Classement de caractères manuscrits

1 Présentation des données

La reconnaissance de caractères manuscrits par un système automatisé est un problème aux multiples applications : reconnaissance des codes postales, création de petits systèmes mobiles de saisie de texte (par exemple, pour les ordinateurs de poche), numérisation de documents manuscrits, ...

Les méthodes les plus performantes pour reconnaître un caractère manuscrit sont basées sur des méthodes d'apprentissage statistique : leur principe commun est de fonder leur prédiction sur la comparaison de l'image du caractère manuscrit à classer à d'autres images de caractères manuscrits pour lesquels la nature du caractère est connue. Typiquement, si une image arrive et qu'elle est très similaire à l'image de nombreux '1' de notre base d'apprentissage, l'algorithme classera l'image dans la catégorie '1'.

Les données considérées ici proviennent de la base MNIST (`http://yann.lecun.com/exdb/mnist/`) sur laquelle des milliers de chercheurs ont travaillé. Elle est constituée de 70 000 chiffres manuscrits au format 28 pixels par 28 pixels où chaque pixel est représenté par un niveau de gris allant de 1 à 256 (i.e. un chiffre manuscrit est donc un vecteur de $\{1, \ldots, 256\}^{28\times 28}$).

Dans ce TD, pour limiter le temps de calcul et la mémoire nécessaire, nous ne considérons que 800 chiffres manuscrits (que des '3' et des '5') : 400 chiffres manuscrits seront utilisés pour apprendre, i.e. calibrer l'algorithme. Ces 400 chiffres manuscrits et la classe ('3' ou '5') qui leur est associée constituent l'ensemble d'apprentissage. Les 400 autres chiffres manuscrits seront uniquement utilisés pour évaluer la qualité des algorithmes. Ces 400 caractères manuscrits constitue l'ensemble de test.

On dispose d'une matrice de taille $(800, 1 + (28 \times 28))$ correspondant aux 800 caractères manuscrits où

- chaque ligne correspond à un caractère manuscrit
- la première colonne contient la classe du caractère (c'est un chiffre qui vaut soit 3 soit 5).
- les colonnes suivantes contiennent les valeurs des 28×28 pixels en commençant par le coin supérieur gauche et parcourant l'image ligne par ligne.

On charge les données dans Scilab, après avoir sauvegardé localement les fichiers `chiffres3.txt` et `chiffres5.txt` contenant les '3' et les '5', à l'aide

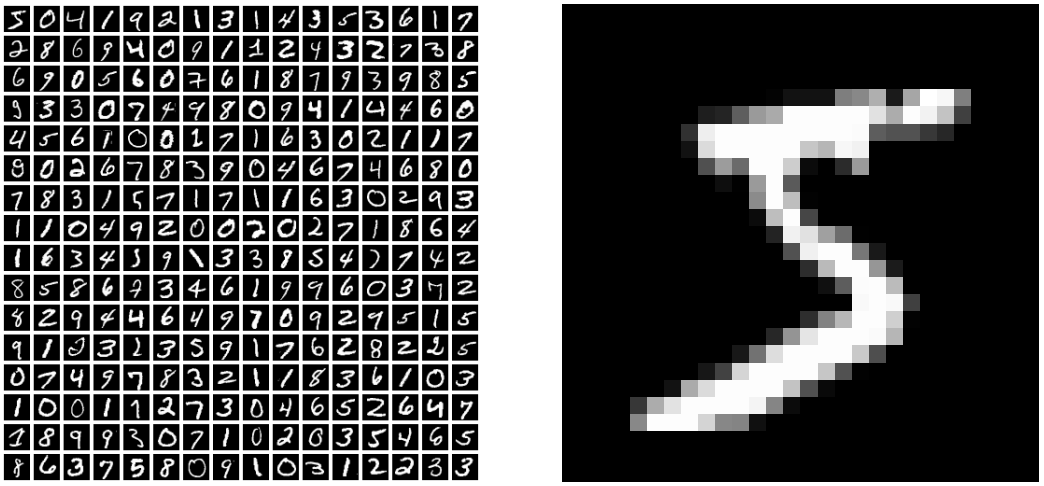

FIGURE 1 – *A gauche :* exemple de caractères manuscrits 28×28. *A droite :* zoom sur le premier caractère manuscrit 28×28. Ce caractère manuscrit appartient à la classe '5'. (Noter que curieusement zoomer sur une image d'un caractère manuscrit ne le rend pas plus lisible)

des commandes :

```
chiffres3=fscanfMat("chiffres3.txt");
chiffres5=fscanfMat("chiffres5.txt");
l=200;
chiffres=[chiffres3(1:l,:);chiffres5(1:l,:);
   chiffres3(l+1:2*l,:);chiffres5(l+1:2*l,:)];
chiffres_sans_label=chiffres(:,2:$);
EnsApp=chiffres(1:2*l,:);
EnsTest=chiffres(2*l+1:4*l,:);
```

En cas de message d'erreur concernant la taille de la pile, on peut utiliser la commande `stacksize(new_size)` avec `new_size` de l'ordre de `10^8`.

Les fonctions nécessaires pour le TD sont accessibles dans le fichier `ppv.sce`. Ce fichier est à sauvegarder localement dans le même répertoire que les fichiers "chiffres3.txt" et "chiffres5.txt". Pour charger le fichier, on utilise la commande `exec nom_du_fichier.sce;` (il faut au préalable vérifier que le répertoire courant contient le fichier menu 'Fichier | changer de répertoire...'). Pour visualiser le fichier, on peut utiliser `scipad ppv.sce;`. Vous pouvez

d'ores et déjà visualiser quelques chiffres avec la commande :
`aff_nb(1:10,chiffres);`

2 Classement par la méthode des k-plus proches voisins

On considère l'algorithme $\hat{g}_k$ des k-plus proches voisins pour la distance euclidienne entre les vecteurs de $\mathbb{R}^{28\times 28}$ que sont les images de caractères manuscrits.

Question 1. *Rappeler comment, en théorie, choisir k pour qu'asymptotiquement (lorsque la taille de l'ensemble d'apprentissage tend vers l'infini) l'algorithme $\hat{g}_k$ soit universellement consistant.*

Soient $(X_1, Y_1), \ldots, (X_n, Y_n)$ les couples images-labels de l'ensemble d'apprentissage et $(X_{n+1}, Y_{n+1}), \ldots, (X_{n+t}, Y_{n+t})$ les couples images-labels de l'ensemble de test. (Ici $n = t = 400$.) Le risque d'une fonction de classification g est $R(g) = \mathbb{P}(Y \neq g(X))$. L'erreur d'apprentissage de l'algorithme $\hat{g}_k$ est $r_a(\hat{g}_k) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{Y_i \neq \hat{g}_k(X_i)}$. Son erreur de test est

$$r_t(\hat{g}_k) = \frac{1}{t}\sum_{i=n+1}^{n+t} \mathbf{1}_{Y_i \neq \hat{g}_k(X_i)}.$$

Ces erreurs sont respectivement les pourcentages de caractères manuscrits de l'ensemble de test et d'apprentissage mal classés par l'algorithme $\hat{g}_k$.

Question 2. *Dans cette question, on raisonne à ensemble d'apprentissage fixé (en termes plus probabilistes, conditionnellement à la valeur de l'ensemble d'apprentissage).*
(a) Quelle est la limite de l'erreur de test de $\hat{g}_k$ lorsque t tend vers l'infini ?
(b) Quelle est la loi limite de $\sqrt{t}\big(r_t(\hat{g}_k) - R(\hat{g}_k)\big)$ lorsque t tend vers l'infini ? On pourra utiliser le théorème centrale limite.
(c) En déduire une borne supérieure sur $|r_t(\hat{g}_k) - R(\hat{g}_k)|$ en fonction de $R(\hat{g}_k)(1 - R(\hat{g}_k))$ qui soit valable asymptotiquement avec probabilité 95% ? On rappelle que $\mathbb{P}(|Z| > 2) \approx 0,05$ pour Z une v.a. de loi normale centrée réduite : $\mathcal{L}(Z) = \mathcal{N}(0, 1)$.

Utiliser la fonction `[table_err,err_ppv]=k_ppv(EnsApp,EnsTest);` pour calculer l'erreur de test des k-p.p.v..

Question 3. *(a) Quelle est l'erreur de test de l'algorithme du plus proche voisin ?*
(b) A combien d'images mal classées correspond-elle ?
(c) Afficher ces images (le code `indices=find(table_err(:,1)==%F)` *recherche dans la première colonne de 'table_err' les indices des points mal classés et en déduire le nombre d'images test de '3' mal classées.*
(d) Afficher les 17 images p.p.v. de la première image mal classée (on pourra utiliser les fonctions `ind_ppv` *et* `aff_nb`*. Que constatez-vous ? Pour quelles valeurs impaires de k entre 1 et 17 l'algorithme des k plus proches voisins classe bien cette image ?*

Question 4. *En utilisant la commande* `plot(err_ppv,'green')`*, tracer l'erreur de test de $\hat{g}_k$ pour les différentes valeurs de k. En utilisant le résultat de la question 2 et en approchant $R(\hat{g}_k)(1-R(\hat{g}_k))$ par $r_t(\hat{g}_k)(1-r_t(\hat{g}_k))$, tracer les 2 courbes montrant l'intervalle de confiance de $R(\hat{g}_k)$ pour toutes les valeurs de k. (On pourra utiliser le produit terme à terme* `err_ppv.*(1-err_ppv)`*.) En réutilisant la fonction* `k_ppv`*, tracer sur la même figure l'erreur d'apprentissage de $\hat{g}_k$ pour les différentes valeurs de k. Pour quelles valeurs de k y a-t-il surapprentissage ? (On dit qu'un algorithme surapprend lorsque son erreur d'apprentissage est significativement plus faible que son erreur de test.)*

Question 5. *(a) Tracer et commenter la courbe représentant l'estimation du risque par validation croisée d'ordre 2 en fonction de k. Attention à (i) ne pas utiliser l'ensemble de test (i.e., n'utiliser que les données de l'ensemble d'apprentissage) et à (ii) s'assurer que les différents blocs contiennent autant de '3' que de '5'. En utilisant de nouveau les courbes d'intervalles de confiance tracées à la question précédente, préciser les valeurs de k pour lesquelles l'estimation du risque par validation croisée d'ordre 2 est raisonnable. Commenter. Quelle est la valeur de k choisie par validation croisée d'ordre 2 ?*
(b) Mêmes questions pour la validation croisée d'ordre 5.
(c) Quelles sont les erreurs de test associées ?