Article KNN

Antoine SERREAU / Corentin BRETONNIERE / Benjamin GUIGON

Janvier 2020

Ce travail est réalisé dans le cadre d'un projet collaboratif comprenant 3 étudiants du MSc Data Management de PSB :

Antoine SERREAU / Corentin BRETONNIERE / Benjamin GUIGON

Nous avons réalisés 2 autres travaux portant sur :

ACP et Les Arbres de Décisions

1 Introduction

Vous vous aventurez dans l'apprentissage du machine learning ? Voici une rapide introduction à l'un des algorithmes du Machine Learning - KNN - qui vous aidera à en saisir les principales dynamiques et notions mathématiques.

L'algorithme K-Nearest Neighbors (ou KNN) est l'un des algorithmes d'apprentissage les plus utilisés en raison de sa simplicité. Alors, qu'est-ce que c'est ?

KNN est un algorithme d'apprentissage non paramétrique. Il utilise des données avec plusieurs classes pour prédire la classification du nouveau point d'échantillonnage. KNN est non-paramétrique car il ne fait aucune hypothèse sur les données étudiées, c'est-à-dire que le modèle est distribué à partir des données.

KNN n'utilise pas les points de données d'entraînement pour faire des généralisations.Il y a donc peu ou pas de phase d'apprentissage explicite, la phase d'apprentissage est assez rapide, KNN conserve toutes les données relatives à l'apprentissage puisqu'elles sont nécessaires pendant la phase de test.

Etant donné que la plupart des données n'obéissent pas aux hypothèses théoriques typiques, comme lorsque nous considérons un modèle de régression linéaire, ce qui rend KNN crucial lorsque l'on étudie des données avec peu ou pas de connaissances préalables.

2 Quand et Pourquoi utiliser KNN ?

KNN peut être utilisé dans les problèmes de régression et de classification prédictive. Cependant, en ce qui concerne les problèmes industriels, il est surtout utilisé dans la classification car il se retrouve dans tous les paramètres évalués lors de la détermination de l'utilisabilité d'une technique.

- Pouvoir de prédiction
- Temps de calcul
- Facilité d'interprétation des résultats

L'algorithme KNN est utilisé en raison de sa facilité d'interprétation et de son faible temps de calcul.

Par exemple, KNN peut être utilisé dans un système bancaire pour prédire si une personne est apte à obtenir un prêt. Ou si elle présente des caractéristiques similaires à celles d'un emprunteur défaillant. Calcul de la cote de crédit - KNN peut aider à calculer la cote de crédit d'une personne en la comparant avec celle de personnes ayant des caractéristiques similaires. L'algorithme KNN est également utilisé pour la reconnaissance vidéo, la reconnaissance d'images, la détection de l'écriture manuscrite et la reconnaissance vocale.

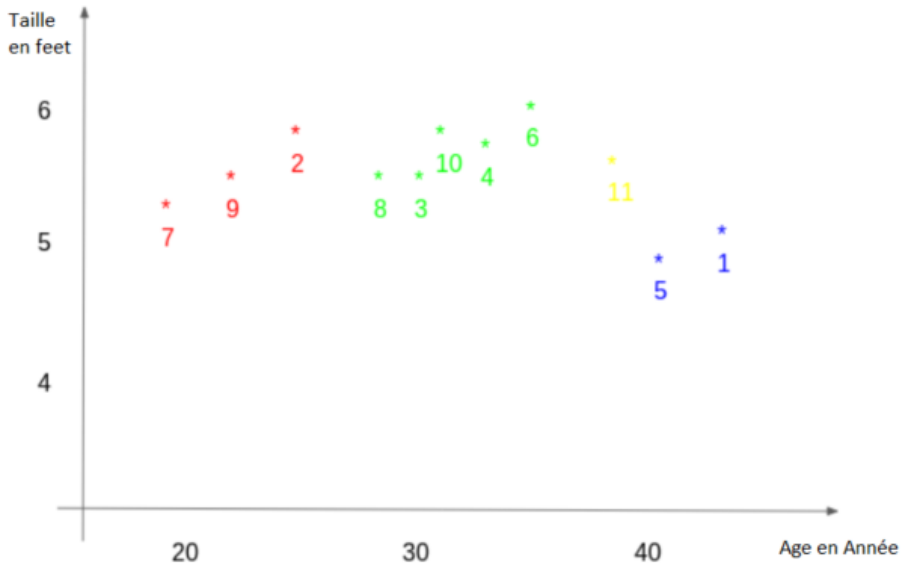
3 Principe de "base" de l'algorithme

Le principe de cet algorithme est de choisir les k données les plus proches du point étudié afin d'en prédire sa valeur.

Commençons par un exemple simple. Considérons le tableau suivant - il se compose de la taille, de l'âge et du poids (valeur cible) pour 10 personnes. Comme vous pouvez le voir, la valeur du poids de ID : 11 est manquante. Nous devons prévoir le poids de cette personne en fonction de sa taille et de son âge.

ID	Taille (feet)	Age	Poids (kg)
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

Pour une meilleure compréhension, voici le graphique de la taille en fonction de l'âge du tableau ci-dessus :



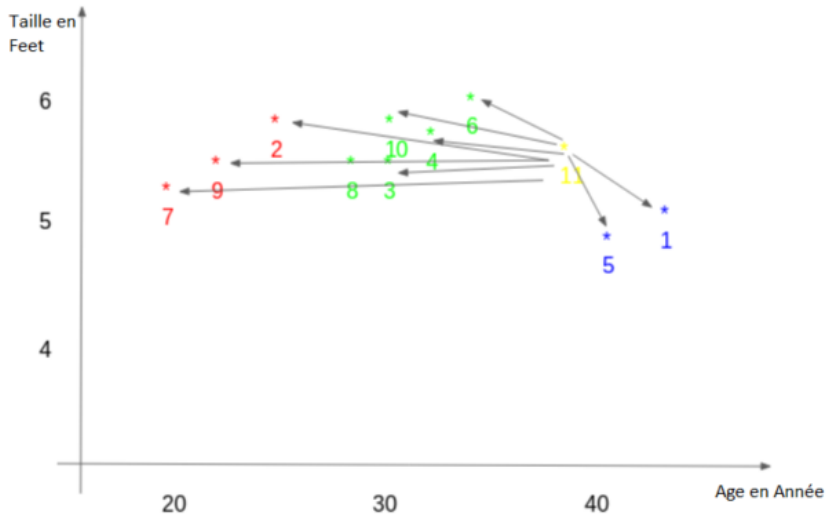
Dans le graphique ci-dessus, l'axe des y représente la taille d'une personne (en feet) et l'axe des x représente l'âge (en années). Les points sont numérotés en fonction des valeurs d'identification. Le point jaune (ID 11) est notre point de test.

Si je vous demande d'identifier le poids de ID11 en fonction du tracé, quelle serait votre réponse ? Vous répondriez probablement que puisque ID : 11 est plus proche des points 5 et 1 (ici en bleu), il doit donc y avoir un poids similaire à ces ID, probablement entre 72 et 77 kg (poids de ID : 1 et ID : 5 du tableau). C'est logique, mais comment pensez-vous que l'algorithme prédit les valeurs ? Nous le découvrirons dans cet article.

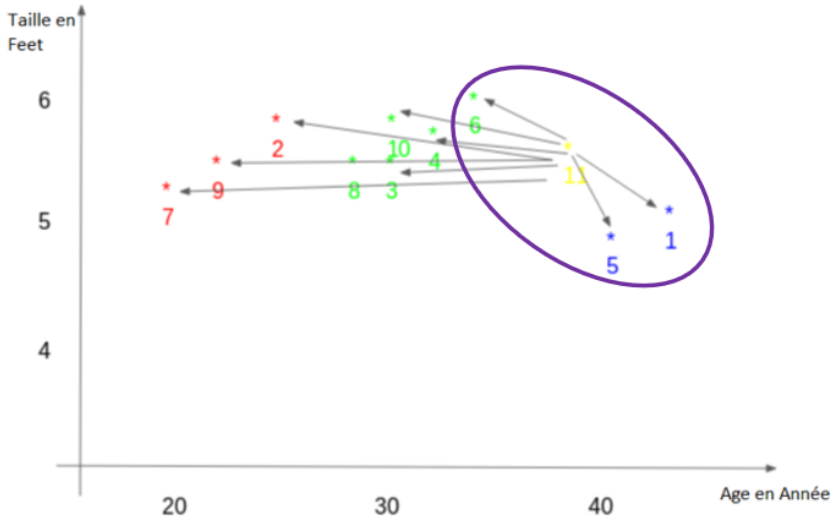
Comme nous l'avons vu ci-dessus, l'algorithme KNN peut être utilisé pour les problèmes de classification et de régression. L'algorithme KNN utilise la "similarité des caractéristiques" pour prédire les valeurs de tout nouveau point de données. Cela signifie qu'une valeur est attribuée au nouveau point en fonction de sa ressemblance avec les points de l'ensemble de formation. D'après notre exemple, nous savons que le point ID : 11 a une taille et un âge similaires à ceux des points ID : 1 et ID : 5, donc le poids serait aussi approximativement le même.

S'il s'agissait d'un problème de classification, nous aurions pris le mode comme prédiction finale. Dans ce cas, nous avons deux valeurs de poids : 72 et 77. Avez-vous une idée de la manière dont la valeur finale sera calculée ? C'est la moyenne des valeurs est considérée comme la prédiction finale. Vous trouverez ci-dessous une explication par étapes :

Tout d'abord, la distance entre le nouveau point et chaque point d'apprentissage est calculée. Nous verrons par la suite qu'il est possible de calculer la distance de différentes façon.



Dans un second temps, les k points de données les plus proches sont sélectionnés (en fonction de la distance). Dans cet exemple, les points 1, 5, 6 seront sélectionnés si la valeur de k est de 3. Nous explorerons plus loin la méthode pour sélectionner la bonne valeur de k plus loin dans cet article.



La moyenne de ces points de données est la prévision finale pour le nouveau point. Ici, nous avons le poids de ID : 11 = (77+72+60)/3 = 69,66 kg.
Voici d'une manière très "généralisée", le fonctionnement de l'algorithhme KNN.

4 Principe de "mathématique" de l'algorithme

Soit un ensemble E contenant n données labellisées :

$$E = (y_i, \vec{x_i})$$

avec i compris entre 1 et n, où y_i correspond à la classe (le label) de la donnée i et où le vecteur $\vec{x_i}$ de dimension $p(\vec{x_i} = (x_1 i, x_2 i, ..., x_p i))$ représente les variables prédictrices de la donnée i.
Soit une donnée u qui n'appartient pas à E et qui ne possède pas de label (u est uniquement caractérisée par un vecteur $\vec{x_u}$ de dimension p).
Soit d une fonction qui renvoie la distance entre la donnée u et une donnée quelconque appartenant à E.
Soit un entier k inférieur ou égal à n.

Voici le principe de l'algorithme de k plus proches voisins :

- On calcule les distances entre la donnée u et chaque donnée appartenant à E à l'aide de la fonction d
- On retient les k données du jeu de données E les plus proches de u
- On attribue à u la classe qui est la plus fréquente parmi les k données les plus proches.

Il existe plusieurs fonctions de calcul de distance, notamment, la distance euclidienne, la distance de Manhattan, la distance de Minkowski, la distance de Hamming... etc. On choisit la fonction de distance en fonction des types de données qu'on manipule. Ainsi pour les données quantitatives (exemple : poids, salaires, taille, montant de panier éléctronique etc...) et du même type, la distance euclidienne est un bon candidat. Quant à la distance de Manhattan, elle est une bonne mesure à utiliser quand les données (variables d'entrées) ne sont pas du même type (exemple : age, sexe, longueur, poids etc...).

La distance Euclidienne est une distance qui calcule la racine carrée de la somme des différences carrées entre les coordonnées de deux points.

$$D_e(x, y) = \sqrt{\sum_{j=1}^{n} (x_j - y_j)^2}$$

La distance de Manhattan est une distance qui calcule la somme des valeurs absolues des différences entre les coordonnées de deux points.

$$D_m(x, y) = \sum_{i=1}^{k} |x_i - y_i|$$

La distance de Hamming est une notion de distance entre deux points données qui est la différence maximale entre leurs coordonnées sur une dimension.

$$D_h(x, y) = \sum_{i=1}^{k} |x_i - y_i|$$

avec $x = y => D = 0$ et $x \neq y => D = 1$

5 Comment choisir la valeur K

Pour obtenir le bon K, il faut exécuter l'algorithme KNN plusieurs fois avec différentes valeurs de K et sélectionner celle qui présente le moins d'erreurs.

Le bon K doit être capable de prédire avec précision des données qu'il n'a jamais vues auparavant. Plus K est proche de 1, moins la prédiction est stable.
Plus la valeur de K augmente, plus votre prédiction devient stable en raison de la majorité des éléments qui compose l'ensemble.
Lorsqu'il y a un nombre croissant d'erreurs, le nombre K est poussé trop loin. Il faut que K soit un nombre impair pour qu'il y ait une majorité.

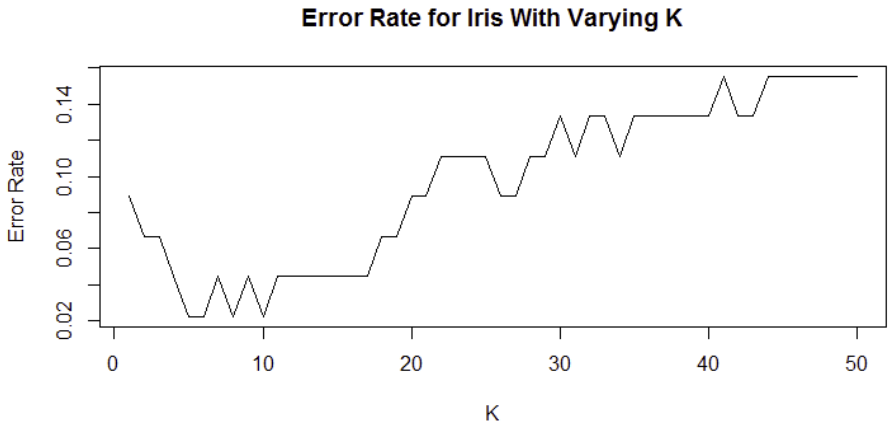
Ci dessous, un exemple produit sous R afin d'identifier quel K choisir.

Nous commencerons par charger le package et diviser l'ensemble des données de la source de donnée Iris en un ensemble de formation et de test.

```
1  library(class) #Contient la fonction KNN
2  set.seed(4948493) #Preparer pour la reproductibilite (seed)
3
4  ir_sample&lt;-sample(1:nrow(iris),size=nrow(iris)*.7)
5  ir_train&lt;-iris[ir_sample,] #S lectionnez les 70% de lignes
6  ir_test&lt;-iris[-ir_sample,] #S lectionnez les 30% de lignes
```

Avec les données séparées, nous pouvons commencer à essayer différents niveaux de K. Nous allons mettre en place une variable de retenue pour stocker les performances de chaque K. Ensuite, nous allons boucler de 1 à 50 (c'est-à-dire le nombre de voisins) et ensuite tracer pour voir lequel effectue le mieux la prédiction.

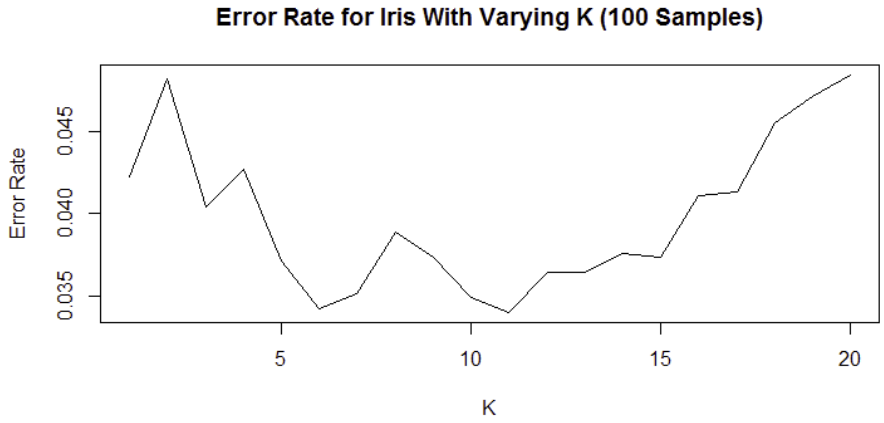
```
1  iris_acc&lt;-numeric() #Variable de retenue
2
3  for(i in 1:50){
4      #Application de KNN avec k = i
5   predict&lt;-knn(ir_train[,-5],ir_test[,-5],
6   ir_train$Species,k=i)
7   iris_acc&lt;-c(iris_acc,
8   mean(predict==ir_test$Species))
9  }
10     #Affichage de k= 1 a 50
11 plot(1-iris_acc,type="l",ylab="Error Rate",
12  xlab="K",main="Error Rate for Iris With Varying K")
```



Nous pouvons voir que n'importe quel nombre de voisins au-delà de 10 semble produire une classification moins précise. Nous pouvons (et devrions) exécuter ce beaucoup plus de fois pour s'assurer que notre K est précis. Nous pouvons exécuter 100 échantillons de l'ensemble de données de l'iris et regarder k = 1 à 20. Une grande partie du code pour cette section est très similaire, juste enveloppé dans une boucle for.

```
trial_sum<-numeric(20)
trial_n<-numeric(20)
set.seed(6033850)
for(i in 1:100){
 ir_sample<-sample(1:nrow(iris),size=nrow(iris)*.7)
 ir_train<-iris[ir_sample,]
 ir_test<-iris[-ir_sample,]
 test_size<-nrow(ir_test)
 for(j in 1:20){
 predict<-knn(ir_train[,-5],ir_test[,-5],
 ir_train$Species,k=j)
 trial_sum[j]<-trial_sum[j]+sum(predict==ir_test$Species)
 trial_n[j]<-trial_n[j]+test_size
 }
}

plot(1-trial_sum / trial_n,type="l",ylab="Error Rate",
 xlab="K",main="Error Rate for Iris With Varying K (100 Samples)")
```



Maintenant, il y a un modèle clair où, moins de 5 voisins sont trop peu nombreux et plus de 11 voisins deviennent rapidement trop nombreux. Basé sur ce graphique, il faurait choisir K = 6.

6 Quelques avantages et inconvénients de KNN

Quelques avantages de KNN

Temps de calcul rapide

Algorithme simple - à interpréter

Polyvalent - utile pour la régression et la classification

Une grande précision - vous n'avez pas besoin de comparer avec des modèles d'apprentissage mieux supervisés

Pas d'hypothèses sur les données - pas besoin de faire des hypothèses supplémentaires, d'ajuster plusieurs paramètres ou de construire un modèle. C'est pourquoi il est crucial dans le cas de données non linéaires.

Quelques inconvénients de KNN

L'exactitude dépend de la qualité des données

Avec des données volumineuses, la phase de prédiction peut être lente

Sensible à l'échelle des données et aux caractéristiques non pertinentes

Nécessite une mémoire importante - nécessité de stocker toutes les données relatives à la formation

Étant donné qu'il stocke toute la formation, il peut être coûteux en termes de calcul

7 Sources

Afin d'avoir des informations sur les notions mathématiques et comprendre plus profondément l'algorithme KNN, vous pouvez consulter ce document portant sur le Traitement Informatique des Données et ce cours approfondi.
https://lipn.univ-paris13.fr/~bennani/tmpc/TID/tid2.pdf

https://perso.math.univ-toulouse.fr/gadat/files/2012/04/KNN_talk.pdf

Vous souhaitez réaliser un TP pour prendre en main cet algorithme ? Comprendre quelle valeur de K choisir ? Ce TP est un bon moyen et celui-ci vous fera pratiquer le langage de programmation R.

http://www.math.u-bordeaux.fr/~mchave100p/wordpress/wp-content/uploads/2013/10/TP1_knn_correction1.pdf

Vous n'êtes pas un adepte de la programmation R ? Pas de problèmme, nous avons ce qu'il vous faut : un exemple à programmer sous le language Python !

https://cache.media.eduscol.education.fr/file/NSI/76/6/RA_Lycee_G_NSI_algo_knn_1170766.pdf

Si vous avez d'approfondir vos connaissances et que vous utilisez l'algorithme KNN, jetez donc un oeil à ce papier de recherche sur l'Advanced KNN, "A-kNN".

https://arxiv.org/pdf/2003.00415.pdf

Un cours entier sur l'algorithme des K plus proches voisins.

https://www.lyceum.fr/1g/nsi/8-algorithmique/3-algorithme-des-k-plus-proches-voisins

Autres sources intéressantes pour encore plus d'informations !

https://moncoachdata.com/blog/algorithme-k-plus-proches-voisins/

https://www.mygreatlearning.com/blog/knn-algorithm-introduction/

8 Liens Github Etudiants

Corentin BRETONNIERE : `https://github.com/CorentinBretonniere/CBRETONNIERE-PSBX`

Benjamin GUIGON : `https://github.com/benjaminguigon/PSBX`

Antoine SERREAU : `https://github.com/aserreau/PSB1`