Apprentissage statistique
TP2 : Plus proche voisin et évaluation de performances

Olivier Schwander <olivier.schwander@lip6.fr>

2020-2021

Plus proche voisin

La classification par la méthode du plus proche voisin est l'une des méthodes d'apprentissage les plus simples. Soit un ensemble d'apprentissage constitué de N observations $X = \{x^{(1)} \dots x^{(N)}\}$ en dimension d et les classes associées $C_1 \dots C_N$. La classification d'une observation x inconnue est réalisée en cherchant le plus proche voisin (souvent au sens de la distance euclidienne, mais pas seulement) parmi les observations X.

Question
Calculer la complexité (le nombre d'opérations nécessaire) de la classification. Quel est le coût de la phase d'apprentissage ?

Question
Calculer la taille du modèle (le nombre de paramètres à conserver en mémoire pour réaliser la classification). Comparer avec le perceptron.

Question
Écrivez une fonction réalisant la classification par plus proche voisin.

Question
Testez la méthode sur des données synthétiques. Ne pas oublier d'ecrire une fonction qui separe les donnees en ensembles d'entrainement et de test.

Question
Tracez la frontière de décision. Commentez.

k-Plus proches voisins

La méthode du plus proche voisin est peu robuste à de petites variations dans les données. Pour atténuer cet effet, on utilise la méthode des k-plus proche voisins : au lieu de regarder uniquement le point le plus proche, on cherche les k points les plus proche, la classe choisie pour l'observation inconnue est alors la classe majoritaire parmi les k voisins.

Plusieurs algorithmes, plus ou moins rapides, existent pour la recherche des k-plus proches voisins. La version la plus simple, linéaire, est la suivante :

- **Entrée** : un point x inconnu et une liste de point $\mathcal{X} = \{X_1, \dots, X_N\}$
- **Initialisation** : soit une liste L contenant les k premiers éléments de $\mathcal{X}$, triés par ordre croissant de leur distance à x.
- **Pour** chaque point X de $\mathcal{X}$:

    - **Si** X est plus proche de x que le dernier point de L **alors :**
        - Supprimer le dernier point de L.
        - Rajouter X dans L de façon à garder la liste triée.
- **Renvoyer** L.

Des variantes plus évoluées sont disponibles dans le module `sklearn.neighbors`.

Question
Écrire une fonction réalisant la classification par k-plus proches voisins.

Pour réduire l'influence des voisins les plus éloignés (et donc portant le moins d'information), on introduit des poids lors du vote. Pour un voisin x_i, le poids w associé est :

$$w = \exp\left(\frac{-D(x, x_i)}{\sigma^2}\right)$$

où σ est un hyper-paramètre à déterminer et D la distance choisie.

Question
Adapter la fonction précédente pour utiliser des poids.

Question
Toujours sur des données synthétiques, tracez les frontière de décision. Commentez.

Comparaison avec le perceptron

Ces expériences se feront sur la base d'images USPS et sur les donnees synthetiques.

On a maintenant étudiés les algorithmes de classification suivants :

- perceptron,
- plus proche voisin,
- k-plus proches voisins sans poids,
- k-plus proches voisins avec poids.

Les deux dernières ont respectivement 1 et 2 hyper-paramètres à déterminer. Une question critique qui se pose lors de la résolution d'un problème d'apprentissage est le choix de la méthode ou du modèle utilisé, ainsi que le choix des paramètres éventuels.

Question
Écrire des fonctions pour le calcul du taux de vrais positifs et de faux positifs. Pour les donnees synthetique, jouer avec plusieurs valeurs de cluster$\backslash_{\text{std}}$ et centers.

Question
Tracer des courbes pour visualiser les taux de vrais positifs et de faux positifs en fonction des méthodes et des paramètres.

Question
Tracer les courbes du temps de calcul pour la classification de plusieurs observations en fonction des méthodes et des paramètres.

— On pourra utiliser la commande `%timeit` dans l'interpréteur `iPython`.
— On fera la mesure sur un nombre suffisant d'observations pour que le temps soit significatif (au moins 10 ou 100, souvent plus, en fonction de la méthode et de son implantation).

Question
Conclure sur les avantages et les inconvénients des différentes méthodes.

Annexe

Génération des points et tracé de la surface de séparation :

```
import numpy as np
from sklearn.datasets import make_blobs
import matplotlib.pylab as plt

X, y  = make_blobs(n_samples=100, n_features=2, centers=2, cluster_std=1.0, center_box=(-10.0, 10.0)

h = .4  # Precision de la grille
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),np.arange(y_min, y_max, h))
data = np.c_[xx.ravel(), yy.ravel()]

# Dans le format de votre classifier
z = classifier.predict(data)
zz = z.reshape(xx.shape)

plt.figure()
plt.contourf(xx, yy, zz, cmap=plt.cm.Paired)
plt.axis('tight')
plt.scatter(X[:,0],X[:,1],c=y)

#plt.savefig("figures/tp3-surface.pdf")
plt.show()
```