Travaux pratiques de
Machine learning
Cycle pluridisciplinaire d'études supérieures
Université Paris sciences et lettres

Joon Kwon

mercredi 30 mars 2022

```
import numpy as np
import matplotlib.pyplot as plt
```

On considère un problème de reconnaissance de caractères. Le jeu de données contient des images de chiffres manuscrits, ainsi que le chiffre correspondant. On souhaite construire un prédicteur qui sache reconnaître le chiffre à partir de l'image. Il s'agit donc d'un problème de classification. On charge le jeu de données.

```
from sklearn import datasets
digits = datasets.load_digits()
X = digits.data
y = digits.target

print(X[1])
```

On peut voir que chaque entrée, par exemple `X[1]` ci-dessus, est un `array` de dimension 1 et de taille 64. Il représente une image de taille 8×8 pixels. Chaque composante est un entier compris entre 0 et 16 et représente l'intensité de gris du pixel correspondant. Pour chaque entrée, on peut utiliser la fonction `.reshape()` afin d'en faire un `array` de dimension 2 et de taille 8×8.

```
print(X[1].reshape(8,8))
```

Cela permet ensuite de visualiser l'image correspondante grâce à la fonction `plt.imshow()`.

```
plt.figure()
plt.imshow(X[1].reshape(8,8),cmap=plt.cm.gray_r)
plt.show()
```

QUESTION 1. — Les différentes classes (0, 1, … , 8 et 9) sont-elles présentes en quantités à peu près égales dans le jeu de données ?

Nous allons construire des prédicteurs kNN. Les prédictions de ces derniers sont lentes à calculer lorsque l'échantillon d'apprentissage est grand. On va donc tenter de limiter la taille de l'échantillon d'apprentissage.

QUESTION 2. — Partitionner le jeu de données en deux échantillons : un échantillon (`X_train`, `y_train`) de taille 200, qui servira pour la validation croisée, et un échantillon de test (`X_test`, `y_test`) qui servira pour le test final. On pensera à utiliser la fonction `train_test_split` vue dans le TP précédent, dans laquelle on pourra spécifier l'argument optionnel `train_size`.

Nous pouvons à présent effectuer une validation croisée sur l'échantillon (`X_train`,`y_train`). Nous allons par exemple considérer l'algorithme kNN avec $k = 5$, et une 7-validation croisée. Nous faisons appel à la fonction `cross_val_score`, fournie par `scikit-learn`.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score

knn = KNeighborsClassifier(n_neighbors=5)
scores = cross_val_score(knn,X_train,y_train,cv=7)
```

QUESTION 3. — Que contient la variable `scores` définie ci-dessus ? En déduire le score de validation de la 7-CV de l'algorithme 5NN.

Pour un ensemble de valeurs de l'hyperparamètre k (par exemple $k \in \{1, \dots, 20\}$), les scores d'apprentissage et de validation de la 7-CV peuvent se calculer comme suit.

```
from sklearn.model_selection import validation_curve
k_range = range(1,21)
train_scores, valid_scores =
 ↪  validation_curve(KNeighborsClassifier(),
 ↪  X_train,y_train,"n_neighbors",k_range,cv=7)
```

Question 4. — Tracer les courbes de validation correspondantes. Y a-t-il besoin d'essayer d'autres valeurs de k ? Définir une variable `k_best` égale à la meilleure valeur de l'hyperparamètre k selon ce qui précède.

On souhaite, pour $k =$ `k_best`, tracer la courbe d'apprentissage correspondant à des tailles d'échantillon d'apprentissage allant de 10 à 150, et multiples de 5. Le code suivant donne les scores 7-CV correspondants.

```
from sklearn.model_selection import learning_curve

train_size_range = range(10,151,5)
train_sizes, train_scores, valid_scores =
 ↪  learning_curve(KNeighborsClassifier(n_neighbors=k_best),X_train,
 ↪  y_train,train_sizes=train_size_range,cv=7)
```

Question 5. — Tracer la courbe d'apprentissage correspondante. Serait-il intéressant d'utiliser des échantillons d'apprentissage de plus grande taille ? Si oui, reprendre depuis la question 2 avec un plus grand échantillon d'apprentissage.

Question 6. — Tracer une courbe d'apprentissage où le score est obtenu par validation simple, et non par validation croisée. Expliquer l'avantage de la validation croisée dans ce contexte.

Question 7. — Entraîner le prédicteur kNN (pour $k =$`k_best`) avec (`X_train`,`y_train`) pour échantillon d'apprentissage. Calculer son score de test avec l'échantillon (`X_test`,`y_test`). Observer également la matrice de confusion sur l'échantillon de test. Quelle est l'erreur de classification la plus fréquente ?

On souhaite à présent ajouter au jeu de données des exemples obtenus par modification des exemples existants, dans l'espoir d'obtenir de meilleurs résultats.

Question 8. — Écrire une fonction `shift_image` qui prend un argument un `array` (nommé `image`) de dimension 2 et de taille 8×8 (représentant une

image), ainsi qu'un `array` (nommé `direction`) contenant des entiers de dimension 1 de taille 2, et qui renvoie l'image translatée par le vecteur de coordonnées `direction` (le contenu de l'image est donc amené à être rogné sur certains côtés). Vérifier le bon fonctionnement de la fonction en visualisant quelques images translatées.

Question 9. — Ajouter au jeu de données la translation de tous les exemples par chacun des vecteurs : $(-1, 0)$, $(1, 0)$, $(0, -1)$, $(0, 1)$, $(-1, -1)$, $(-1, 1)$, $(1, -1)$ et $(1, 1)$. Construire alors un prédicteur kNN en choisissant l'hyperparamètre k par validation croisée, et observer son score sur un échantillon de test. Comparer le prédicteur obtenu à celui de la question 7. On pourra dans un premier temps utiliser un échantillon de validation croisée de taille 400, et décider ensuite s'il est intéressant d'en augmenter la taille.