Projet complémentaire
Développement en C++ d'un alogrithme des *k plus proches voisins* pour la classification de chiffres

Christian Raymond & Éric Anquetil

10 octobre 2019

1 Introduction

L'objectif est de programmer en C++ de A à Z un classifieur de type kppv (k plus proches voisins). Un classifieur est un outil qui permet d'apprendre, à partir de données, à classer des objets. Dans notre exemple, l'objectif est de reconnaître à partir de caractéristiques extraites d'un tracé sur tablette tactile, le chiffre qui a été écrit par l'utilisateur. Un classifieur est alors en général entraîné avec des données (d'apprentissage) pour lesquels nous connaissons la classe (ici le chiffre de 0 à 9 qui a été dessiné) et il apprend à faire le lien entre ce qu'il observe, les caractéristiques (ici, de type courbure, centre de gravité, points singuliers, *etc.*) et la classe. Un kppv est un classifieur qui fait parti de la famille des classifieurs paresseux : il ne produit pas de modèle à partir des données : lorsque on l'interroge sur une nouvelle instance à classer, il la compare (mesure de distance ou de similarité) avec toutes les instances de sa base de référence (base d'apprentissage) et renvoi comme étiquette l'étiquette obtenue par le vote des k plus proches voisins (vote éventuellement pondéré par la distance entre le voisin et l'instance en question).

2 Objectif du projet C++

L'objectif est de développer un classifieur knn en C++ qui fonctionne en ligne de commande. Ce classifieur devra prendre en entrée, au minimum, deux fichiers et le paramètre k :

1. un fichier contenant les données d'apprentissage (instances de références)
2. un fichier contenant les données de test, des exemples que l'on veut étiqueter automatiquement et sur lesquelles on veut évaluer la performance de votre classifieur (pourcentage de bonnes étiquettes prédites)
3. k, le nombre de voisin que l'on fait voter pour la prédiction d'étiquette

Le classifieur devra :

— lire et stocker les données en mémoire
— faire la prédiction de la classe de chaque exemple de test suivant le principe du knn
— produire une sortie (console ou fichier) de l'étiquette prédite accompagné de sa probabilité
— produire une évaluation objective du classifieur :

1. le pourcentage de bonne classification sur la base de test
2. une matrice de confusion montrant les confusions du classifieurs (la diagonale indique les bonnes réponses)

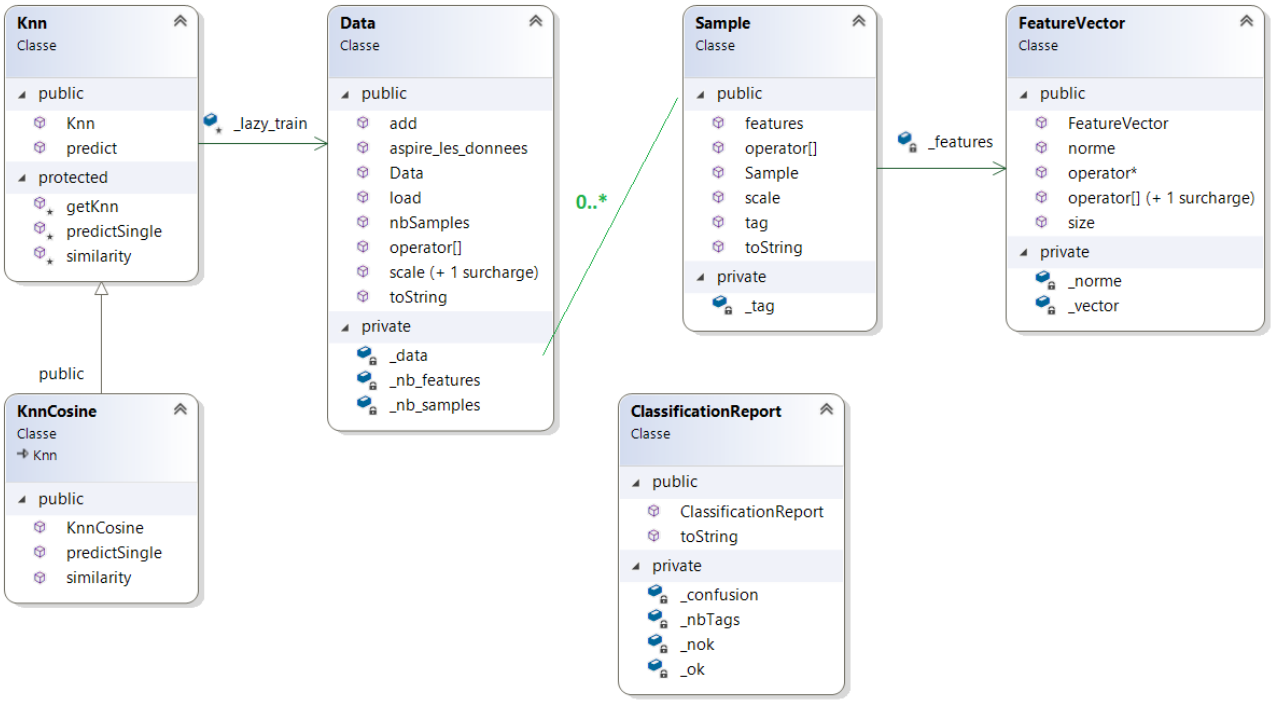

FIGURE 1 – schéma UML de modélisation suggérée

3 Données

Nous fonctionnerons comme un challenge scientifique : nous vous donnerons le fichier d'apprentissage, le fichier de test sera dévoilé à la fin du projet pour évaluation finale et classement des différents participants en fonction de l'efficacité de leur kppv.

Les données seront sous la forme d'un fichier texte :

1. les deux premières lignes donnent la quantité de données à lire :
   (a) la première ligne contient un nombre i : le nombre de ligne de données
   (b) la deuxième ligne contient un nombre j : le nombre de caractéristiques qui décrivent chaque exemple (tracé sur tablette)
2. les lignes suivantes sont les données elles-mêmes, elles sont constituées de valeurs numériques séparées par des blancs : la première valeur, un numéro de 0 à 9 indique la classe, les autres valeurs sont les j caractéristiques extraites du tracé qui caractérisent l'exemple.

4 Proposition de modélisation

Afin de vous faciliter la tâche, nous vous suggérons une modélisation dans la figure 1 : un découpage entre classe, celui-ci n'est pas figé et vous pouvez librement l'adapter.

FeatureVector : l'objet est prévu pour stocker un vecteur de caractéristique et proposer des méthodes pour l'exploiter, par exemple des méthodes permettant de calculer sa norme, de calculer des produits scalaires, *etc.*

Sample : est prévu pour stocker un exemple : le vecteur de caractéristique ainsi que sa classe (étiquette)

Data : est prévu pour stocker un corpus complet, celui d'apprentissage ou de test

Knn : est la classe principale de l'application, elle met en œuvre le principe du kppv à travers ses méthodes. Elle est abstraite car les méthodes sur lesquelles un kppv est ajustable sont

virtuelles : notamment la fonction de comparaison entre deux exemples. En effet quelle va être la meilleure fonction de comparaison entre deux exemples pour notre problème ? une distance l2 ? une similarité cosine, ce sera à vous d'en proposer et d'en tester.

KnnCosine : est une exemple de réalisation concrète de Knn implémentant la comparaison utilisant la similarité Cosine.

ClassificationReport : est une classe permettant de comparer une liste de prédictions avec une liste d'étiquettes connues pour évaluer la performance du classifier, elle les compare et calcule le pourcentage de réussite ainsi que la matrice de confusion.

5 Livrables

Vous devrez rendre (sur moodle) avant la dernière séance qui servira de créneau d'évaluation :

1. votre code C++ (seulement les sources) accompagné d'un Makefile permettant de le compiler facilement
2. une présentation électronique (format pdf) d'environ 6 pages qui serviront à la présentation de votre projet présentant entre autres :
   — le modèle UML de votre solution
   — le détail de votre meilleur kppv (quel k, quelle mesure de distance/similarité, stratégie de vote, *etc.*)
   — les résultats obtenus