Cours 6

Classification supervisée (compléments)

Plan du cours

1. Généralités sur l'apprentissage automatique

2. Classer sans apprendre (classifieur de Bayes)

3. Classer sans apprendre mais en réglant des paramètres (Plus proches voisins)

4. Introduction à la Fouille de Textes

5. Classification supervisée (compléments)

   - les arbres de décision
   - les réseaux de neurones artificiels
   - les SVM (Support Vector Machine)

Rappels : classification supervisée

On dispose d'un ensemble de données d'apprentissage : $S = \{(x_i, u_i)\}_{1...m}$

On cherche à induire un modèle/une méthodologie pour **prédire la classe** *u** d'une nouvelle donnée *y*.

Le **classifieur de Bayes**

- est adapté aux données qualitatives/symboliques (seulement)
- n'est pas paramétrable
- peut s'avérer performant dans certains cas

Le **classifieur k-PPV** (*k*-Plus Proches Voisins)

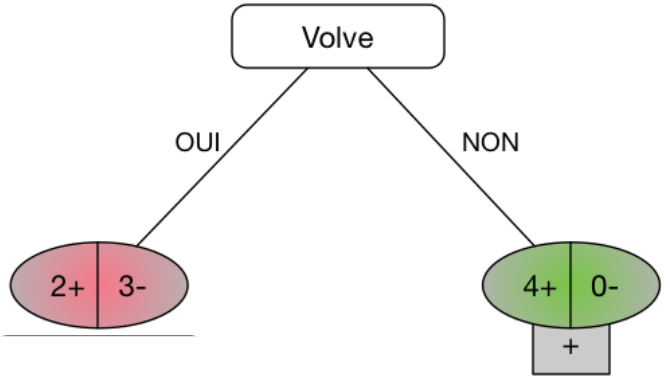
- est adapté aux données quantitatives
- paramétrable (k, choix de la distance, pondération des descripteurs)
- propose un « début » d'explication de la décision (voisins)

→ toujours pas d'apprentissage d'un modèle de classification réutilisable !

Les « arbres de décision »

Principe : apprendre un ensemble de règles de décisions, structuré sous forme d'un arbre (le modèle), à partir d'exemples étiquetés.

Champignons	Hauteur	Couleur	Dessous	Anneau	Volve	Classse
C1	grand	blanc	lamelles	non	non	+
C2	moyen	blanc	mousse	non	non	+
C3	petit	marron	lamelles	oui	oui	+
C4	petit	noir	lamelles	non	oui	+
C5	grand	blanc	mousse	non	non	+
C6	petit	blanc	lamelles	non	non	+
C7	grand	blanc	mousse	oui	oui	-
C8	petit	marron	mousse	oui	oui	-
C9	moyen	marron	lamelles	non	oui	-



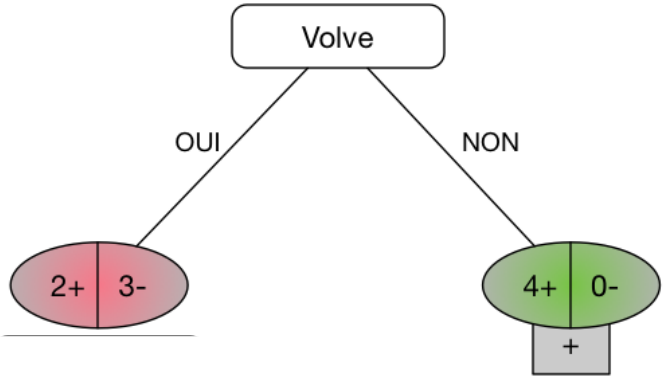
Decouper(S) :

- Choisir un descripteur discriminant : S=S'US"

Les « arbres de décision »

Principe : apprendre un ensemble de règles de décisions, structuré sous forme d'un arbre (le modèle), à partir d'exemples étiquetés.

Champignons	Hauteur	Couleur	Dessous	Anneau	Volve	Classse
C1	grand	blanc	lamelles	non	non	+
C2	moyen	blanc	mousse	non	non	+
C3	petit	marron	lamelles	oui	oui	+
C4	petit	noir	lamelles	non	oui	+
C5	grand	blanc	mousse	non	non	+
C6	petit	blanc	lamelles	non	non	+
C7	grand	blanc	mousse	oui	oui	-
C8	petit	marron	mousse	oui	oui	-
C9	moyen	marron	lamelles	non	oui	-



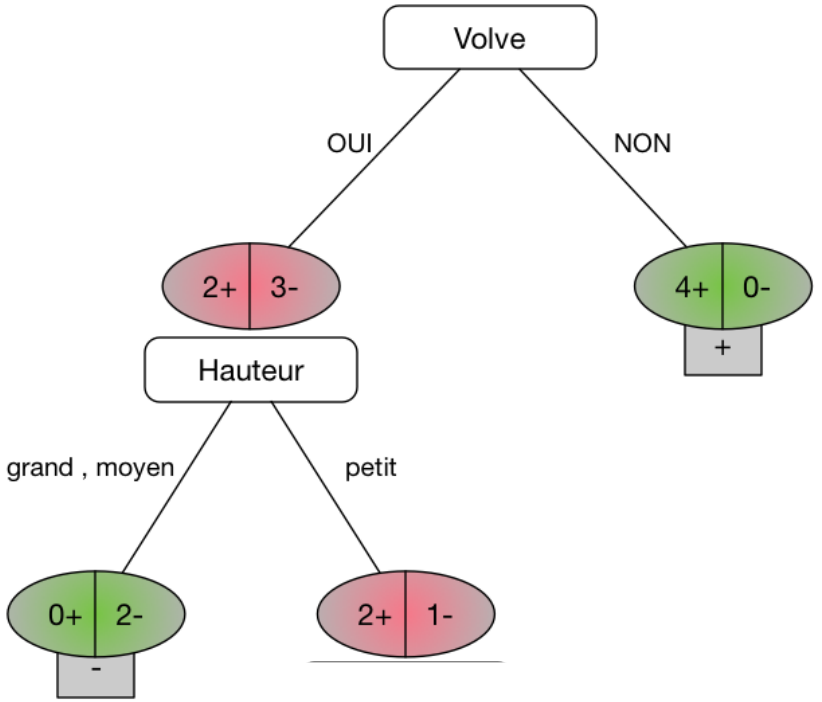
Decouper(S) :

- Choisir un descripteur discriminant : S=S'US"
- Analyser la « pureté » des deux sous-ensembles d'exemples S' et S"
- Pour chaque sous-ensemble S' insuffisamment « pure » : **Decouper(S')**

Les « arbres de décision »

Principe : apprendre un ensemble de règles de décisions, structuré sous forme d'un arbre (le modèle), à partir d'exemples étiquetés.

Champignons	Hauteur	Couleur	Dessous	Anneau	Volve	Classse
C1	grand	blanc	lamelles	non	non	+
C2	moyen	blanc	mousse	non	non	+
C3	petit	marron	lamelles	oui	oui	+
C4	petit	noir	lamelles	non	oui	+
C5	grand	blanc	mousse	non	non	+
C6	petit	blanc	lamelles	non	non	+
C7	grand	blanc	mousse	oui	oui	-
C8	petit	marron	mousse	oui	oui	-
C9	moyen	marron	lamelles	non	oui	-



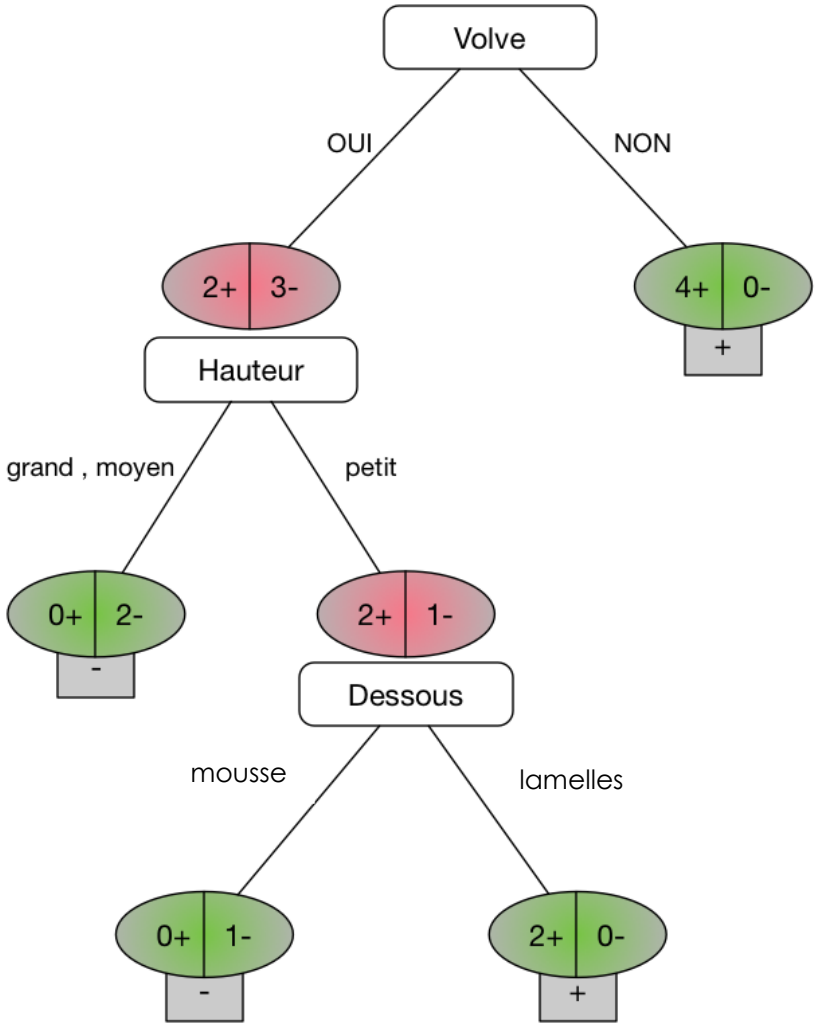
Decouper(S) :

- Choisir un descripteur discriminant : S=S'US"
- Analyser la « pureté » des deux sous-ensembles d'exemples S' et S"
- Pour chaque sous-ensemble S' insuffisamment « pure » : **Decouper(S')**

Les « arbres de décision »

Principe : apprendre un ensemble de règles de décisions, structuré sous forme d'un arbre (le modèle), à partir d'exemples étiquetés.

Champignons	Hauteur	Couleur	Dessous	Anneau	Volve	Classse
C1	grand	blanc	lamelles	non	non	+
C2	moyen	blanc	mousse	non	non	+
C3	petit	marron	lamelles	oui	oui	+
C4	petit	noir	lamelles	non	oui	+
C5	grand	blanc	mousse	non	non	+
C6	petit	blanc	lamelles	non	non	+
C7	grand	blanc	mousse	oui	oui	-
C8	petit	marron	mousse	oui	oui	-
C9	moyen	marron	lamelles	non	oui	-



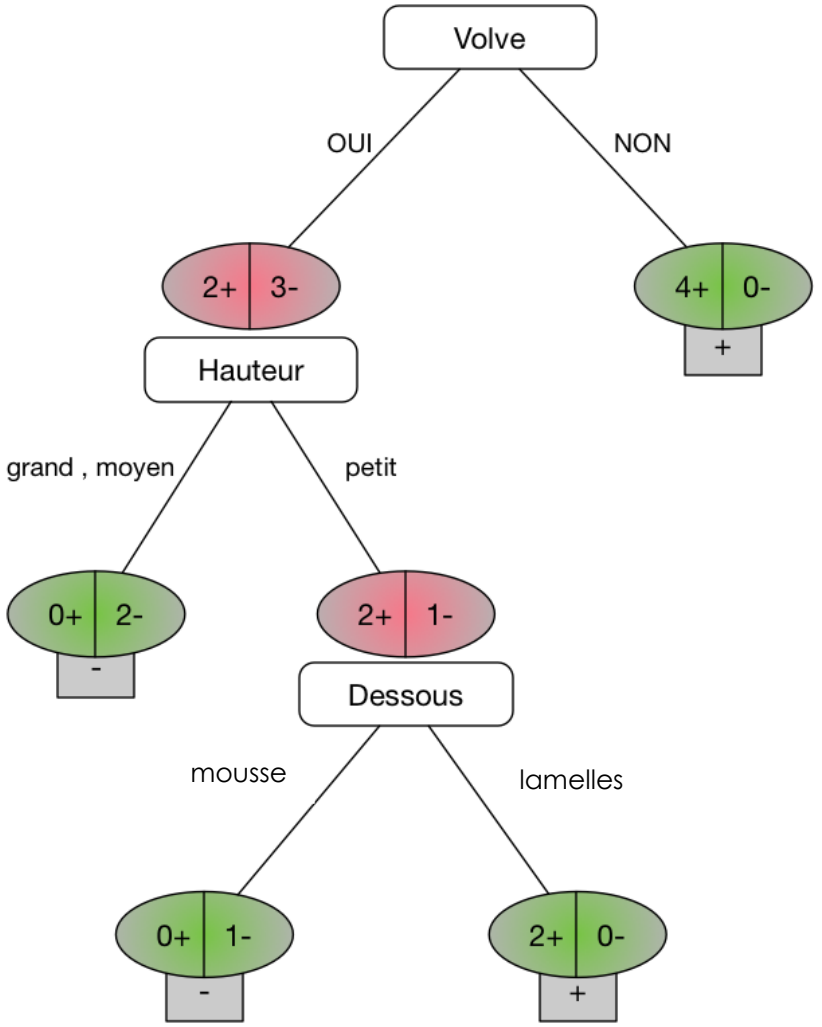
Decouper(S) :

- Choisir un descripteur discriminant : S=S'US"
- Analyser la « pureté » des deux sous-ensembles d'exemples S' et S"
- Pour chaque sous-ensemble S' insuffisamment « pure » : **Decouper(S')**

Les « arbres de décision »

Principe : apprendre un ensemble de règles de décisions, structuré sous forme d'un arbre (le modèle), à partir d'exemples étiquetés.

Champignons	Hauteur	Couleur	Dessous	Anneau	Volve	Classse
C1	grand	blanc	lamelles	non	non	+
C2	moyen	blanc	mousse	non	non	+
C3	petit	marron	lamelles	oui	oui	+
C4	petit	noir	lamelles	non	oui	+
C5	grand	blanc	mousse	non	non	+
C6	petit	blanc	lamelles	non	non	+
C7	grand	blanc	mousse	oui	oui	-
C8	petit	marron	mousse	oui	oui	-
C9	moyen	marron	lamelles	non	oui	-
C10	moyen	blanc	lamelles	oui	non	?



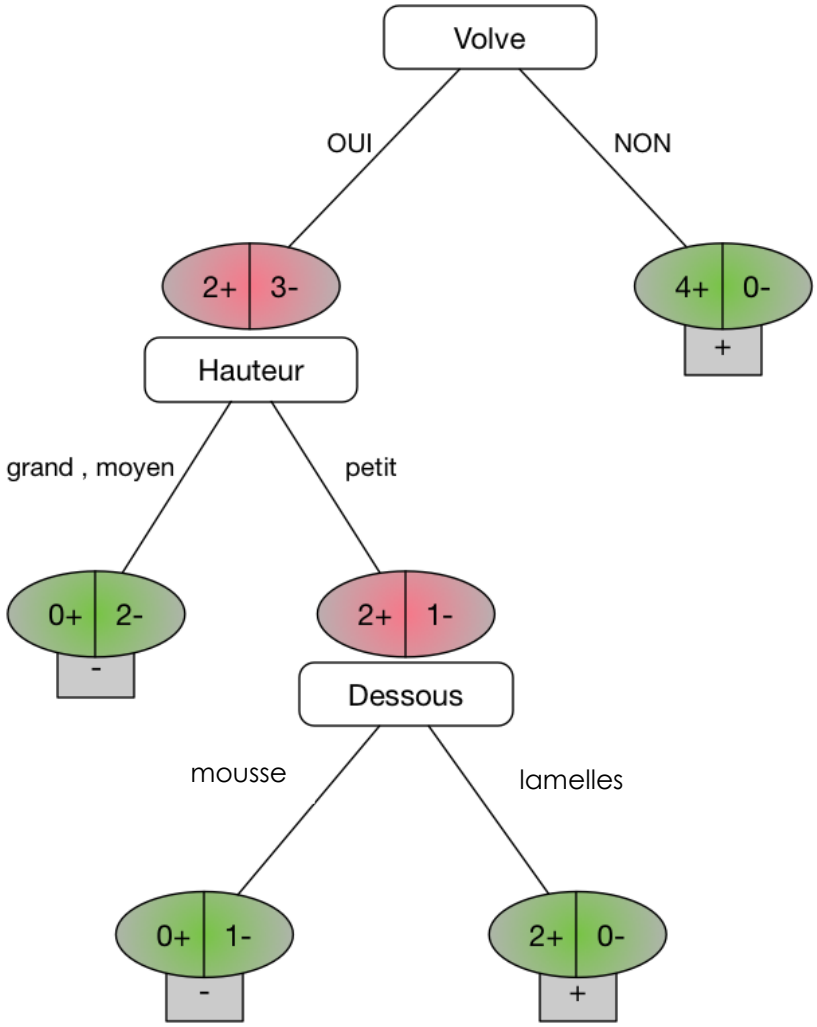
Decouper(S) :

- Choisir un descripteur discriminant : S=S'US"
- Analyser la « pureté » des deux sous-ensembles d'exemples S' et S"
- Pour chaque sous-ensemble S' insuffisamment « pure » : **Decouper(S')**

Les « arbres de décision »

Principe : apprendre un ensemble de règles de décisions, structuré sous forme d'un arbre (le modèle), à partir d'exemples étiquetés.

$$Gini(S) = \sum_{i=1}^{k} \frac{|S_i|}{|S|} \times \left(1 - \frac{|S_i|}{|S|}\right)$$



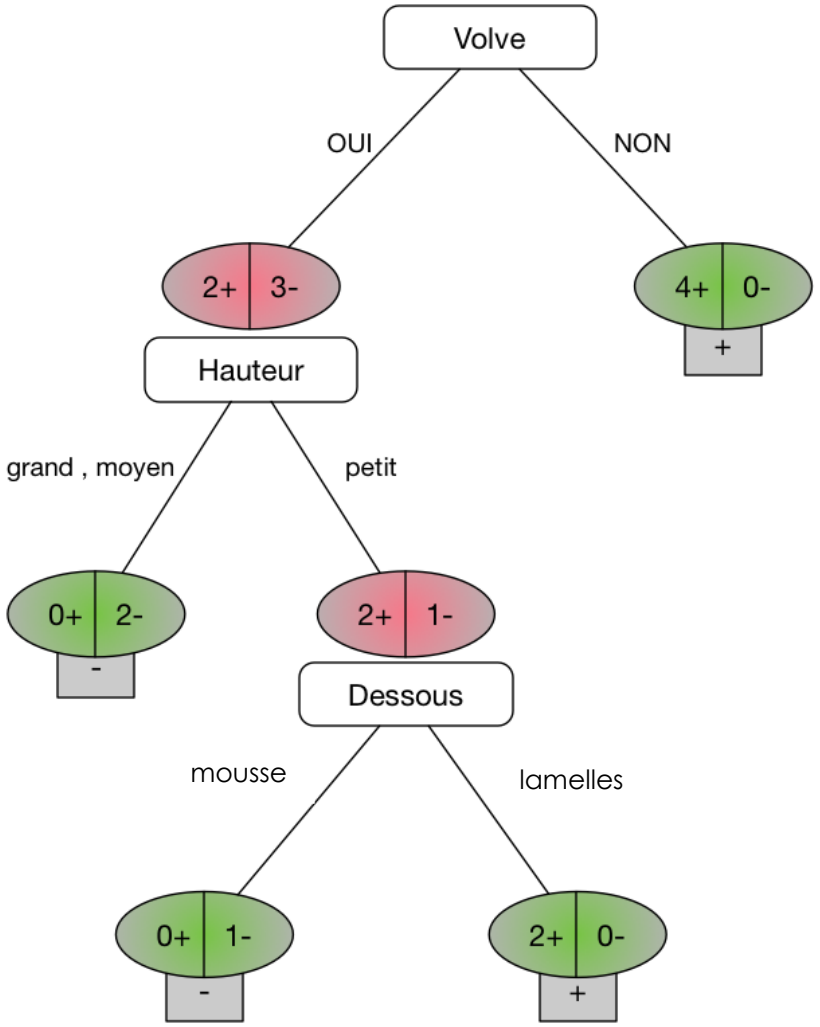
Decouper(S) :

- Choisir un descripteur discriminant : S=S'US"
- Analyser la « pureté » des deux sous-ensembles d'exemples S' et S"
- Pour chaque sous-ensemble S' insuffisamment « pure » : **Decouper(S')**

Les « arbres de décision »

Principe : apprendre un ensemble de règles de décisions, structuré sous forme d'un arbre (le modèle), à partir d'exemples étiquetés.

Seuil(s) et/ou taille minimale pour découpage



Decouper(S) :

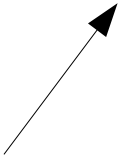
- Choisir un descripteur discriminant : S=S'US"
- Analyser la « pureté » des deux sous-ensembles d'exemples S' et S"
- Pour chaque sous-ensemble S' insuffisamment « pure » : **Decouper(S')**

Les « arbres de décision »

Package R : librairie 'rpart' (Recursive partitioning)

Phase d'apprentissage du modèle :

```
arbre <-rpart(Classe~.,data=champi,method="class",
                control=rpart.control(minsplit=2))
```



Attention : par défaut
minsplit=20

Les « arbres de décision »

Package R : librairie 'rpart' (Recursive partitioning)

Phase d'apprentissage du modèle :

```
arbre <-rpart(Classe~.,data=champi,method="class",
                control=rpart.control(minsplit=2))
```

Phase de prédiction/décision :

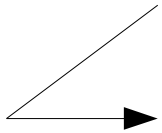
```
predict(arbre,vector,type="class")
```

Un ou plusieurs vecteurs
à classer

Les « arbres de décision »

Discussion :

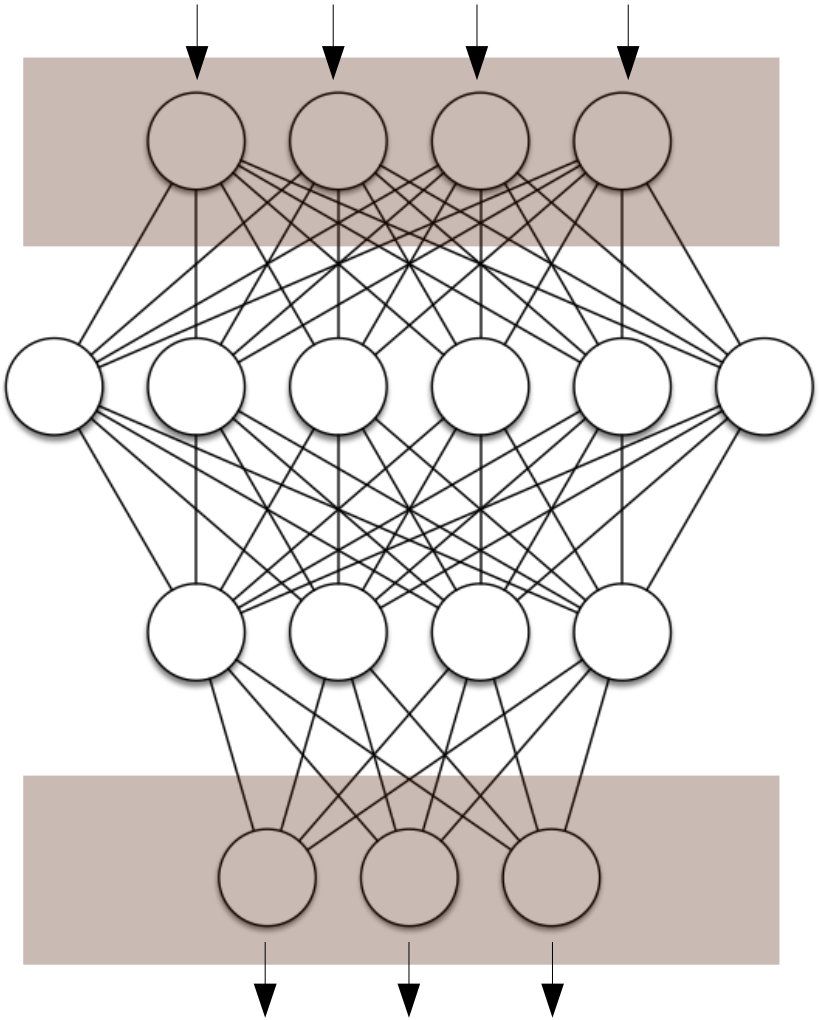
- Apprentissage d'un modèle (arbre) explicatif et réutilisable (prédiction rapide)
- L'algorithme d'apprentissage réalise une sélection automatique des descripteurs utiles pour discriminer les classes
- Adaptée principalement aux données qualitatives mais adaptable facilement aux données quantitatives (discrétisation à la volée)
- La construction de l'arbre est généralement suivie d'une étape d'**élagage** (suppression de sous-arbres trop « collés » aux données d'apprentissage => généralisation)



Problème de sur-adéquation aux données d'apprentissage (*over-fitting*)

Les « réseaux de neurones artificiels »

Principe : reproduire le fonctionnement du cerveau humain = neurones / connexions synaptiques / impulsions électriques



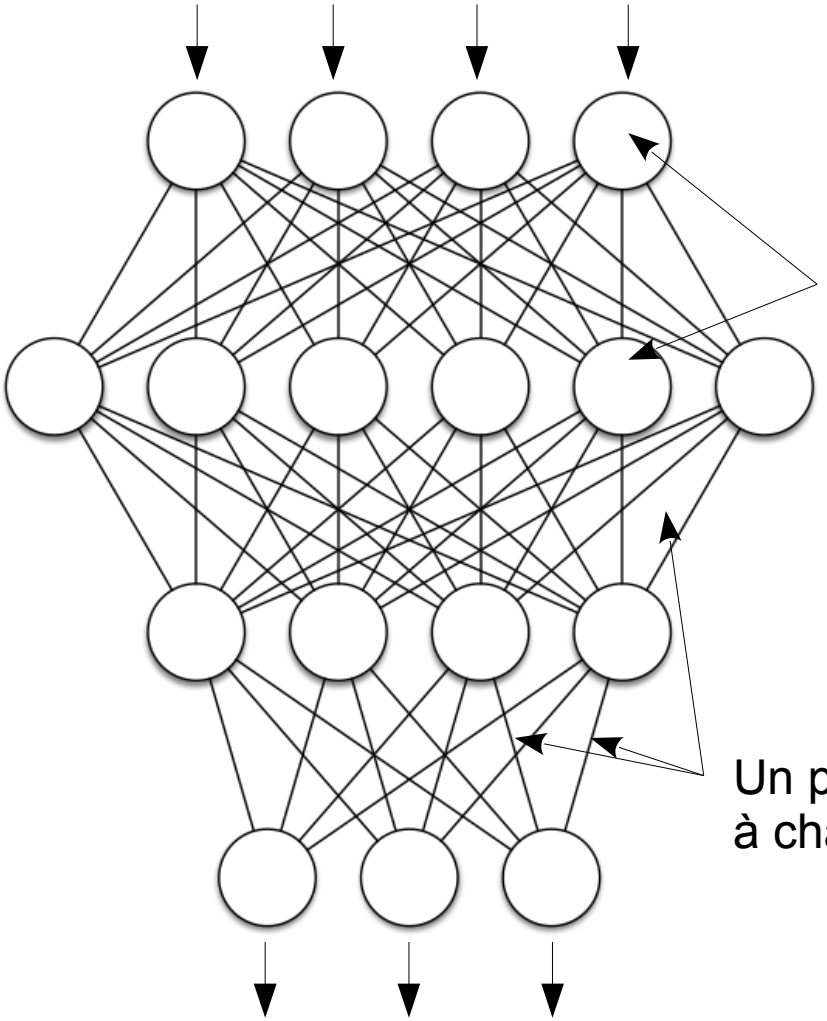
Couche d'entrée

Couche(s) cachée(s)

Couche de sortie (décision(s))

Les « réseaux de neurones artificiels »

Principe : reproduire le fonctionnement du cerveau humain = neurones / connexions synaptiques / impulsions électriques

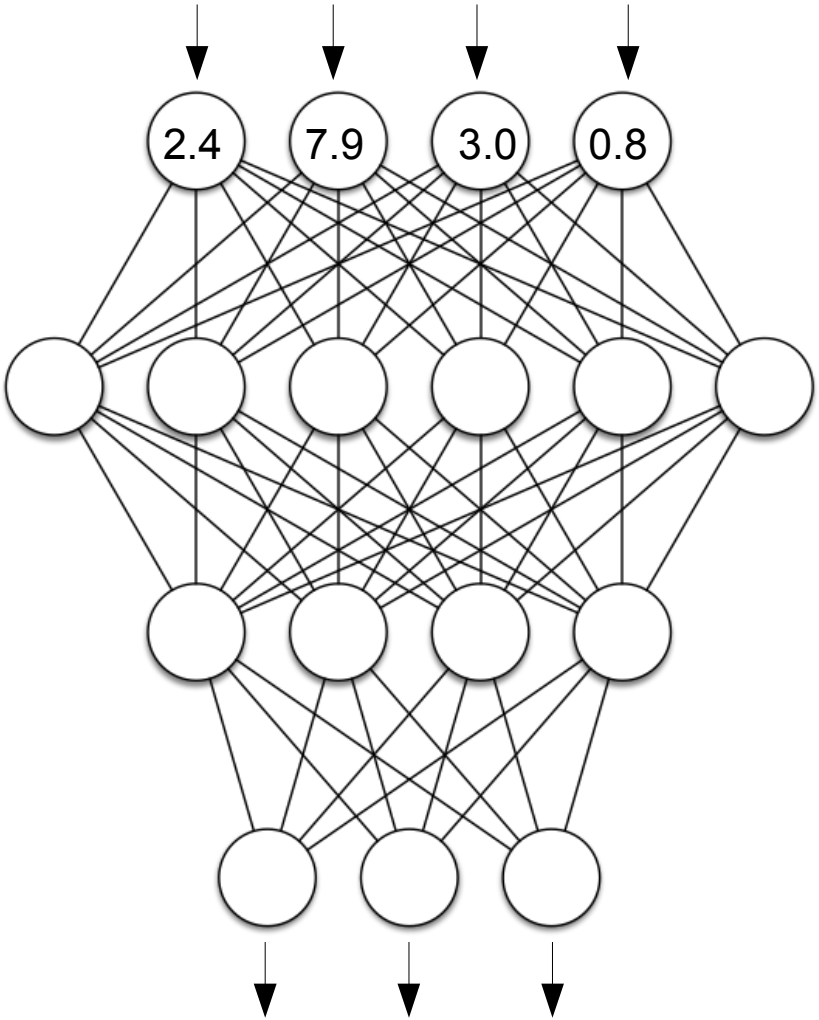


Une fonction d'activation *f()* associée à chaque neurone

Un poids *w* associée à chaque connexion

Les « réseaux de neurones artificiels »

Principe : reproduire le fonctionnement du cerveau humain = neurones / connexions synaptiques / impulsions électriques

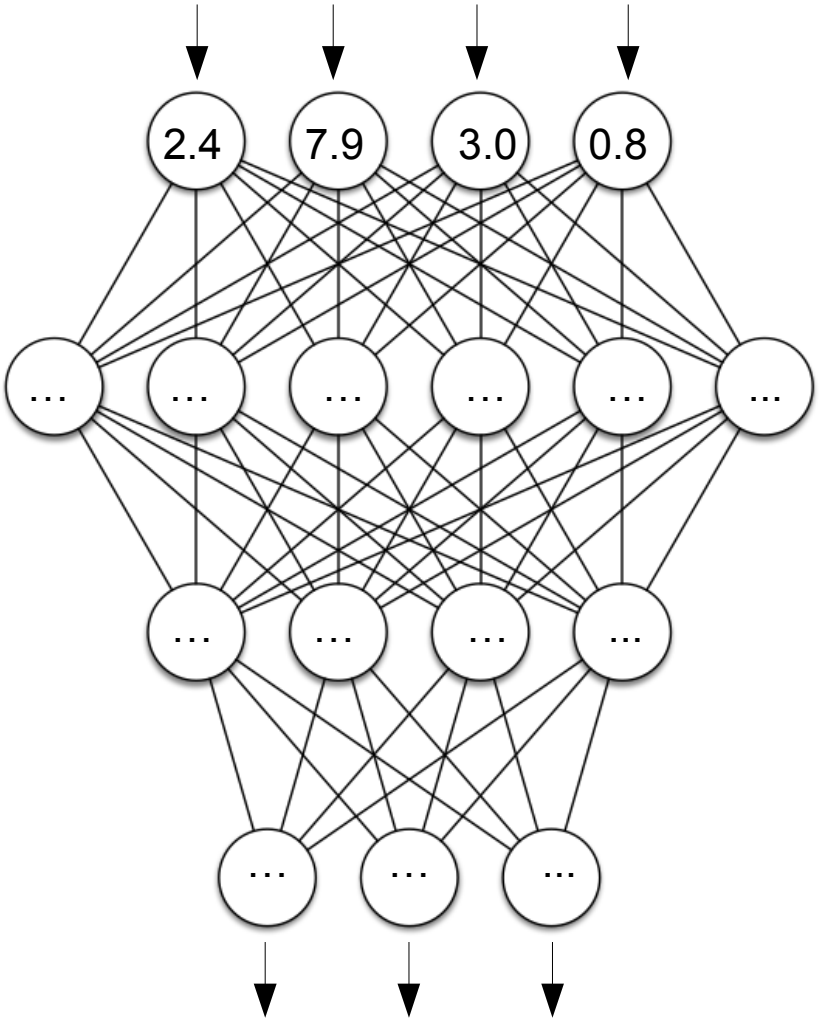


Phase d'apprentissage :

- Présenter un exemple

Les « réseaux de neurones artificiels »

Principe : reproduire le fonctionnement du cerveau humain = neurones / connexions synaptiques / impulsions électriques

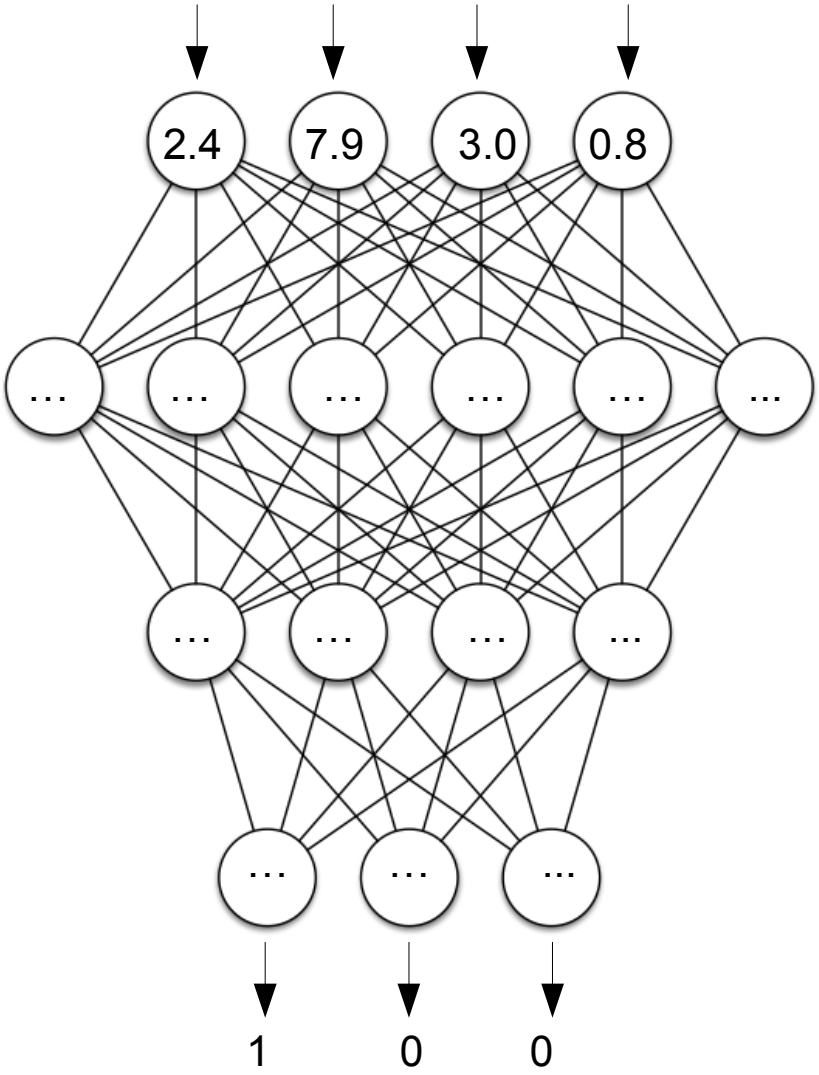


Phase d'apprentissage :

- Présenter un exemple
- Faire transiter dans le réseau

Les « réseaux de neurones artificiels »

Principe : reproduire le fonctionnement du cerveau humain = neurones / connexions synaptiques / impulsions électriques



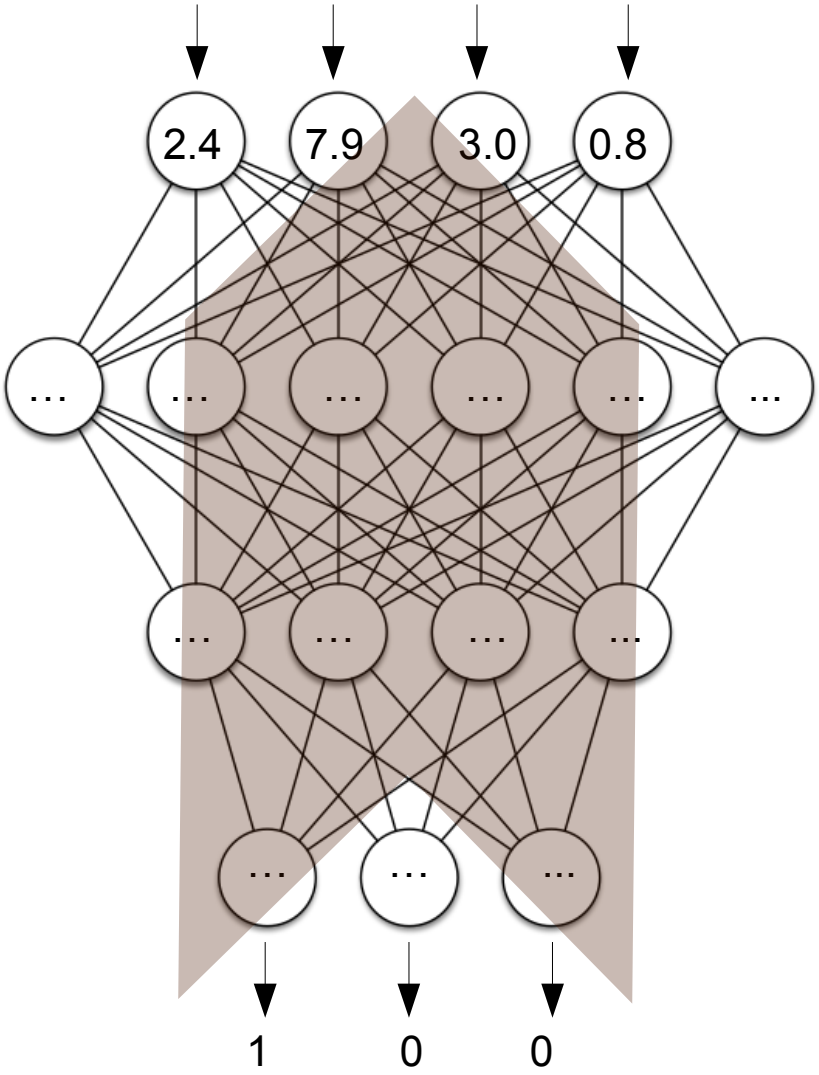
Phase d'apprentissage :

- Présenter un exemple
- Faire transiter dans le réseau
- Comparer :

  sortie obtenue vs. Sortie attendue

Les « réseaux de neurones artificiels »

Principe : reproduire le fonctionnement du cerveau humain = neurones / connexions synaptiques / impulsions électriques



Phase d'apprentissage :

FAIRE

- Présenter un exemple
- Faire transiter dans le réseau
- Comparer :
  sortie obtenue vs. Sortie attendue
- Corriger les poids du réseau en conséquence

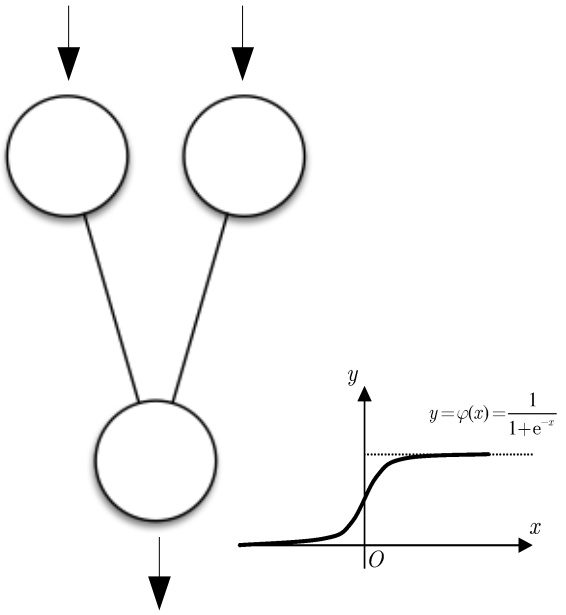
JUSQU'A convergence

Les « réseaux de neurones artificiels »

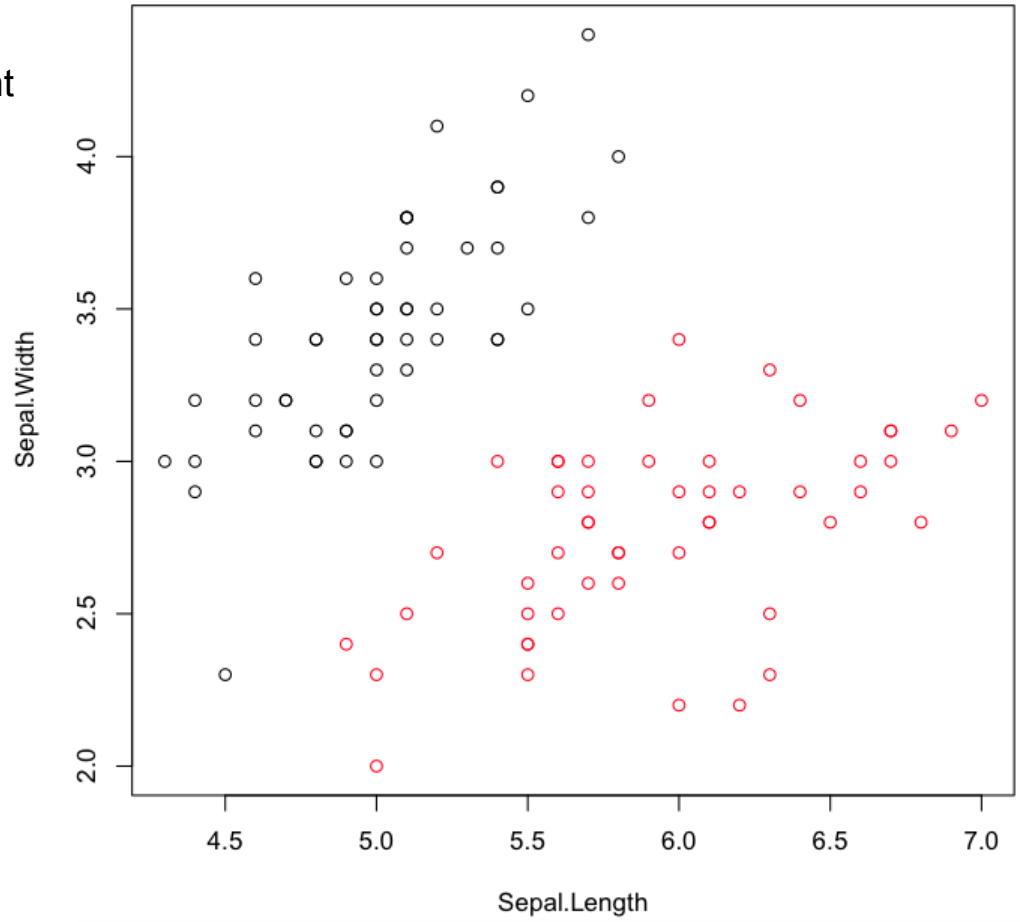
Exemple : 2 classes d'iris à discriminer

49	5.3	3.7	setosa
50	5.0	3.3	setosa
51	7.0	3.2	versicolor
52	6.4	3.2	versicolor
53	6.9	3.1	versicolor
54	5.5	2.3	versicolor
55	6.5	2.8	versicolor
56	5.7	2.8	versicolor
57	6.3	3.3	versicolor
58	4.9	2.4	versicolor

Classe codée
Numériquement
(ex. 0 et 1)



~ 1 ou 0

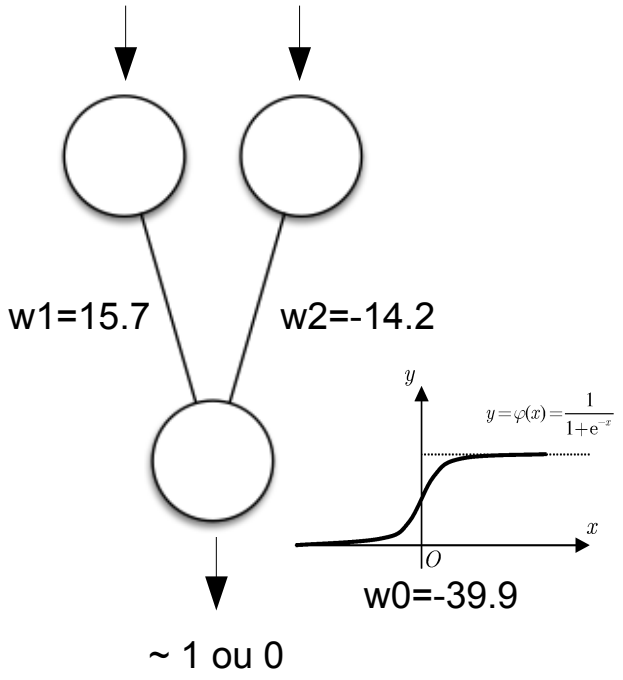


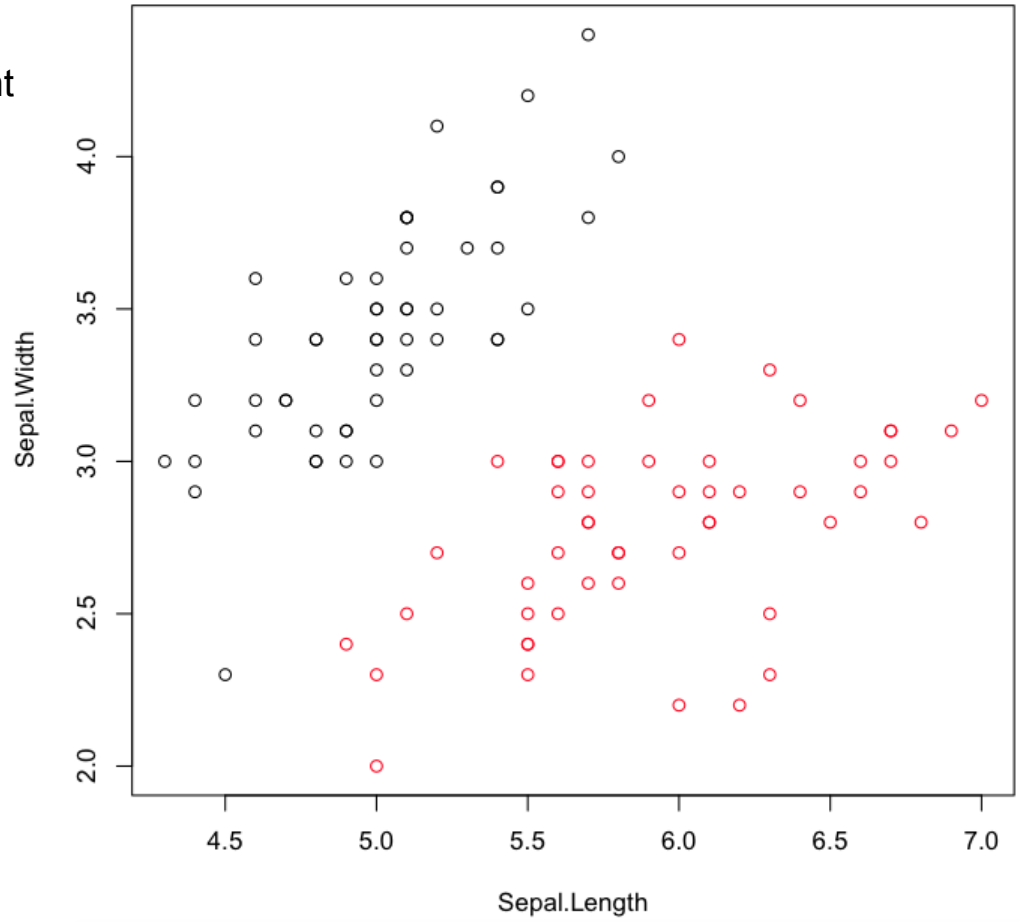
Les « réseaux de neurones artificiels »

Exemple : 2 classes d'iris à discriminer

49	5.3	3.7	setosa
50	5.0	3.3	setosa
51	7.0	3.2	versicolor
52	6.4	3.2	versicolor
53	6.9	3.1	versicolor
54	5.5	2.3	versicolor
55	6.5	2.8	versicolor
56	5.7	2.8	versicolor
57	6.3	3.3	versicolor
58	4.9	2.4	versicolor

Classe codée
Numériquement
(ex. 0 et 1)



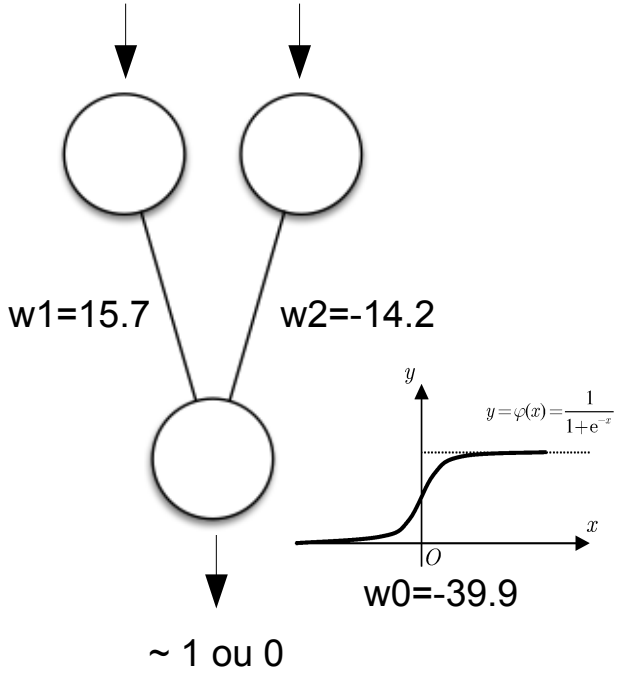


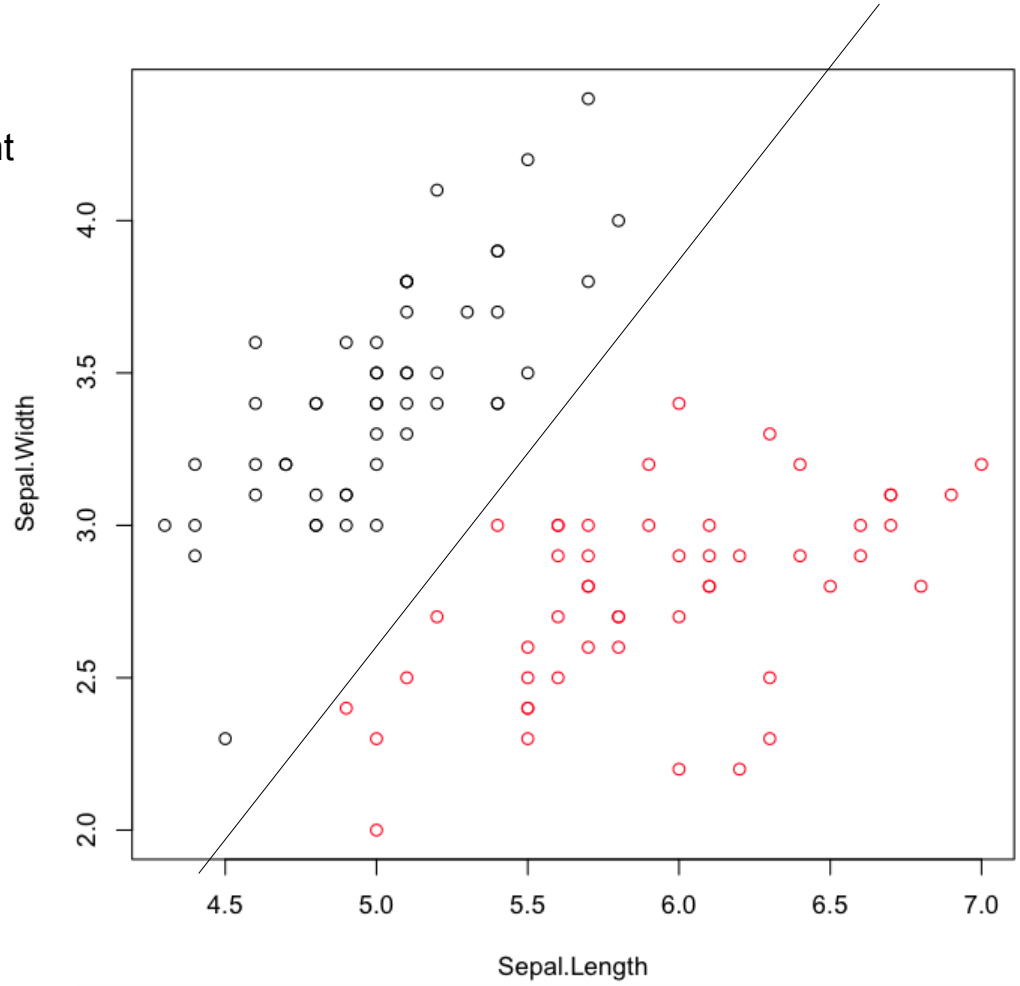
Les « réseaux de neurones artificiels »

Exemple : 2 classes d'iris à discriminer

49	5.3	3.7	setosa
50	5.0	3.3	setosa
51	7.0	3.2	versicolor
52	6.4	3.2	versicolor
53	6.9	3.1	versicolor
54	5.5	2.3	versicolor
55	6.5	2.8	versicolor
56	5.7	2.8	versicolor
57	6.3	3.3	versicolor
58	4.9	2.4	versicolor

Classe codée
Numériquement
(ex. 0 et 1)





Les « réseaux de neurones artificiels »

Package R : librairie 'nnet' (neural networks)

Phase d'apprentissage du modèle :

```
nn <-nnet(x=iris[1:100,1:2],y=c(rep(0,50),rep(1,50)),size=0,skip=TRUE)
```

- Tableau des données d'apprentissage (sans l'étiquette)
- Sorties attendues (classe encodée)
- Nb. couches cachées
- Liaisons directes Couche d'entrée → sortie

Phase de prédiction/décision :

```
predict(nn,vector)
```

- Un ou plusieurs vecteurs à classer

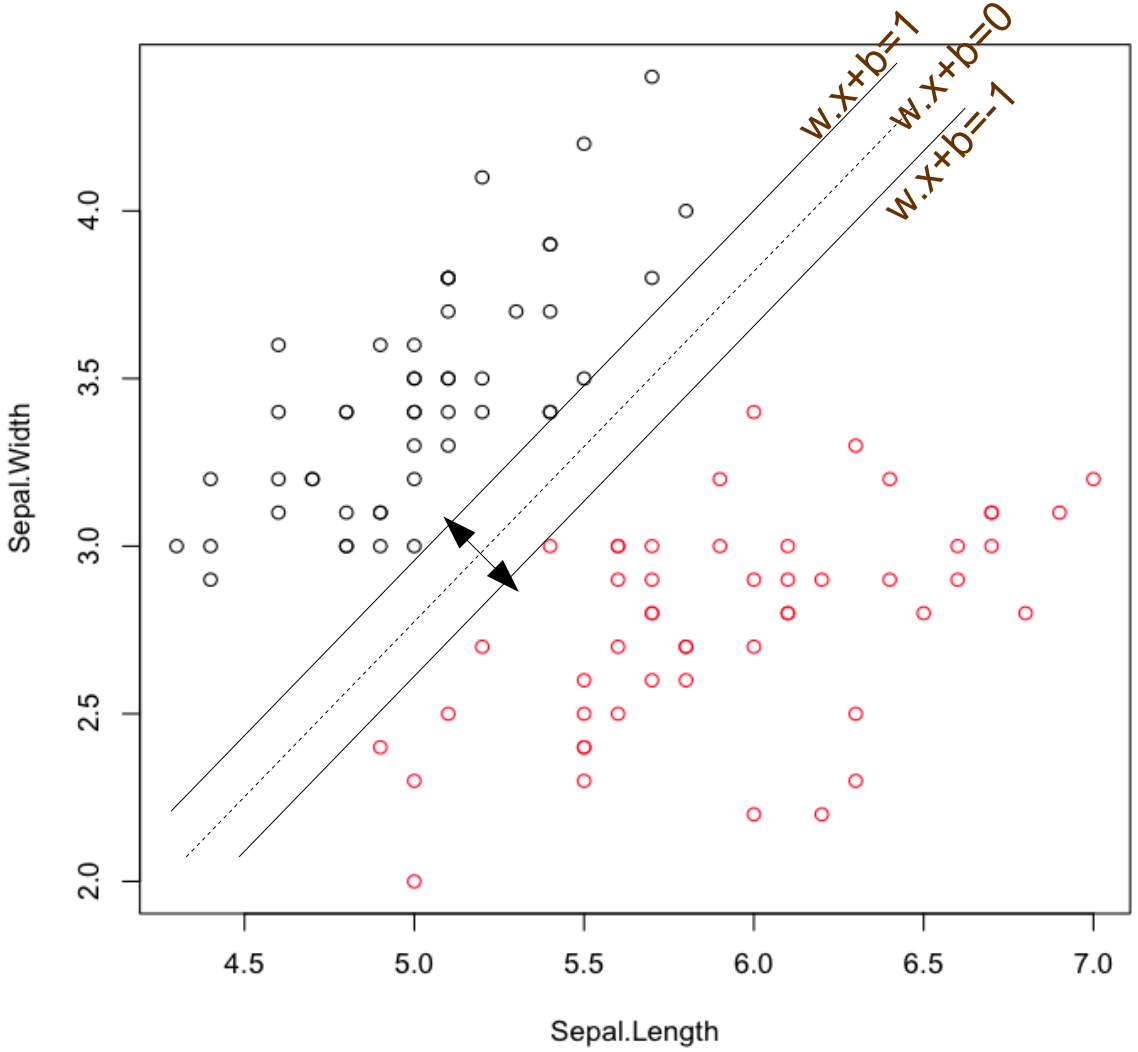
Les « réseaux de neurones artificiels »

Discussion :

- Apprentissage d'un modèle (réseau de neurones) peu explicatif mais réutilisable (prédiction rapide)
- Paramètres à choisir : structure du réseau & fonctions d'activation
- Paramètres à apprendre : poids des connexions
- Sans couche cachées → modèle = séparateur linéaire
- Avec couche(s) cachée(s) → séparateurs non-linéaires
- Adaptée aux données quantitatives
- Deep-Learning : Réseaux de neurones profonds et/ou récurrents

Les SVM *support vector machine* / séparateur à vaste marge

Principe : apprendre un séparateur linéaire qui maximise la marge entre les 2 classes d'exemples à séparer



Formalisation du problème :

Minimisation de l'expression

$$\frac{1}{2} w.w + C \sum_{k=1}^{R} \epsilon_k$$

Sous *R* contraintes à satisfaire (les points sont bien classés)

→ se réécrit comme une expression

$$f(\{< x_i.x_j >\}_{(i,j)})$$

Les SVM *support vector machine* / séparateur à vaste marge

Package R : librairie 'e1071'

Phase d'apprentissage du modèle :

```
model <-svm(x=iris[1:100,1:2],y=c(rep(0,50),rep(1,50)),type='C',kernel='linear')
```

- Tableau des données d'apprentissage (sans l'étiquette)
- Sorties attendues (classe encodée)
- Paramètre de l'algo.
- Type de noyau

Phase de prédiction/décision :

```
predict(model,vector)
```

- Un ou plusieurs vecteurs à classer

Questions ?