

Université de Tours – Département informatique – Antenne de Blois

Année universitaire 2019-2020

Reconnaissance et forme d’analyse d’image

K- Plus Proche Voisins (KNN)



Réalisé par : Ridha Kchouk Supérvisé par : Nicolas Ragot

# Introduction

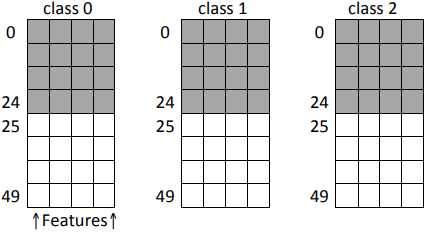
Les k plus proches voisins, est une méthode d’apprentissage supervisé. En abrégé k-NN ou KNN, de l'anglais k-nearest neighbors. En reconnaissance et forme l'algorithme des k plus proches voisins (k-NN) est utilisée pour la classification et la régression.

Dans le cadre de ce projet, nous disposons de données d’apprentissage constitué par un couple (vecteur) de cinq chaque couple désigne des caractéristiques appartenant à une classe de fleur (les iris de Fisher). Les quatre premiers éléments correspondent aux caractéristiques des fleurs (longueur et largeur des sépales, longueur et largeur des pétales) tandis que le dernier élément du couple représente l’espèce (sa classe) de la fleur respectivement.

Le but de ce projet est de monté étape par étape un algorithme K-NN qui aura pour but de classer l'entrée dans la catégorie à laquelle appartient les k plus proches voisins dans l'espace des caractéristiques identifiées par apprentissage.

La première question concernant le TP ne sera pas détaillée dans ce rapport pour la raison a laquel il s’agit de créer un projet java et d’y rajouter une classe sur lequel sera édité le code source.

Les données seront découpées sous forme de deux catégories soit un jeu d’apprentissage et un jeu de d’entrainement, voici une description des données :





# 1ère partie : k-PPV

Pour commencer il faut créer une fonction prend en entrée un vecteur, et un tableau de résultat vide qui sera retourné en résultat contenant les distances par rapport aux autres vecteurs du jeu de données.

La distance euclidienne est calculée de la façon suivante :

Par la suite, une deuxième fonction est nécessaire. La deuxième retourne la classe d'un vecteur en utilisant le tableau des distances précédemment calculé. Cette classe est celle correspondant à la classe du voisin le plus proche (correspondant à valeur minimale dans le tableau des distances). Une fois que la classe d’un vecteur est connu il est désormais possible de faire une matrice de confusion sur lequel sera calculer le taux de reconnaissance et sont pourcentage d’erreur.

Une troisième fonction consistera à retourner la classe majoritaire des K plus proche voisin cela se fera en triant le tableau des distances en ordre croissant ainsi les K premier élément seront donc les K voisins plus proche. Le trie du tableau de distance se fait toujours par rapport à un vecteur X. Une fois le tableau obtenu il faut connaitre quelle classe est majeur (apparaissant le plus) et la retourné. Si les classes apparaissent le même nombre de fois alors une de ces classes sera choisie aléatoirement.  
L’algorithme est le suivant :

Soit «  » un ensemble de données

Soit une fonction de définition de distances « d », (distance euclidienne).

Soit un entier « K »

Pour une nouvelle observation « X » (notre vecteur) dont on veut prédire sa variable de sortie « y » il faut :

1. Calculer toutes les distances par rapport au vecteur « X » du jeu de donnée « D »
2. Retenir les « K » observation du jeu de données « D » les proches de « X » en utilisant la fonction de calcul de distance « d »
3. Prendre les « y » des « K » observation retenues.
4. Les « y » (sortie) représente les classes du jeu de données, sélectionner la classe majoritaire
5. Retourner la valeur calculée dans l’étape 4 comme étant la valeur qui a été prédite par K-PPV pour l’observation X

# 2ème partie : Weka

La seconde partie du rapport consistera à analyser ces données sous Weka qui est une suite de logiciels d’apprentissage automatique. Sur Weka nous verrons la méthode d’apprentissage supervisé K-NN ainsi qu’un réseau de neurone.

Les paramètres d'un réseau de neurones sont généralement les poids des connexions. Dans ce cas, ces paramètres sont appris pendant la phase de formation. Ainsi, l'algorithme lui-même (et les données d'entrée) ajuste ces paramètres.

Les hyper paramètres sont généralement le taux d'apprentissage, la taille des lots ou le nombre d'époques (Epoch). Ils sont appelés "hyper" parce qu'ils influencent la façon dont les paramètres seront appris. Il est possible d’optimiser ces hypers paramètres comme on le souhaite (en fonction de vos possibilités) : recherche par grille, recherche aléatoire, à la main, à l'aide de visualisations... L'étape de validation permet à la fois de savoir si nos paramètres ont été suffisamment appris et de savoir si les hyper paramètres sont bons. Par conséquent avoir « 3 hidden units » n’est pas forcement obligatoire, agir sur le nombre de « hidden layers » fera qu’influencer légèrement le résultat sur le taux erreur de classification sur ce jeu de données.

# Documentation et code source

Une documentation java doc a été généré pour k-PPV, détaillant les fonctions et leurs paramètres. Cette documentation est disponible dans le répertoire « doc » du projet (kPPV.html)

Le code source est disponible sur GitHub ici : [MASTER SOURCE CODE](https://github.com/ridhakchouk/kPPV.git)