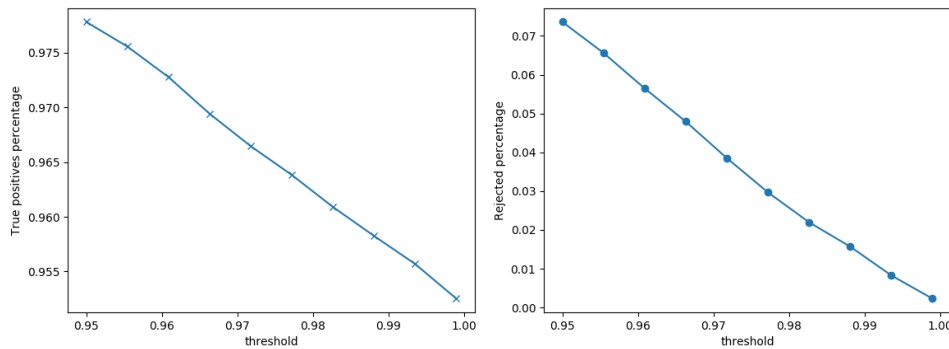


FIGURE 4.2 – A Ajouter

On remarque dans les figures (numéro) 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 (cf. figure). Il est donc intéressant de chercher la valeur de k optimale pour notre algorithme.

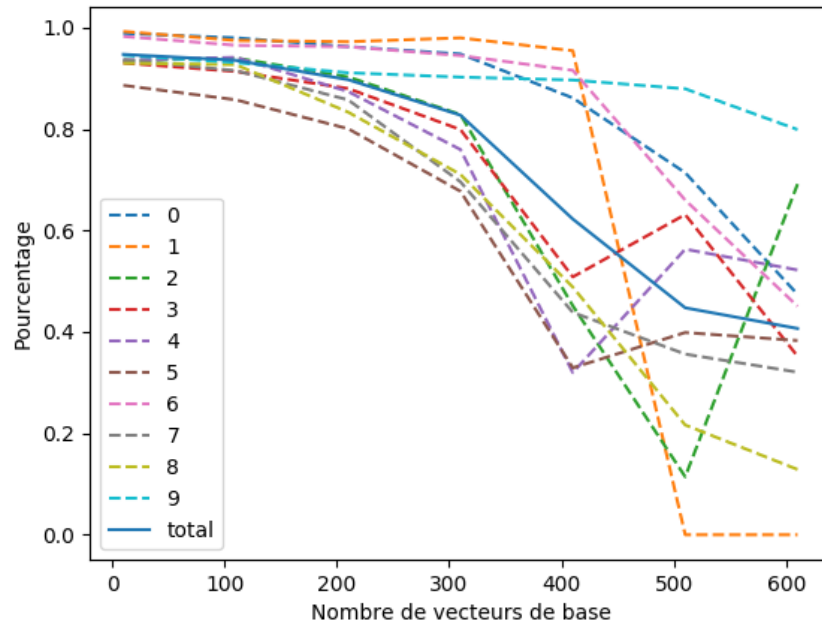


FIGURE 4.3 – A Ajouter

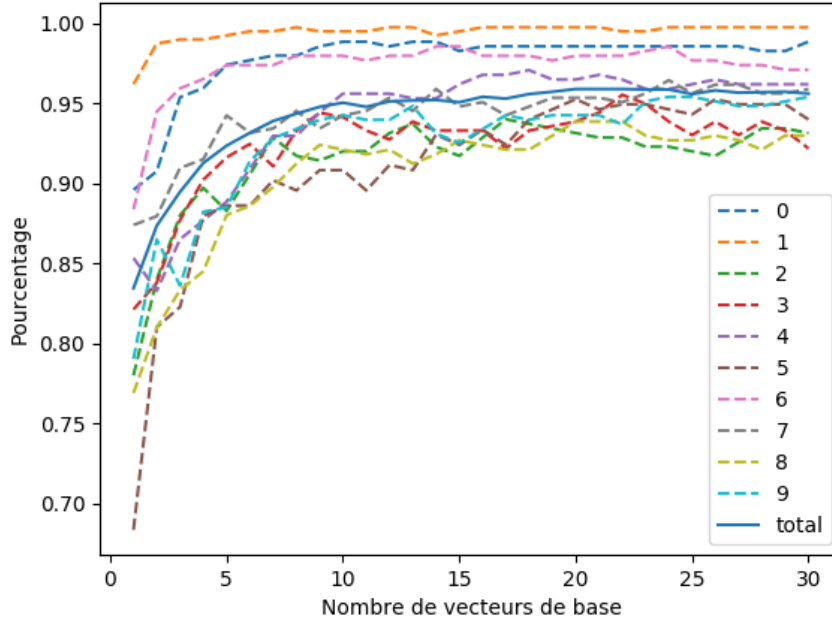


FIGURE 4.4 – A Ajouter

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 $k=20$.

4.0.4 Tests globaux

Chapitre 5

Conclusion

Annexe A

Annexe 1

...