1 Apprentissage automatique

Le problème de l'apprentissage automatique "On dit qu'un programme apprend d'une expérience E en vue d'une tâche future T et d'une mesure de performance P, si sa performance sur T mesurée par P, s'améliore avec l'expérience E"(Tom Mitchell 1998)

Exemple :

1. **programme** = logiciel email comme Gmail
2. **expérience E** = étiquetage des mails comme spam ou non spam **par l'utilisateur** ce qui sera pris en compte ensuite par le logiciel qui va donc "apprendre" de cette expérience

   L'ensemble des mails étiquetés s'appelle **la base d'apprentissage**
3. **tâche T** = **classement** des futurs mails comme spams ou non spams **par le logiciel**
4. **mesure de performance P** = $\dfrac{\text{nb mails correctement classés}}{\text{nombre total de prédictions}}$

Nous allons étudier un exemple de programme d'apprentissage automatique, l'algorithme des k plus proches voisins.

Dans un premier temps (k = 1) nous allons étudier l'algorithme du plus proche voisin

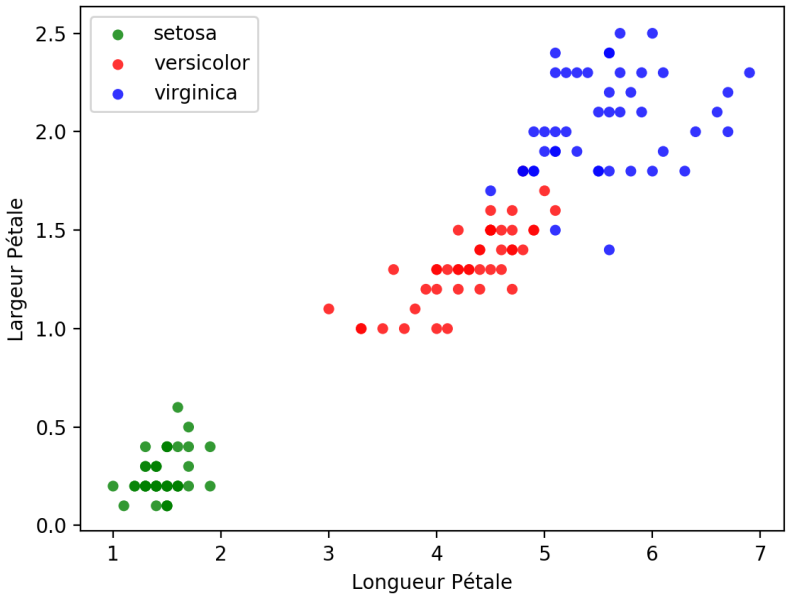
2 Algorithme du plus proche voisin

Nous disposons d'un fichier .csv contenant 150 mesures concernant des fleurs (Trois variétés d'Iris : Iris-setosa, Iris-versicolor et Iris-virginica).
Pour chaque fleur on a mesuré la longueur et la largeur de la sépale ainsi que la longueur et la largeur de la pétale (voir ci-dessous les premières lignes du fichier)

```
LongueurSepale,LargeurSepale,LongueurPetale,LargeurPetale,Variete
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
```

Nous allons diviser ensuite cet ensemble en deux (de manière aléatoire) une **base d'apprentissage** et **une base de test**

Pour mieux **visualiser** l'algorithme du plus proche voisin on ne va considérer dans un premier temps que 2 mesures, la longueur et la largeur de la pétale que l'on peut **considérer alors comme les coordonnées d'un point dans le plan** au risque que plusieurs fleurs ont les mêmes coordonnées
Le classement futur d'une Iris dans l'une des 3 classes, Iris-setosa, Iris-versicolor et Iris-virginica, en fonction de ses coordonnées longueur et la largeur de la pétale se fera suivant l'algorithme du plus proche voisin



Algorithme 1 : Algorithme du plus proche voisin

ppv (B,X)
Données : Une base d'apprentissage B, des éléments X de classe inconnue
Résultat : On attribue une classe à chaque élément à classer

```
1  début
2      pour chaque x ∈ X faire
3          d_min ← inf
4          b_min ← φ
5          pour chaque b ∈ B faire
6              d ← dist(b, x)
7              si d < d_min alors
8                  d_min ← d
9                  b_min ← b
10             fin
11         fin
12         attribuer à x la même classe que celle de b_min
13     fin
14 fin
```

La distance utilisée dans l'algorithme est la plus part du temps **la distance euclidienne**, mais on peut changer la distance (voir TP)

Pour mesurer la performance de l'algorithme on va l'exécuter sur la base de test en comparant les classements obtenus aux vraies classes et calculer le rapport $\frac{\text{succès de la prévision}}{\text{nb total de cas}}$

3 Algorithme des k plus proches voisins

On va affiner l'algorithme précédent en prenant par exemple pour $k = 3$, les trois plus proches voisins, puis considérer pour ces trois plus proches voisins **la classe majoritaire**, puis attribuer à l'élément à classer cette dernière
D'où l'algorithme :

Algorithme 2 : Algorithme des k plus proches voisins

```
kppv (B,X,k)
Données : Une base d'apprentissage B, des éléments X de classe inconnue,k
          un entier
Résultat : On attribue une classe à chaque élément à classer
1  début
2      pour chaque x ∈ X faire
3          d_min ← [inf] * k
4          b_min ← [φ] * k
5          pour chaque b ∈ B faire
6              d ← dist(b, x)
7              insérer d et b à leur place dans les listes d_min et b_min(tri par insertion)
8          fin
9          c ← classeMajoritaire(b_min)
10         attribuer à x la classe c
11     fin
12 fin
```

Pour insérer d et b à leur place dans les listes $d_{\min}$ et $b_{\min}$ on va procéder comme pour **le tri pour insertion**
$d_{\min}$ et $b_{\min}$ sont à priori des listes de longueur k, mais on va les considérer comme des listes de longueur $k + 1$ et insérer d et b en position k

Puis procéder comme pour le tri par insertion pour insérer d et b à leur place

Quant à la fonction `classeMajoritaire(`$b_{\min}$`)` elle retourne la classe ayant apparu le plus grand nombre de fois dans la liste $b_{\min}$

Algorithme 3 : insertion

`insert` (d,b,$d_{\min}$,$b_{\min}$)

Données : Une distance d, un élément b de la base, la liste $d_{\min}$ des k distances les plus petites classées dans l'ordre, la liste $b_{\min}$ correspondante à la liste précédente

Résultat : On insère d et b à leur place dans les listes si ils ont leur place

```
1  début
2      d_min[k] ← d
3      b_min[k] ← b
4      pour j ∈⟦0; k⟦ faire
5          si d_min[j] < d_min[j − 1] alors
6              //On échange les valeurs en position j et j-1 dans d_min
7              temp ← d_min[j]
8              d_min[j] ← d_min[j − 1]
9              d_min[j − 1] ← temp
10             //On échange les valeurs en position j et j-1 dans b_min
11             temp ← b_min[j]
12             b_min[j] ← b_min[j − 1]
13             b_min[j − 1] ← temp
14         sinon
15             sortir de la boucle
16         fin
17     fin
18 fin
```

4 Mesure de performance

On se rend bien compte de manière expérimentale que cette méthode risque de ne pas être fiable à "la frontière" des classes 'versicolor' et 'virginica'
Voici quelques résultats expérimentaux en testant l'algorithme avec une base de test extraite de la base d'apprentissage, ce qui nous permet de comparer les prédictions fournies par l'algorithme aux vrais résultats

Ensuite on calcule le rapport $P = \dfrac{\text{nb de prévisions exactes}}{\text{nb total de prévisions}}$ et on note $|B|$ le nombre d'éléments de la base apprentissage

$\lvert B\rvert$	1	3	5
75	0,92	0,97	0,97
145	0,93	0,93	0,93

En conclusion :

Augmenter la taille de la base d'apprentissage n'entraîne pas forcément de meilleures prédictions