Rapport Projet K-nn

Nicolas Vanderstigel

I. Base

Afin de réaliser ce projet je me suis basés sur le K-nn réalisé lors du TD5 pour les iris de fleurs. J'avais à ce moment-là décidé d'utiliser une classe pour manipuler plus facilement les coordonnées de chaque fleur, c'est pourquoi j'ai à nouveau travaillé avec une classe bien que cela ne soit pas forcément nécessaire je trouvais cela plus lisible.

II. Améliorations apportées

Une fois mon algorithme K-nn du TD5 récupéré je l'ai testé et adapté pour ce nouveau problème (6 coordonnées et non plus 4) et la précision n'était pas au rendez-vous. Je savais que le gros de mon travail aller se concentrer là-dessus.

J'ai donc décidé de l'améliorer via 3 axes : vitesse d'exécution (bien que cela ne soit pas pris en compte je voulais avoir un algorithme bien optimisé), choix de la structure du vote et précision du K.

Cela m'a permis d'obtenir une précision sur *PreTest.csv* et data.csv entre 85 et 90%.

1. Distance euclidienne ou de Manhattan ?

Ma première distance était la distance euclidienne mais j'ai voulu tester mon algorithme avec la distance de Manhattan (que j'avais vu sur internet). Obtenant des résultats similaires (moyenne de 82.24% pour euclidienne et 82.16% pour Manhattan), j'ai finalement opté pour la distance euclidienne bien qu'il n'y a que très peu de différence entre les deux. La distance de Manhattan aurait été plus intéressante si nous avions eu 20 dimensions car elle converge plus rapidement pour plus de dimensions. J'ai décidé de supprimer la partie du code sur la distance de Manhattan car je ne m'en servais pas du tout. Cependant cela a été une nouvelle façon de calculer une distance que j'ai découvert.

2. Un vote pondéré

J'avais également la possibilité d'agir sur la manière dont je sélectionnais le label final et ce via un système de vote. Le premier système était basé sur le nombre d'occurrences de chaque label et je sélectionnais celui ayant le plus d'occurrences.

Cependant ce système facile à mettre en œuvre offre le même poids au label de dernière position et à celui de première position. J'ai donc décidé de pondérer cette valeur en ajoutant un poids. Ainsi la première valeur du tableau gardant les K meilleurs résultats aura un poids bien plus important que la dernière. Cela permet d'avoir plus de précision car la 18eme fleur la plus proche de notre fleur inconnue a moins de chance d'être du même type que la fleur inconnue que la 1ere fleur la plus proche.

3. Amélioration du K

Afin de garder le K offrant la meilleure précision, j'ai opté pour une approche brute en testant de déterminer les classes de notre fichier *preTest.csv* à partir du fichier *data.csv*, cela m'a permis de déterminer quel k était le plus précis. En faisant varier la valeur de notre K (entre 3 et 18) via une boucle *for ()*, j'ai finalement gardé k = 18 comme le k me donnant la plus grande précision.

Je suis assez satisfait de la vitesse d'exécution totale de mon code. Car en entrainant mon algorithme pour 15 k différents, en testant le meilleur k sur les deux fichiers donnés et en remplissant les classes inconnues de notre fichier finalTest.csv, je compile le tout en moins de 8 minutes. J'en suis assez satisfait.

N'hésitez pas à regarder mon code car j'ai rajouté beaucoup de commentaires permettant de comprendre mon code sans trop de difficultés.