TD-COURS : Algorithme des k plus proches voisins

I-Introduction

L'algorithme des k plus proches voisins appartient à la famille des algorithmes d'apprentissage automatique (machine learning).

L'idée d'apprentissage automatique ne date pas d'hier, puisque le terme de machine learning a été utilisé pour la première fois par l'informaticien américain Arthur Samuel en 1959.

Les algorithmes d'apprentissage automatique ont connu un fort regain d'intérêt au début des années 2000 notamment grâce à la quantité de données disponibles sur internet.

L'algorithme des k plus proches voisins est un algorithme d'apprentissage supervisé, il est nécessaire d'avoir des données labellisées. À partir d'un ensemble E de données labellisées, il sera possible de classer (déterminer le label) d'une nouvelle donnée (donnée n'appartenant pas à E).

LE PRINCIPE : « Dis moi qui sont tes voisins et je te dirais qui tu es »

Ce genre d'algorithme permet par exemple de prédire le comportement d'une personne en s'intéressant à son milieu.

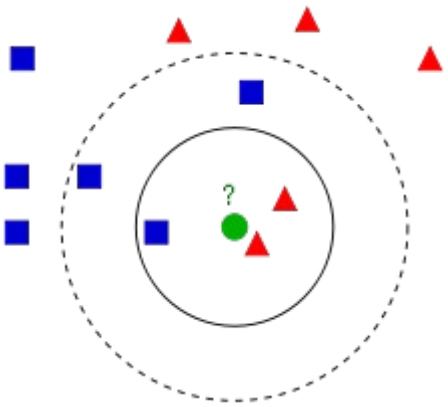
Il peut être utilisé par les géant de la vente sur Internet pour prédire si vous seriez intéressé par un produit.

En effet, en disposant de vos données (age, sexe, derniers achats, historique de navigation…..) et en les comparant à celles des autres clients qui ont ou non acheté le produit, l'algorithme peut essayer de prédire si vous seriez ou non intéressé par le produit en vous classant dans la classe de ceux qui l'ont acheté ou dans celle de ceux qui ne l'ont pas acheté.

Cet algorithme peut également être utilisé par les réseaux sociaux afin de proposer aux utilisateurs du contenu ciblé.

Exemple :

Question : A quelle classe appartient le point vert ? Celle des carré bleus ou celle des triangle rouge ?



L'échantillon de test (point vert) pourrait être classé soit dans la première classe de carré bleu ou la seconde classe de triangles rouges.

Si l'on considère les 3 plus proches voisins soit k = 3 (cercle en ligne pleine) il est affecté à la seconde classe car il y a deux triangles et seulement un carré dans le cercle considéré.

Si l'on considère les 5 plus proches voisins soit k = 5 (cercle en ligne pointillée) il est affecté à la première classe (3 carrés face à deux triangles dans le cercle externe).

A retenir :
L'algorithme des k plus proches voisins consistent donc à regarder la classe des k plus proches voisins d'un élément et de le ranger dans la classe majoritairement représentée parmi ses k plus proches voisins.

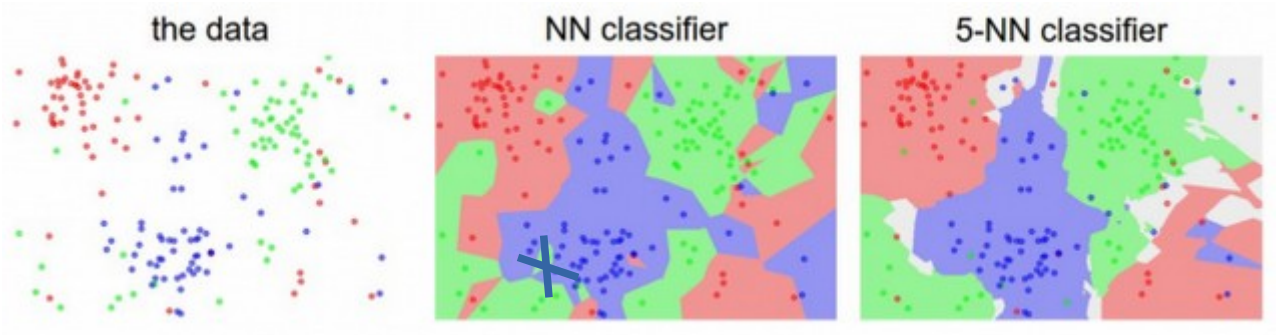
Choix du nombre k de plus proches voisins :

Le choix de la valeur k à utiliser pour effectuer une prédiction avec k-NN (k nearest neighbors en anglais), varie en fonction du jeu de données.

En règle générale, moins on utilisera de voisins (un nombre k petit) plus on sera sujet au sous apprentissage (underfitting).

Par ailleurs, plus on utilise de voisins (un nombre k grand), plus on sera fiable dans notre prédiction.

Toutefois, si on utilise k nombre de voisins avec k = N et N étant le nombre d'observations, on risque d'avoir du overfitting et par conséquent un modèle qui se généralise mal sur des observations qu'il n'a pas encore vu.



L'image ci-dessus à gauche représente des points dans un plan 2D avec trois types d'étiquetages possibles (rouge, vert, bleu).

Pour le 5-NN classifieur, les limites entre chaque région sont assez lisses et régulières.

Quant au N-NN Classifier, on remarque que les limites sont "chaotiques" et irrégulières. Cette dernière provient du fait que l'algorithme tente de faire rentrer tous les points bleus dans les régions bleues, les rouges avec les rouges etc… c'est un cas d'overfitting.

Par un exemple, un nouveau point à l'endroit désigné par la croix sera classer en vert par le N-NN classifieur alors qu'il semble beaucoup plus logique de le classer bleu.

Pour plus de précision sur le sujet du machine leaurning, on pourra visionner la vidéo :

II-Méthodologie

Nous allons dans ce TD-COURS nous intéresser à prédire la classe d'un nouvel élément dont la position est connue à partir d'une liste d'éléments dont les coordonnées et les classes sont connues.

Pour cela, il faudra :

1. Évaluer la distance qui sépare ce nouvel éléments de chaque éléments de la liste.
2. Stocker ses valeurs de distances à chaque élément de la liste dans une liste de distances.
3. Choisir les k plus proches éléments de la liste de notre nouvel élément. (Les k plus proches voisins)
4. Assigner une classe au nouvel élément à partir de la classe majoritairement représentée dans les k plus proches voisins.

Concrètement au niveau de la programmation, il faudra créer 3 fonctions.

1	Une fonction **`distance`** qui calculera la distance entre deux éléments dont on connaît toutes les caractéristiques. Cette fonction prendra en argument deux éléments et retournera un nombre qui sera la distance entre ces deux éléments.
2	Une fonction **`kplusprochesvoisins`** qui déterminera les k plus proches voisins d'un nouvel élément. Cette fonction prendra en argument la listes des éléments connus, le nouvel élément et le nombre de plus proches voisins que l'on souhaite sélectionner. Elle retournera une liste composée des k plus proches voisins de notre nouvel élément.
3	Une fonction **`predire`** qui déterminera la classe du nouvel élément. Cette fonction prendra en argument la listes des éléments connus, le nouvel élément et le nombre de plus proches voisins que l'on souhaite sélectionner. Elle retournera la classe supposée du nouvel élément.

III-Présentation du problème que nous allons traiter avec l'algorithme des k plus proches voisins

Vous disposez d'un fichier `Parc_Naturel.csv` déposé dans Accès à NSI. Ce fichier contient une liste de villes ou villages de la région Auvergne avec leurs coordonnées GPS (latitude et longitude) ainsi qu'une mention sur leur appartenance à un parc naturel (celui des volcans d'auvergne, celui du livradois Forez ou aucun des deux).

Les premières lignes du fichier sont donc :

```
"Localité","Lattitude","Longitude","Parc naturel"
"Auzon",45.3904, 3.3708,"Parc naturel du livradois forez"
"Vezezoux",45.4034, 3.3471,"Aucun"
"Apchat",45.3781, 3.1612,"Parc naturel des volcans d'auvergne"
```

Le but de ce TD-COURS sera donc d'implémenter l'algorithme des k plus proches voisins afin de déterminer l'appartenance ou non de la ville de Sainte-Florine dont la latitude est 45,4021° et la longitude 3,3176° à un des parc naturel.

Puis même question pour la ville de Saint Germain l'Herm de latitude 45,4572° et de longitude 3,545° et pour la ville de Saint Victor la rivière de latitude 45,5502° et de longitude 2,9412°.

1. Vous allez donc commencer votre programme par lire le fichier csv et créer une liste nommée **`echantillon`** dont chaque élément est une ligne du fichier csv.

IV-La fonction distance

L'algorithme des kppv utilise la notion de distance. Plusieurs distances peuvent être utilisées comme des distances entre des mots en comptant le nombre de lettres communes à deux mots ou encore des distances sémantiques sur le sens des mots (Chat sera proche de félin car presque synonyme), ou plus simplement des distance géométriques.

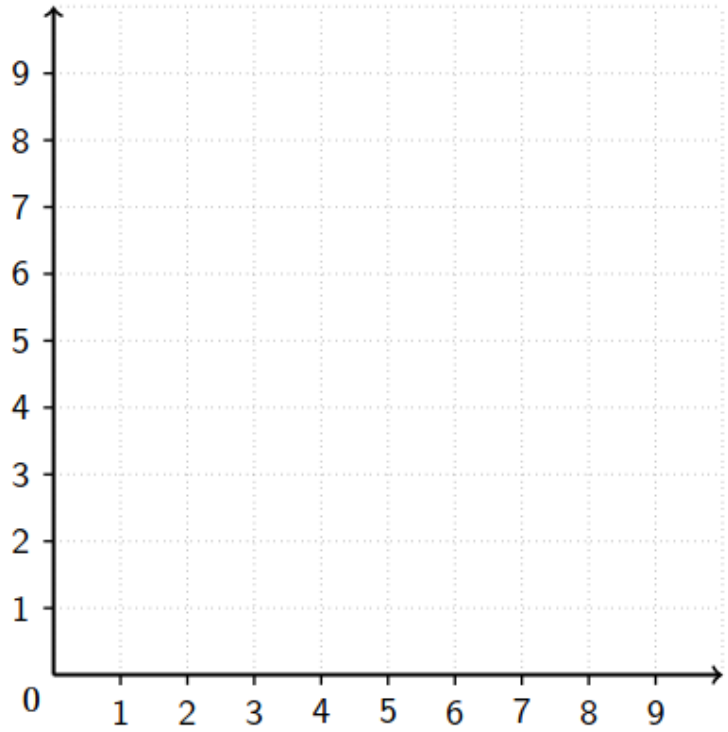
Dans notre problème, nous utiliserons la distance euclidienne calculer à partir des coordonnées des villes. Il est certes pas mathématiquement rigoureux de calculer des distances euclidiennes à partir de

coordonnées GPS qui sont des angles mais pour simplifier notre problème nous nous satisferont de cette imprécision.

Distance euclidienne :

Rappel de mathématiques : Distance euclidienne :
Soit deux points A et B de coordonnées respectives $(x_A ; y_A)$ et $(x_B ; y_B)$.
On appelle distance euclidienne entre A et B, notée AB, le nombre réel positif défini par :

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$



1. Placer sur le repère ci-dessus les points A(1;1) ; B(1 ; 5) ; C(5 ; 1) ; D(5 ; 5) et E(7 ; 9).

2. Calculer les distances euclidiennes AB, AC ,BC, DA, EA, CE et EB.

AB =
AC =
BC =
DA =
EA =
CE =
EB =

Il existe également d'autre distances géométriques, par exemple :

Distance de Tchebychev:
Soit deux points A et B de coordonnées respectives $(x_A ; y_A)$ et $(x_B ; y_B)$.
On appelle distance de Tchebychev entre A et B, notée $d_{Tch}(AB)$, le nombre réel positif défini par :

$$d_{Tch}(A,B) = \max(|x_B - x_A| ; |y_B - y_A|)$$

3. Calculer les distances de Tchebychev $d_{Tch}(A,B)$, $d_{Tch}(A,C)$,$d_{Tch}(B,C)$, $d_{Tch}(D,A)$, $d_{Tch}(E,A)$, $d_{Tch}(C,E)$ et $d_{Tch}(EB)$.

$d_{Tch}(A,B)$ =
$d_{Tch}(A,C)$ =
$d_{Tch}(B,C)$ =
$d_{Tch}(D,A)$ =
$d_{Tch}(E,A)$ =
$d_{Tch}(C,E)$ =
$d_{Tch}(E,B)$ =

Fonction distance :

Créer une fonction **`distance`** qui calculera la distance euclidienne entre deux villes de la liste **`echantillon`**.
Cette fonction prendra en argument deux éléments de la liste **`echantillon`** et retournera un nombre flottant qui sera la distance euclidienne entre ces deux éléments (villes).

V-La fonction kplusprochevoisins

Cette fonction prendra en argument la liste de donnée **`echantillon,`** le nouvel élément que l'on cherche à classer **`x`**, et le nombre **`k`** de plus proches voisins que l'on veut selectionner.

On aura donc :

```
Def kplusprochesvoisins(echantillon,x,k) :
      ……..
      Return kppv
```

Cette fonction devra retourner une liste de k éléments nommé **`kppv`** issus de la liste **`echantillon`** qui sont les k plus proches voisins de l'élément **`x`**.

Pour réussir cela, il faudra suivre les étapes suivantes décrites dans le tableau ci-dessous :

1	Créer une liste vide nommée **kppv**.
2	Créer une liste vide nommée **liste_des_distances**.
3	Calculer la distance entre **x** et chaque élément de la liste **echantillon** puis l'ajouter dans la liste **liste_des_distances**.
3	Commencer une boucle sur le nombre **k** de plus proche voisins cherchés.
4	Boucler sur la liste **echantillon** pour chercher successivement les plus proches voisins de x. On gardera l'indice de l'élément de la liste **echantillon** qui le plus proche de **x** puis on ajoutera cet élément dans la liste **kppv**. Attention, il faut faire attention à ne pas sélectionner **k** fois le plus proches éléments mais bien à sélectionner les **k** plus proches éléments. Pour ce faire, lorsque l'on boucle sur la liste **echantillon** pour sélectionner un des **k** plus proches voisins, il faut vérifier que celui-ci n'est pas déjà dans la liste **kppv**.
5	On retourne la liste **kppv**.

Une fois votre fonction créer, tester la en faisant afficher la liste **kppv** qu'elle a retourner pour vérifier si elle a bien fait ce qui était demandé.

Pour vérification, les 5 plus proches voisins de Sainte florine sont Vezezoux, Auzon, Esteil, Vichel et Azerat.

VI-La fonction predire

Cette fonction prendra en argument la liste de donnée **echantillon,** le nouvel élément que l'on cherche à classer **x**, et le nombre **k** de plus proches voisins que l'on veut selectionner.

On aura donc :

```
Def predire(echantillon,x,k) :
    ……….
    Return parc
```

Cette fonction devra retourner le nom du parc naturel auquel devrait appartenir le nouvel élément **x**.

Pour cela cette fonction commencera par utiliser la précédente (fonction **`kplusprochesvoisins`**) pour connaître les **`k`** plus proches voisins de **`x`** puis elle regardera qu'elle est le parc naturel majoritairement représenté dans les **`k`** plus proches voisins et retournera ce résultat comme prédiction.

Pour vérification, la ville de Sainte-Florine devrait être classer dans le parc naturel « Aucun », la ville de Saint-Germain l'Herm devrait être classer dans le parc naturel « Livradois Forez » et la ville de Saint-Victor la rivière devrait être classer dans le parc naturel « Volcans d'Auvergne ».

Vous écrirez ensuite un programme principal permettant à l'utilisateur :

- de rentrer un nom de ville, une latitude et une longitude et un nombre k de plus proches voisins
- de connaître la prédiction de l'algorithme pour les valeurs entrées.

VII-Prolongement : choix de la distance

Vous allez créer une fonction **`distance_Tchebychev`** qui calcule la distance de Tchebychev entre deux ville puis vous modifierez l'algorithme pour qu'il fonctionne avec cette nouvelle distance.

Vous pourrez également modifiez le programme principal afin qu'il permettent à l'utilisateur de choisir la distance à utiliser (distance euclidienne ou distance de Tchebychev).

VIII-Prolongement : représentation graphique

En vous inspirant du code de la fonction **`Trace`** utiliser dans le TD sur la complexité des algorithmes de tri, faire en sorte d'afficher graphiquement les villes de la liste **`echantillon`** et le nouvel élément en utilisant une couleur différentes par parc naturel comme dans l'exemple ci-dessous.