APPRENTISSAGE AUTOMATIQUE SUPERVISE: ALGORITHME DES K PLUS PROCHES VOISINS

Informatique Tronc Commun

E. CLERMONT

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

La méthode des **"k plus proches voisins" (k-nearest neighbors ou k-NN ou KNN)** fait partie des méthodes les plus simples **d'apprentissage supervisé** pouvant être utilisée pour les cas de **régression et de classification**.

Il ne présente pas réellement de phase d'entrainement, qui est confondue avec la phase de prédiction : il utilise toutes les données à chaque calcul .
=> Cet algorithme est qualifiée de paresseux (Lazy Learning)
=> Plus le jeu de données grandit, moins l'algorithme KNN est efficace.

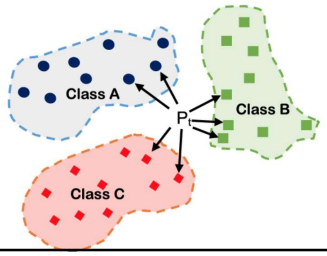
Il est cependant **couramment utilisé pour les systèmes de recommandation simples**, la reconnaissance de modèle, la fouille de données, les prévisions des marchés financiers ou la détection d'intrusion.
Remarque : possible d'utiliser l'algorithme des k plus proches voisins à des fins de régression en statistiques (on cherche à déterminer une valeur à la place d'une classe).

De nombreuses sociétés (exemple les GAFAM) utilisent les données concernant leurs utilisateurs afin de "nourrir" des algorithmes de machine learning qui permettront à ces sociétés d'en savoir toujours plus sur nous et ainsi de mieux cerner nos "besoins" en termes de consommation.

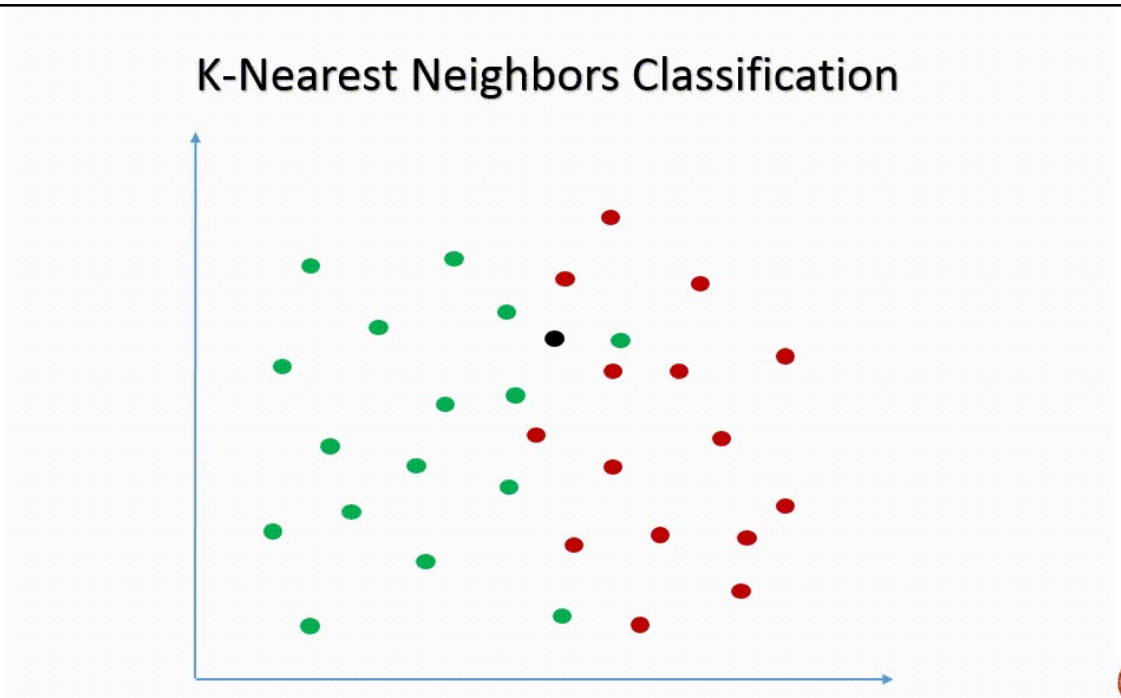
ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Pour les problèmes de classification, l'étiquette de classe est affectée à l'échantillon analysé sur la base d'un vote à la majorité : on utilise l'étiquette de la classe la plus fréquemment représentée autour de l'échantillon.
Pour les problèmes de régression avec une fonction f à valeur continue, on procède de la même manière mais on calcule la prédiction sur la valeur de la moyenne de f sur les k plus proches voisins.

L'algorithme knn nécessite d'avoir des données labellisées.
À partir d'un ensemble E de données labellisées, il sera possible de classer (déterminer le label) une nouvelle donnée (donnée n'appartenant pas à E).
Pour prédire la classe d'une nouvelle donnée d'entrée, il va chercher ses k voisins les plus proches (en utilisant la distance euclidienne, ou autres) et choisira la classe des voisins majoritaires.



3



4

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Le jeu de données comporte pour chaque donnée :
- une ou plusieurs caractéristiques,
- Une classe (ou étiquette ou label) pour chaque donnée.

Soit $k \in N$.

L'algorithme des k plus proches voisins prédit la classe d'une nouvelle donnée x de la façon suivante :

1- Calculer les distances de x à toutes les données d'entraînement.

2- Trouver les k données d'entraînement les plus proches de x (en termes de distance).

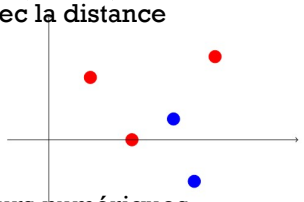
3- Trouver la classe majoritaire $c \in Y$ parmi ces k données les plus proches de x.

4- Prédire que x est de classe c.

5

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Exemple dans le plan : Implémentation de l'algorithme dans le plan, avec la distance euclidienne.
On suppose qu'il n'y a que 2 classes d'objets :
les points rouges et les points bleus.



L'ensemble d'apprentissage (la base) sera constituée d'une liste :
- De points p du plan euclidien, représenté comme un couple de valeurs numériques qui correspondent aux abscisses et aux ordonnées de chaque point.
- e est une étiquette, pouvant valoir 'r' ou 'b', pour rouge et bleu pour chaque point .

Abscisse	Ordonnée	étiquette
3	0	r
1	2	r
4	1	b
6	4	r
5	-2	b
…	…	…

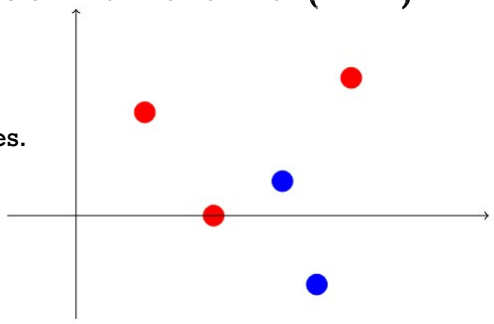
L'algorithme prendra en paramètres :
- l'ensemble d'apprentissage
- un point du plan dont on veut décider s'il est rouge ou bleu,
- un entier k supposé impair.

On attribuera la couleur associée à la couleur majoritaire parmi les k plus proches voisins, comme k est impair on n'a pas à se préoccuper des cas d'égalité.

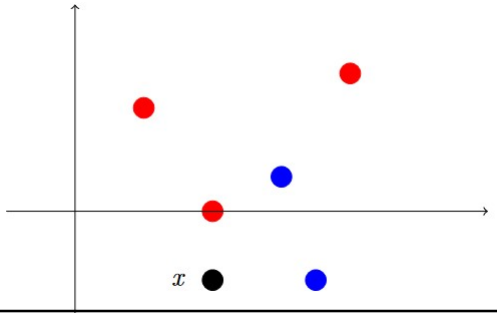
6

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

On dispose de données
dont les classes (rouge ou bleues) sont connues.



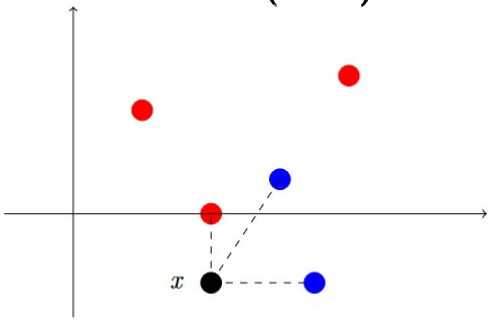
On cherche à déterminer la classe d'une nouvelle donnée x.



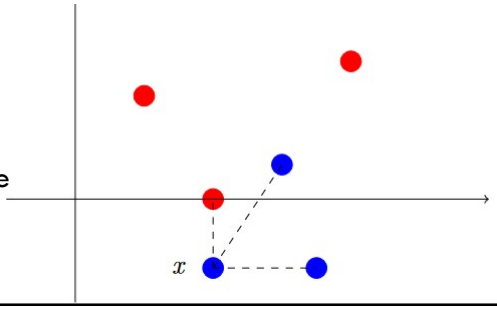
7

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

On calcule la distance de x à toutes les données
et on détermine les k plus proches voisins.
⇒ Fonction distance de calcul de distance
⇒ Fonction voisins de détermination des k plus proches voisins



On associe à x la classe majoritaire de ses plus proches voisins.
⇒ Fonction majoritaire pour déterminer la classe majoritaire des k voisins les plus proches
⇒ Fonction knn qui détermine la classe de x



8

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

On a besoin de mesurer la distance entre 2 données.

On utilise souvent la **distance euclidienne** :

$$d(x,y) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}$$

x,y sont des vecteurs

Exercice :

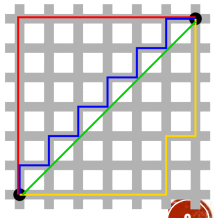
On représente un vecteur x ∈ X par la liste de ses coordonnées.

Écrire une fonction **distance(x, y)** renvoyant la distance euclidienne entre 2 vecteurs x et y.

```
def distance(x, y):
    dist= 0
    for i in range(len(x)):
        dist = dist + (x[i] - y[i])**2
    dist = dist**.5
    return dist

print( distance( [1,1] , [2,2] ) )
```

25ca-2711538



On peut utiliser d'autres distances,
par exemple la **distance de Manhattan** :

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

9

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Étape 1 : déterminer les k plus proches voisins.

Soient : x un vecteur (sous forme de liste), donnée à classifier
X une liste de vecteurs qui correspond au jeu de données,
k un entier qui correspond au nombre de plus proches voisins
dist une fonction de calcul de distance.

Écrire une **fonction voisins(x, X, k, dist)** renvoyant la liste des k plus proches voisins de x dans X pour la distance d.

```
def voisins(x, X, k, dist=distance):
    listeDistanceIndices=[] # liste de listes[distance, indice]
    for i in range ( len(X)) :
      d = dist( x, X[i])
      listeDistanceIndices.append( [d, i])
    listeDistanceIndices.sort()

    lesVoisins = [] # liste qui ne contiendra que les indices
    for i  in range( k ):
      lesVoisins.append( listeDistanceIndices[i][1] )

    return lesVoisins
```

10

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Étape 1 : déterminer les k plus proches voisins.

Autre solution : on peut aussi trier les données d'entraînement par ordre croissant de distance à x et prendre les k premières.

```
def voisins(x, X, k, dist=distance):
    # renvoie les k plus proches voisins de x dans X
    indices = sorted(range(len(X)), key=lambda i: dist(x, X[i]))
    return indices[:k]
```

11

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Étape 2 : Déterminer la classe majoritaire

Écrire une **fonction majoritaire(L)** renvoyant l'élément le plus fréquent de la liste L,. Cette fonction doit avoir une complexité linéaire.

```
def majoritaire(L):
    compte = {} # compte[e] = nombre d'occurrences de e dans L
    for e in L:
        if e in compte:
            compte[e] += 1
        else:
            compte[e] = 1
    kmax = L[0]
    for k in compte:
        if compte[k] > compte[kmax]:
            kmax = k
    return kmax
```

12

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Étape 2 : Déterminer la classe majoritaire

Écrire une **fonction majoritaire(L)** renvoyant l'élément le plus fréquent de la liste L, en complexité linéaire.

```
#version courte
def majoritaire(L):
    compte = {}
    for e in L:
        compte[e] = compte.get(e, 0) + 1
    return max(compte, key=compte.get)
```

13

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Étape 3: prédire la classe de x selon l'algorithme knn

Écrire une **fonction knn(x, X, Y, k)** qui retourne la classe de x.

```
def knn(x, X, Y, k, dist = distance):
    """ Prédit la classe de x avec l'algorithme KNN
        x : nouvelle,  donnée
        X : données d'entraînement
        Y : étiquettes des données d'entraînement
        k : nombre de voisins à considérer
    """
    V = voisins(x, X, k, distance)
    return majoritaire([Y[i] for i in V])
```

14

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Evaluation d'un modèle

On suppose que l'on possède des données X avec des étiquettes Y .
On veut savoir si l'algorithme KNN est un bon classifieur pour ces données.

Pour cela, on partitionne les données en 2 ensembles :

- Ensemble d'entraînement Xtrain (de classes Ytrain) : données parmi lesquelles on va chercher les k plus proches voisins.
- Ensemble de test Xtest (de classes Ytest) : données utilisées pour évaluer le modèle, en comparant les classes prédites par KNN avec les classes réelles.

15

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Evaluation d'un modèle

Définir la **fonction separer** qui prend en entrée le jeu de données X(liste), les classes associées Y (liste), p la proportion entre volume de données d'entrainement et de test.
Cette fonction retourne :

- Le jeu de données d'entrainement Xtrain et les classes Ytrain associées
- Le jeu de données de test Xtest et les classes Ytest associées

```
def separer(X, Y, p):
    Xtrain, Xtest, Ytrain, Ytest = [], [], [], []
    for i in range(len(X)):
        if i <= p * len(X):
            Xtrain.append(X[i])
            Ytrain.append(Y[i])
        else:
            Xtest.append(X[i])
            Ytest.append(Y[i])
    return Xtrain, Xtest, Ytrain, Ytest

#constitution des données d'apprentissage et de test (ratio .7)
Xtrain, Xtest, Ytrain, Ytest = separer(X, Y, 0.7)
```

16

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Evaluation d'un modèle

La précision d'un modèle correspond à la proportion de données de test bien classées par rapport au nombre total de données.
L'erreur est égale à 1 - précision.

```
def precision(k, Xtest, Ytest, Xtrain, Ytrain):
    n = len(Xtest)
    p = 0
    for i in range(n):
        if knn(Xtest[i], Xtrain, Ytrain, k) == Ytest[i]:
            p += 1
    return p/n
```

17

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Evaluation d'un modèle

La **matrice de confusion** est une matrice carrée dont les lignes et les colonnes sont les classes possibles.
La case (i, j) contient le nombre de données de test de classe i qui ont été prédites comme appartenant à la classe j.
Exemple : Dans la matrice suivante,

```
array(  [[21, 0, 0],
         [ 0, 29, 1],
         [ 0, 2, 23]])
```

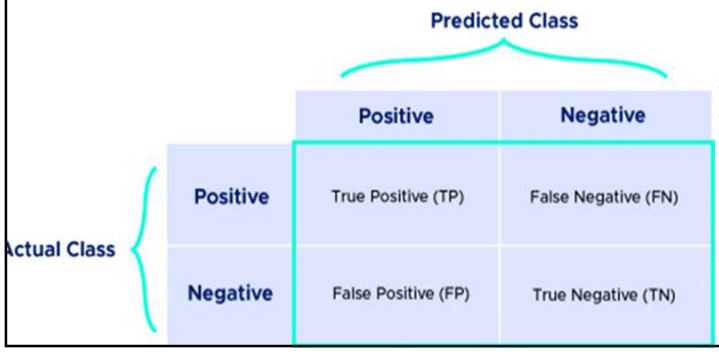
on remarque que :
21 données de classe 0 ont été prédites comme appartenant à la classe 0,
2 données de classe 1 ont été prédites comme appartenant à la classe 2,...

Question
Comment obtenir la précision à partir de la matrice de confusion ?

18

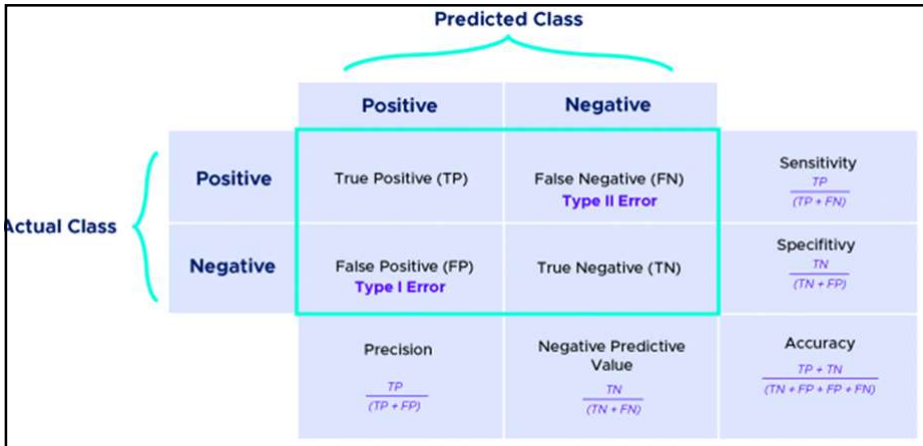
ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Matrice de confusion (ou matrice de contingence) : matrice qui mesure la qualité d'un système de classification.
disposition de tableau spécifique qui permet de visualiser les performances d'un algorithme, généralement un apprentissage supervisé.
Elle met en valeur les prédictions correctes et incorrectes
et surtout donne surtout un indice sur le type d'erreurs commises.
Pour calculer une matrice de confusion, on a besoin d'un **ensemble de données de test et un autre de validation qui contient les valeurs des résultats obtenus.**
Chaque colonne du tableau contient une classe prédite par l'algorithme et les lignes des classes réelles.



4 catégories :
•**TP (True Positive)** : les cas où la prédiction est positive, et où la valeur réelle est effectivement positive.
•**TN (True Negative)** : les cas où la prédiction est négative, et où la valeur réelle est effectivement négative.
•**FP (False Positive)** : les cas où la prédiction est positive, mais où la valeur réelle est négative.
•**FN (False Negative)** : les cas où la prédiction est négative, mais où la valeur réelle est positive.

19



Sensibilté (sensitivity) ou taux de vrais positifs : nombre de prédictions positives correctes / nombre total de données positives.

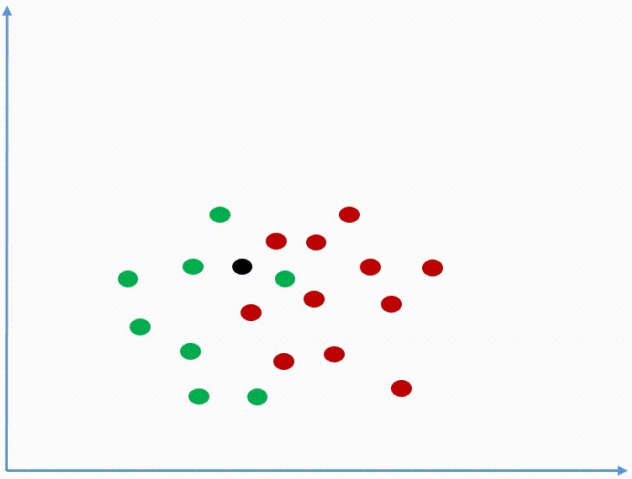
Spécificité (specificity) ou taux de vrais négatifs : nombre de prédictions négatives correctes / nombre total de données négatives

La précision d'un modèle correspond à la proportion de données de test bien classées par rapport au nombre total de données.
L'erreur est égale à 1 - précision.

Taux de reconnaissance global (*accuracy*) quantifie globalement le modèle et qui se généralise facilement à plus de 2 classes.

Taux de faux positifs TFP : nombre de prédictions positives incorrectes / nombre total de données négatives
TFP = 1-Spécificité
Taux de faux négatifs TFN : nombre de prédictions négatives incorrectes / nombre total de données positives
TFP = 1-Sensibilité

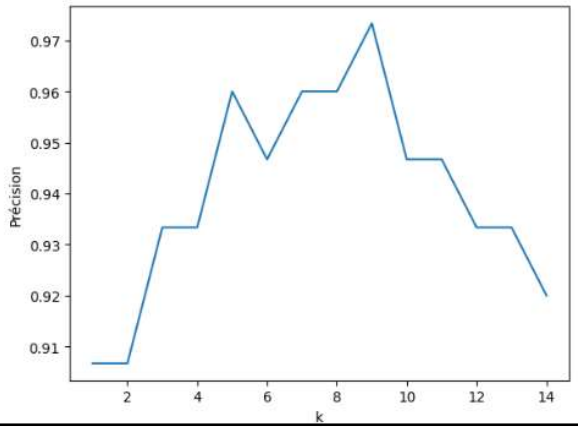
Choice of value of K



ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Evaluation d'un modèle

Comment choisir la valeur de k dans l'algorithme des k plus proches voisins ?
On affiche la précision en fonction de k pour choisir la valeur de k qui donne la meilleure précision.

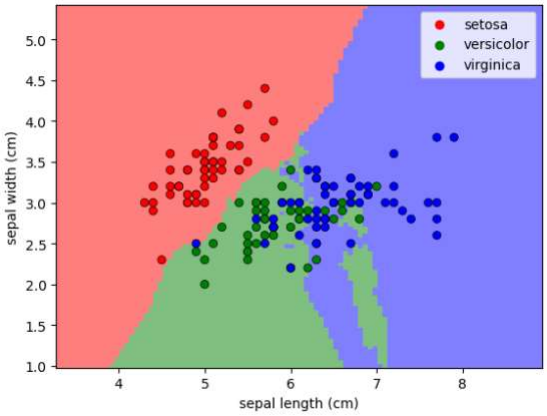


22

ALGORITHME DES K PLUS PROCHES VOISINS (KNN)

Evaluation d'un modèle

Comment choisir la valeur de k dans l'algorithme des k plus proches voisins ?
On peut aussi visualiser la frontière de décision (decision boundary) permettant de voir à quelle classe est associée chaque point de l'espace des données :



APPRENTISSAGE AUTOMATIQUE SUPERVISE: ALGORITHME DES K PLUS PROCHES VOISINS

Informatique Tronc Commun

E. CLERMONT