Classification supervisée - TP2

Rémy Degenne

2 Février 2022

Mise en place: dans le dossier du cours, créer un notebook (avec `jupyter notebook`) et lui donner un titre.

1 Documentation

Documentation de scikit-learn: `https://scikit-learn.org/0.24/index.html`
Documentation de matplotlib: `https://matplotlib.org/stable/tutorials/introductory/usage.html`
Documentation de numpy: `https://numpy.org/doc/1.18/`

2 Imports

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

from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import Perceptron, SGDClassifier
from sklearn.neural_network import MLPClassifier

from sklearn.datasets import load_digits, make_classification, fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.preprocessing import StandardScaler
from sklearn.utils import shuffle
```

3 Classer des chiffres

- Charger l'ensemble d'exemples `digits` de scikit-learn à l'aide de la fonction `load_digits`.
- Placer les données dans un tableau `X` et les étiquettes dans un vecteur `Y`.
- Séparer les données en un ensemble d'entrainement et un ensemble de test (40% des données) à l'aide de la fonction `train_test_split`.

On pourra utiliser la fonction suivante pour observer des exemples des données qu'on vient de charger:

```
def plot_examples(examples, labels, pred=False):
    _, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
    nb_pxl = len(examples[0].flatten())
    side_len = int(np.sqrt(nb_pxl)) # assume square images
    for ax, image, label in zip(axes, examples, labels):
        ax.set_axis_off()
        image = image.reshape((side_len, side_len))
        ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
        if pred:
            ax.set_title("Prediction: %i" % label)
        else:
            ax.set_title("Training: %i" % label)
```

1. Que sont les données ? Et leurs étiquettes ?
2. Pourquoi est-ce qu'on coupe les exemples en un ensemble d'entrainement et un ensemble de test ?
3. Combien de classes (= étiquettes différentes) dans cet ensemble d'exemples ?
4. Est-ce que ce nombre de classes est un problème pour la classification avec un classeur des plus proches voisins ? Et pour un perceptron ?
5. Si c'est un problème, proposer des solutions.

On va utiliser des classeurs perceptron et perceptron multi-couches (MLP) sur ces exemples.

3.1 Le perceptron

```
per = Perceptron(tol=0.001, max_iter=1000, shuffle=True, eta0=1.0, random_state=0)
```

`tol` est un paramètre qui définit le critère d'arrêt de l'apprentissage: l'algorithme arrête de changer les poids (avec une descente de gradient) si la performance sur les données d'apprentissage ne s'améliore pas plus que `tol`.
`eta0` est le paramètre qui règle la taille du pas de gradient effectué à chaque étape de l'apprentissage.

1. Rappeler la manière dont un perceptron prédit l'étiquette d'une donnée.
2. Entrainer le classeur et obtenir son score sur les données d'entrainement et de test (utiliser `per.score()`).
3. Utiliser la fonction suivante et expliquer ce que représente le résultat:

```
def plot_confusion_matrix(Y, pred):
    disp = ConfusionMatrixDisplay.from_predictions(Y, pred)
    disp.figure_.suptitle("Confusion Matrix")
    plt.show()
```

4. La fonction `partial_fit` permet de faire une itération de la règle d'entrainement (utiliser chaque donnée d'entrainement une fois pour changer les poids). A l'aide de cette fonction, construire un graphe du score du perceptron sur les données d'entrainement et de test au cours de l'apprentissage.
5. Qu'observez vous sur ce graphe concernant le score d'apprentissage ? Et le score de test ?

3.2 Le perceptron multi-couches

```
mlp = MLPClassifier(
    hidden_layer_sizes=(15,10,), activation="relu",
    solver="sgd", learning_rate="constant", learning_rate_init=0.001,
    max_iter=2000, verbose=False)
```

`hidden_layer_sizes=(15,10,)` indique le nombre de neurones dans chaque couche "cachée" du réseau de neurones.
`activation` décide la fonction d'activation de chaque neurone.
`solver` indique la méthode d'apprentissage des poids.
`learning_rate_init` indique la taille de chaque pas de gradient dans la méthode des gradients stochastiques.

1. Entrainer le classeur et obtenir son score sur les données d'entrainement et de test.
2. Afficher la matrice de confusion du classeur. Comparer avec le perceptron.
3. Essayer plusieurs architectures (valeurs de `hidden_layer_sizes`).
4. Tracer une courbe des erreurs d'entrainement et de test au fil de l'apprentissage, comme effectué pour le perceptron plus haut.

5. Faire varier `learning_rate_init` et observer l'effet sur l'apprentissage.

6. Changer le paramètre `learning_rate`: essayer les valeurs `adaptive` et `invsvaling`. Chercher dans la documentation ce que veulent dire ces paramètres.

4 Classer des chiffres en plus haute définition

On va utiliser ici le jeu de données MNIST, qui contient des images de chiffres écrits à la main, représentés par des images en nuances de gris de 28x28 pixels. On peut obtenir ces données grâce à la fonction suivante:

```
mnist = fetch_openml("mnist_784", version=1, as_frame=False)
```

L'obtention du jeu de données peut prendre du temps: il est conseillé d'écrire cette ligne dans une cellule à part et de ne l'exécuter qu'une fois.

1. Combien de données sont présentes dans mnist ? Combien de classes ? Afficher quelques exemples à l'aide de la fonction `plot_examples` définie plus haut.

2. Séparer l'ensemble de données en données d'entrainement et de test. Attention: l'ensemble de données est très gros. Si vous prenez toutes les données, l'apprentissage des classeurs peut être très lent. Utilisez les paramètres `train_size=..., test_size=...` de la fonction `train_test_split` pour obtenir des ensembles d'entrainement et de test de la taille voulue.

3. Entrainer un perceptron sur ces données. Quel est son score ?

4. Entrainer des MLP divers sur ces données. Est-il possible de faire mieux que le perceptron ?

5. Le code suivant permet de recentrer les valeurs des pixels autour de zéro. Est-ce que l'apprentissage d'un MLP est plus efficace après ce changement ?

   ```
   scaler = StandardScaler()
   X_train = scaler.fit_transform(X_train)
   X_test = scaler.transform(X_test)
   ```