

# Apprentissage supervisé

## I Algorithme des $k$ plus proches voisins

On va chercher à apprendre à un ordinateur à reconnaître par lui-même des « objets », c'est-à-dire à prédire à quelle « classe » ils appartiennent. On suppose que

- un objet est défini par un certain nombre de caractéristiques : une couleur, une forme, une marque, ...
- les différentes classes forment une partition de l'ensemble des différents objets : elles sont donc deux à deux disjointes et tous les objets considérés appartiennent bien à une des classes.
- on dispose d'un ensemble d'objets  $E$  pour lesquels on connaît leur classe respective
- on dispose d'un ensemble d'objets  $X$  que l'on va chercher à reconnaître en cherchant des « similitudes » avec les objets de  $E$ .

Pour mesurer les similitudes entre deux objets, on a besoin d'une *distance*, ie une application telle que, pour tous objets  $x, y$  et  $z$

$$d(x, y) = 0 \Leftrightarrow x = y \quad d(x, y) = d(y, x) \quad d(x, y) \leq d(x, z) + d(z, y)$$

Le principe de l'algorithme des  $k$  plus proches voisins est le suivant :

- on fixe au départ un entier  $k$ , compris entre 1 et  $n = \text{len}(E)$
- pour un objet  $x$  de  $X$ , on détermine les  $k$  éléments de  $E$  les plus proches de  $x$  (au sens de la distance  $d$ )
- on prédit que la classe de l'objet  $x$  est la classe majoritaire parmi les  $k$  objets de  $E$  précédents.

## II Un exemple dans le plan

### 1. Codage de l'algorithme

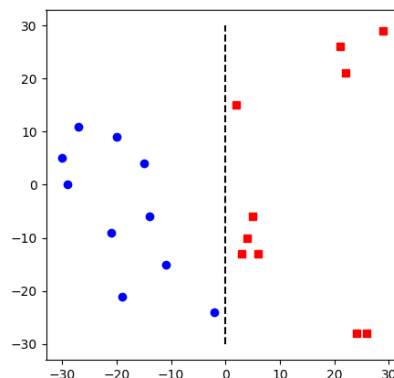
On considère un ensemble de points du plan divisé en deux sous-ensembles :

- 10 points ronds (et bleus)
- 10 points carrées (et rouges)

Les points ronds ont une abscisse  $x < 0$ , les rouges ont une abscisse  $x > 0$

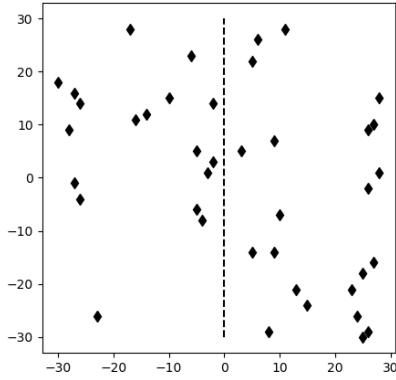
Un point est représenté par un triplet  $(x, y, c)$  où  $x, y$  sont les coordonnées du point et  $c$  est sa classe (0 pour un point bleu, 1 pour un point rouge).

L'ensemble  $E$  est par exemple le suivant :

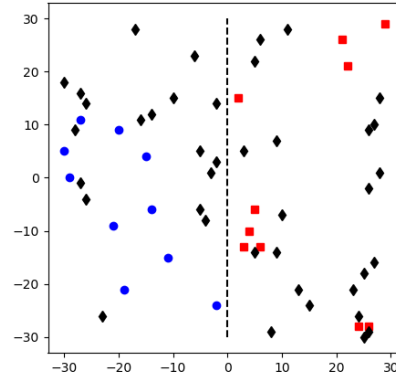


L'ensemble  $E$

On rajoute alors 40 points pour définir l'ensemble  $X$  :



L'ensemble  $X$



Les ensembles  $E$  et  $X$

On va donc chercher à prédire le signe de l'abscisse de ces points en fonction de leurs plus proches voisins. Pour cet exemple on utilisera la distance euclidienne canonique : si  $X = (x_1, x_2)$  et  $Y = (y_1, y_2)$ ,

$$d(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

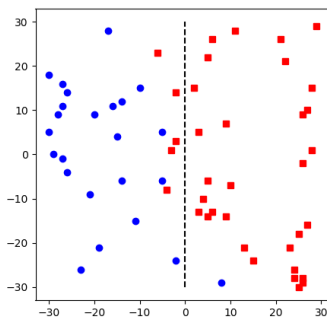
```
def d(x, y) :
    return ((x[0] - y[0]) ** 2 + (x[1] - y[1]) ** 2) ** 0.5
```

La fonction suivante détermine la classe d'un point  $x$  de  $X$  en fonction de l'échantillon  $E$  avec l'algorithme de  $k$  plus proches voisins :

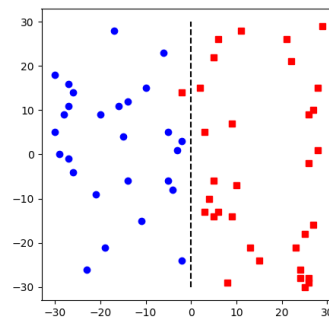
```
def kNN(x, E, k) :
    L = [(d(x, (p[0], p[1])), p[2]) for p in E]
    L.sort()
    test = 0
    for i in range(k) :
        if L[i][1] == 1 :
            test += 1
        else :
            test -= 1
    if test > 0 :
        return (x[0], x[1], 1)
    else :
        return (x[0], x[1], 0)
```

- La liste  $L$  initiale contient des couples  $(d(x, p), c)$  où  $p$  est un point de  $E$  et  $c$  est la couleur du point  $p$
- La ligne  $L.sort()$  permet de trier la liste  $L$  selon sa première coordonnée, donc suivant les distances à  $x$  croissantes.
- Les lignes suivantes servent à déterminer, parmi les  $k$  plus proches voisins, s'il y a plus de points bleus (et  $test$  est alors  $> 0$ ) ou de points rouges (et  $test$  est alors  $< 0$ ) ; pour éviter les ambiguïtés, on choisira  $k$  impair.
- On renvoie alors un nouveau point, avec les mêmes coordonnées et avec la couleur prédite.

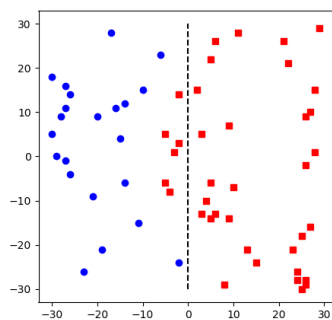
En modifiant la valeur de  $k$ , on obtient les résultats suivants :



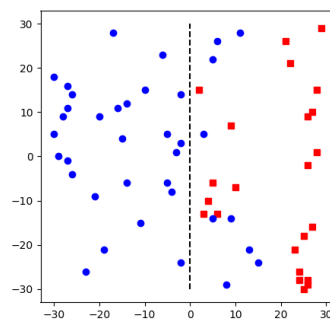
$k = 1$



$k = 11$



$k = 7$



$k = 19$

## 2. Matrice de confusion

On voit sur l'exemple précédent que le choix du paramètre  $k$  a une incidence sur les résultats :

- pour  $k = 1$ , la couleur donnée est celle du point le plus proche, ce qui donne 6 erreurs dans notre exemple
- pour  $k = 7$ , on a toujours 6 erreurs (mais pas sur les mêmes points)
- pour  $k = 11$ , il n'y en a qu'une
- pour  $k = 19$  (le maximum), la couleur donnée est la couleur opposée au point le plus loin, ce qui donne 9 erreurs.

On pourrait imaginer qu'augmenter la valeur de  $k$  donne de meilleurs résultats, mais comme on le voit sur cet exemple, ce n'est pas toujours le cas. Afin de déterminer la meilleure valeur de  $k$ , il faut faire un certain nombre de tests.

Pour mesurer l'efficacité de l'algorithme (et du choix de  $k$ ), on utilise la *matrice de confusion* définie par  $M = (m_{i,j})_{1 \leq i,j \leq p} \in \mathcal{M}_p(\mathbb{R})$ , où  $p$  est le nombre de classes et

$m_{i,j}$  est le nombre d'objets de classe réelle  $i$  dont l'algorithme a prédit la classe  $j$

Dans notre exemple  $M \in \mathcal{M}_2(\mathbb{R})$  et

- $m_{1,1}$  est le nombre de points de  $X$  d'abscisses  $< 0$  et prédits dans la classe « ronde »
- $m_{1,2}$  est le nombre de points de  $X$  d'abscisses  $< 0$  et prédits dans la classe « carrée »
- $m_{2,1}$  est le nombre de points de  $X$  d'abscisses  $> 0$  et prédits dans la classe « ronde »
- $m_{2,2}$  est le nombre de points de  $X$  d'abscisses  $> 0$  et prédits dans la classe « carrée »

Dans une prédiction parfaite, cette matrice serait diagonale. On cherche donc une valeur de  $k$  pour laquelle la matrice de confusion est la plus proche d'une matrice diagonale.

Avec les valeurs de  $k$  précédentes, on obtient les matrices suivantes :

$$M_1 = \begin{pmatrix} 13 & 5 \\ 1 & 21 \end{pmatrix} \quad M_7 = \begin{pmatrix} 12 & 6 \\ 0 & 22 \end{pmatrix} \quad M_{11} = \begin{pmatrix} 17 & 1 \\ 0 & 22 \end{pmatrix} \quad M_{19} = \begin{pmatrix} 18 & 0 \\ 9 & 13 \end{pmatrix}$$

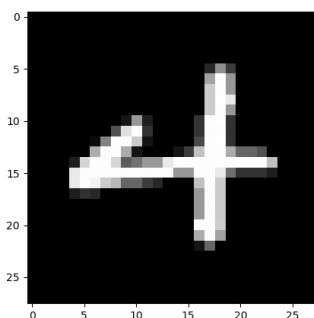
## III Reconnaissance de caractères

En utilisant l'algorithme des  $k$  plus proches voisins, on peut apprendre à un ordinateur à reconnaître des chiffres sur une image, pour lire le code postal sur une enveloppe ou la plaque minéralogique d'une voiture par exemple.

La reconnaissance des chiffres est un problème à 10 classes (donc la matrice de confusion sera une matrice de  $\mathcal{M}_{10}(\mathbb{R})$ ).

On suppose disposer d'un échantillon d'images de chiffres et du chiffre correspondant à partir duquel on va lire les autres images. Un tel échantillon est par exemple disponible dans le **dataset MNIST784** qui contient 70000 images de chiffres.

Une image, en noir et blanc, de chiffre est représentée par une matrice de taille  $28 \times 28$  pixels codés par des entiers de  $\llbracket 0, 255 \rrbracket$  (0 pour du noir et 255 pour du blanc).



C'est un 4

On peut calculer la distance entre deux images  $X$  et  $Y$  par

$$d(X, Y) = \sqrt{\sum_{1 \leq i, j \leq 28} (x_{i,j} - y_{i,j})^2}$$

Si on considère un ensemble  $E$ , choisi aléatoirement, composé de 600 images à partir desquelles on fait des tests pour reconnaître 200 images (de l'ensemble  $X$ ), on peut obtenir, avec seulement  $k = 3$ , la matrice de confusion suivante :

$$M_3 = \begin{pmatrix} 22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 15 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 22 & 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 17 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 12 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 28 & 0 & 1 \\ 0 & 1 & 0 & 3 & 0 & 0 & 0 & 1 & 6 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 21 \end{pmatrix}$$

Les principales erreurs concernent la reconnaissance du chiffre 8. Plus précisément, si on cherche, avec cet échantillon à reconnaître des 8, le taux de bons résultats est, en faisant le test sur 100 images représentant effectivement un 8 :

- 78% pour  $k = 3$
- 72% pour  $k = 7$
- 72% pour  $k = 12$

On voit à nouveau qu'il peut être difficile de déterminer la meilleure valeur de  $k$  en fonction de l'échantillon  $E$  choisi.