Big Data Analytics

Jour 2 — Algorithmes 1/2

François-Marie Giraud

https://www.orsys.fr/

Apprentissage supervisé

Apprentissage supervisé

Naive Bayes

Rappels - Probabilités

Rappels :

- Une variable aléatoire
  $A \in \mathbb{R}$
- Probabilité
  $0 \leq P(A \in [a_1\ a_2]) \leq 1$
- Probabilité conditionnelle
  $P(\ A > 0 \mid B < -3\ )$
- Évènements indépendants
  $P(A|B) = P(A)$ et $P(B|A) = P(B)$
- Probabilité jointe
  $P(A, B) = P(B|A) * P(A)$
  $P(A, B) = P(A|B) * P(B)$
- A et B Indépendants
  $\iff P(A, B) = P(A) * P(B)$

Théorème de Bayes

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

Naive Bayes

Prenons un exemple :
Soit une base de donnée de fruits contenant uniquement des bananes ,oranges ,courgettes.
Chaque élément possède des caractéristiques couleur, taille, sucré.
Appliquer Naive Bayes, c'est chercher le maximum de vraisemblance d'un élèments dont on ne connait pas la nature mais dont on connait les caractéristiques.
On cherche donc quelle est la plus grande probabilité :

- P(banane | jaune,long,sucré)
- P(orange | jaune,long,sucré)
- P(tomate | jaune,long,sucré)

Naive Bayes

Naive = toutes les variables sont considérée indépendantes, donc :

$$P(banane|jaune, long, sucre) =$$

$$\frac{P(jaune|banane)*P(long|banane)*P(sucre|banane)*P(banane)}{P(jaune)*P(long)*P(sucre)}$$

Pour estimer les différentes probabilités, on 'compte' dans notre base de donnée de fruits :

$$P(sucre|banane) = \frac{card(banane\ ET\ sucre)}{card(banane)}$$

Apprentissage supervisé

Regression Linéaire

Regression Linéaire

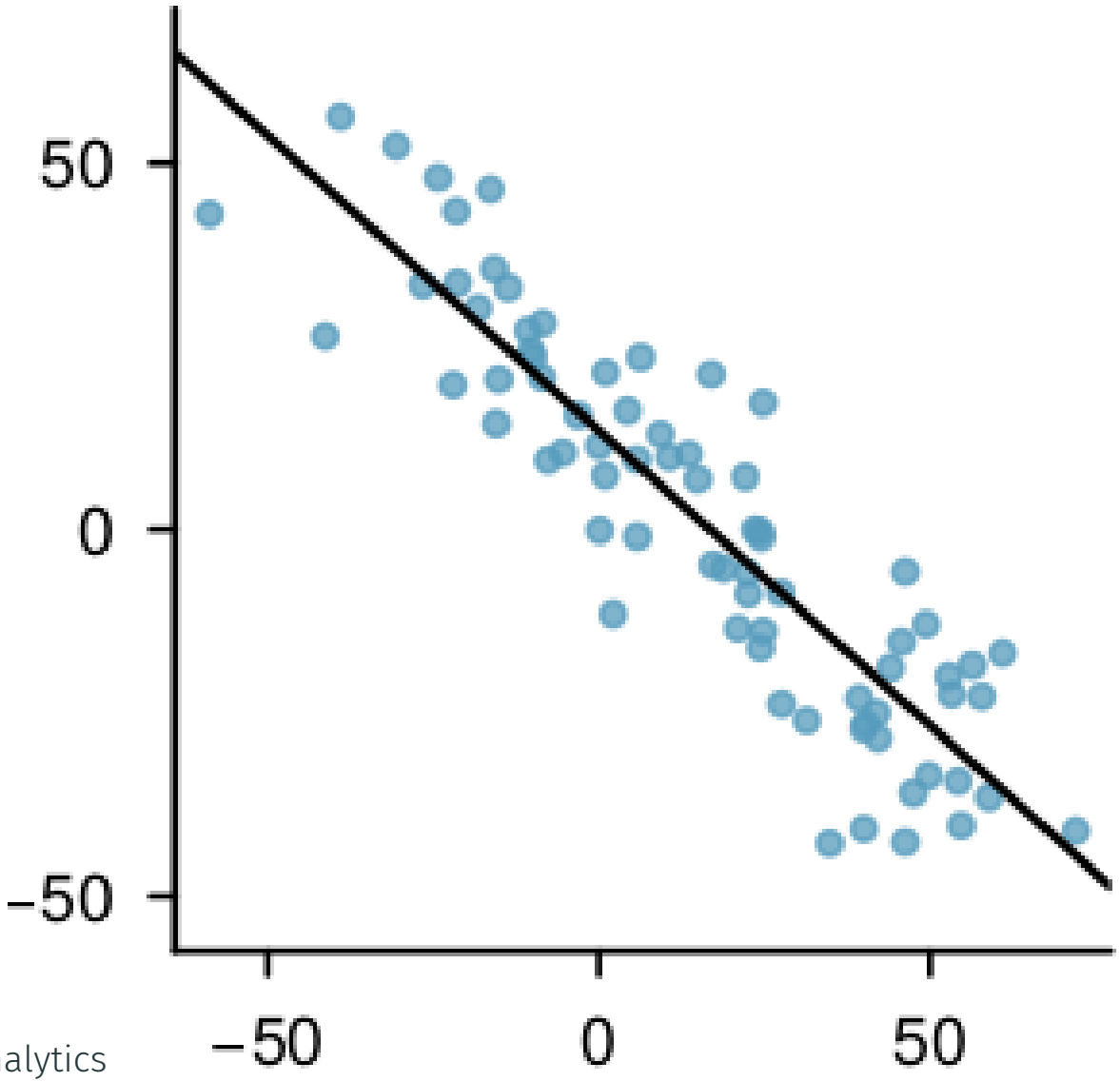
Définition du problème :

Soit $\{(x_i, y_i)\}_{i \in \mathbb{R}}$ un ensemble de données tel que $\forall i, x_i \in \mathbb{R}$ et $y_i \in \mathbb{R}$

Trouver $\phi^*(x_i) = y_i^*$ telle que

$\forall i, y_i^* - y_i \to 0$
sous la contrainte que ϕ^* soit une fonction linéaire (affine)

Regression Linéaire



Apprentissage supervisé

Fonction de Coût/Erreur

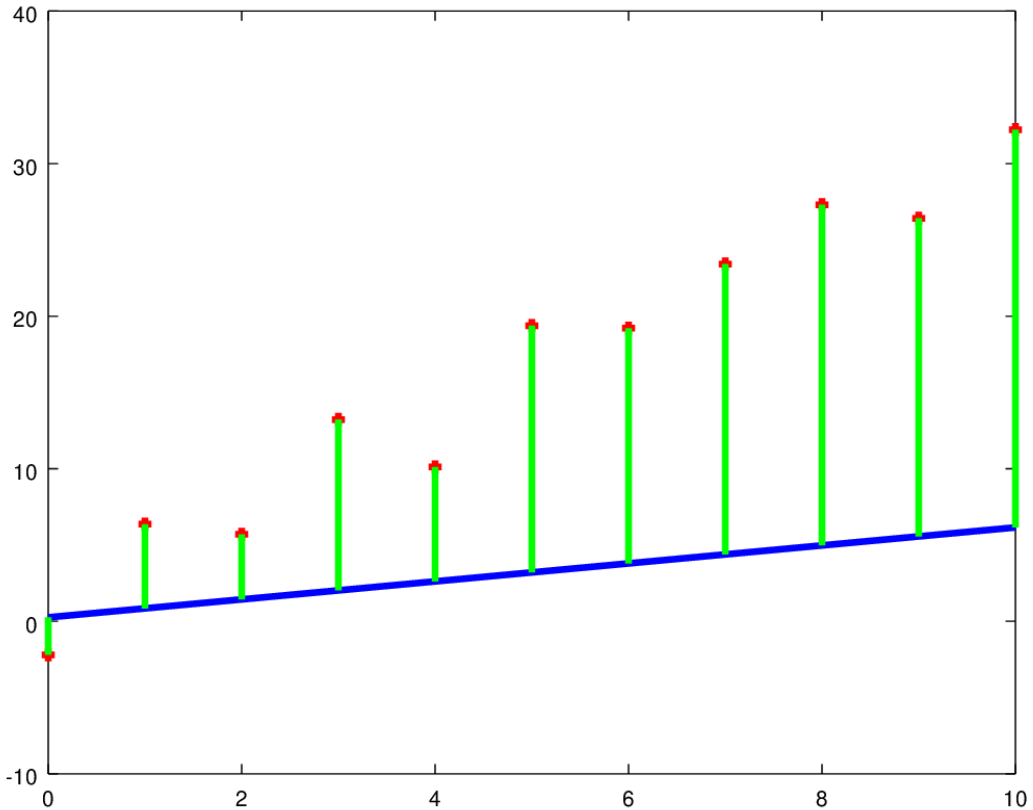
Fonction de Coût/Erreur

Erreur moyenne :

$$\frac{1}{n} \sum_{i=[1..n]} \sqrt{(\hat{y}_i - y_i)^2}$$

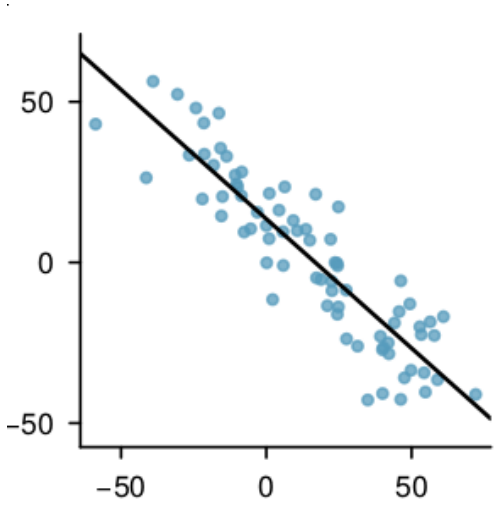
Critère des moindres carrés

$$\frac{1}{n} \sum_{i=[1..n]} (\hat{y}_i - y_i)^2$$



Fonction de Coût/Erreur

$$y = a.x + b$$



$$E_\Omega = \frac{1}{2n} \sum_{i=[1..n]} (\hat{y}_i - y_i)^2$$

Apprentissage supervisé

Optimisation

Optimisation

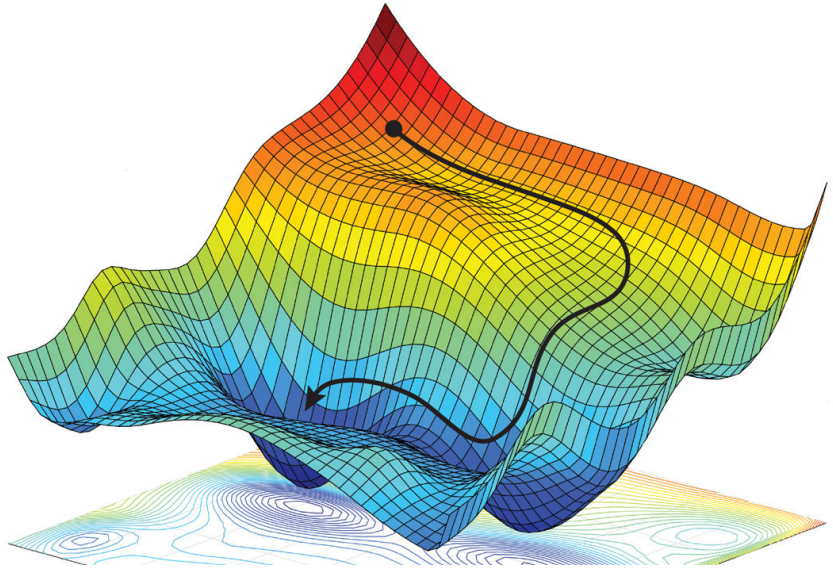
Calcul du gradient de l'erreur par rapport aux paramètres :

$$\frac{\partial Err}{\partial w_i}$$

Mise à jour :

$$w_i' = w_i - \gamma * grad$$

où : $0 < \gamma < 1$ (learning rate)



Optimisation

1 - initialisation aléatoire du modèle
2 - Tant que(critère arret == 0)

- Selection aléatoire d'un **batch** de données
- **Forward** : Passe avant du **batch** dans le modèle
- Calcul de l'erreur par rapport aux sorties attendues
- **Backward** : Rétropropagation du gradient de l'erreur en fonction des paramèrtres dans le modèle (mise à jour du modèle)
- Calcul critère arret

Apprentissage supervisé

Gradient de l’erreur

Gradient et mise à jour

$$\boxed{y = a.x + b}$$

$$E_\Omega = \frac{1}{2n} \sum_{i=[1..n]} (\hat{y}_i - y_i)^2$$

$$E_\Omega = \frac{1}{2n} \sum_{i=[1..n]} (\hat{y}_i - (a.x_i + b))^2$$

...

$$\frac{\partial E_\Omega}{\partial a} = \frac{1}{n} \sum_{i=[1..n]} (a.x_i + b - \hat{y}_i).x$$

$$\frac{\partial E_\Omega}{\partial b} = \frac{1}{n} \sum_{i=[1..n]} (a.x_i + b - \hat{y}_i)$$

$$\boxed{U^{2\,\prime} = 2U' * U}$$

M.A.J :

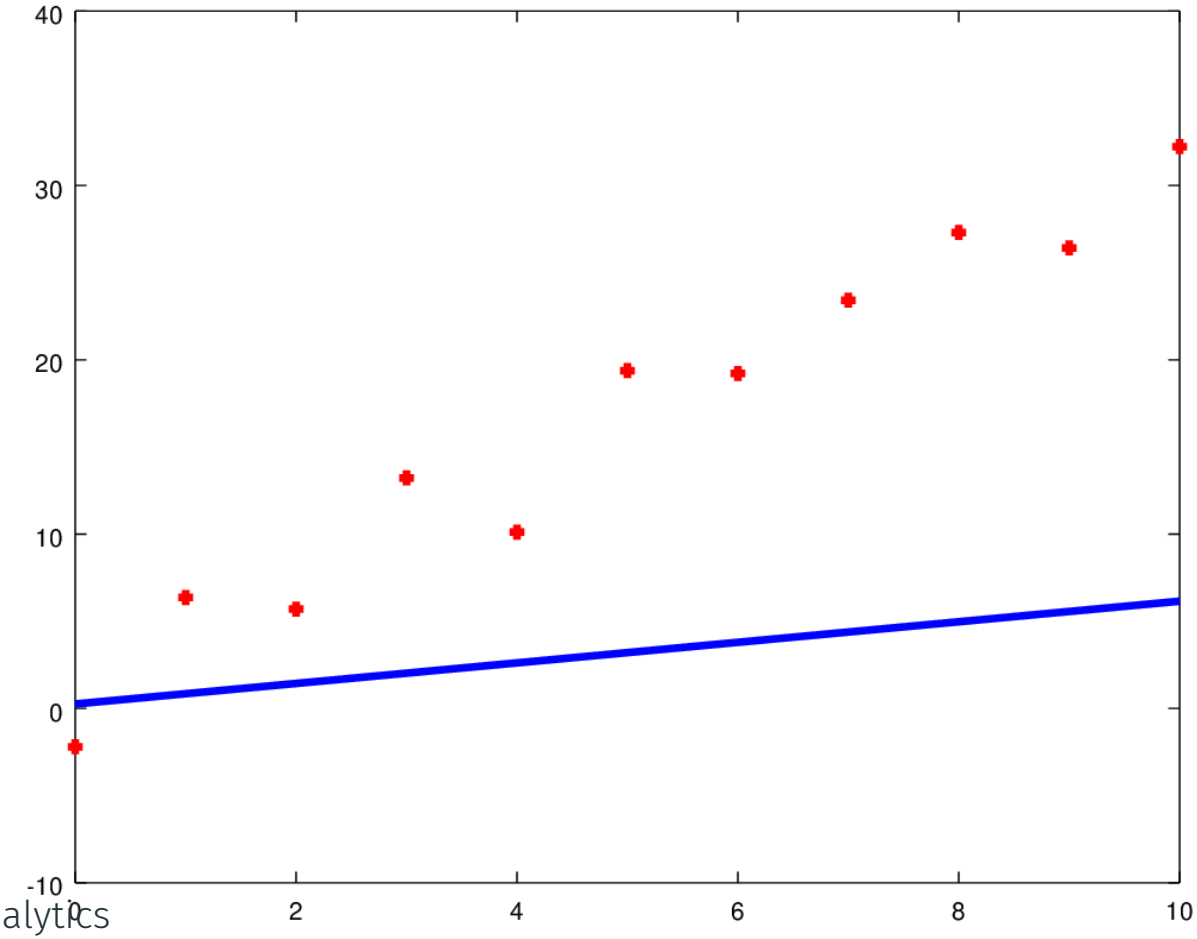
$$a \leftarrow a - \gamma . \frac{\partial E_\Omega}{\partial a}$$

$$b \leftarrow b - \gamma . \frac{\partial E_\Omega}{\partial b}$$

où $1 > \gamma > 0$ (learning rate)

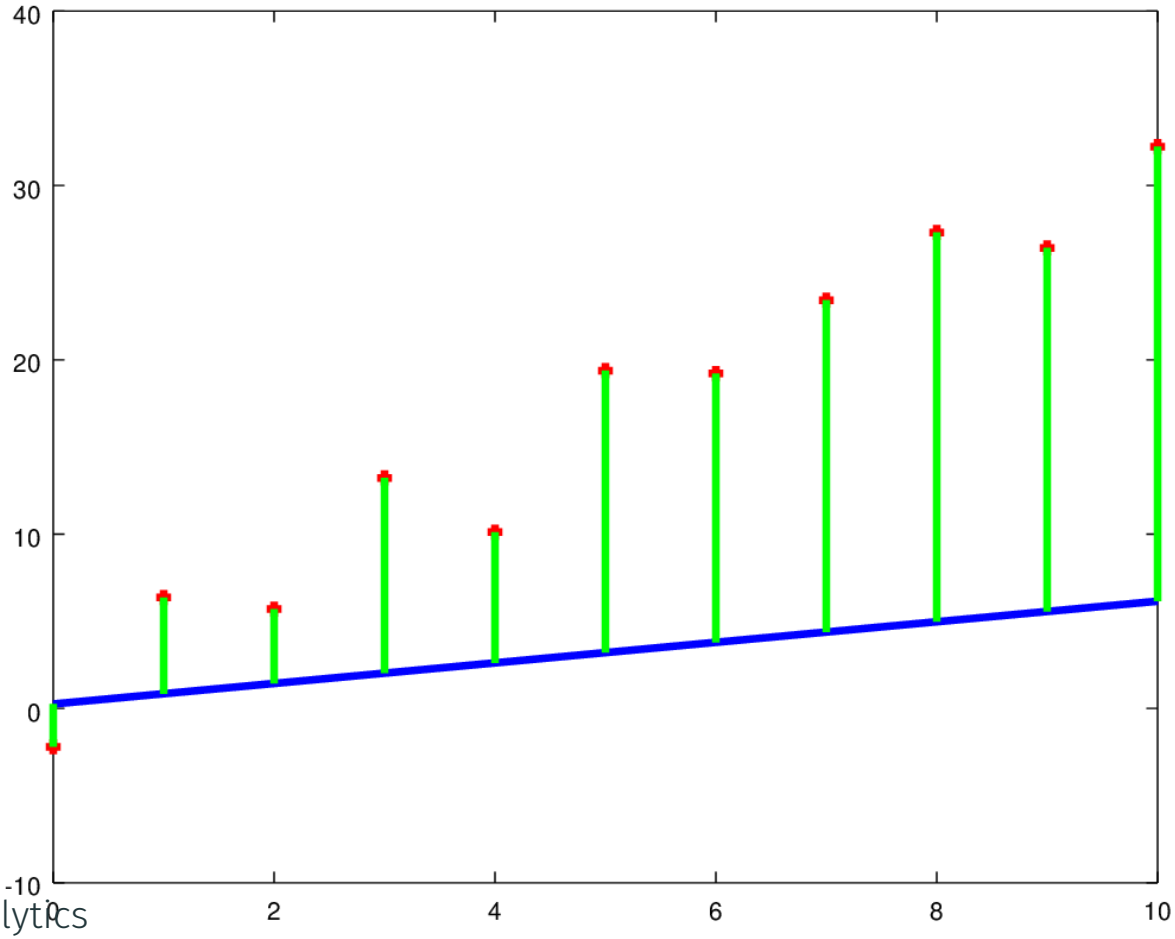
Initialisation au hasard (γ = 0.01)

- a = 0.58 ($\hat{a} = 3.0$)
- b = 0.25 ($\hat{b} = 0.5$)



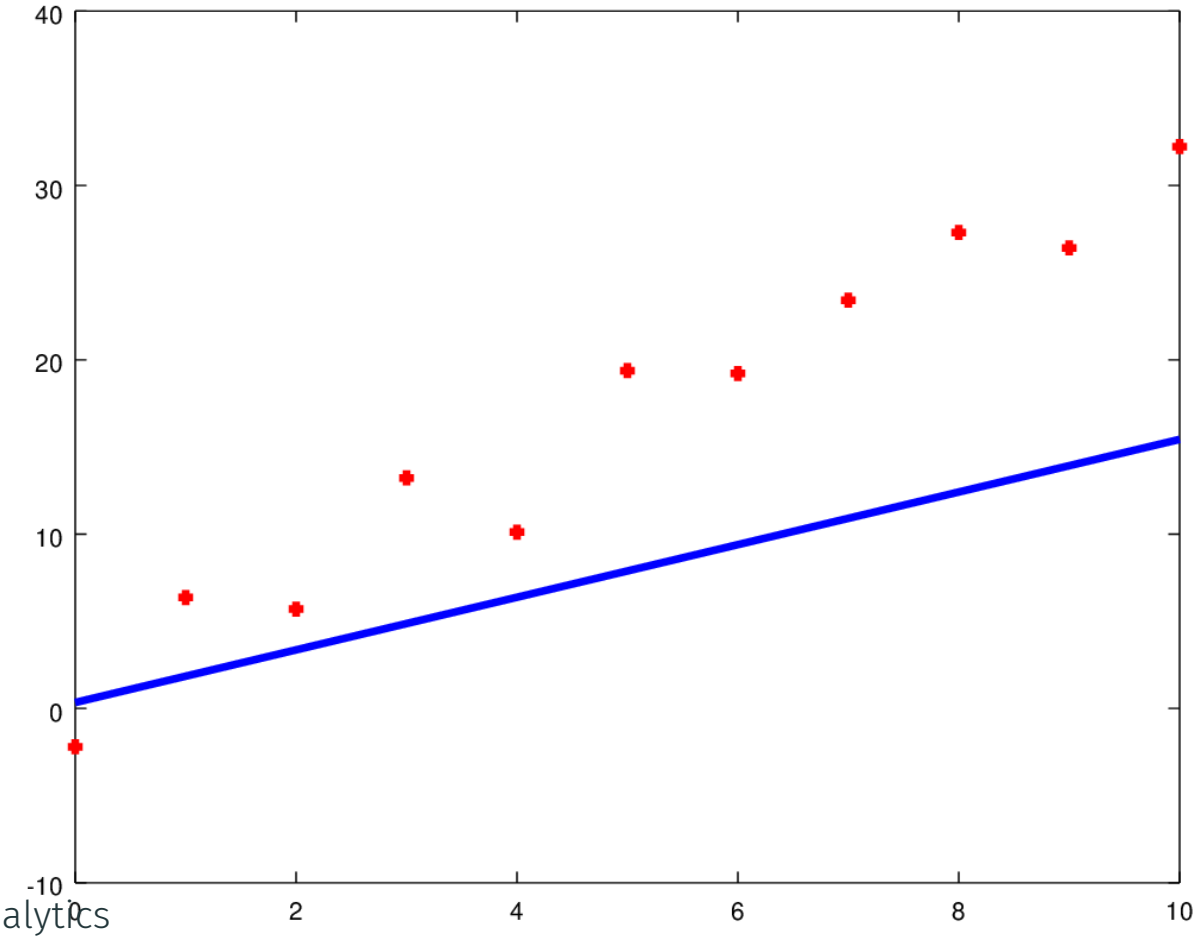
Gradient et mise à jour

- a = 0.58 ($\hat{a} = 3.0$)
- b = 0.25 ($\hat{b} = 0.5$)



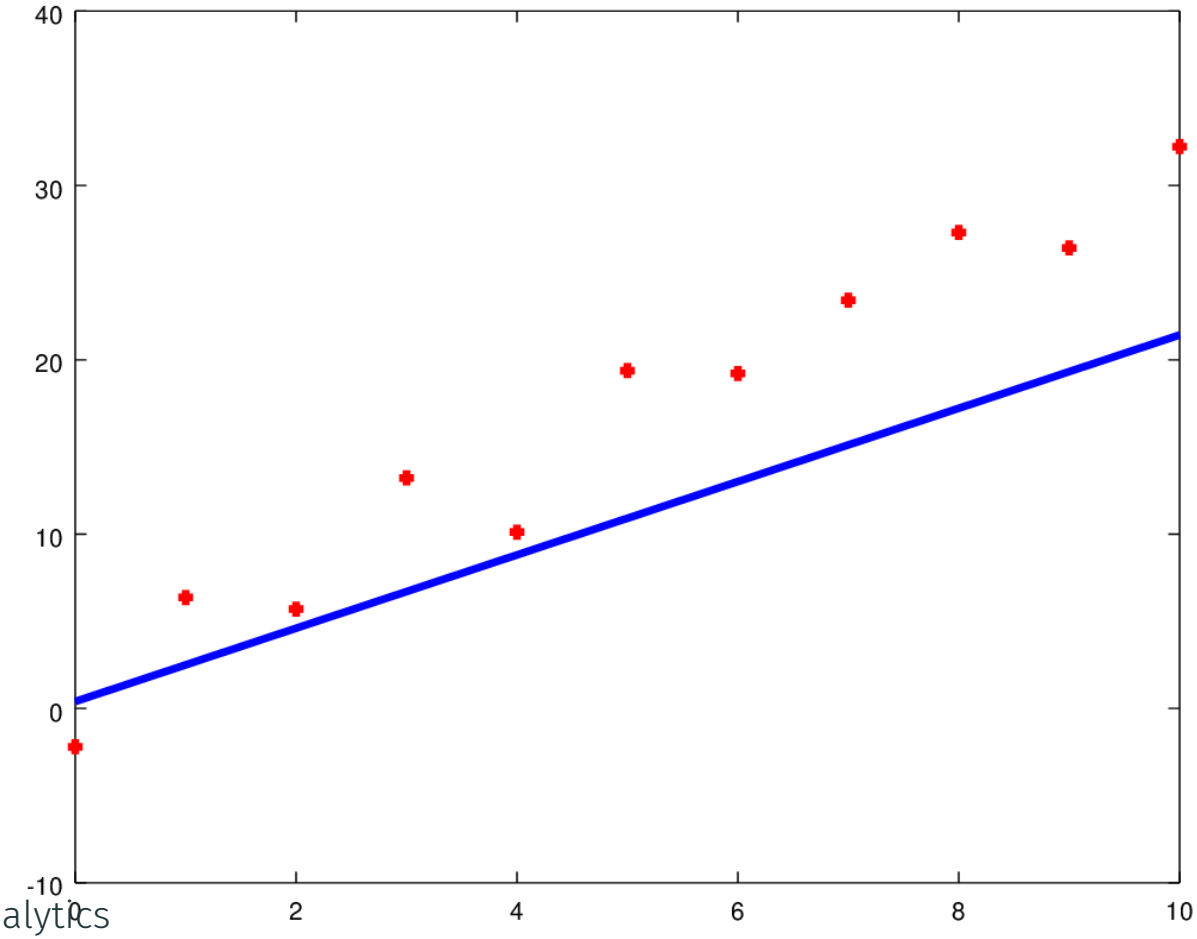
Gradient et mise à jour

- a = 1.50 ($\hat{a} = 3.0$)
- b = 0.35 ($\hat{b} = 0.5$)



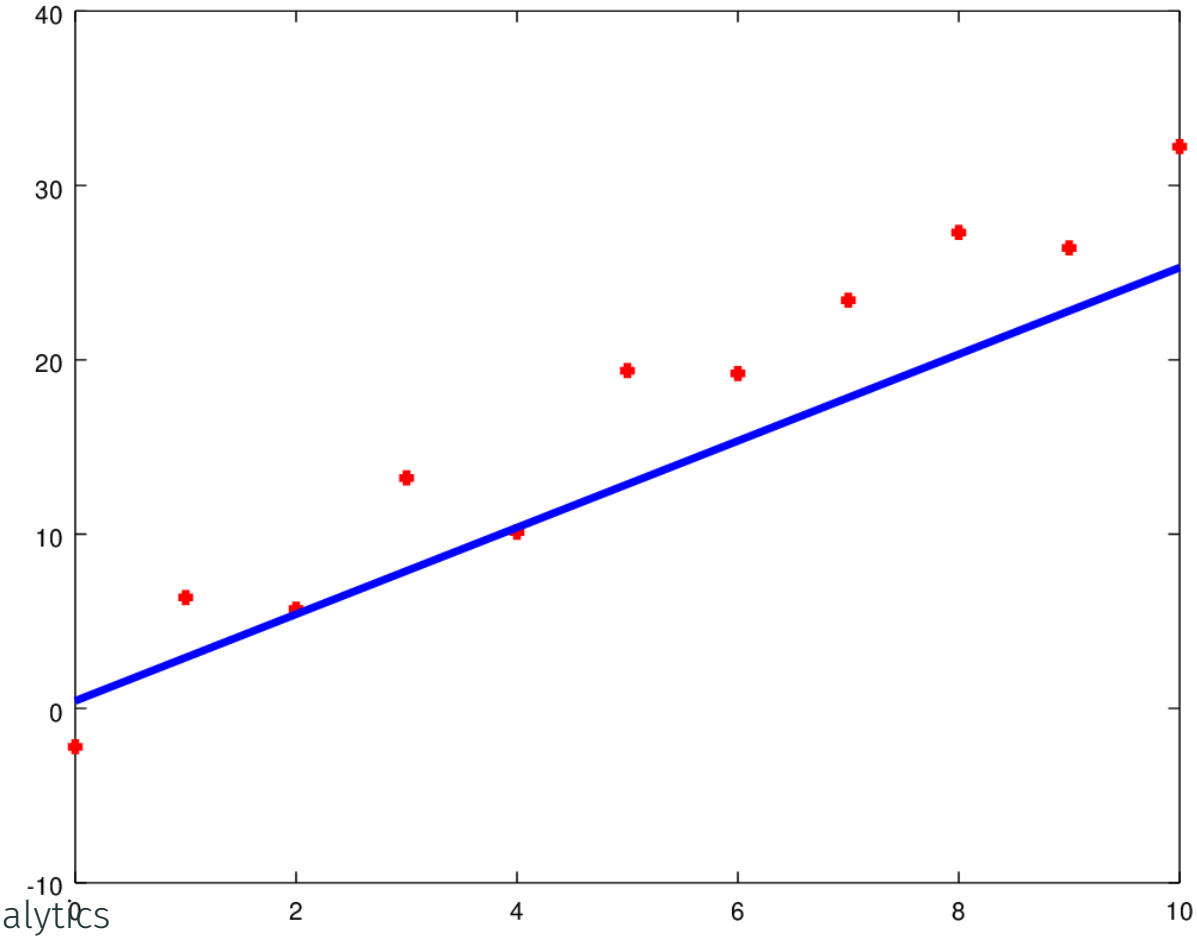
Gradient et mise à jour

- a = 2.10 ($\hat{a} = 3.0$)
- b = 0.40 ($\hat{b} = 0.5$)



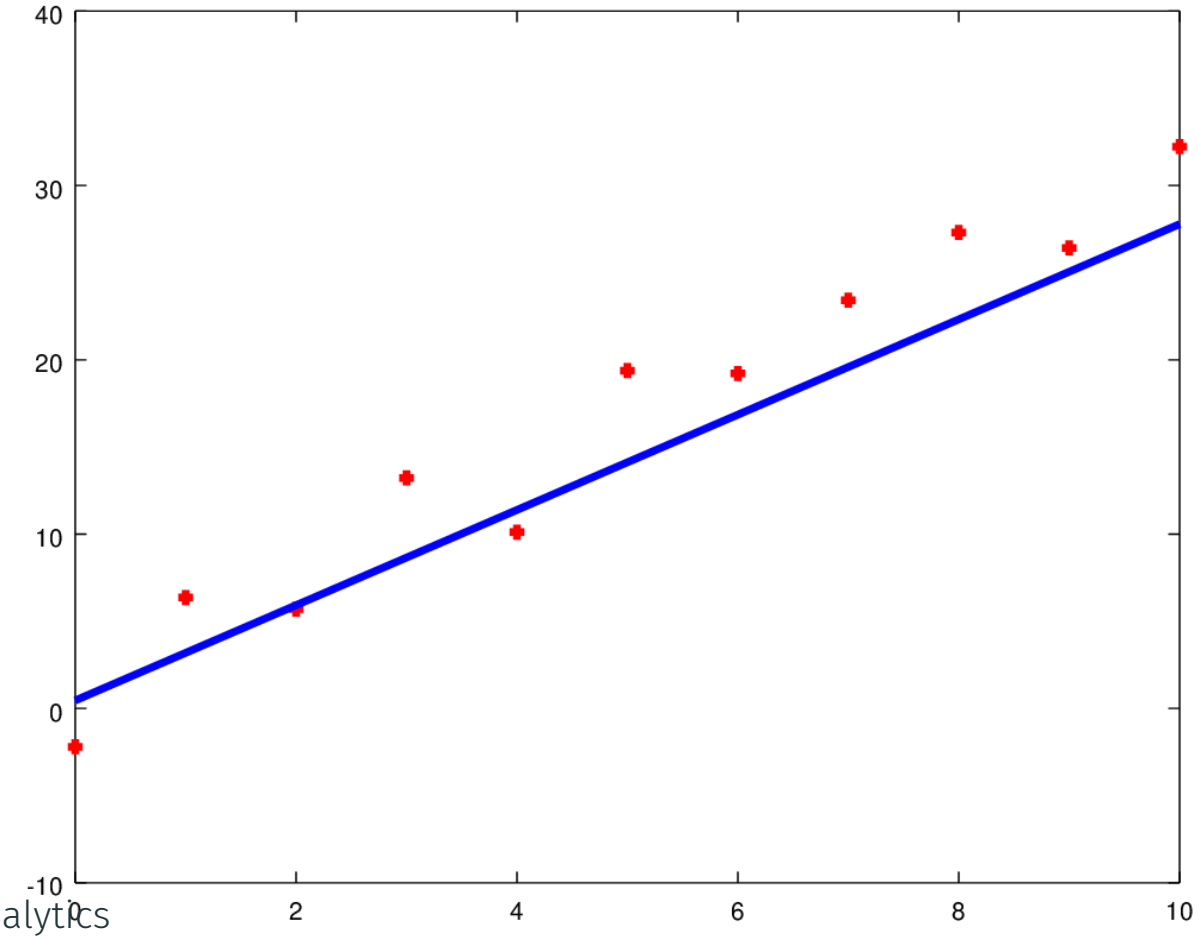
Gradient et mise à jour

- a = 2.48 ($\hat{a} = 3.0$)
- b = 0.43 ($\hat{b} = 0.5$)



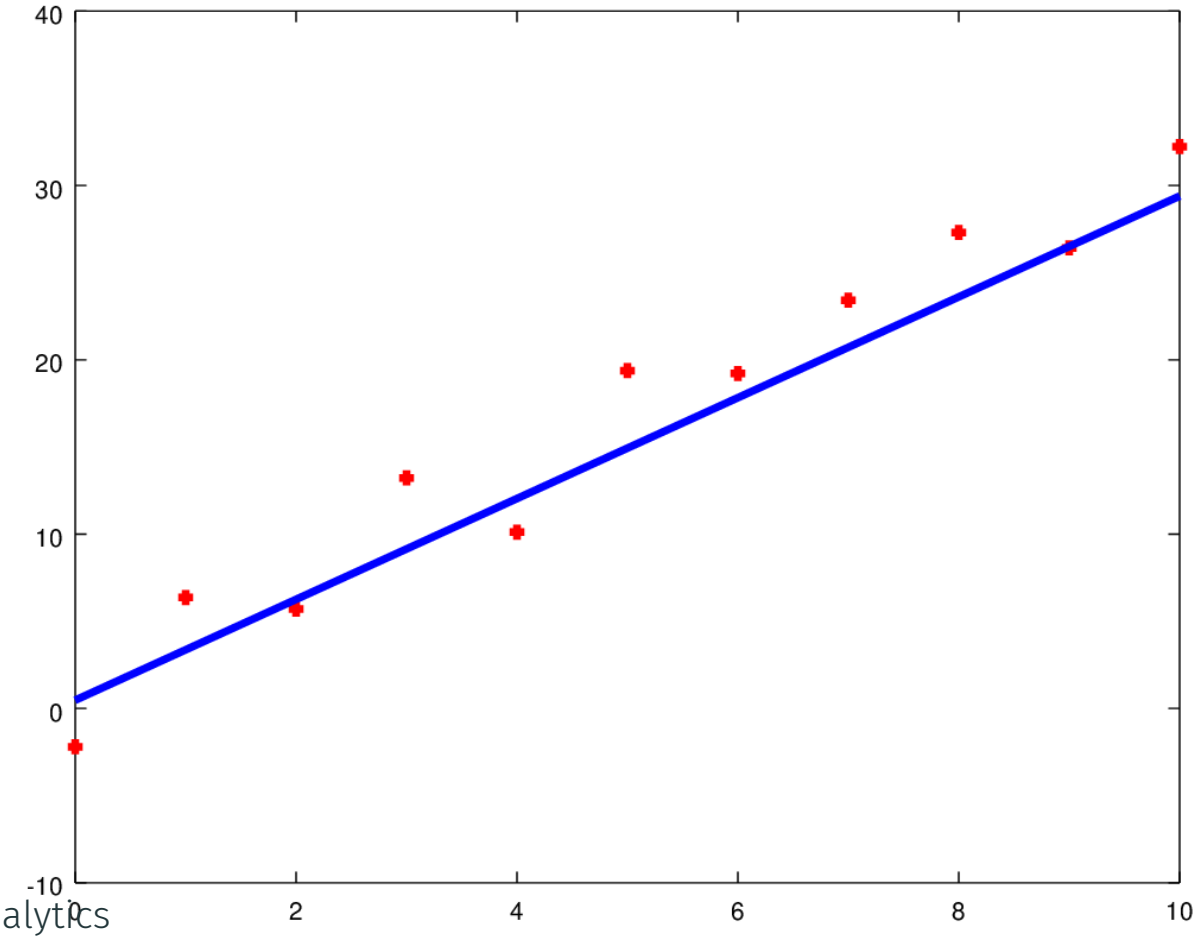
Gradient et mise à jour

- a = 2.73 ($\hat{a} = 3.0$)
- b = 0.46 ($\hat{b} = 0.5$)



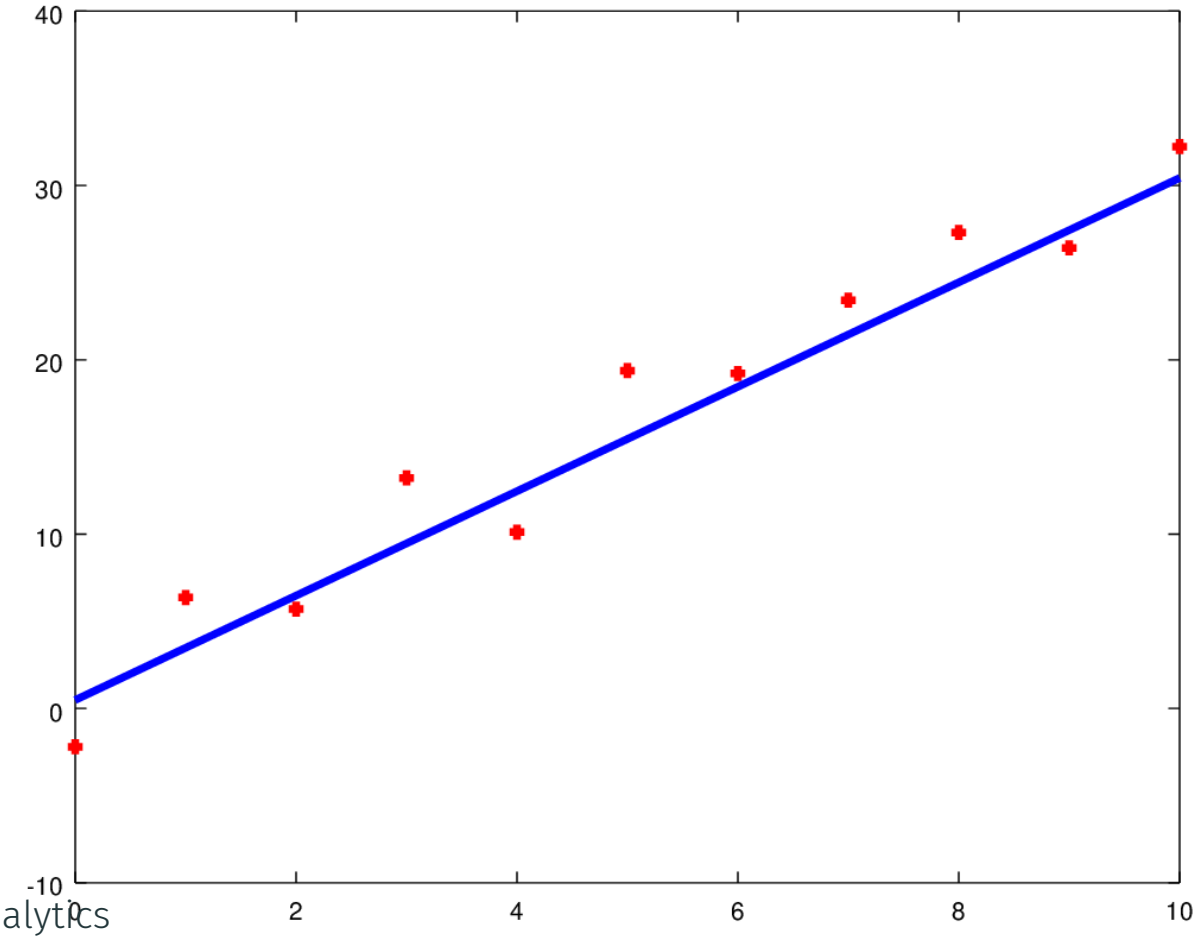
Gradient et mise à jour

- a = 2.89 ($\hat{a} = 3.0$)
- b = 0.47 ($\hat{b} = 0.5$)



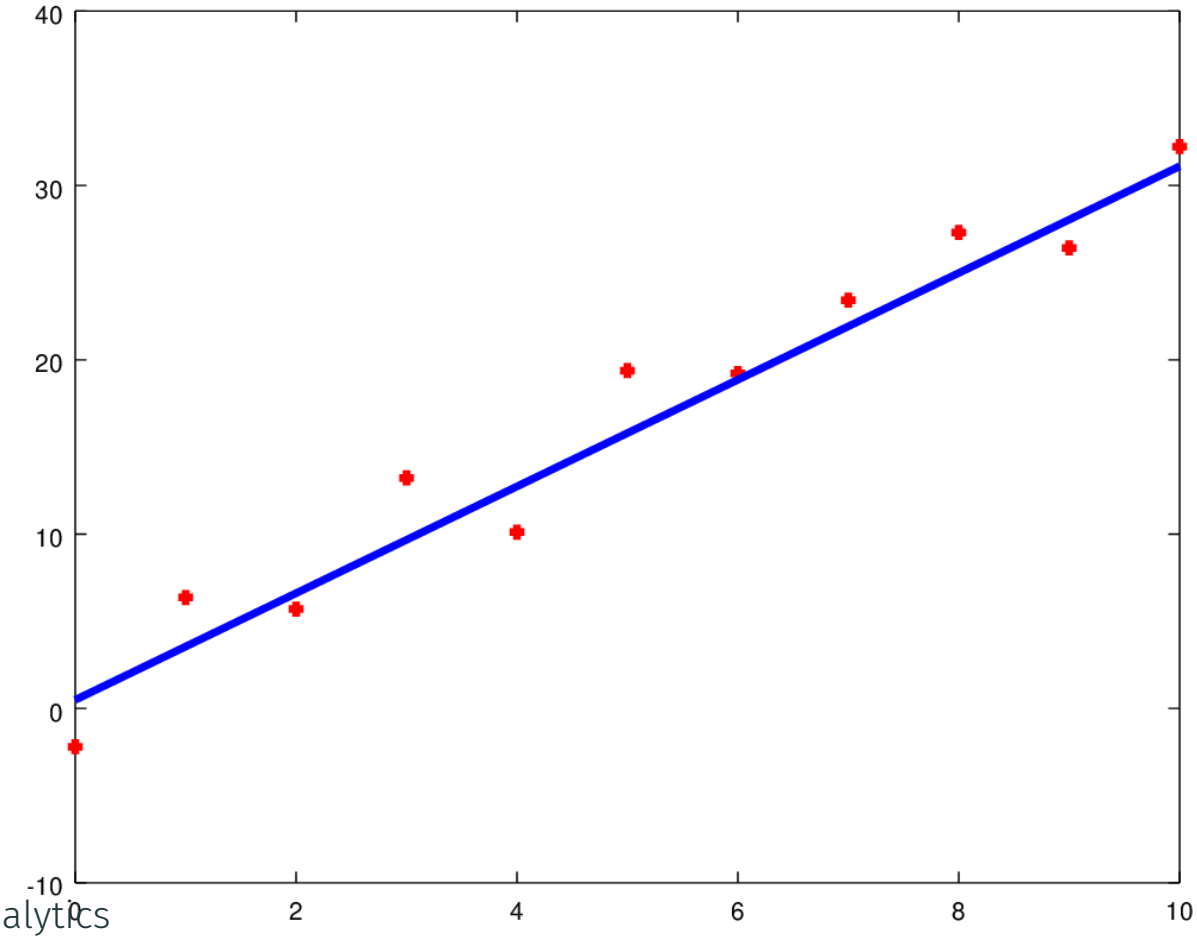
Gradient et mise à jour

- a = 2.99 ($\hat{a} = 3.0$)
- b = 0.48 ($\hat{b} = 0.5$)



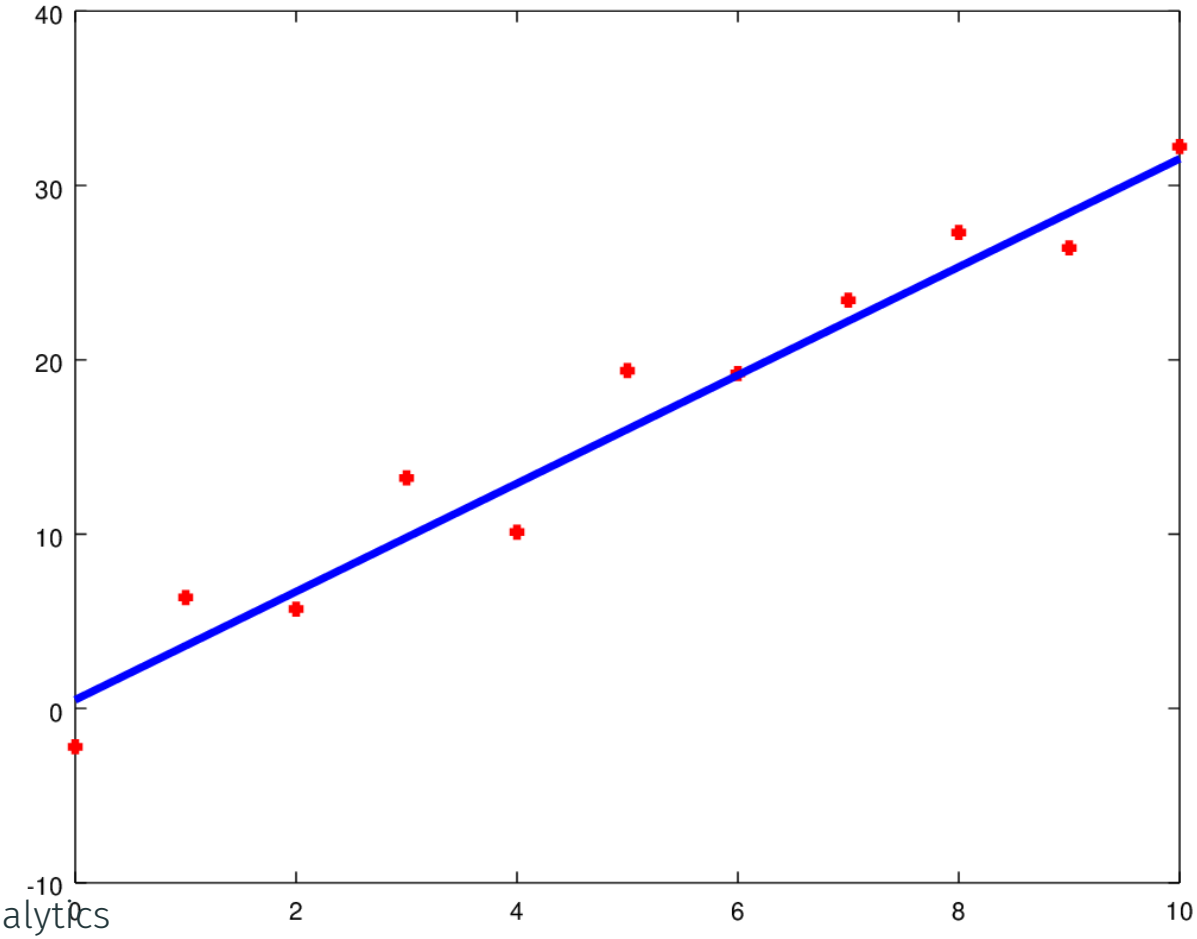
Gradient et mise à jour

- a = 3.06 ($\hat{a} = 3.0$)
- b = 0.49 ($\hat{b} = 0.5$)



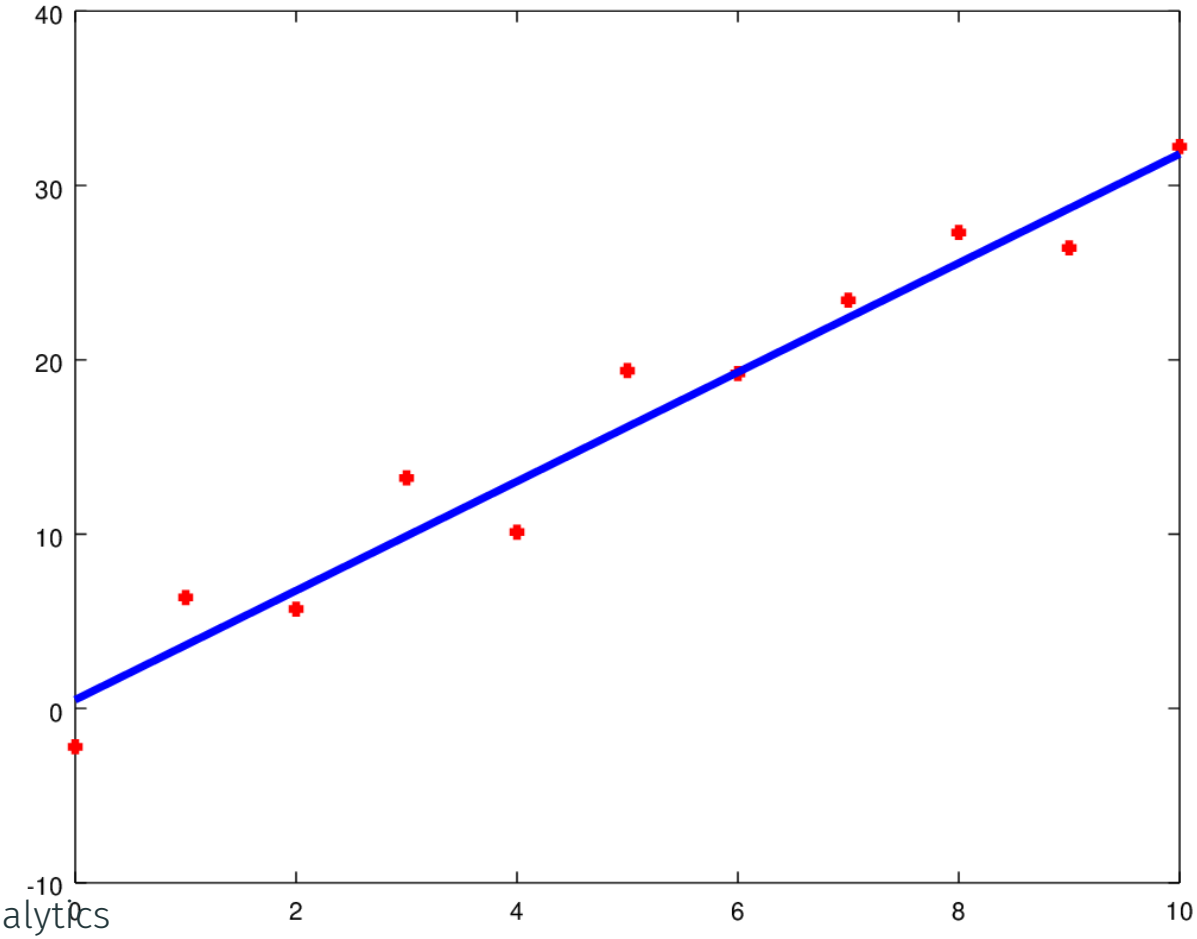
Gradient et mise à jour

- a = 3.10 ($\hat{a} = 3.0$)
- b = 0.49 ($\hat{b} = 0.5$)



Gradient et mise à jour

- a = 3.13 ($\hat{a} = 3.0$)
- b = 0.50 ($\hat{b} = 0.5$)

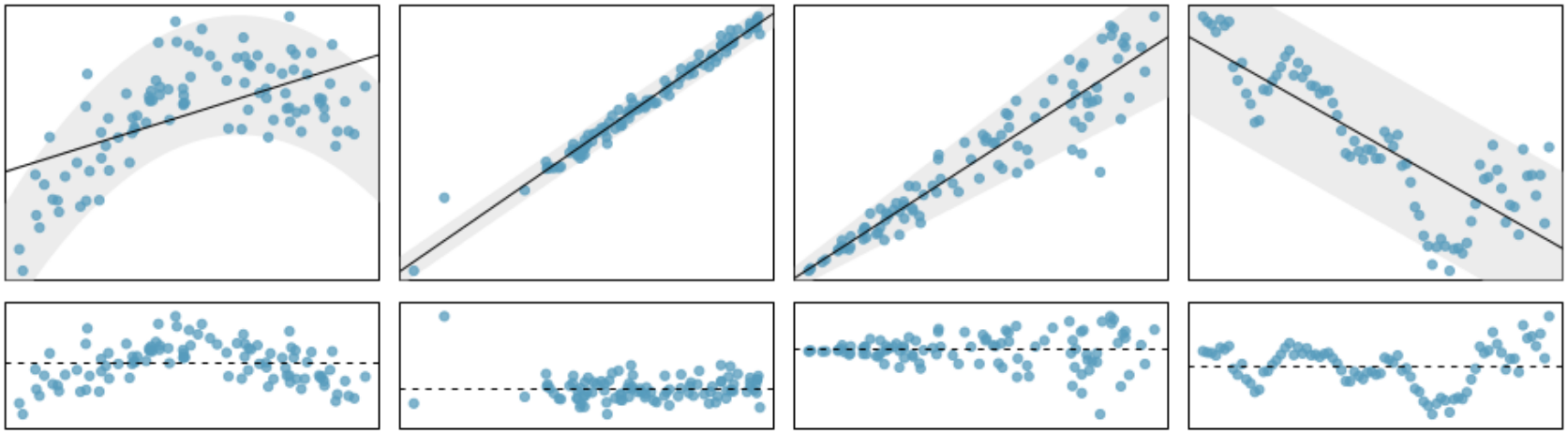


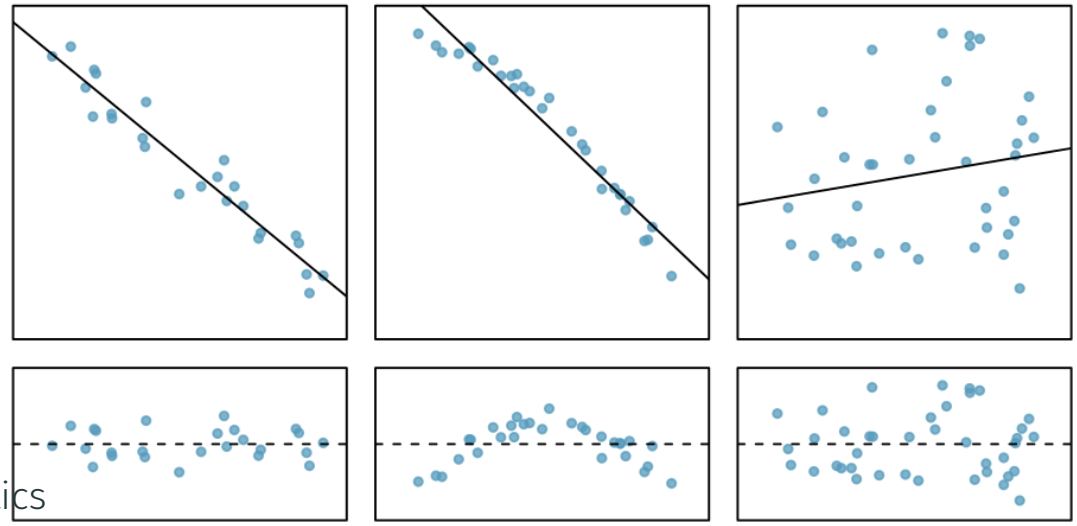
Apprentissage supervisé

Régression Polynomiale

Regression Polynomiale

Limites de la regression linéaire

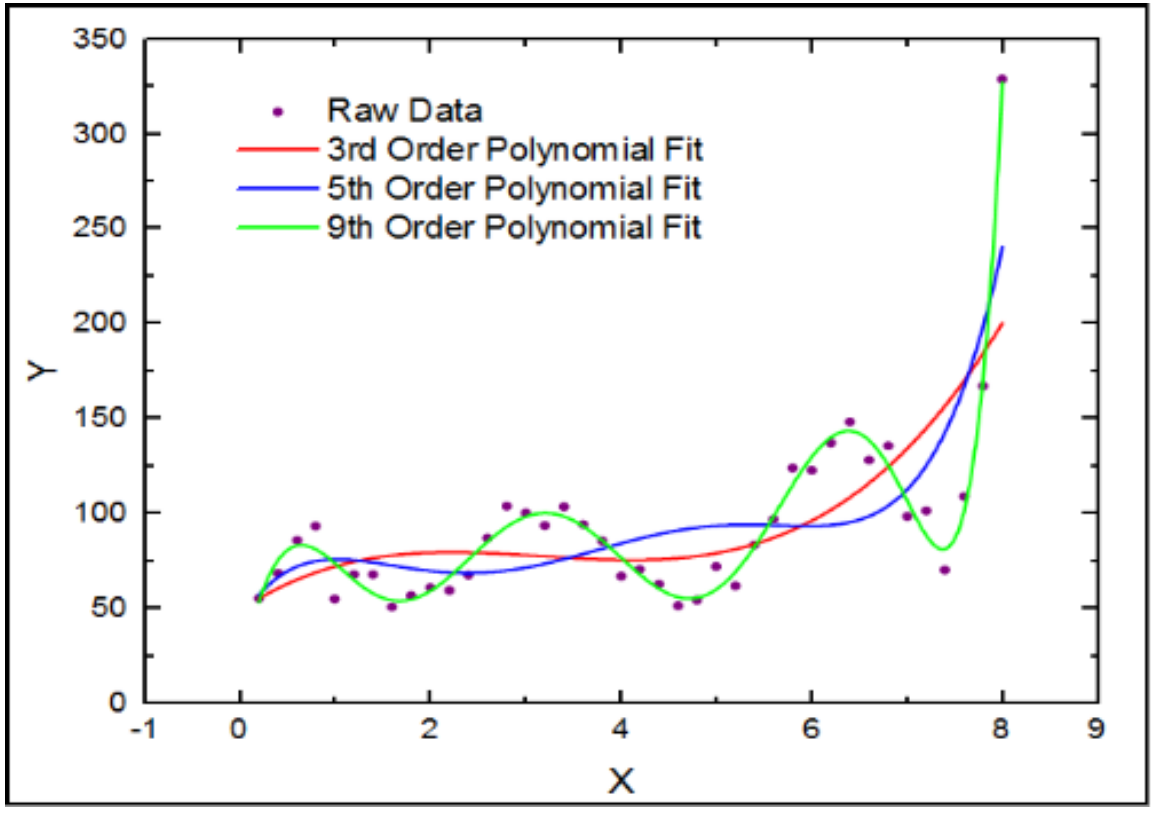




Régression Polynomiale

$$Y = a_0 + a_1 * X + a_2 * X^2 + a_3 * X^3 + ... + a_n * X^n$$

$$Y = \sum_{k \in [0..n]} a_k * X^k$$



Gradient et mise à jour

$$Y = \sum_{k \in [0..n]} a_k * X^k$$

$$E_\Omega = \frac{1}{2n} \sum_{i=[1..n]} (\hat{Y}_i - Y_i)^2$$

$$\frac{\partial E_\Omega}{\partial a_k} = \frac{1}{n} \sum_{i=[1..n]} (Y_i - \hat{Y}_i).X^k$$

M.A.J :

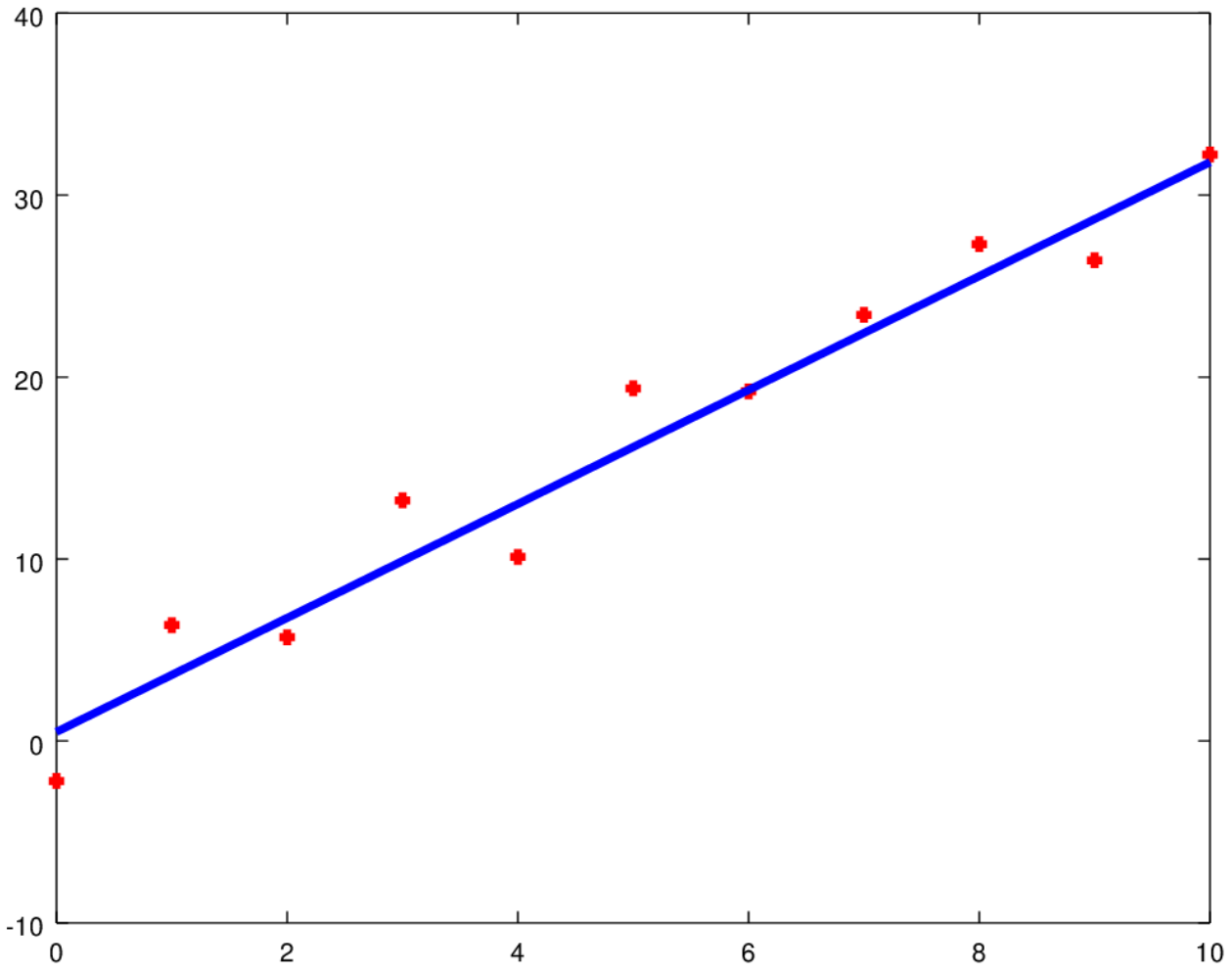
$$a_k \leftarrow a_k - \gamma . \frac{\partial E_\Omega}{\partial a_k}$$

où $1 > \gamma > 0$ (learning rate)

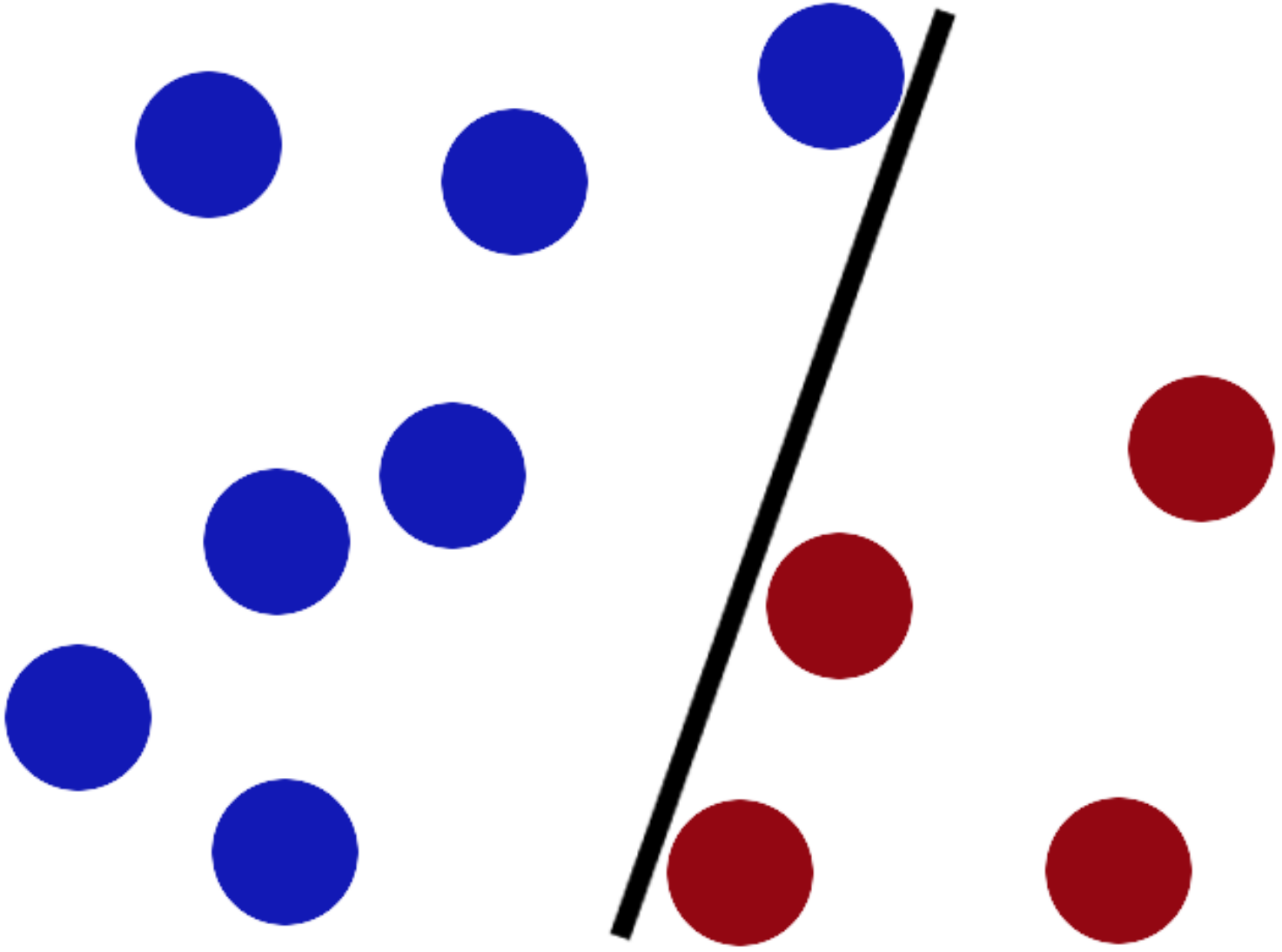
Apprentissage supervisé

Réseau de Neurones

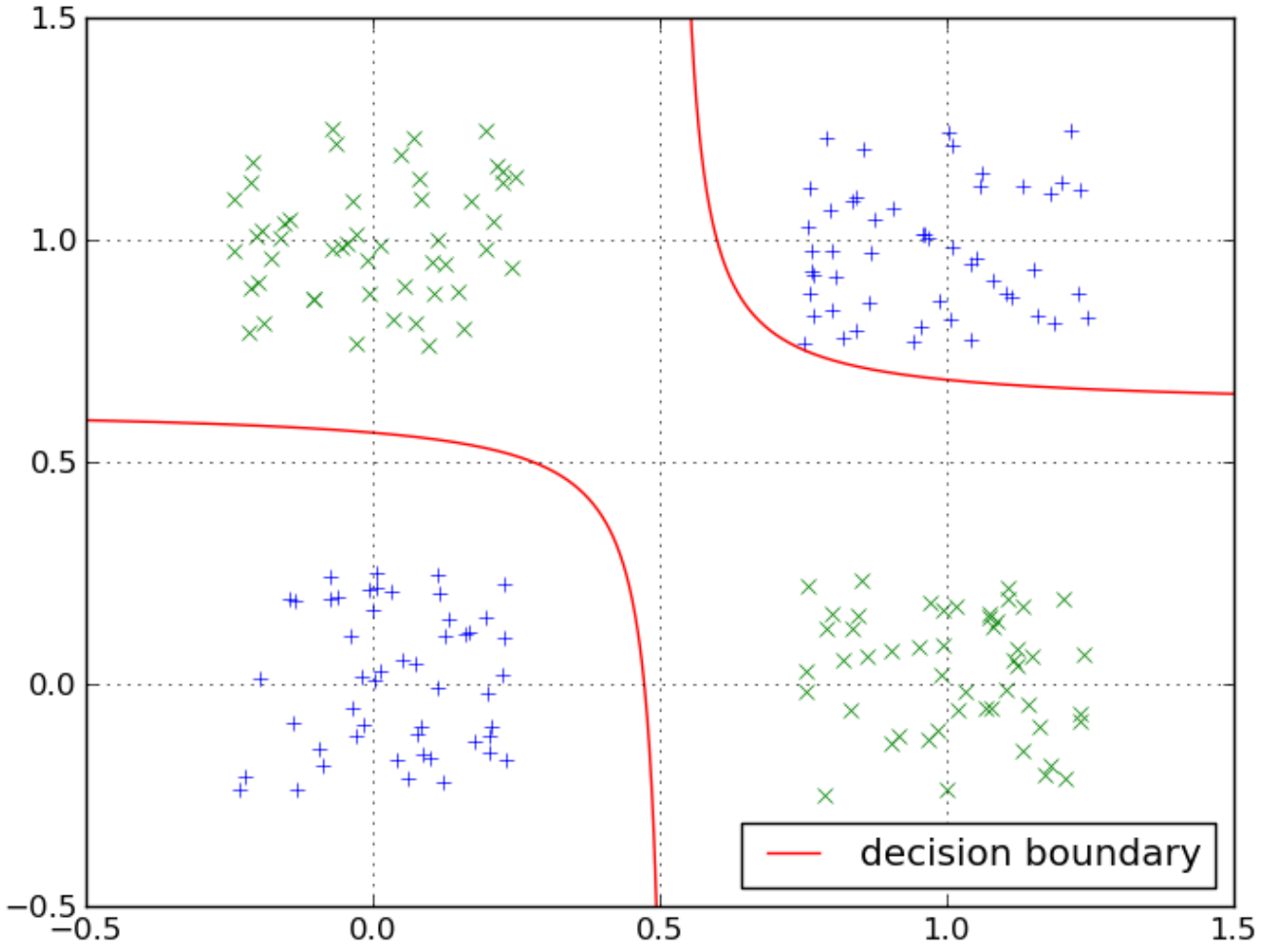
Lien avec la régression linéaire



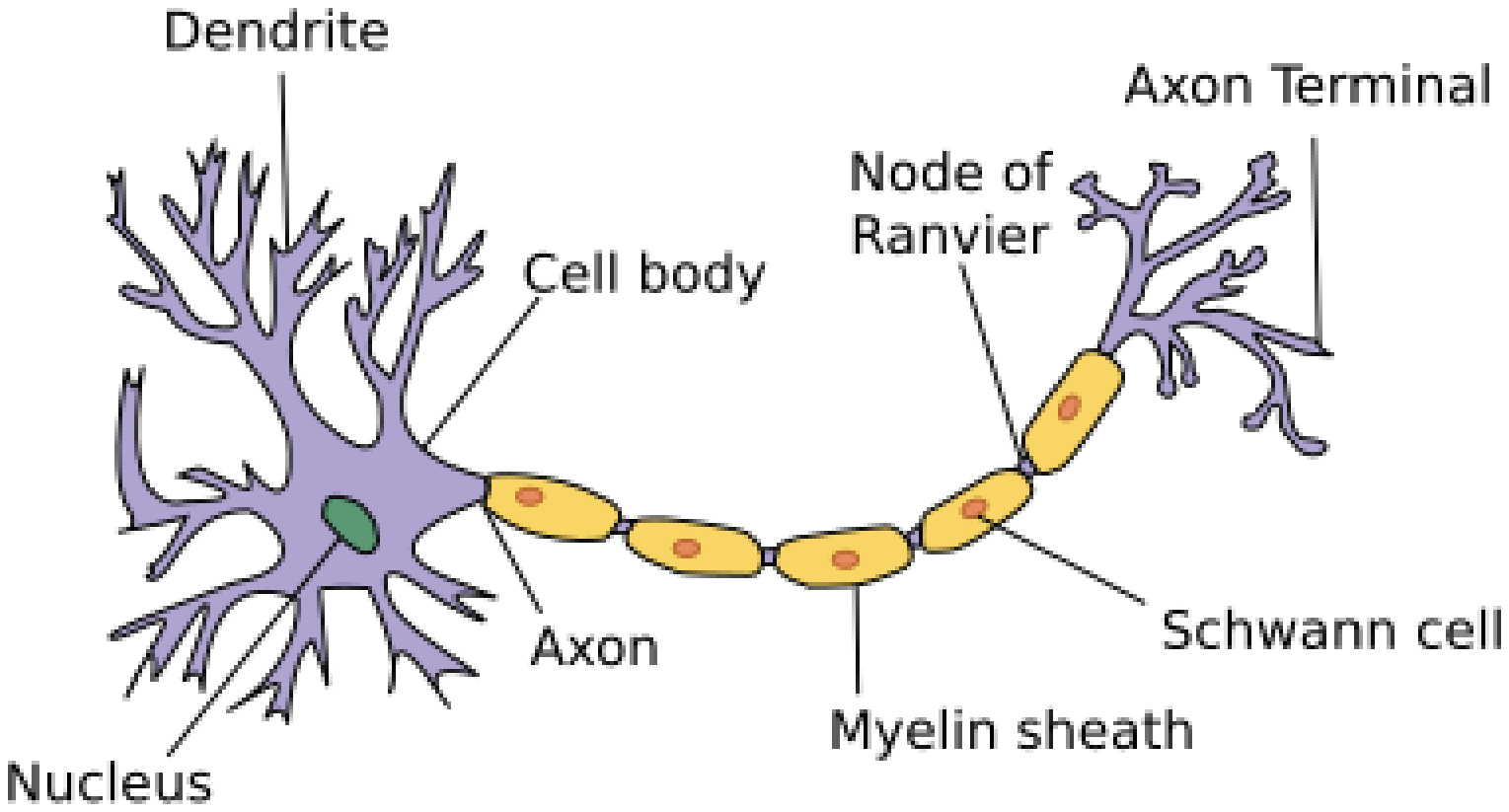
Lien avec la régression linéaire



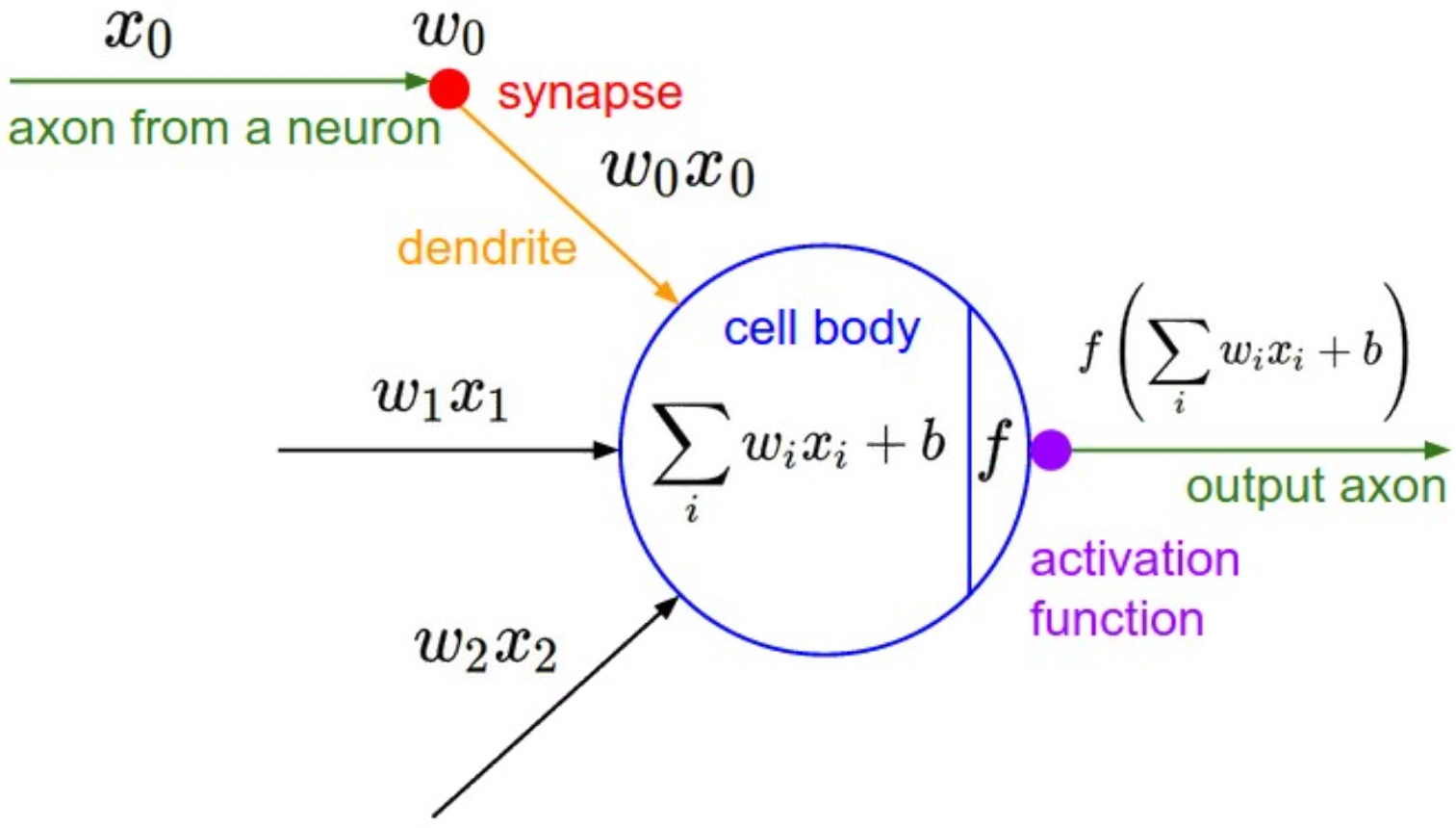
Lien avec la régression linéaire



Réseau de Neurones



Réseau de Neurones



Réseau de Neurones

$$\sigma\left(\begin{bmatrix} x_1 & x_2 & \dots & x_d \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{d1} & w_{d2} & \dots & w_{dn} \end{bmatrix} + \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}\right) = \begin{bmatrix} o_1 & o_2 & \dots & o_n \end{bmatrix}$$

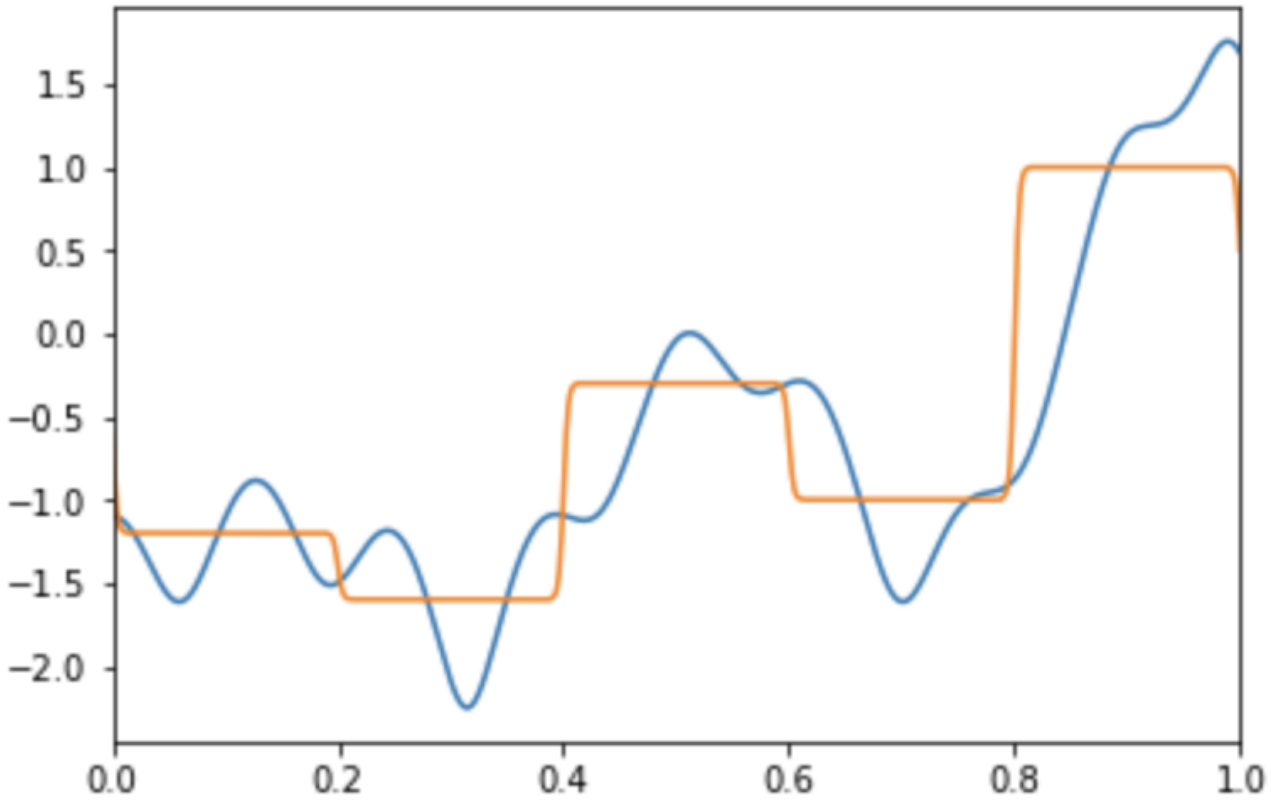
où :
X est une donnée en entrée de dimension **d**,
w et *b* sont les paramètres à trouver des **n** neurones de notre modèle.
σ la fonction d'activation et
O la sortie du réseau

Réseaux de neurones : Fonction d'activation σ

- Sigmoïde
- Tanh
- Softmax
- ReLU
- ...

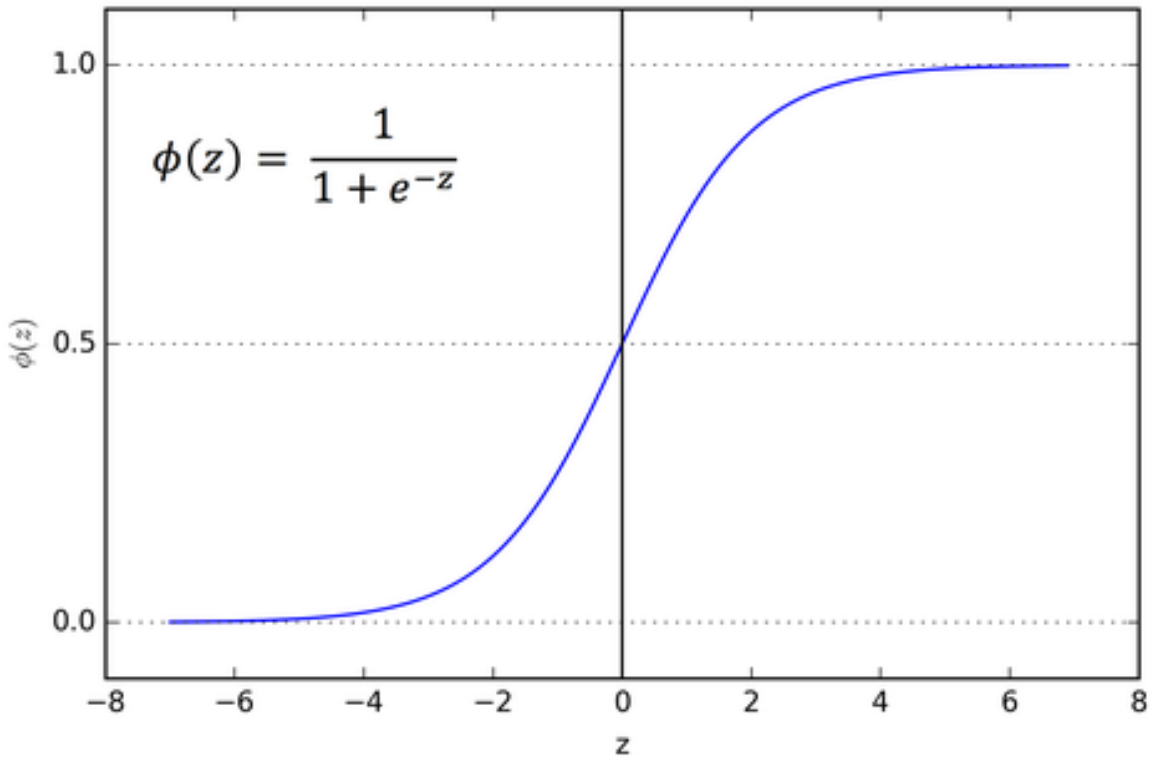
Réseaux de neurones : un Potentiel infini

1991 : **Kurt Hornik** : Théorème d'approximation universelle



Réseaux de neurones : Fonction d'activation σ

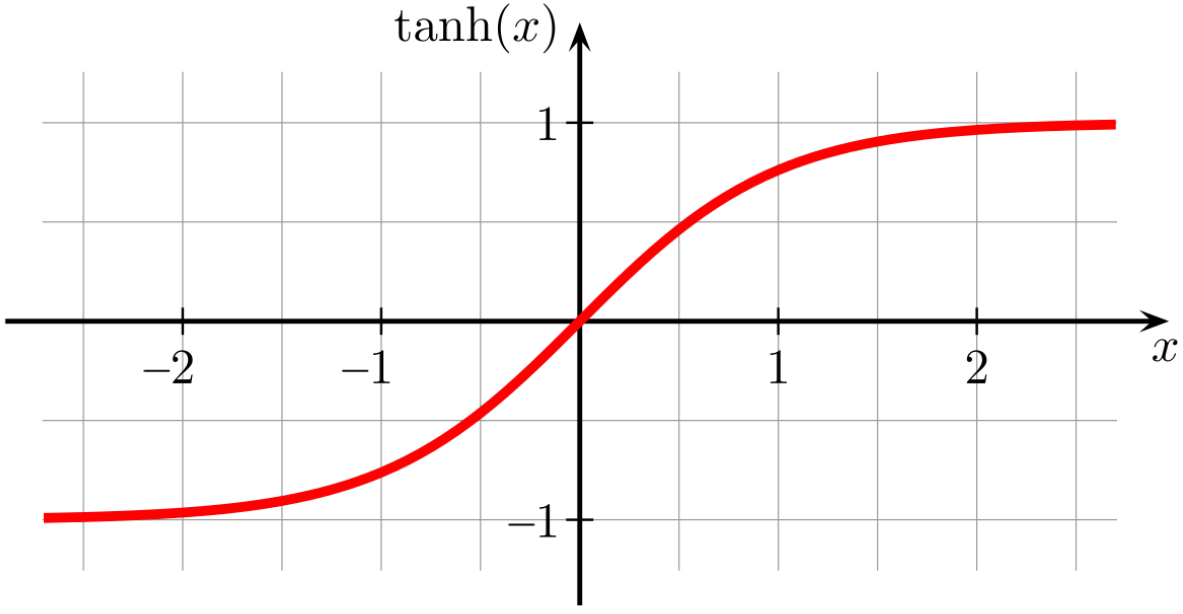
Sigmoïde



$$\frac{\partial \phi(x)}{\partial x} = \phi(x) * (1 - \phi(x))$$

Réseaux de neurones : Fonction d'activation σ

$$\tanh(x) = \frac{1 - \exp -2 * x}{1 + \exp -2 * x}$$



$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh^2(x)$$

Réseaux de neurones : Fonction d'activation σ

$$Softmax(x_j) = \frac{\exp x_j}{\sum_{i=1}^{n} \exp x_i}$$

donc :

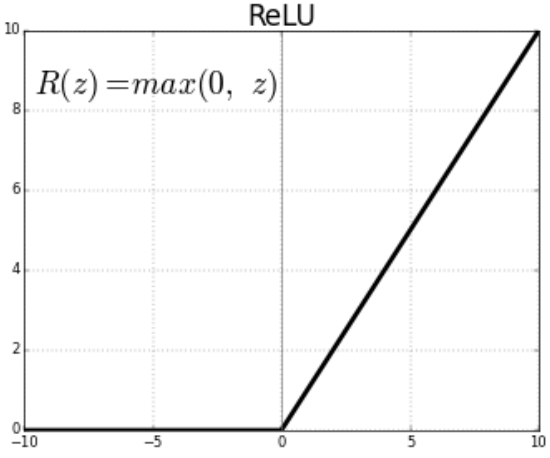
$$\sum_{j=1}^{n} Softmax(x_j) = 1$$

dérivée (ou jacobien car le softmax est une fonction de $\mathbb{R}^n \to \mathbb{R}^n$) :

$$D_j S_i = S_i(\delta_{ij} - S_j)$$

où $D_j S_i$ est la dérivée partielle de la i-ième sortie par rapport à la j-ième entrée
δ_{ij} est le delta de Kronecker

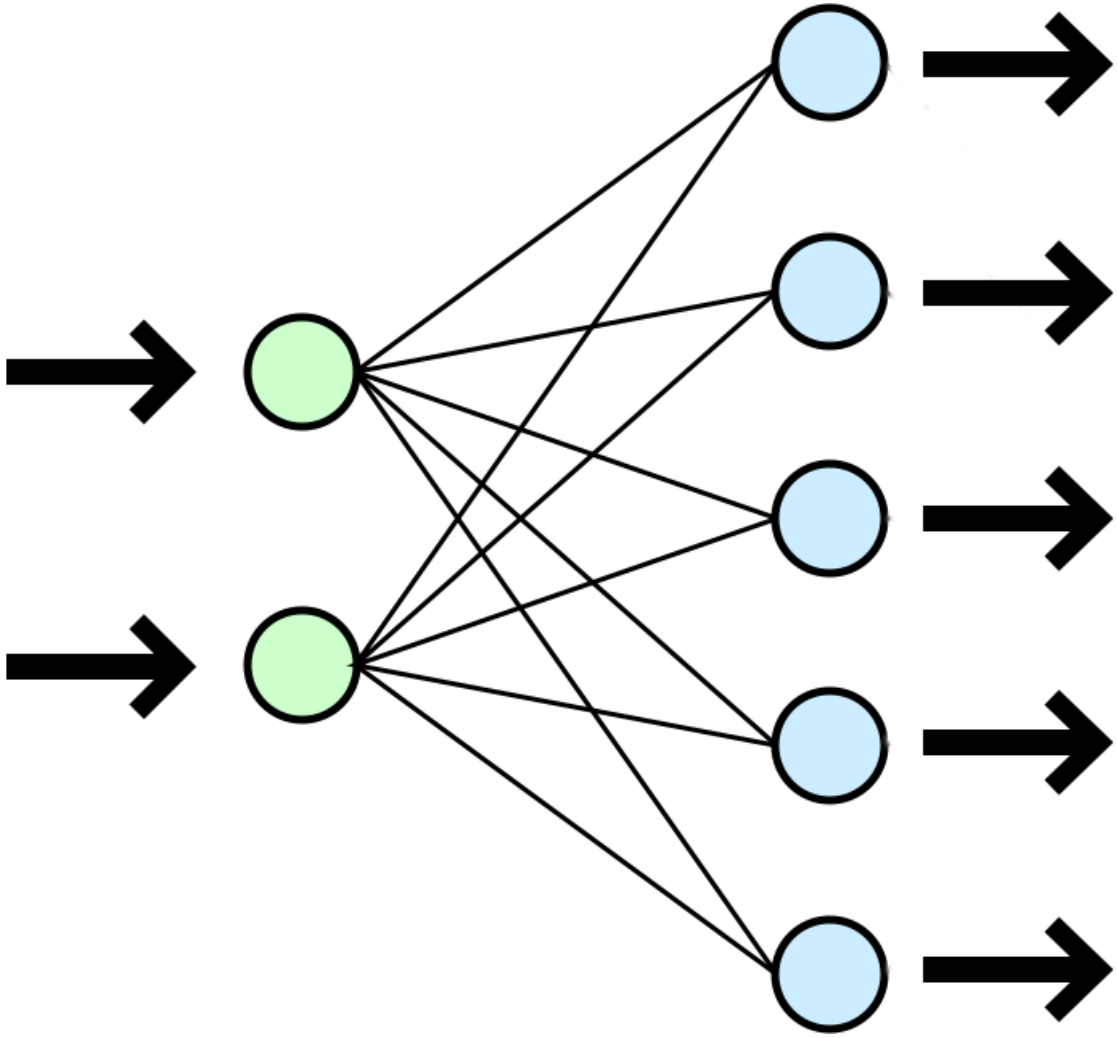
Réseaux de neurones : Fonction d'activation σ



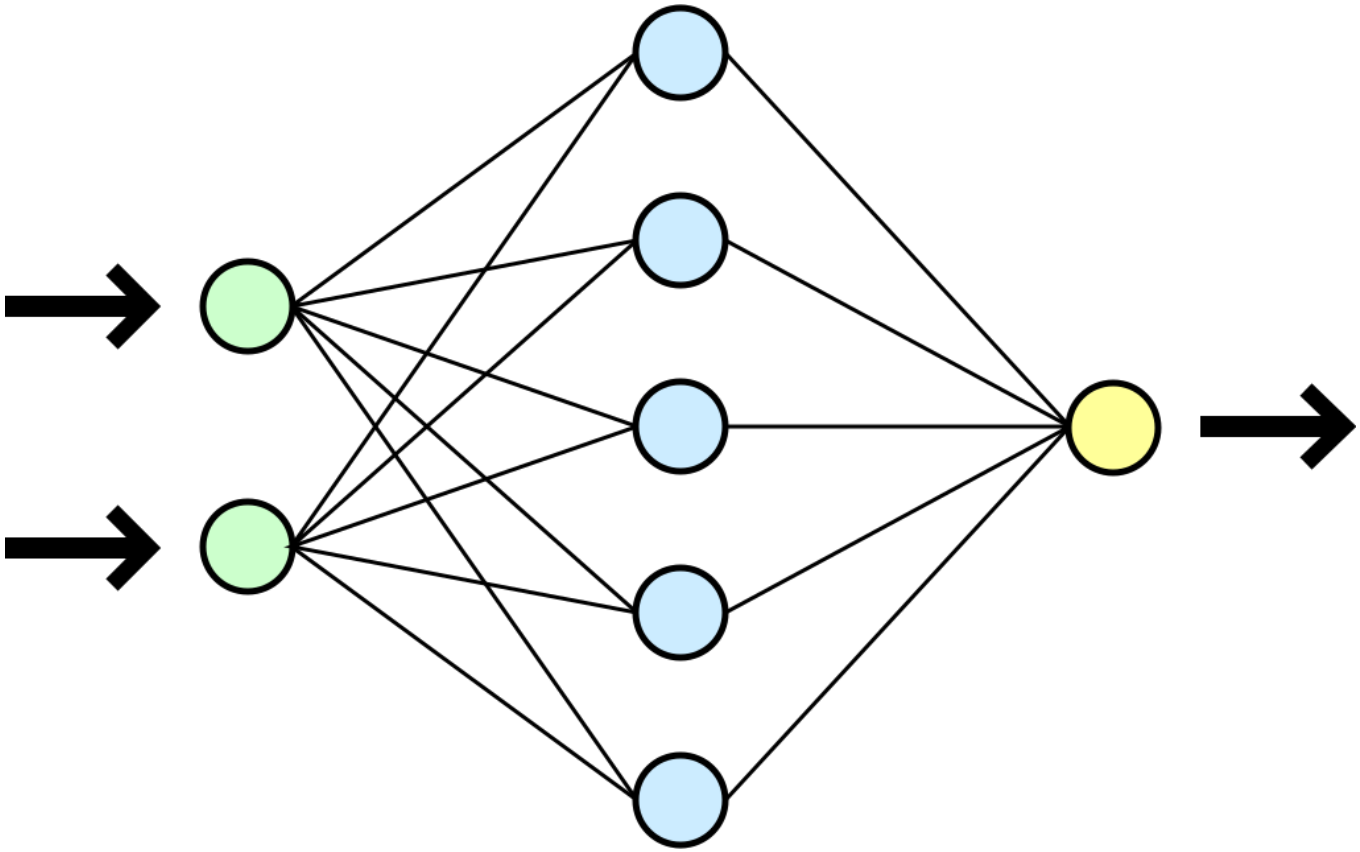
$$n_1(x) = Relu(-5x - 7.7)$$
$$n_2(x) = Relu(-1.2x - 1.3)$$
$$n_3(x) = Relu(1.2x + 1)$$
$$n_4(x) = Relu(1.2x - .2)$$
$$n_5(x) = Relu(2x - 1.1)$$
$$n_6(x) = Relu(5x - 5)$$

$$Z(x) = -n_1(x) - n_2(x) - n_3(x) + n_4(x) + n_5(x) + n_6(x)$$

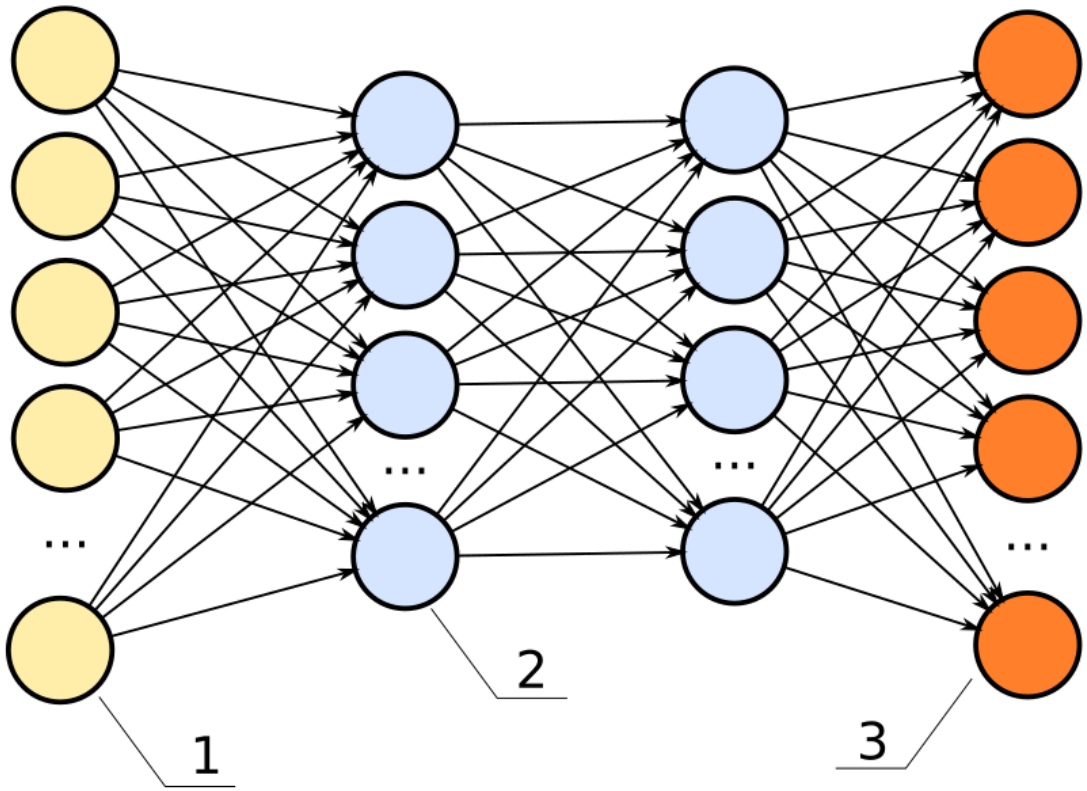
Réseau de Neurones



Réseau de Neurones



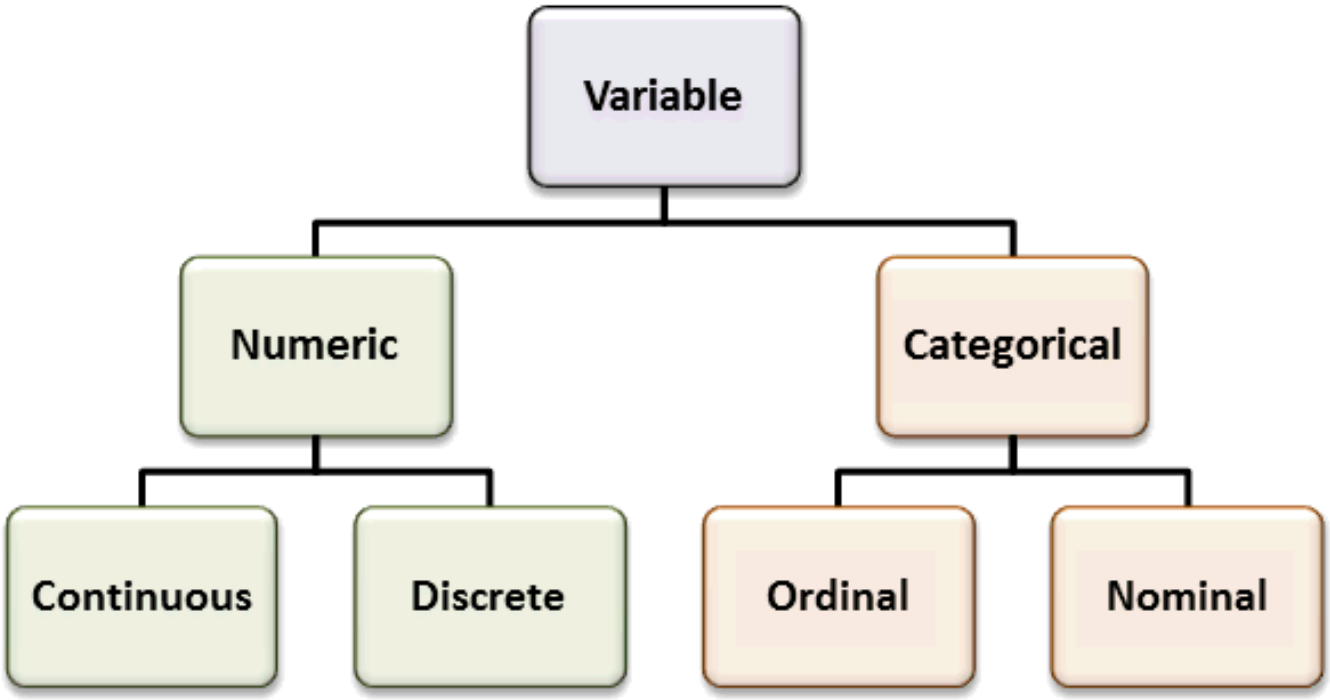
Réseau de Neurones



Apprentissage supervisé

Classification

Classification

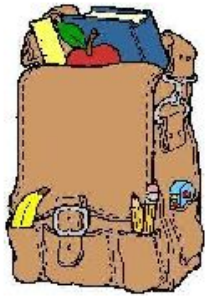


Classification

Classification


001.ak47


002.american-flag


003.backpack


004.baseball-bat


005.baseball-glove


006.basketball-hoop


007.bat


008.bathtub


009.bear


010.beer-mug


011.billiards


012.binoculars

Classification

≈ Distance entre la sortie et la cible ?

Sortie :

0.0	0.1	0.4	0.0	0.0	0.2	0.1	0.0	0.2	0.0

Cible :

0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Apprentissage supervisé

Régression Logistique

Introduction

Équivalent de la régression linéaire mais quand la sortie est binaire.
L'approche est bayésienne, basée sur une étude statistique.

Le cas à 2 classes

On s'intéresse à faire une régression de l'"*évidence*" de la variable aléatoire cible :

$$\ln \frac{P(1|X)}{1-P(1|X)} = b_0 + b_1 x_1 + ... + b_d x_d$$

où

- $X = [x_1, ..., x_d]$ un exemple de la base
- $B = [b_1, ..., b_d]$ l'ensemble des paramètres de notre modèle
- $P(1|X)$ est la probabilité que X soit de classe 1

Les coéfficients B sont alors estimés par descente de gradient

Multinomial logistic Regression

Problème à K classes :

- (K-1) prédicteurs binaire en utilisant une classe 'pivot'
- Maximum de vraisemblance pour la prédiction finale

Avez-vous des questions ?

Apprentissage supervisé

Travaux Pratiques : Régression Linéaire

Apprentissage supervisé

Travaux Pratiques — Apprentissage Surpervisé

Régression Linéaire

Regression Linéaire - TP

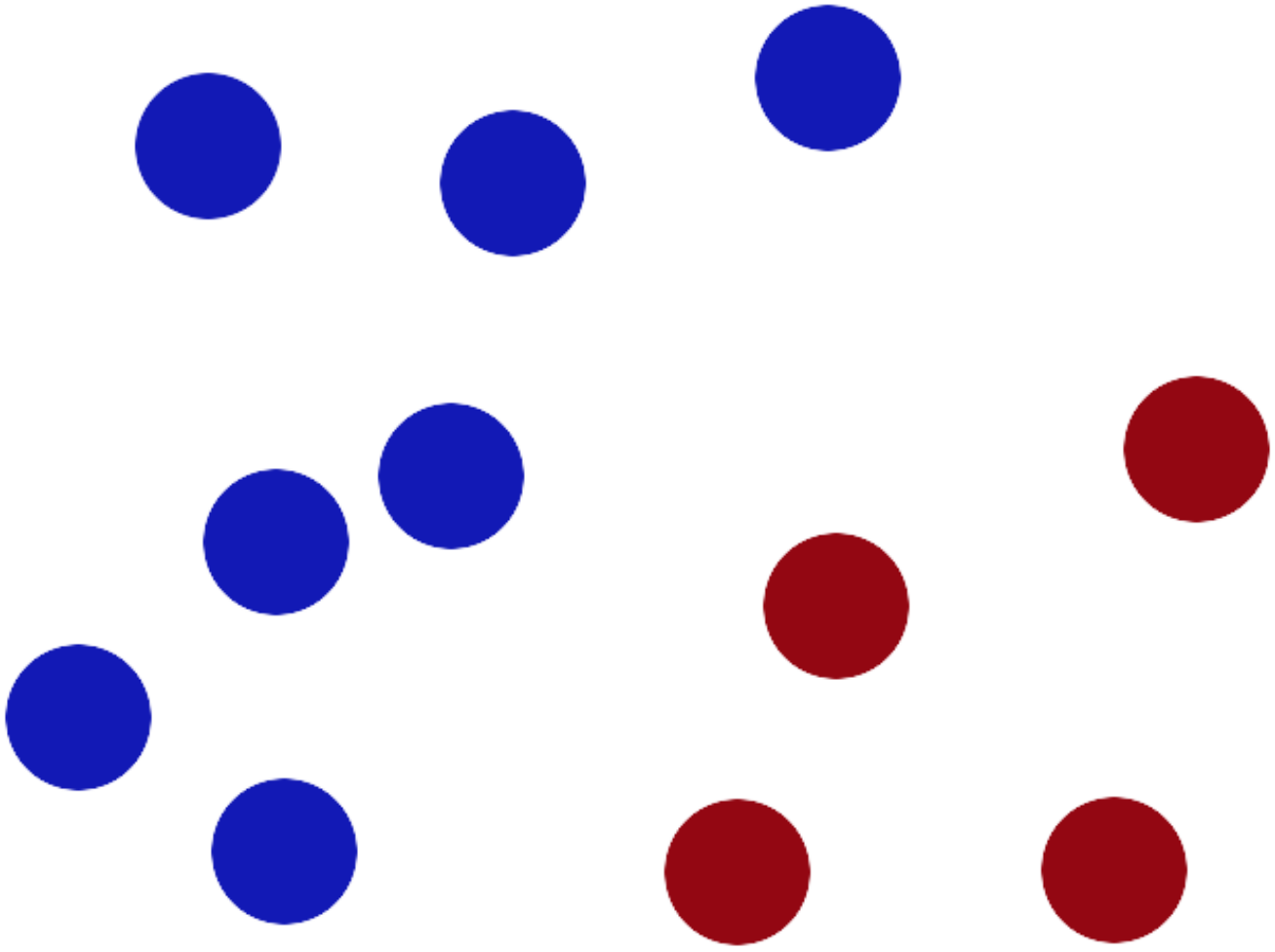
Régression Polynomiale

Régression Polynomiale - Tutoriel

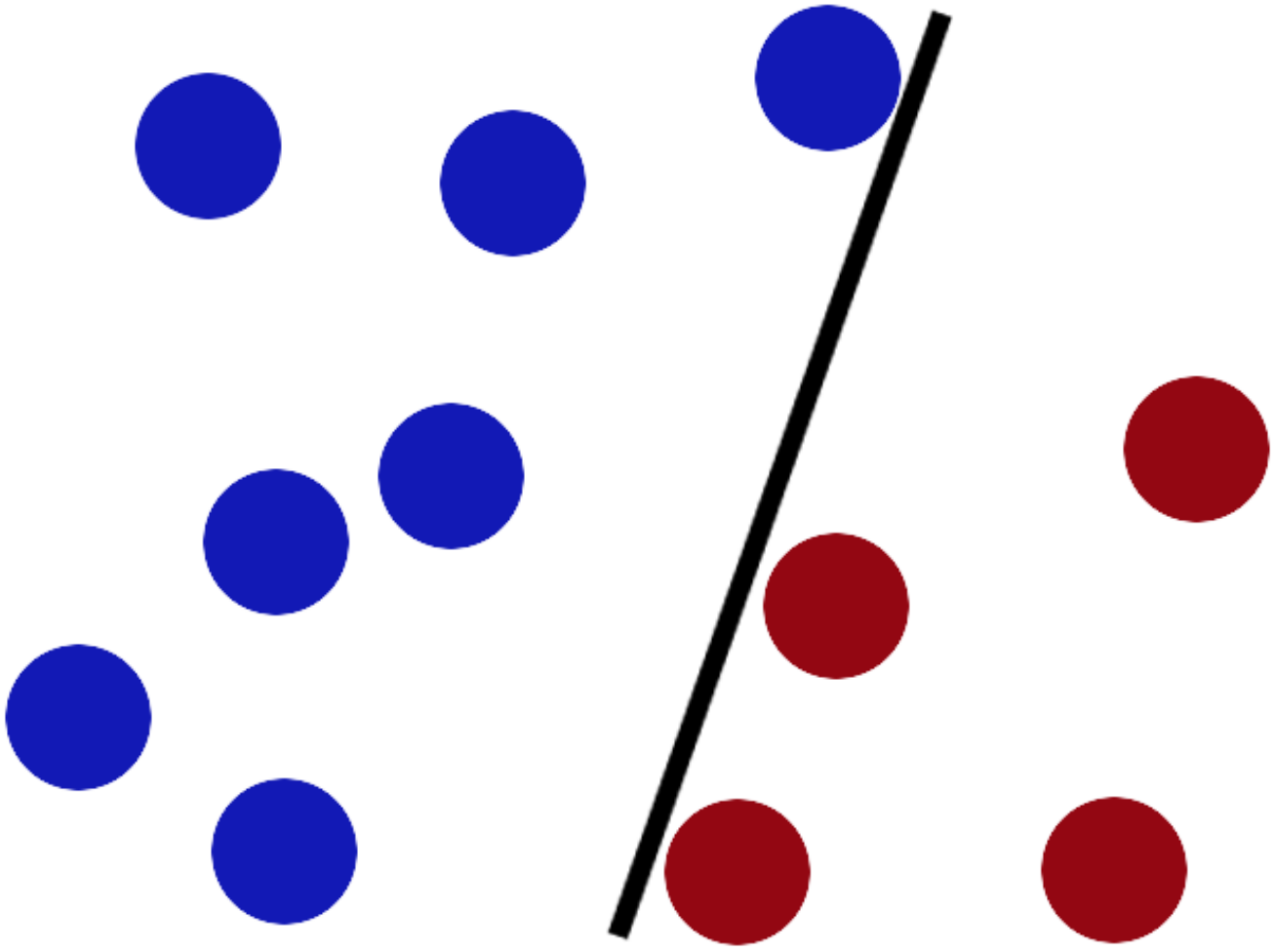
Apprentissage supervisé

Support Vector Machine

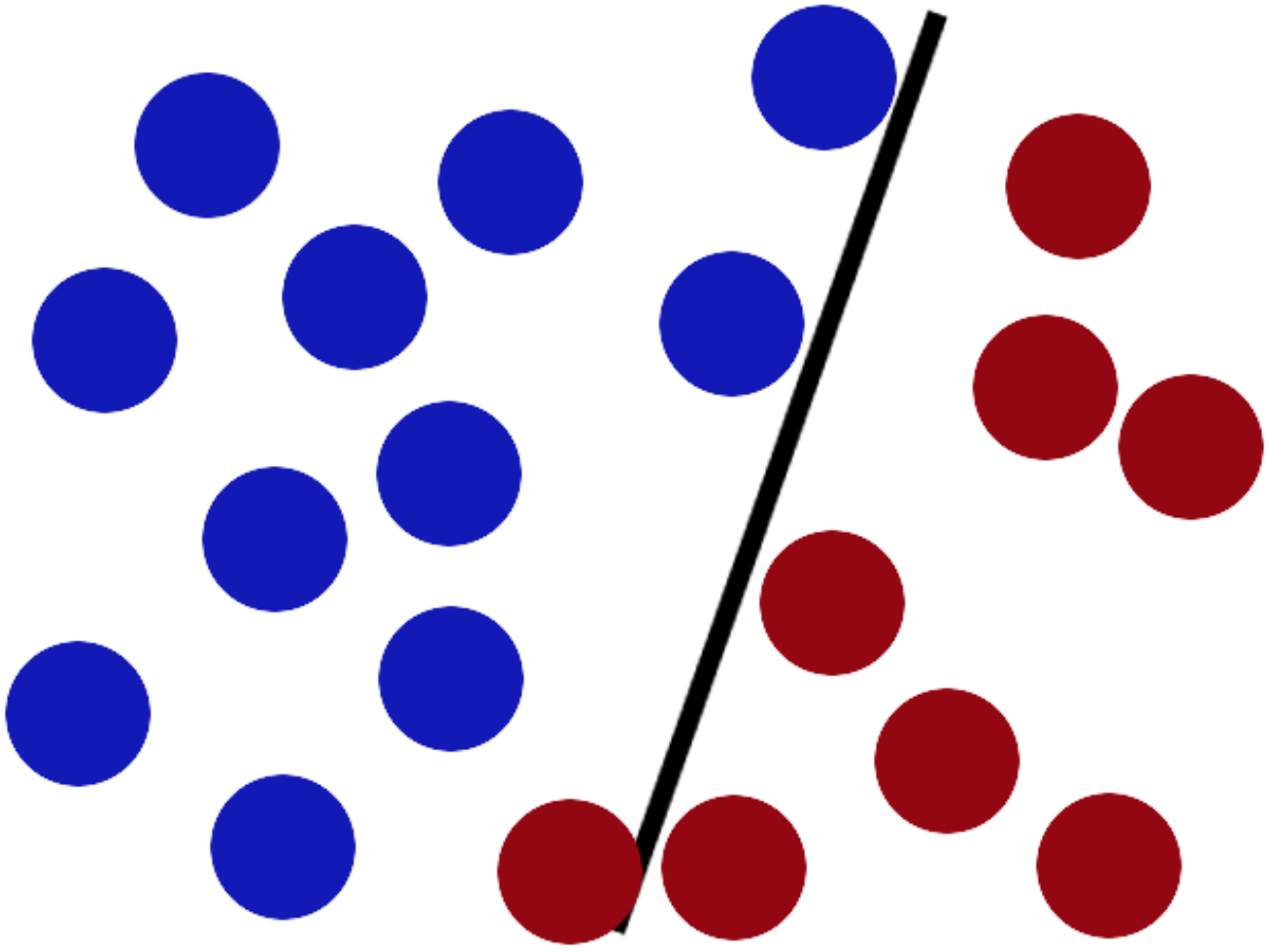
Support Vector Machine



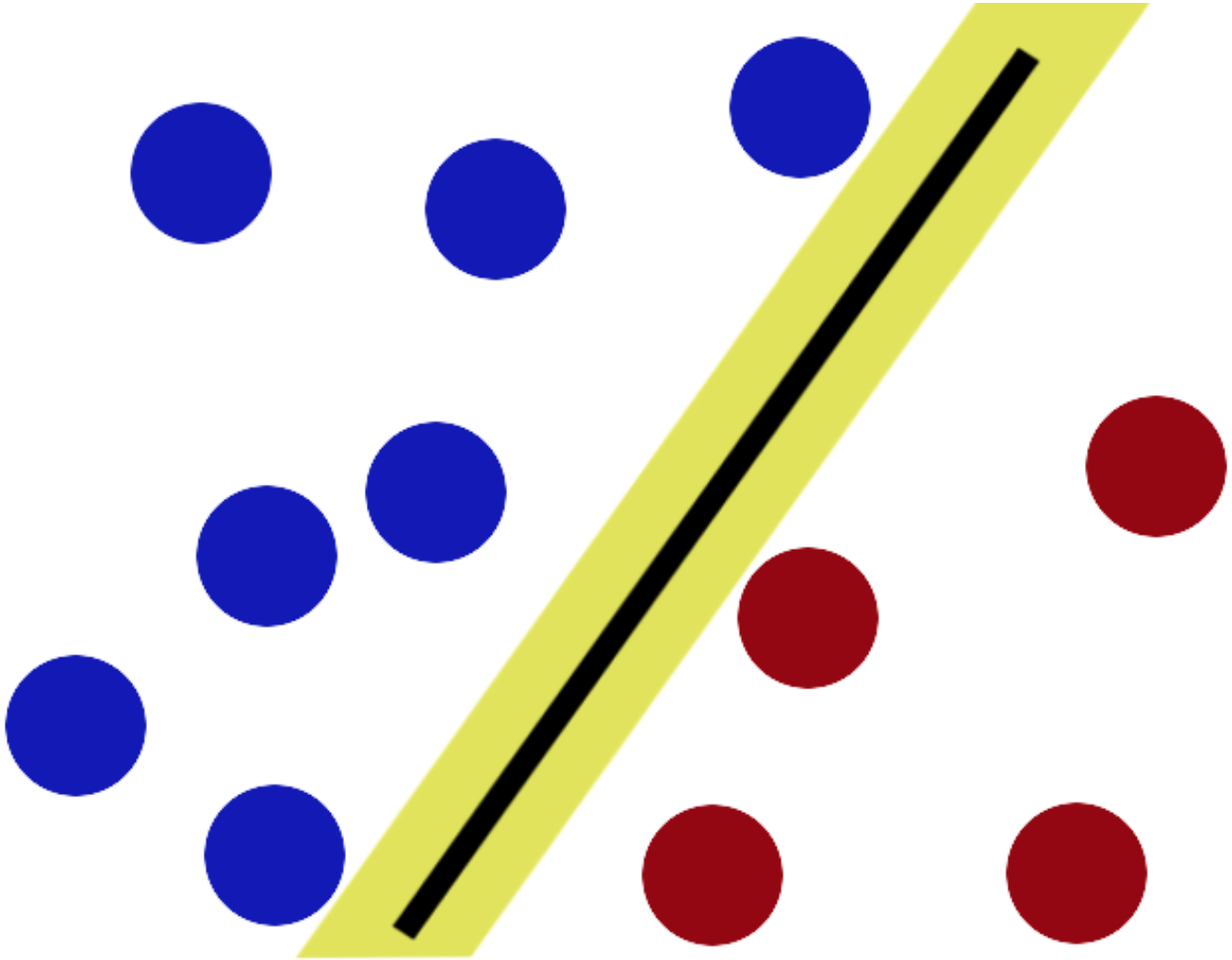
Support Vector Machine



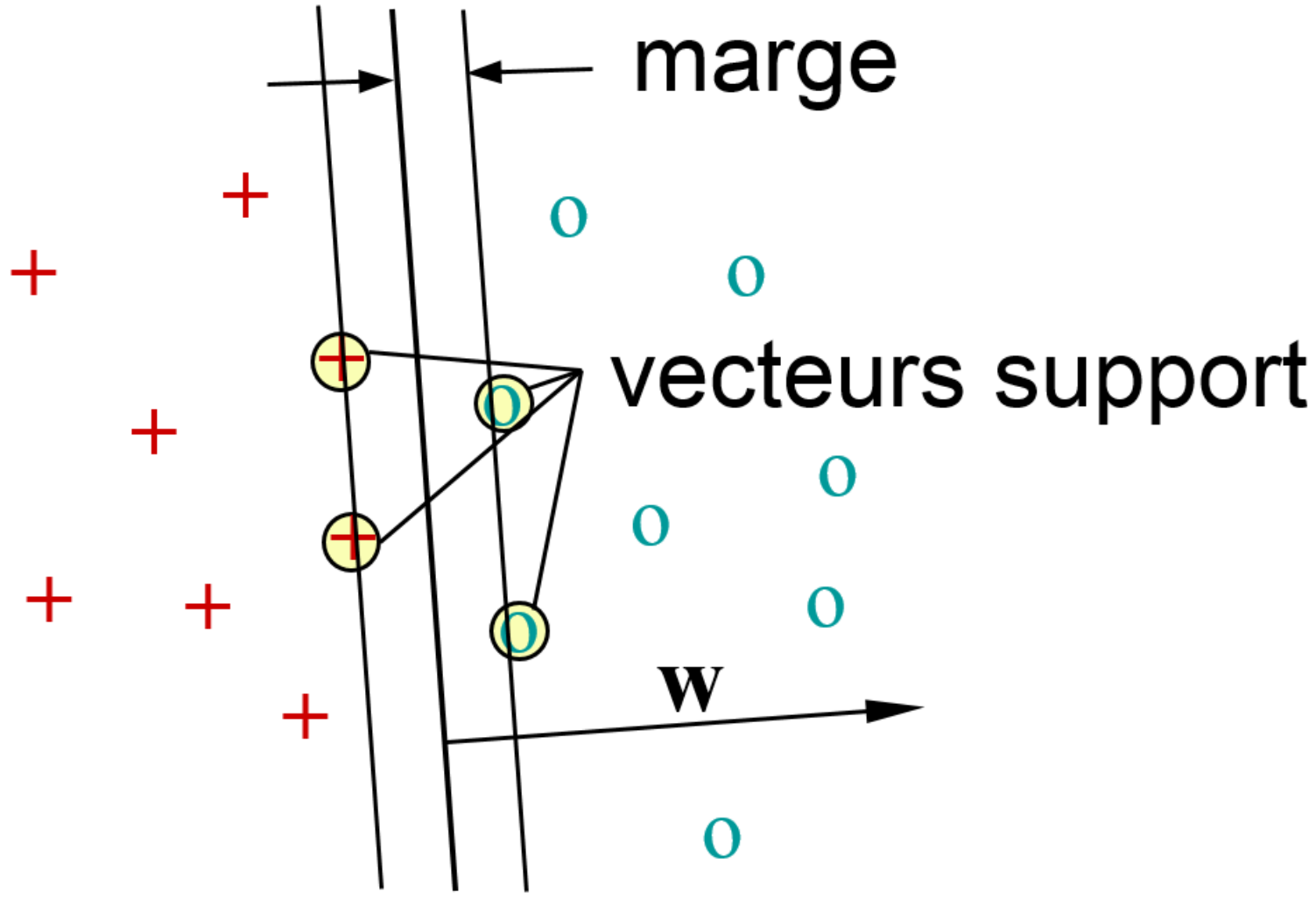
Support Vector Machine



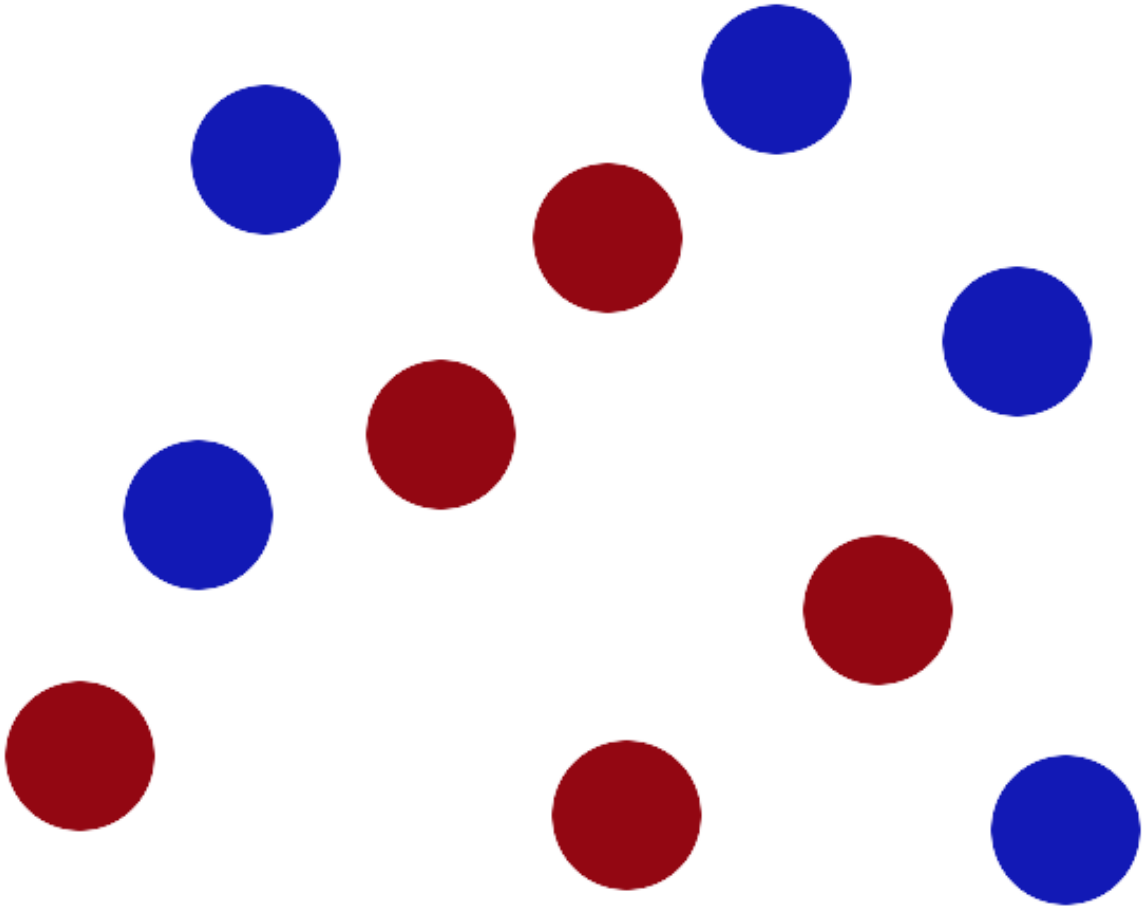
Support Vector Machine



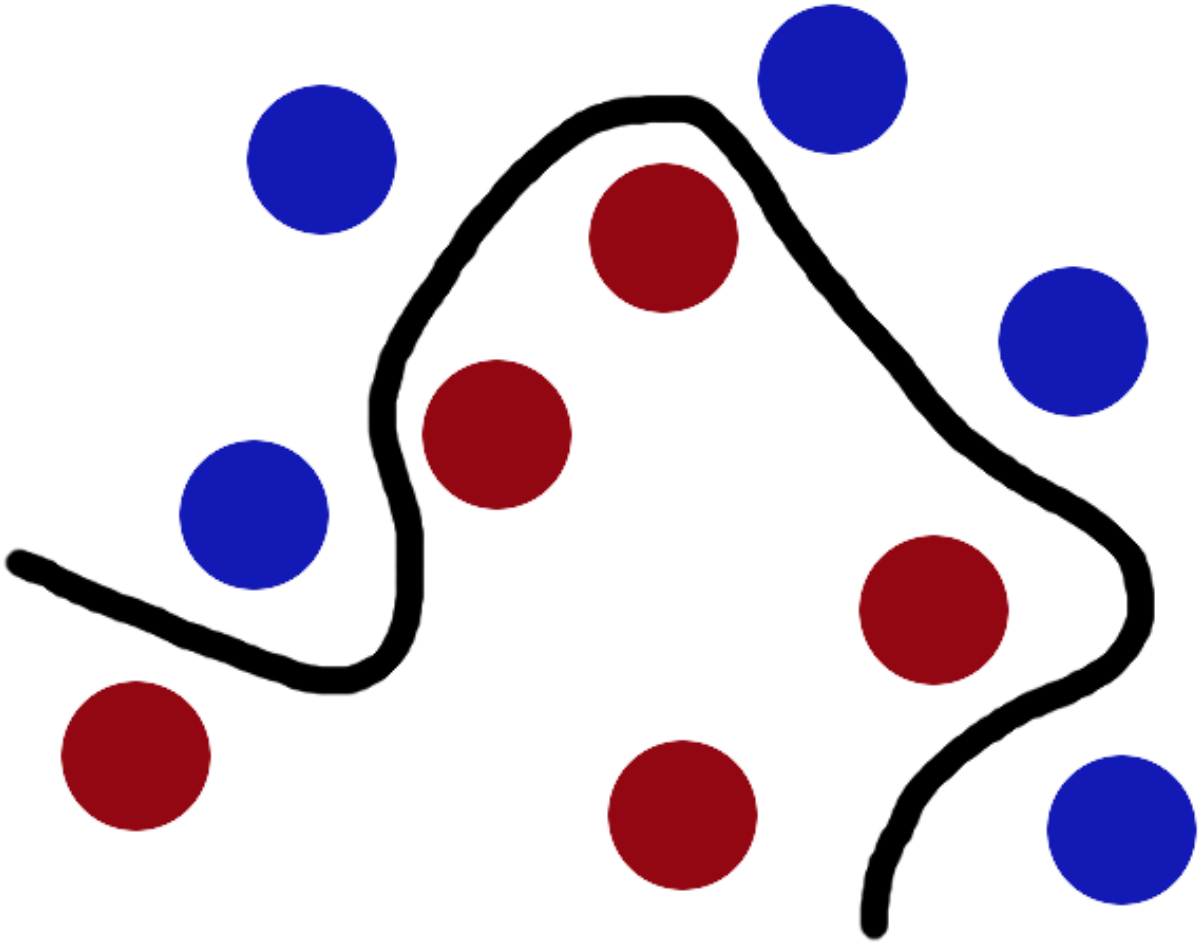
Support Vector Machine



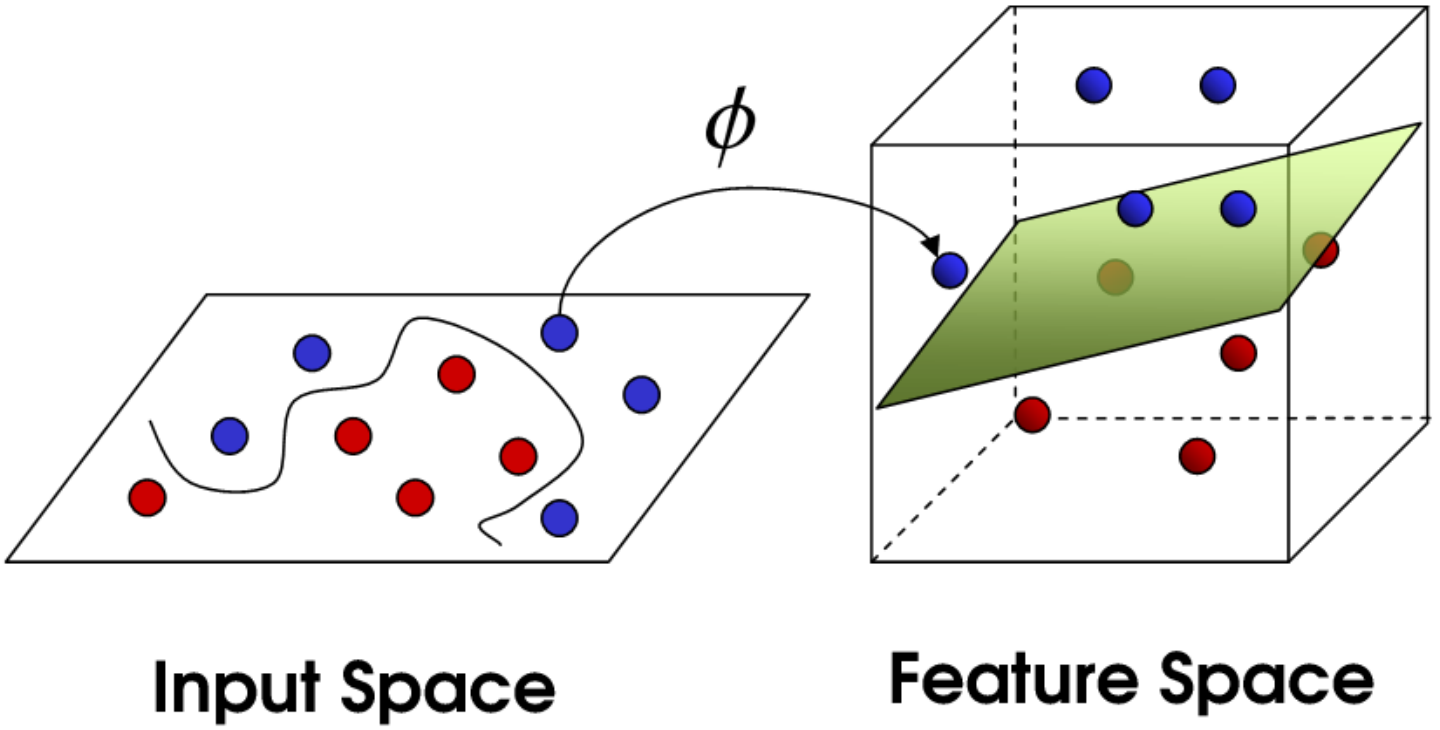
Support Vector Machine



Support Vector Machine



Support Vector Machine



Support Vector Machine

Généralisation à un problème de régression logistique à $K > 2$ classes :

- One Vs All : K modèles. Agréagation par meilleur score.
- One Vs One : $\frac{K(K-1)}{2}$ modèles. Vote majoritaire.

Apprentissage supervisé

Démo Sklearn

SVM

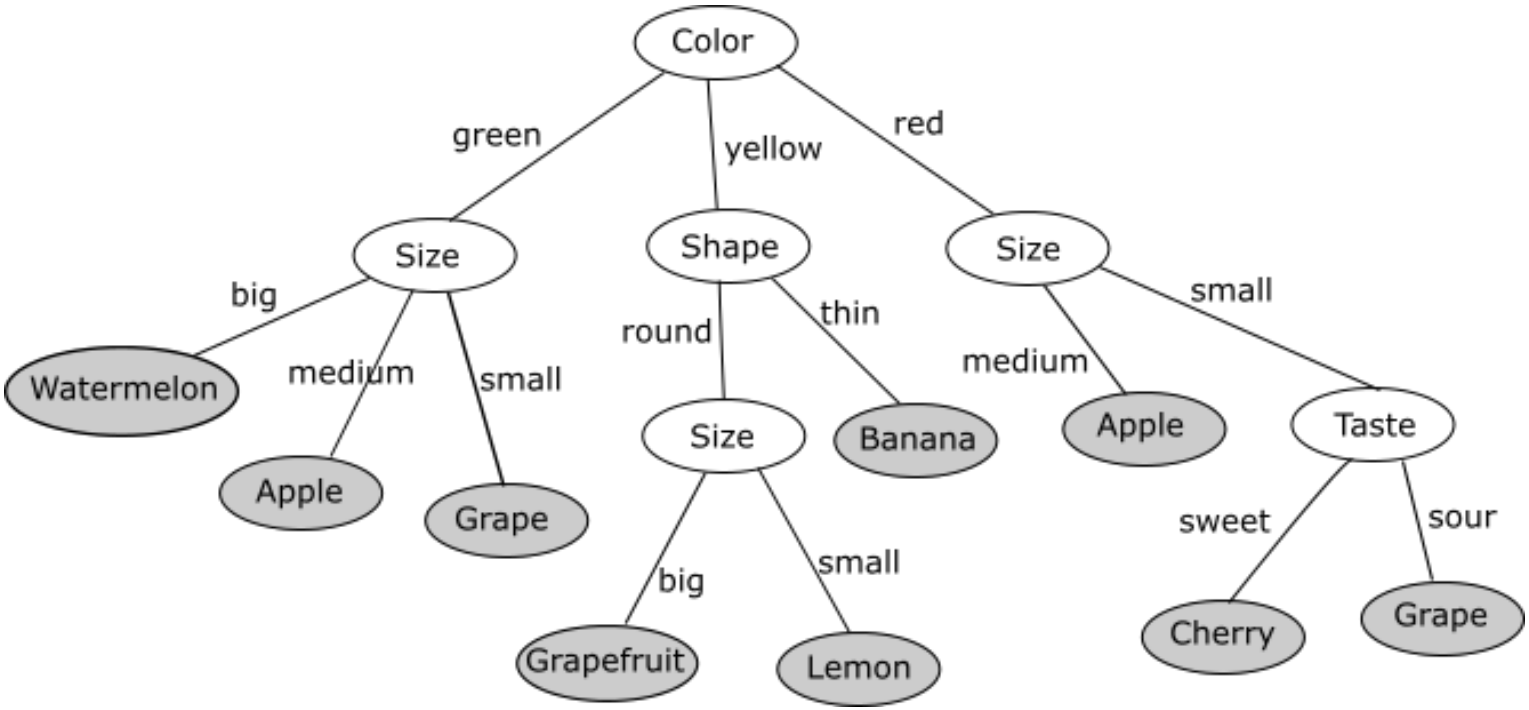
SVM Tutoriel

Apprentissage supervisé

Arbres de décision

Introduction

Modèle de classification ou regréssion qui classe un input dans une de ses feuilles pour rendre sa prédiction :



Avantages

Les arbres de décision

- gèrent les inputs numériques comme catégoriels

Avantages

Les arbres de décision

- gèrent les inputs numériques comme catégoriels
- ne nécessitent pas que la variable d'output soit normalement distribuée (regression linéaire)

Avantages

Les arbres de décision

- gèrent les inputs numériques comme catégoriels
- ne nécessitent pas que la variable d'output soit normalement distribuée (regression linéaire)
- sont interprétables

Avantages

Les arbres de décision

- gèrent les inputs numériques comme catégoriels
- ne nécessitent pas que la variable d'output soit normalement distribuée (regression linéaire)
- sont interprétables
- sont très rapides durant l'inférence

Avantages

Les arbres de décision

- gèrent les inputs numériques comme catégoriels
- ne nécessitent pas que la variable d'output soit normalement distribuée (regression linéaire)
- sont interprétables
- sont très rapides durant l'inférence
- ne nécessitent pas de normalisation des données

Avantages

Les arbres de décision

- gèrent les inputs numériques comme catégoriels
- ne nécessitent pas que la variable d'output soit normalement distribuée (regression linéaire)
- sont interprétables
- sont très rapides durant l'inférence
- ne nécessitent pas de normalisation des données
- leur apprentissage est hautement parallèlisable

Avantages

Les arbres de décision

- gèrent les inputs numériques comme catégoriels
- ne nécessitent pas que la variable d'output soit normalement distribuée (regression linéaire)
- sont interprétables
- sont très rapides durant l'inférence
- ne nécessitent pas de normalisation des données
- leur apprentissage est hautement parallèlisable

→ Couteau-suisse du machine learning tabulaire.

Désavantages

- peuvent overfit les données, mais l'ensembling résoud ce problème

Désavantages

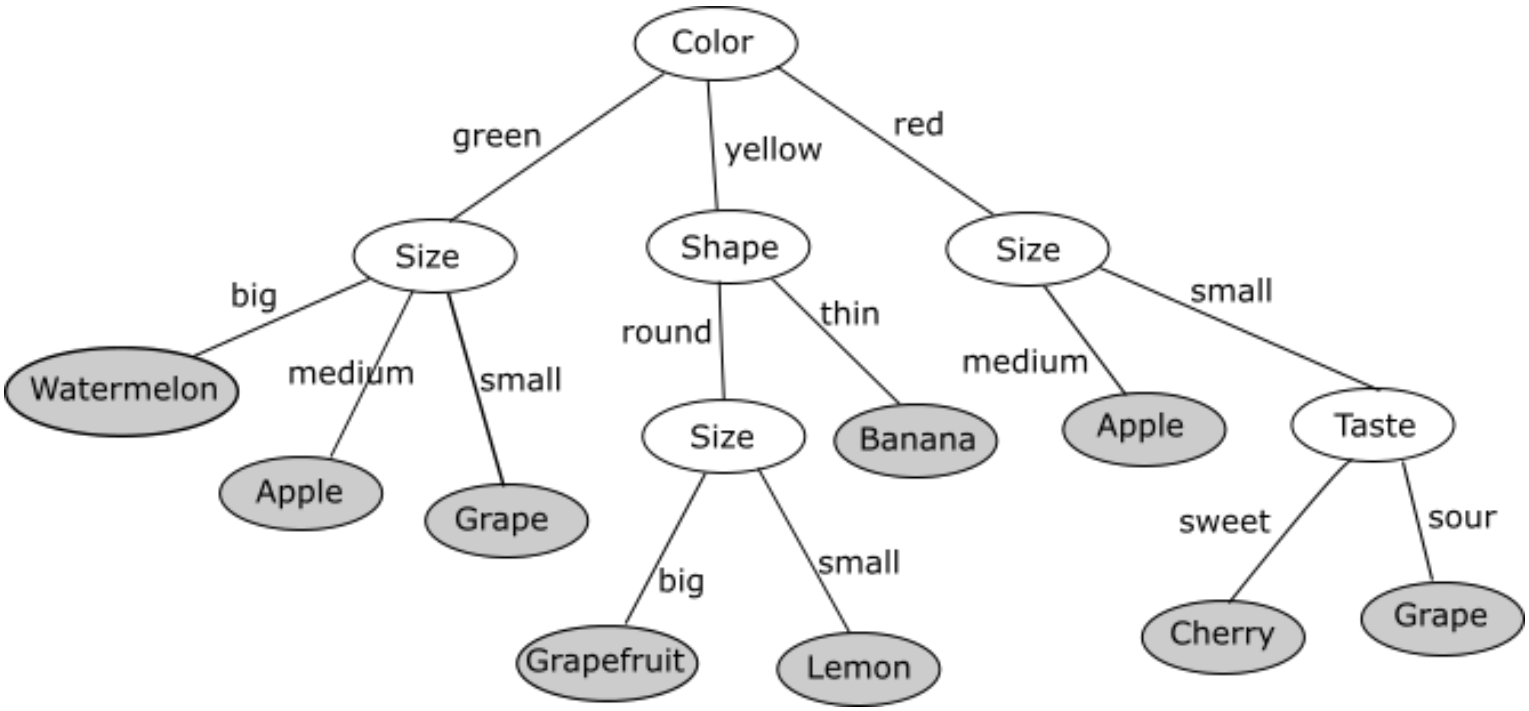
- peuvent overfit les données, mais l'ensembling résoud ce problème
- sont sensibles aux déséquilibres de classe

Désavantages

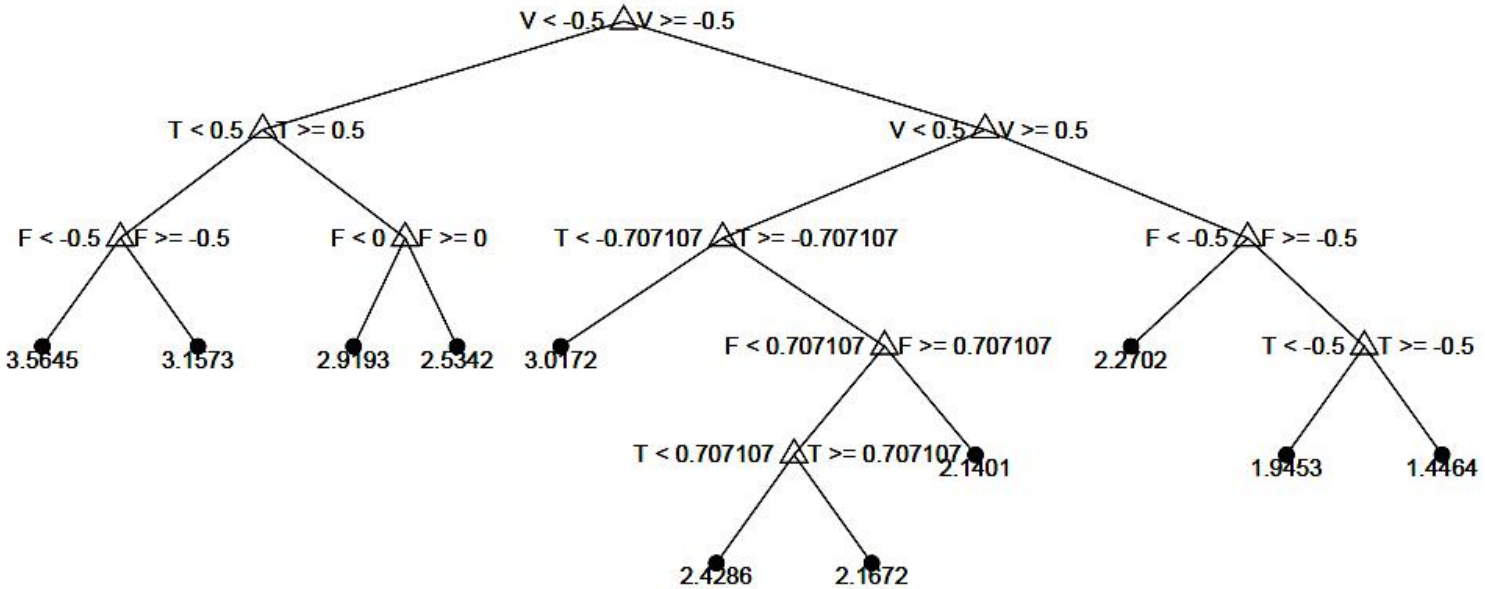
- peuvent overfit les données, mais l'ensembling résoud ce problème
- sont sensibles aux déséquilibres de classe

→ Si les classes ne sont pas équilibrées, peut-être les resampler.

Arbres de classification



Arbres de régression



Apprendre un arbre de décision

Approche « top-down », procédure récursive :

- créer un nœud de départ qui contient toutes les instances du training set

Apprendre un arbre de décision

Approche « top-down », procédure récursive :

- créer un nœud de départ qui contient toutes les instances du training set
- tant qu'il reste des nœuds non-traités :

Apprendre un arbre de décision

Approche « top-down », procédure récursive :

- créer un nœud de départ qui contient toutes les instances du training set
- tant qu'il reste des nœuds non-traités :
  - choisir un nœud non traité

Apprendre un arbre de décision

Approche « top-down », procédure récursive :

- créer un nœud de départ qui contient toutes les instances du training set
- tant qu'il reste des nœuds non-traités :
  - choisir un nœud non traité
  - si le nœud remplit des conditions de feuille finale, ne rien faire

Apprendre un arbre de décision

Approche « top-down », procédure récursive :

- créer un nœud de départ qui contient toutes les instances du training set
- tant qu'il reste des nœuds non-traités :
  - choisir un nœud non traité
  - si le nœud remplit des conditions de feuille finale, ne rien faire
  - sinon, créer deux branches à partir du nœud non traité pour répartir les instances dans deux nouveaux nœuds

Apprendre un arbre de décision

Approche « top-down », procédure récursive :

- créer un nœud de départ qui contient toutes les instances du training set
- tant qu'il reste des nœuds non-traités :
  - choisir un nœud non traité
  - si le nœud remplit des conditions de feuille finale, ne rien faire
  - sinon, créer deux branches à partir du nœud non traité pour répartir les instances dans deux nouveaux nœuds

Conditions de feuilles finales : contient n_{min} éléments, est déjà à profondeur p_{max}, splitterait sans décroître assez l'entropie…

Décision rendue

En fonction de la tâche, une fois arrivé dans la feuille de fin :

Classification classe majoritaire

Régression moyenne des valeurs cibles

Splits possibles

Splits possibles d'une feature donnée :

Catégorielle chaque catégorie vs le reste

Ordinale/Continue milieu de chaque valeur ou quantiles

Évaluation de la qualité d'un split

En fonction de la tâche :

Régression coût si on rendait la moyenne des instances comme résultat

$$Loss = \sum |\hat{y} - y| \approx variance$$

Classification Entropie de Shannon :

$$Loss = -\sum_{x \in X} P_x * \log_2(P_x)$$

$= 0 \Rightarrow$ il n'y a pas d'incertitude
maximale quand on a une distribution uniforme

Exemple — démarrage

ID, jardinage, jeux vidéos, chapeaux, âge

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 13 \\ 2 & 0 & 1 & 0 & 14 \\ 3 & 0 & 1 & 0 & 15 \\ 4 & 1 & 1 & 1 & 25 \\ 5 & 0 & 1 & 1 & 35 \\ 6 & 1 & 0 & 0 & 49 \\ 7 & 1 & 1 & 1 & 68 \\ 8 & 1 & 0 & 0 & 71 \\ 9 & 1 & 0 & 1 & 73 \end{bmatrix}$$

Première étape : création du nœud de départ

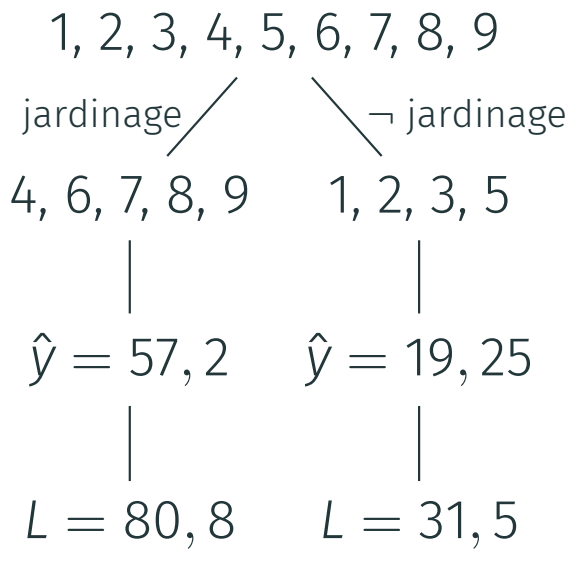
1, 2, 3, 4, 5, 6, 7, 8, 9

Exemple — split

ID, jardinage, jeux vidéos, chapeaux, âge

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 13 \\ 2 & 0 & 1 & 0 & 14 \\ 3 & 0 & 1 & 0 & 15 \\ 4 & 1 & 1 & 1 & 25 \\ 5 & 0 & 1 & 1 & 35 \\ 6 & 1 & 0 & 0 & 49 \\ 7 & 1 & 1 & 1 & 68 \\ 8 & 1 & 0 & 0 & 71 \\ 9 & 1 & 0 & 1 & 73 \end{bmatrix}$$

Split du premier nœud. Il faut tester 3 splits. Split sur jardinage :



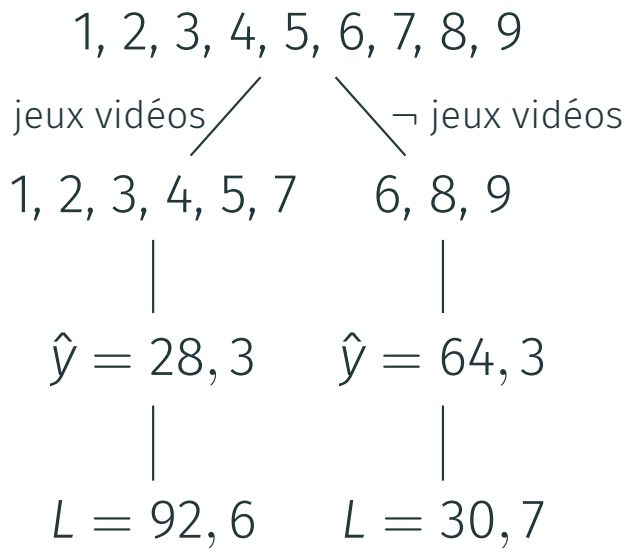
Loss totale : $122,3$

Exemple — split

ID, jardinage, jeux vidéos, chapeaux, âge

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 13 \\ 2 & 0 & 1 & 0 & 14 \\ 3 & 0 & 1 & 0 & 15 \\ 4 & 1 & 1 & 1 & 25 \\ 5 & 0 & 1 & 1 & 35 \\ 6 & 1 & 0 & 0 & 49 \\ 7 & 1 & 1 & 1 & 68 \\ 8 & 1 & 0 & 0 & 71 \\ 9 & 1 & 0 & 1 & 73 \end{bmatrix}$$

Split du premier nœud. Il faut tester 3 splits. Split sur jeux vidéos :



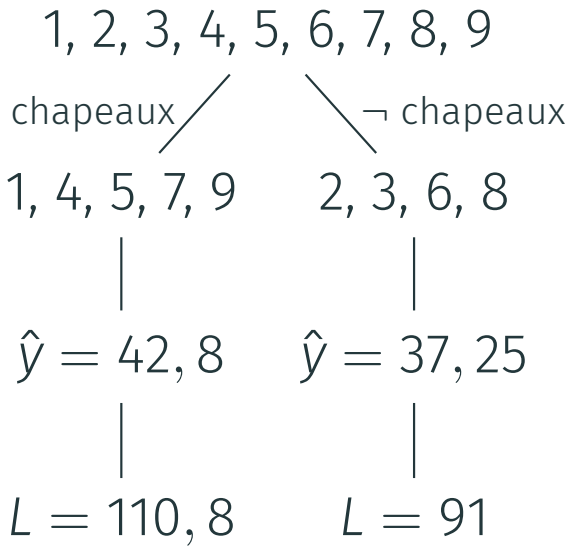
Loss totale : 123, 3

Exemple — split

ID, jardinage, jeux vidéos, chapeaux, âge

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 13 \\ 2 & 0 & 1 & 0 & 14 \\ 3 & 0 & 1 & 0 & 15 \\ 4 & 1 & 1 & 1 & 25 \\ 5 & 0 & 1 & 1 & 35 \\ 6 & 1 & 0 & 0 & 49 \\ 7 & 1 & 1 & 1 & 68 \\ 8 & 1 & 0 & 0 & 71 \\ 9 & 1 & 0 & 1 & 73 \end{bmatrix}$$

Split du premier nœud. Il faut tester 3 splits. Split sur chapeaux :



Loss totale : $201,8$

Exemple — split

ID, jardinage, jeux vidéos, chapeaux, âge

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 13 \\ 2 & 0 & 1 & 0 & 14 \\ 3 & 0 & 1 & 0 & 15 \\ 4 & 1 & 1 & 1 & 25 \\ 5 & 0 & 1 & 1 & 35 \\ 6 & 1 & 0 & 0 & 49 \\ 7 & 1 & 1 & 1 & 68 \\ 8 & 1 & 0 & 0 & 71 \\ 9 & 1 & 0 & 1 & 73 \end{bmatrix}$$

122,3 jardinage

123,3 jeux vidéos

201,8 chapeaux

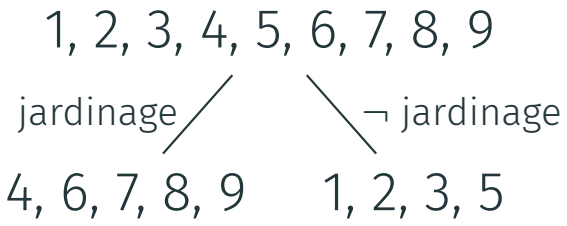
→ On split donc sur jardinage

Exemple — split

ID, jardinage, jeux vidéos, chapeaux, âge

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 13 \\ 2 & 0 & 1 & 0 & 14 \\ 3 & 0 & 1 & 0 & 15 \\ 4 & 1 & 1 & 1 & 25 \\ 5 & 0 & 1 & 1 & 35 \\ 6 & 1 & 0 & 0 & 49 \\ 7 & 1 & 1 & 1 & 68 \\ 8 & 1 & 0 & 0 & 71 \\ 9 & 1 & 0 & 1 & 73 \end{bmatrix}$$

Résultat après le premier split :



À vous de jouer !

Limiter l'overfit

Fait par :

- la profondeur maximum
- le nombre minimum d'instances dans chaque feuille
- une baisse d'entropie maximale à chaque split
- le nombre minimum d'instances pour split
- le pruning

Apprentissage supervisé

Random Forest

Introduction

- les arbres de décision overfit facilement
- ils sont rapides à apprendre
- en combiner beaucoup est faisable et réduit la variance

→ création d'une forêt (ensemble d'arbres) aléatoire

But

Produire des arbres décorrélés et moyenner leurs prédictions pour réduire la variance.

Outil 1 — bagging (row sampling)

Boostrap **agg**regat**ing** (Bagging) :

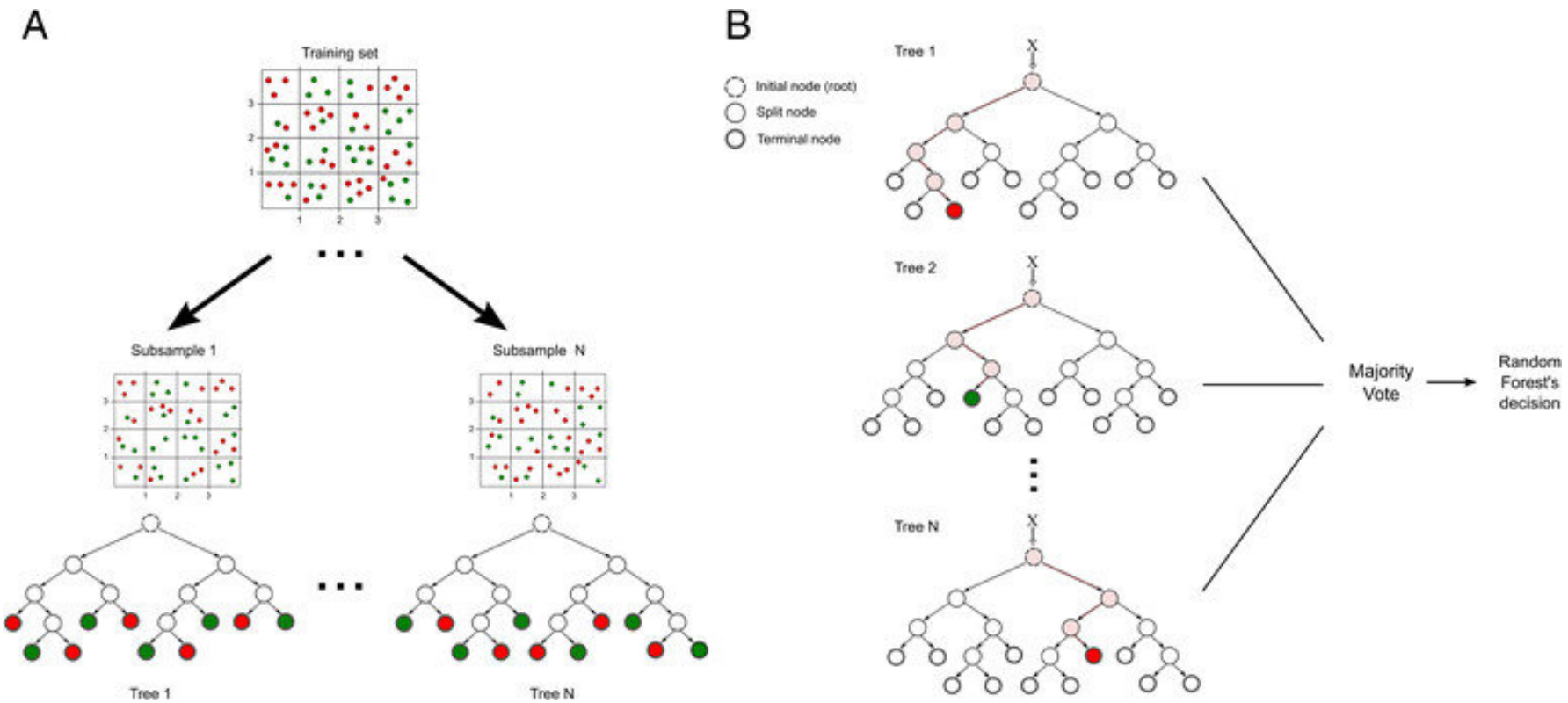
- tirer un échantillon du dataset avec replacement
- entraîner un arbre sur cet échantillon
- répéter B fois

Le bagging s'appelle aussi row sampling.

Outil 2 — random subspace method (column sampling)

- à chaque split, considérer seulement un sous-ensemble des features
- valeurs conseillées :
  - classification : $|\sqrt{m}|$ features par split
  - regréssion : $|\frac{m}{3}|$ features par split, 5 exemples par node minimum

Random Forest



Random Forest

- Pas de sur-apprentissage en augmentant le nombre d'arbres
- Une fois appris, le modèle est très rapide

Conclusion

- les arbres sont interprétables, rapides à entraîner, combinables.
- random forest combine des arbres faibles en un prédicteur versatile

Avez-vous des questions ?

Apprentissage supervisé

Démo Sklearn

Random Forest

Random Forest - Tutoriel

Apprentissage supervisé

Travaux Pratiques : Random Forest

Random Forest

Random Forest - TP

Avez-vous des questions ?