TP : Base de l'apprentissage
La Classification : K-NN

Pierre Mahé
pierre.mahe@univ-tln.fr

21 novembre 2022

Dans ce TP, nous allons nous intéresser au fonctionnement de classification de données et plus particulièrement au K-NN.

Les TPs seront notés, ils seront donc a terminer et à envoyer à l'adresse email : pierre.mahe@univ-tln.fr. Le compte rendu devra contenir : le code, figures ainsi qu'un court un résumé des explorations ainsi que des observations.

Exercice 1

Dans cet exercice, nous allons nous intéresser à la classification binaire par K-NN. Nous allons travailler sur des données synthétiques. Ces données x sont composées de 2 features, chaque donnée pourra être représenté comme un point en 2 dimensions, à chaque donnée sera associé un : $y = \{0, 1\}$. Dans un premier temps, nous allons implémenter l'algorihtme de K-NN. Puis dans un second temps, nous allons implémenter les métriques d'évaluations de bases.

1.1) Nous allons implémenter l'algorithme de K-NN. Pour allons commencer par découper l'algorithme principal en fonction élémentaire. Une fois implémenté, nous les rassemblerons pour former l'algorithme complet. Le train set peut être généré par la fonction generate_trainset() qui retourne un tableau de points, ainsi que le label associé a chaque point. Le test set est lui générer par la fonction generate_testset(). Dans cet exercice, nous ferons abstraction du set de validation.

Les fonctions élémentaires sont :

- dist(x_1, x_2) : qui calcule la distance Euclidien entre 2 points.
- nearest_neighbor(k, dataset, x) : qui retourne l'indice des K points les plus proches de x.
- dominant label(dataset, y) : qui retour le label majoritaire dans un set de données.
- display_classification(dataset, y) : qui affiche le dataset ainsi que le label associé pour chaque point. Cet affichage est fait à l'aide de la fonction plt.scatter de matplotlib, utiliser l'argument color ou marker pour différentier les 2 classes.

Note : les fonctions et prototypages sont donnés à titre indicatif, ils peuvent être modifiés au besoin.

1.2) Utilisez ces fonctions pour implémenter l'algorithme complet de K-NN. Vérifier le bon fonctionnement du modèle en utilisant le dataset retourné par la fonction generate_testset().

1.3) Affichez le résultat pour K = 1, 3, 7, comparer les résultats différents, quelles observations pouvez vous en faire ?

1.4) Pour évaluer les performances de votre modèle, nous allons maintenant implémenter les métriques suivantes :

- Accurracy
- Precision
- Recall

Quels scores obtenez-vous ? Faite varier le nombre de points support K, quelle valeur de K vous semble préférable ?

1.5) Vérifiez votre implémentation des métriques d'évaluations en utiliser la bibliothèque scikit-learn. Si vos codes fonctionne correctement, les résultats devraient être identiques.

1.6) Implémentez la variante du K-NN : la Distance-weighted Nearest Neighbor. Pour rappel, dans cette variante, le vote du label majoritaire des K plus proche voisins est pondéré par l'inverse de la distance aux carrées.

Exercice 2

Dans cet exercice, nous allons mettre en place un traitement du début à la fin. Le but de l'exercice est d'implémenter un classifier pour déterminer l'espèce de pingouins. Le dataset sera composé de 3 espaces de pingouins : Adélie, Chinstrap, Gentoo. Pour effectuer cette tâche, le classifier aura à sa disposition 4 caractéristiques pour chaque pinguins : la masse, la longueur des ailes, la longueur et la hauteur du bec. Tout d'abord, nous allons préparer le dataset et nous familiariser avec ce dernier.

2.1) Avant tout entraînement, il faut étudier les dataset : calculez le nombre d'échantillons par classe, calculez la moyenne et l'écart-type pour chacune des classes. Vous afficherez aussi les différents points selon leurs certaines de leurs caractéristiques pour voir s'il semble y avoir des caractéristiques plus discriminante que d'autre.

2.2) Pour pouvoir une recherche d'hyper-paramètres rigoureuse et une évaluation des performances correctes. Découpez le dataset en trois dataset : train set, valid set et test set. Veuillez a fabriquer un test set pertinant pour la suite (d'une taille suffisante, balancé...).

2.3) En se basant sur l'exercice 1, programmez une classification binaire par KNN avec N dimensions. Attention, pour le tester, il faudra n'utiliser uniquement pas 2 classes et non 3. Pour cela, vous pouvez retirer un espace du dataset, juste pour cette question, ou bien utiliser la fonction make_blobs de sklearn.

2.4) Étendez maintenant, le classifier programmé à la question précédente pour qu'il puis faire une classification pour les 3 espèces avec l'approche One-vs-All.

2.5) Pour évaluer les performances de votre modèle, programmé une fonction pour calculer la matrice de confusion et l'afficher.

2.6) Utilisez le validation set pour déterminer le K optimal. Quelles sont mes métriques qui vous semble les plus pertinente à utiliser ?

2.7) Maintenant, nous allons faire l'évaluation de votre modèle, quelle performance obtenez-vous sur le test set ? Tracer la matrice de confusion pour le test set.