Algorithme des K plus proches voisins (K-NN)

Principe

Objectif et nécessité

L'algorithme des k plus proches voisins (k-nearest neighbors : kNN en anglais) est un algorithme de machine learning. Il s'agit d'un algorithme de classification supervisé. On dispose donc d'un jeu de données labellisées. C'est-à-dire que chaque donnée à un label qui fait appartenir la donnée à un nombre fini de catégories (au moins deux).

Pour cet algorithme, il faut pouvoir définir une distance (au sens de norme en mathématiques) entre deux données. Donc les données ont des paramètres qui définiront cette distance Exemple distance Euclidienne, distance de Manhattan, distance de Levenstein, etc… .

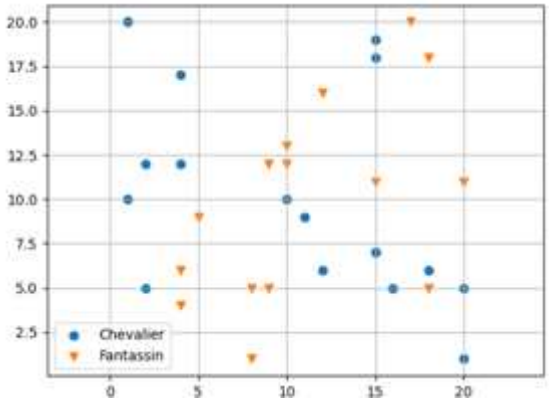
L'objectif est de définir la catégorie à laquelle appartient une nouvelle donnée dont on ne connait pas le label. Pour cela on utilise la norme entre deux données.

Principe

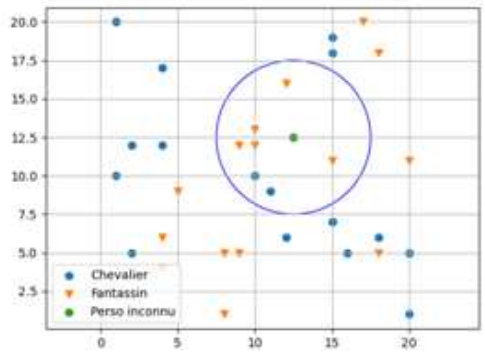
- ☞ On cherche les k plus proches voisins de la donnée à catégoriser parmi toutes les données du jeu de données d'entrainement.
- ☞ On dénombre, parmi les k plus proches voisins, le nombre d'occurrences pour chaque catégorie du jeu d'entrainement.
- ☞ La catégorie qui comptabilise le plus grand nombre d'occurrence est la catégorie à laquelle appartient la donnée que l'on cherche à labelliser.

Exemple

Dans cet exemple, on considère un jeu de rôle disposant de chevaliers et de fantassins. Chaque personnage possède une note de courage et une note de force. On peut ainsi représenter les données d'entrainement sous la forme ci-contre :



On souhaite savoir si un nouveau personnage, de courage et de force égal à 12,5 est un chevalier ou un fantassin. Dans ce cas, la méthode des k plus proches voisins (par la distance Euclidienne) est appropriée.



- ☞ **On établi la liste de toutes les distances entre la donnée non labellisée (à catégoriser) et les données du jeu d'entrainement. On retient en fait dans cette liste le couple (distance et catégorie)**
- ☞ **On trie cette liste (dans l'ordre croissant des distances) puis ont ne retient que les k premiers éléments de cette liste : Les k plus proches voisins.**
- ☞ **On dénombre pour chaque catégorie le nombre d'occurrences parmi les k plus proches voisins.**

La catégorie ayant le plus grand nombre d'occurrences parmi ces k plus proche voisins est la catégorie à laquelle on fera appartenir la donnée dont le label est inconnu.

Dans cet exemple, pour k = 7, on peut alors estimer que le personnage inconnu est un fantassin. Notez bien que si l'on choisit k = 13, le résultat est différent et le personnage inconnu peut alors être estimé être un chevalier.

L'influence du paramètre k de cet algorithme (qui est arbitraire) est donc très importante.

Performances de prédiction de l'algorithme

Objectif

L'exemple précédent nous montre que cet algorithme (comme tous les algorithmes d'IA) ne fournit qu'une prédiction. Mais ne détermine pas de manière sure la catégorie de la données de label inconnu. Il est donc intéressant de pouvoir déterminer la qualité de cette prédiction.

Principe

Pour cela il faut pouvoir disposer d'un jeu de données de test. Ce jeu de données différents du jeu de données d'entrainement est un ensemble de données pour lesquelles ont connait la catégorie pour chacune d'elles.

Pour chacune des données du jeu de test on applique l'algorithme des K plus proches voisins et l'on compare la prédiction issue de l'application de l'algorithme à la catégorie issue du label propre à chacune des données du jeu de test qui sont toues labellisées.

Plus le nombre de fois où la catégorie issue de la prédiction est identique à celle donnée par le label des données du jeu de test est importante, meilleure est la performance (ou qualité) de l'algorithme.

Score

Une 1ière grandeur pour quantifier cette performance peut-être le pourcentage de prédictions exactes :

$$\textbf{Score} = \frac{\textbf{Nombre de catégorisations exactes}}{\textbf{Nombre total du jeu de test}}$$

Matrice de confusion

Grâce à la fonction score, on peut connaître la qualité de la prédiction de l'algorithme. Cependant, il pourrait être très intéressant de savoir où les erreurs sont produites et comment. Pour cela, on va utiliser la matrice de confusion.

On définit un tableau à deux entrées qui a autant de lignes et de colonnes qu'il existe de catégories pour les données. On a donc une matrice n×n où n est le nombre de catégories.

Les valeurs $C_{i,j}$ dont les indices dans cette matrice, sont i et j, représentent le nombre d'occurrences où la catégorie issue des labels du jeu de test sont l'indice i (ligne) et la catégorie issue de la prédiction de l'algorithme sont l'indice j (colonne).

Exemple d'une matrice de confusion non normalisée :

Chaque élément du tableau, de la ligne $i \in \{1, \ldots, n\}$ et la colonne $j \in \{1, \ldots, n\}$ correspond au nombre de fois où la fonction de prédiction a produit la sortie j alors que la vraie étiquette était i.

		Catégories issues des prédictions	
		Chevalier	Fantassin
Catégories issues des labels du jeu de données test	Chevalier	45	5
	Fantassin	12	28

Exemple d'une matrice de confusion normalisée :

Même matrice mais le nombre de prédiction est divisé par le nombre d'occurrences de la catégorie dans le jeu de test. (Somme des nombres de la ligne).

		Catégories issues des prédictions	
		Chevalier	Fantassin
Catégories issues des labels du jeu de données test	Chevalier	90 %	10 %
	Fantassin	30 %	70 %