Chapitre 4

Algorithme des k plus proches voisins

4.1 Heuristique

On considère une équipe de rugby professionnelle. Celle-ci est composée de quinze joueurs habilement numéroté de un à quinze que l'on subdivise en plusieurs catégories :

1$^{\text{res}}$ lignes : numéros 1 à 3 ; notés L_1.

2$^{\text{es}}$ lignes : numéros 4 et 5 ; notés L_2.

3$^{\text{es}}$ lignes : numéros 6 à 8 ; notés L_3.

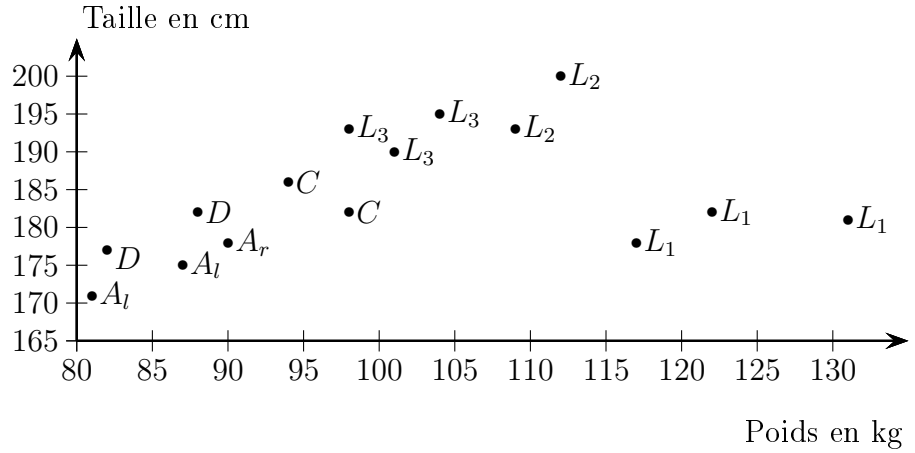
Les demis de mêlée et d'ouverture : numéros 9 et 10 ; notés D.

Les centres : numéros 12 et 13 ; notés C.

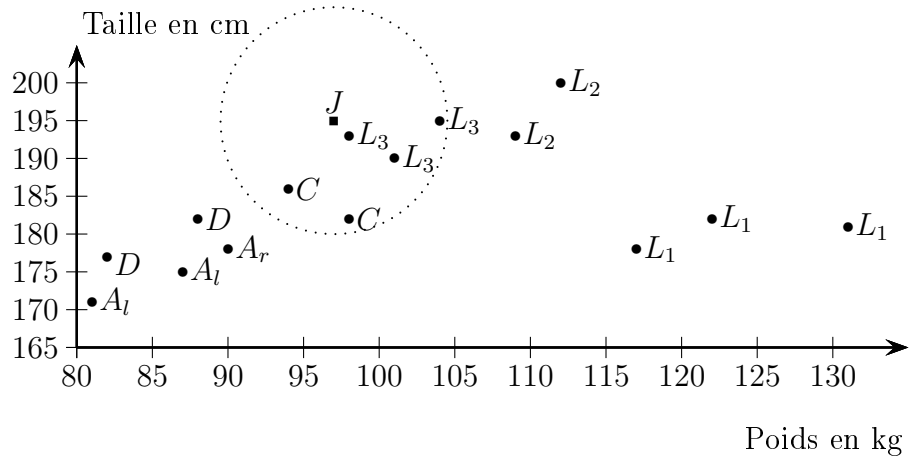
Les ailiers : numéros 11 et 14 ; notés A_l.

L'arrière : numéro 15 ; noté A_r.

Pour attribuer un poste à un joueur (outre la technique, la vitesse, l'intelligence de jeu...), deux critères *a priori* sont le poids et la taille du joueur. On considère que l'on a la répartition suivantes en fonction des postes :



On considère alors un nouveau joueur n'ayant pas encore de poste au sein de l'équipe et on cherche à lui en donner un. En première approche, on va lui affecter un poste correspondant à sa taille et son poids. Le joueur mesurant 1m93 et pesant 97kg, on le positionne à l'aide du point J dans notre graphe des répartitions.



Pour déterminer quel poste lui attribuer, on peut naturellement regarder les postes de ses plus proches voisins. En regardant ses cinq voisins les plus proches, on voit qu'il y a les deux centres et les trois troisièmes lignes ; il serait donc de façon quasi certaine sur l'un de ces deux postes et plus probablement troisième ligne.

4.2 Histoire et utilisation

Dans un rapport de la faculté de médecine aéronautique de la US Air Force publié en 1951, Fix et Hodges introduisirent une méthode pour la classification des motifs, connue depuis sous la règle des k plus proches voisins. Il s'écrit en abrégé k-NN ou KNN, de l'anglais k-nearest neightbors. Il s'agit d'un des algorithmes de machine learning essentiel dans le milieu de l'intelligence artificielle.

Le principe peut être résumé par « Dis-moi qui sont tes amis, et je te dirai 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 par exemple être utilisé par des géants de la vente afin de prévoir si vous seriez ou non intéressé par un produit. En effet, en disposant de vos données et en les comparant à celle d'un client qui a acheté un produit, un algorithme peut tâcher de prédire si vous seriez intéressé ou non par le produit. Cela vaut aussi pour les opinions, les centres d'intérêts, etc qui sont ensuite exploités par certains réseaux sociaux ou services de vidéos à la demande à des fins de personnalisations de contenus.

Bien évidemment, le champ d'application des KNN ne s'arrête aux exemple décrits ci-dessus, il en existe de nombreux autres, notamment en sciences.

De manière générale, cet algorithme peut être utilisé selon deux objectifs :

Les régressions : on calcule la moyenne des valeurs des k plus proches voisins d'un élément et on attribue cette moyenne à l'élément étudié.

Les classifications : on cherche le résultat majoritaire des classes d'appartenance des k plus proches voisins d'un élément. On attribue la classe de l'élément suivant le résultat obtenu.

Les algorithmes à base d'apprentissage apportent une réponse plausible, mais pas nécessairement exacte, à un problème auquel il est difficile d'appliquer un algorithme traditionnel.

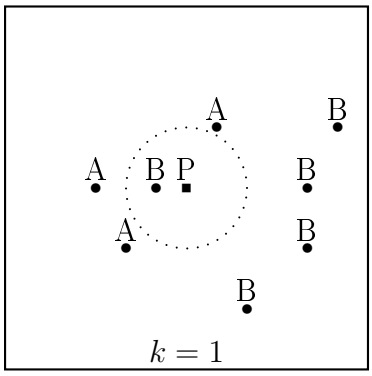
4.3 Paramétrage

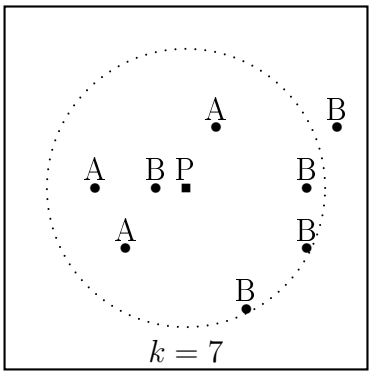
4.3.1 Le nombre de voisins k

Le choix du nombre de voisin k n'est pas anecdotique, il fait radicalement changer les résultats que procurent cette méthode :

— Si k est trop petit, on utilise peu de valeurs donc il suffit d'un intrus (le bruit) pour modifier la classification.

— Si k est trop grand, l'influence de chaque point devient faible et on prend en compte des éléments peu significatifs car trop éloignés.





Dans les deux cas ci-dessus, le nombre de voisins considérés nous donne en réponse B alors que A semblerait être la réponse la plus naturelle. Cette méthode n'a donc rien d'absolu mais fournit de bonnes premières approches.

4.3.2 Distances

Les distances sont des fonctions mathématiques prenant en argument deux objets et donnant en sortie un réel représentant le degré de similarité entre eux. Il existe de nombreuses distances et le choix de celle-ci n'est pas anodin. Certaines prennent en compte tous les paramètres équitablement, d'autres donnent plus de poids à certaines données. En voici quelques exemples classiques.

Distance euclidienne : en dimension deux avec $A\,(x_A\,;y_A)$ et $B\,(x_B\,;y_B)$, on a :

$$d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}.$$

En dimension d quelconque avec $X\,(x_1\,;\ldots\,;x_d)$ et $Y\,(y_1\,;\ldots\,;y_d)$, on a :

$$d(X,Y) = \left(\sum_{k=1}^{d}(x_k - y_k)^2\right)^{\frac{1}{2}}.$$

Distance de Manhattan : en dimension deux :

$$d(A,B) = |x_A - x_B| + |y_A - y_B|,$$

et en dimension d :

$$d(X,Y) = \sum_{k=1}^{d}|x_k - y_k|.$$

Aussi appelée taxi-distance, c'est la distance entre deux points parcourue par un taxi lorsqu'il se déplace dans une ville où les rues sont agencées selon un réseau ou quadrillage comme à Manhattan.

Distance de Tchebychev : en dimension deux :

$$d(A,B) = \max(|x_A - x_B|, |y_A - y_B|),$$

et en dimension d :

$$d(X,Y) = \max_{1 \leq k \leq d} |x_k - y_k|.$$

C'est la distance entre deux points donnée par la différence maximale entre leurs coordonnées sur une dimension.

Distance de Hamming : notion de distance entre deux chaînes de caractères.

4.4 Exercices

Exercice 4.1. [Des mots] On se propose d'utiliser l'algorithme du plus proche voisin pour corriger un mot faux donné par un utilisateur. La distance entre deux mots, la distance de Hamming, consiste à comptabiliser le nombre de différences de caractères entre les deux mots :

— d(and,end)=1 car ces deux mots font la même longueur et ils diffèrent en un caractère.

— d(assert,finally)=7 car le plus long fait 7 caractères et ils diffèrent pour tous les caractères.

On donne ci-dessous la liste des 29 mots réservés de Python :

and	assert	break	class	continue	def	del	elif	else	except
exec	finally	for	form	global	if	import	in	is	lambda
not	or	pass	print	raise	return	try	while	yield	

1. **Calculs de distances**
   (a) Calculer la distance d(assert,except).
   (b) Citer les mots de la liste à distance 1 de « if ».
2. **Recherche du plus proche voisin**
   (a) Quel est le plus proche voisin du mot « ou » ? Justifier la réponse.
   (b) Existe-t-il des mots qui ont plusieurs « plus proche voisin » ?
   (c) Quel est le plus proche voisin de « form » ? Qu'en penser ?

Exercice 4.2. [Des iris] Nous allons utiliser un jeu de données relativement connu dans le monde du machine learning : le jeu de données « iris ». En 1936, Edgar Anderson a collecté des données sur 3 espèces d'iris : iris setosa, iris virginica et iris versicolor. Pour chaque iris étudié, Anderson a mesuré :

— la largeur des sépales ;

— la longueur des sépales ;

— la largeur des pétales ;

— la longueur des pétales ;

— l'espèce (iris setosa, iris virginica ou iris versicolor).

On trouvera 150 de ces mesures dans le fichier `iris.csv` (disponible dans ce dossier avec les images des iris) et par soucis de simplification, nous nous intéresserons uniquement à la largeur, la longueur des pétales et l'espèce d'iris. Dans ce fichier, les mesures sont en millimètres et l'espèce de l'iris est un chiffre avec 0 pour iris setosa, 1 pour iris versicolor et 2 pour iris virginica.

Dans cet exercice, on considère pour l'algorithme knn la distance euclidienne.

1. Récupérer les descripteurs du fichier iris.csv. Quels sont ceux qui nous intéressent ?
2. Construire en Python la table de données associées.
3. Afficher sur un graphique la largeur des pétales de chaque iris en fonction de leur longueur. On affichera les différentes espèces à l'aide de différentes couleurs.
4. (a) Ajouter au graphique le point de coordonnées $(20\,;5)$ et de couleur orange.
   (b) Déterminer visuellement la classe du point que vous venez de rajouter avec l'algorithme des 3 plus proches voisins.
5. (a) Ajouter au graphique le point de coordonnées $(51\,;17)$ et de couleur noire.
   (b) Déterminer visuellement la classe du point que vous venez de rajouter avec l'algorithme des 3 plus proches voisins.
6. Implémenter une fonction `classe(x,y)` qui prédit l'espèce de l'iris dont la longueur du pétale mesure `x` cm et la largeur du pétale mesure `y` cm grâce à la méthode des 3 plus proches voisins.
7. Tester votre code avec quelques points au hasard et vérifier graphiquement.
8. Modifier votre fonction pour prendre en paramètre `k` afin d'appliquer la méthode des k plus proches voisins.