UNIVERSITÉ SORBONNE PARIS NORD | Institut GALILÉE Université Sorbonne Paris Nord

Master EID²

Exploration Informatique des Données et Décisionnel
UE Science des Données Numériques
Fiche TP N° : 4

Les TP sont notés. Les étudiants doivent sauvegarder le code de chaque exercice et doivent présenter le résultat au chargé du TP.

L'objectif de ce TP est de programmer et de tester deux algorithmes de classification, très simples mais très efficaces : l'algorithme du Plus Proche Voisin (PPV) et le Classifieur Bayesien Naïf (CBN). Nous n'étudions ici que les versions les plus simple de ces algorithmes.

Pour ce TP nous aurons besoin d'importer sklearn et numpy. Les test pourront se faire sur les données (data) prédéfinies de sklearn qui sont fournies avec leurs étiquettes de classe (target), par exemple :

```
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

Plus Proche Voisin

L'algorithme du Plus Proche Voisin est un algorithme très simple de classification qui repose sur le principe suivant : la classe de chaque donnée de test (à classer) doit être la classe de la donnée la plus proche (le plus similaire) parmi les données d'apprentissage.

Liste de fonctions utiles :
- `metrics.pairwise.euclidean_distances` : calcule des distances entre données.
- `argsort`, `argmin`, `argmax` : renvoie les indices des valeurs ordonnées, minimales et maximales.
- `neighbors.KNeighborsClassifier` : l'algorithme des K Plus Proches Voisins de sklearn.

1. Créez une fonction PPV(X,Y) qui prend en entrée des données X et des étiquettes Y et qui renvoie une étiquette, pour chaque donnée, prédite à partir du plus proche voisin de cette donnée. Ici on prend chaque donnée, une par une, comme donnée de test et on considère toutes les autres comme données d'apprentissage. Cela nous permet de tester la puissance de notre algorithme selon une méthode de validation par validation croisée (cross validation) de type "leave one out".
2. La fonction PPV calcule une étiquette prédite pour chaque donnée. Modifiez la fonction pour calculer et renvoyer l'erreur de prédiction : c'est à dire le pourcentage d'étiquettes mal prédites.

3. Testez sur les données Iris.
4. Testez la fonction des K Plus Proches Voisins de sklearn (avec ici $K = 1$). Les résultats sont-ils différents ? Testez avec d'autres valeurs de K.
5. **BONUS** : Modifiez la fonction PPV pour qu'elle prenne en entrée un nombre K de voisins (au lieu de 1). La classe prédite sera alors la classe *majoritaire* parmi les K voisins.

Classifieur Bayesien Naïf

L'algorithme du Classifieur Bayesien Naïf est un algorithme de classification basé sur le calcul de probabilité d'appartenance à chaque classe. C'est à dire que la donnée de test (à classer) sera affectée à la classe la plus probable. Les probabilités d'appartenances à chaque classe sont calculés à partir des données d'apprentissage de la façon suivante :

$$classe(x) = argmax_{\omega_k}\{\prod_i P(x_i/\omega_k)P(\omega_k)\} \quad (1)$$

Ici $P(\omega_k)$ est la probabilité *à priori* d'appartenir à la classe k. autrement dit c'est la probabilité d'obtenir une donnée de la classe k si on tire une donnée au hasard. $P(x_i/\omega_k)$ est la probabilité qu'une donnée x ait la valeur x_i pour la variable i, si on connaît sa classe ω_k. Ici nous allons calculer cette probabilité en calculant la distance entre la donnée x_i et chaque barycentre des classes (c'est à dire la moyenne de la classe), divisée par la somme des distances entre cette donnée et chaque barycentre.

Liste de fonctions utiles :
- `mean`, `sum` : calculent la moyenne et la somme d'une liste de valeur.
- `unique` : renvoie la liste des valeurs d'une liste, mais sans répétitions de valeurs.
- `asarray` : transforme une liste en vecteur.
- `nom_d_un_vecteur.prod` : fait le produit des valeurs d'un vecteur.
- `naive_bayes.GaussianNB` : l'algorithme du Classifieur Bayesien Naïf de sklearn.

1. Créez une fonction CBN(X,Y) qui prend en entrée des données X et des étiquettes Y et qui renvoie une étiquette, pour chaque donnée, prédite à partir de la classe la plus probable selon l'équation (1). Ici encore, on prend chaque donnée, une par une, comme donnée de test et on considère toutes les données comme données d'apprentissage. Il est conseillé de calculer d'abord les barycentres et les probabilités à priori $P(\omega_k)$ pour chaque classe, puis de calculer les probabilités conditionnelles $P(x_i/\omega_k)$ pour chaque classe et chaque variable.
2. La fonction CBN calcule une étiquette prédite pour chaque donnée. Modifiez la fonction pour calculer et renvoyer l'erreur de prédiction : c'est à dire le pourcentage d'étiquettes mal prédites. Testez sur les données Iris.
3. Testez la fonction du Classifieur Bayesien Naïf inclut dans sklearn. Cette fonction utilise une distribution Gaussienne au lieu des distances aux barycentres. Les résultats sont-ils différents ?
4. **BONUS** : Modifiez la fonction CBN pour qu'elle utilise une distribution Gaussienne pour les lois de probabilités au lieu de simple distance au barycentre.