

Apprentissage Supervisé

Blaise Hanczar

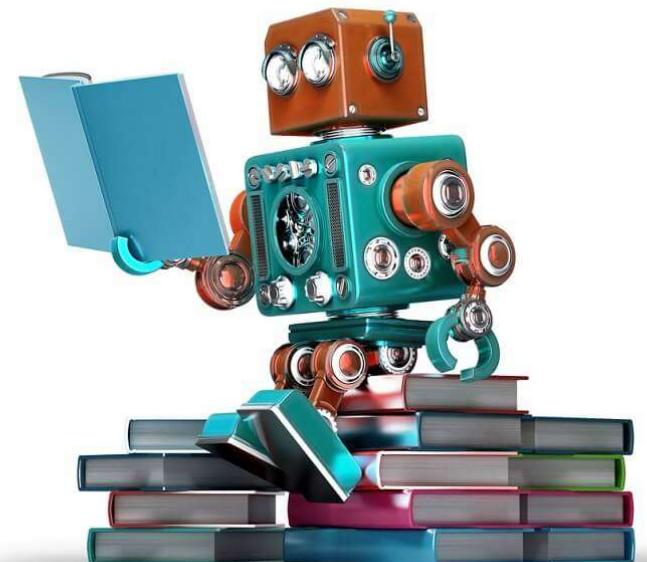


Laboratoire IBISC
Informatique, Bioinformatique et Systèmes Complexes



Apprentissage supervisé

- Notions de bases
- Régression linéaire / logistique
- K plus proche voisins
- Arbres de décision
- Machines à vecteur de support
- Méthodes d'ensemble
 - Random forest
 - Boosting
- Apprentissage profond
 - Perceptron multicouche (MLP)
 - Réseaux de convolution (CNN)
 - Réseaux récurrents (RNN, LSTM, GRU)



Notions de bases

Régression Linéaire

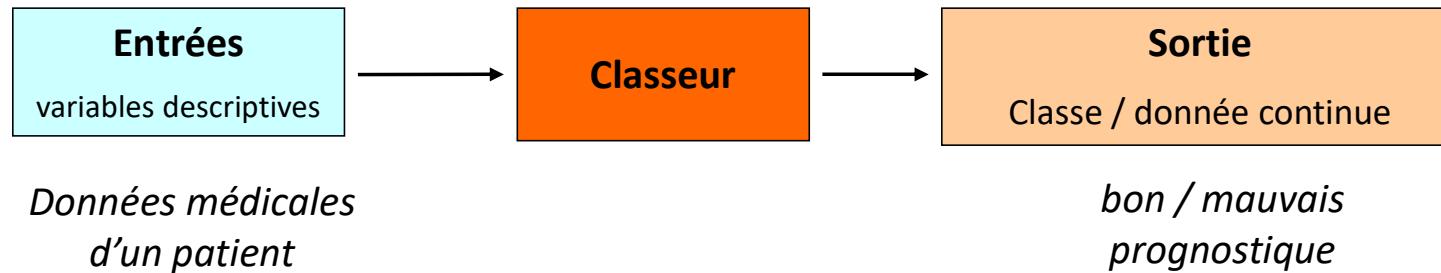
Régression Logistique

Apprentissage Automatique Machine Learning

- Apprentissage supervisée :
 - Inférence d'une fonction à partir de données étiquetées
 - Objectif de faire des prédictions sur des nouvelles données
- Apprentissage non supervisée :
 - Inférence d'un fonction à partir de données non étiquetées
 - Identifier des structures des les données
- Apprentissage par renforcement
 - Apprendre à partir d'expériences
 - Trouver un comportement maximisant une fonction de récompense

Apprentissage supervisée

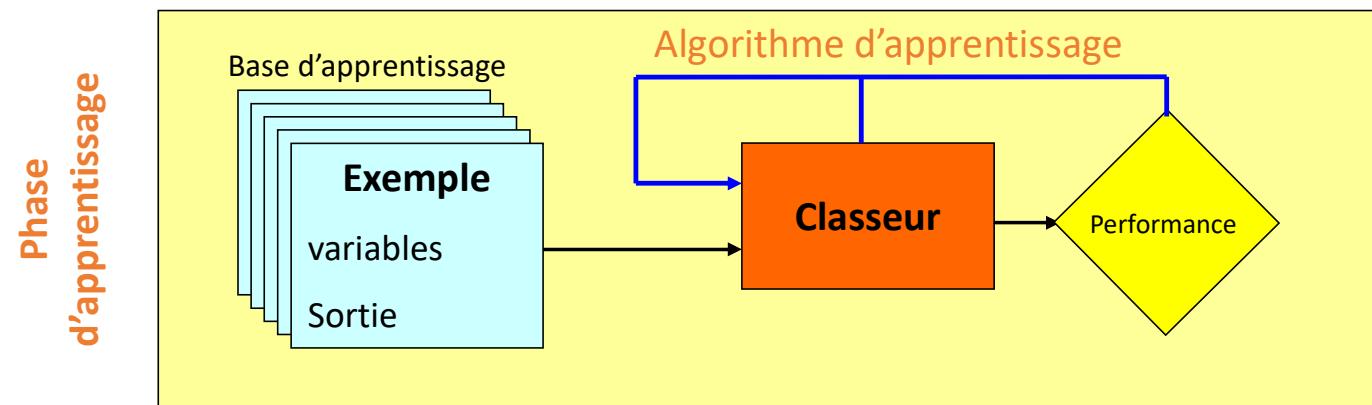
Objectif : Apprendre un modèle prédictif (classeur) à partir d'un ensemble d'exemples d'apprentissage.



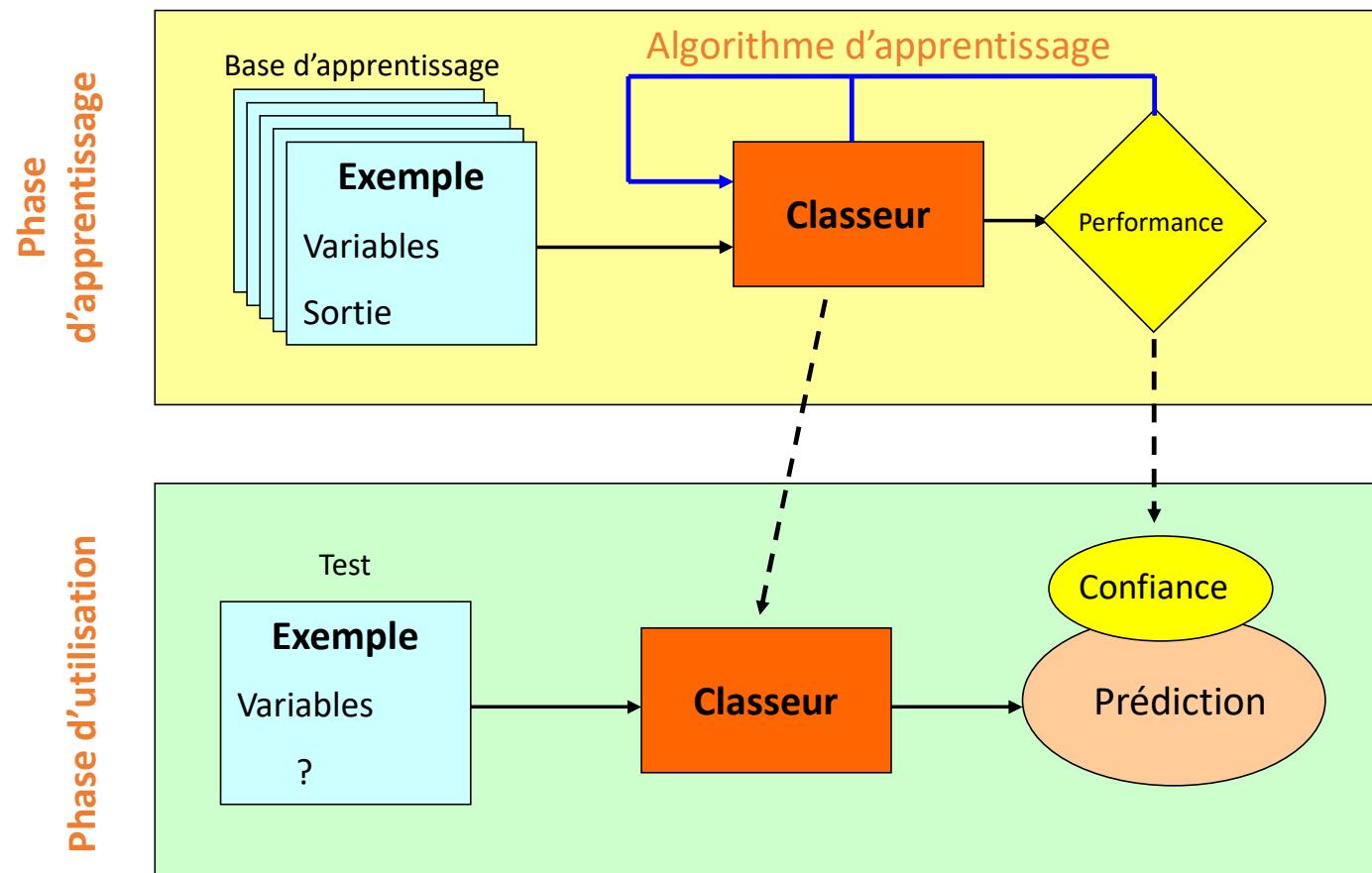
Classification: sortie discrète (classe)

Régression: sortie continue

Construction d'un modèle

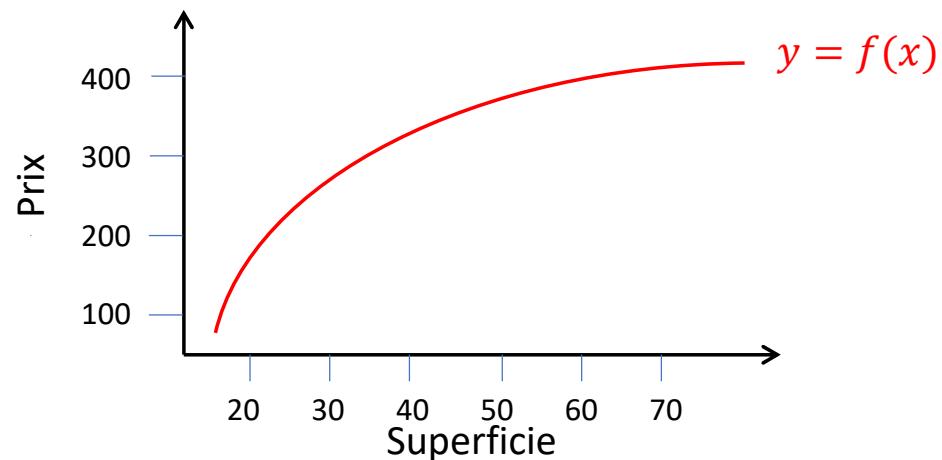


Utilisation du modèle



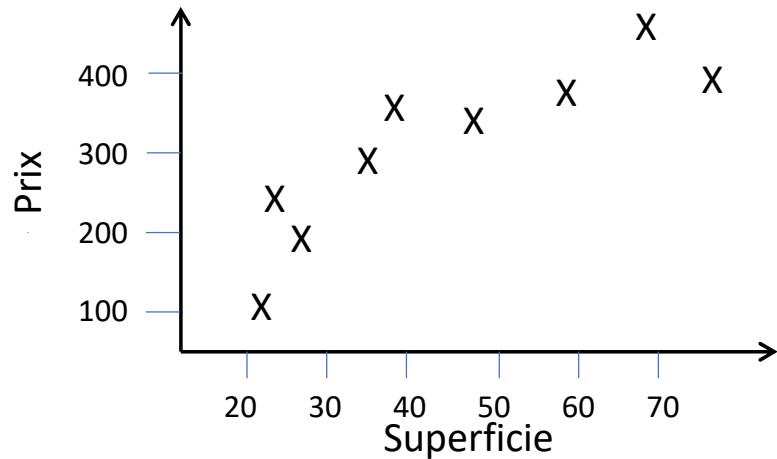
Problème de régression

Prédiction du prix d'un logement à partir de sa superficie



Problème de régression

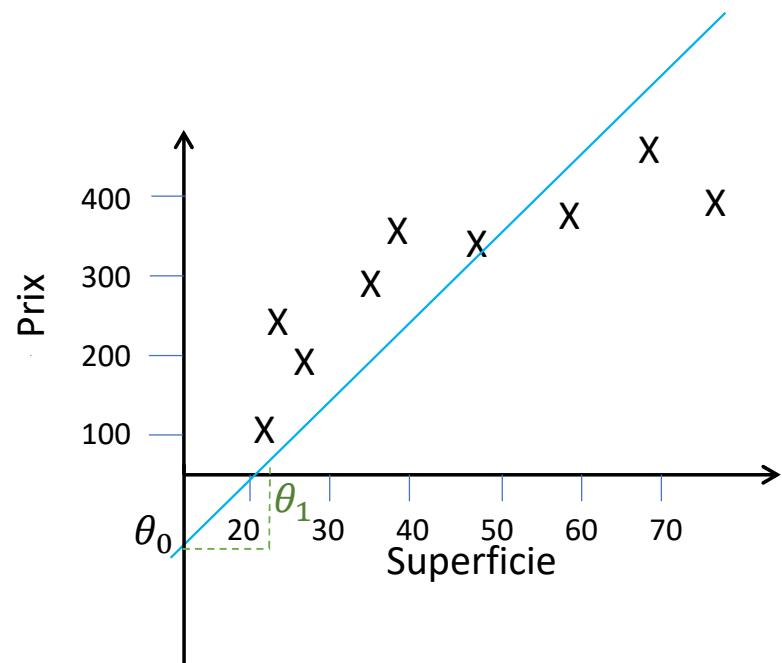
Base d'apprentissage $S=\{X_i, Y_i\}$ de m exemples



Superficie	Prix
21	110
23	240
27	190
36	280
39	350
48	340
60	360
68	440
77	390

Problème de régression

Choisir un modèle

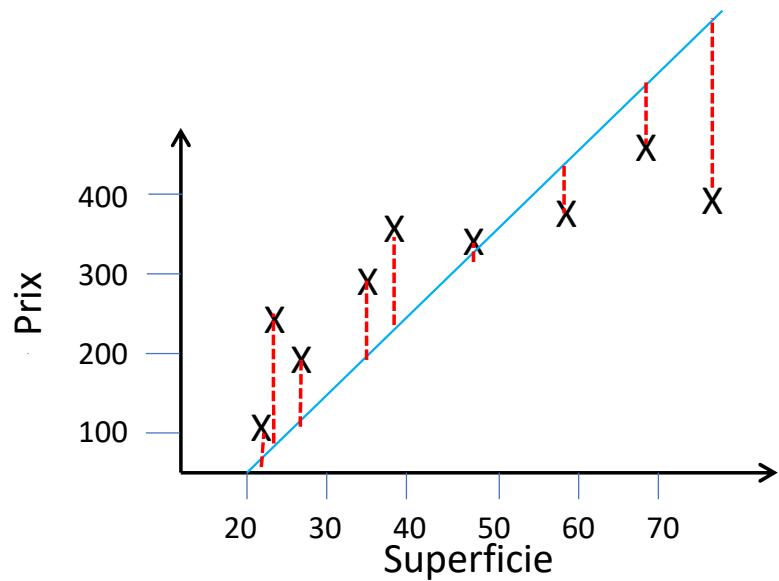


Modèle linéaire:

$$h(x) = \theta_0 + \theta_1 x$$

Problème de régression

Evaluer l'erreur du modèle



Modèle linéaire:

$$h(x) = \theta_0 + \theta_1 x$$

Fonction de cout :

$$J = \sqrt{\frac{1}{m} \sum_{i=1}^m (\hat{f}(x_i) - y_i)^2}$$

Objectif :

$$\theta_0^*, \theta_1^* = \operatorname{argmin}_{\theta_0, \theta_1} (J)$$

Descente du gradient

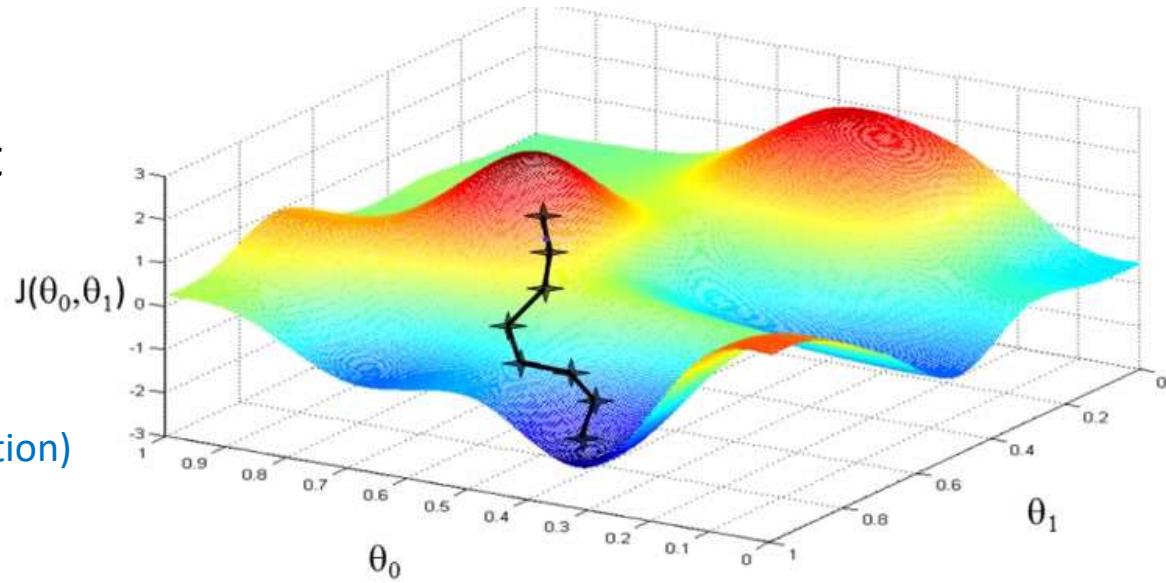
Algorithme de descente du gradient

- Initialisation des poids θ
- Répéter

$$\theta^{new} := \theta - \alpha \frac{\partial L(\theta)}{\partial \theta}$$

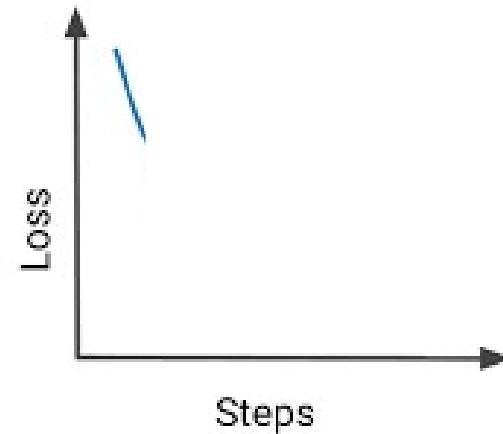
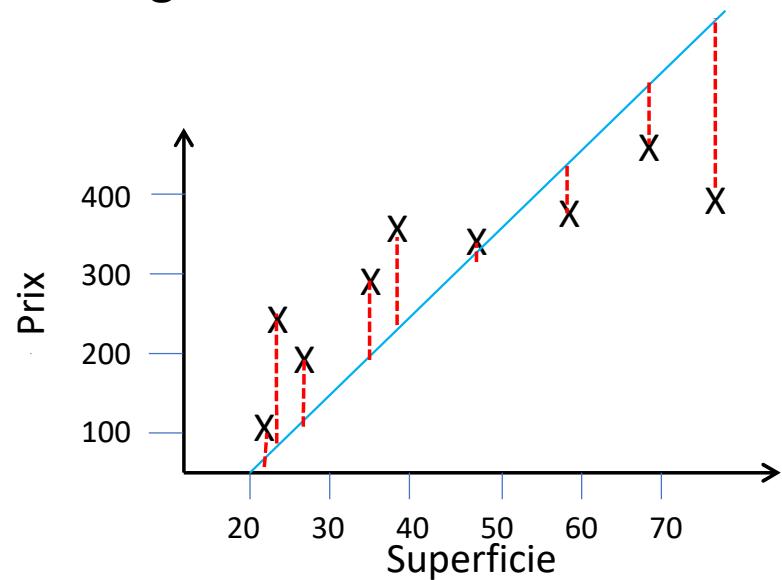
Gradient (direction)
Pas d'apprentissage

Jusqu' à $J(\theta)$ ne diminue plus



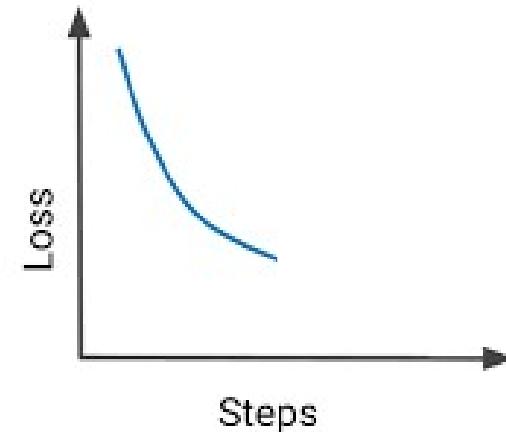
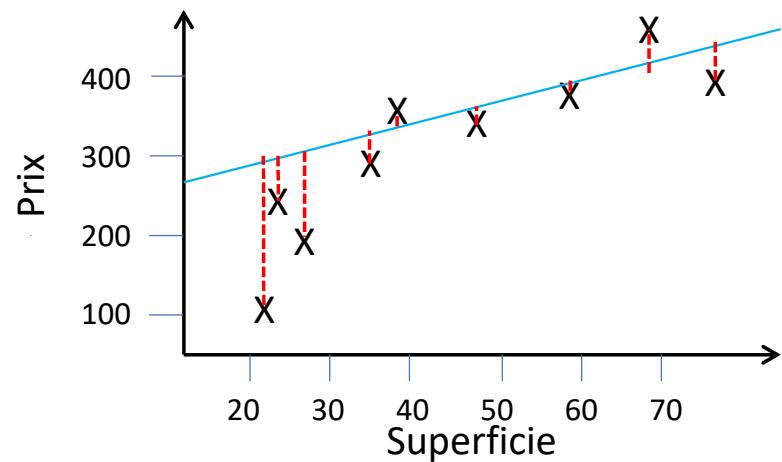
Problème de régression

Apprentissage du modèle en fonction des erreurs sur la base d'apprentissage



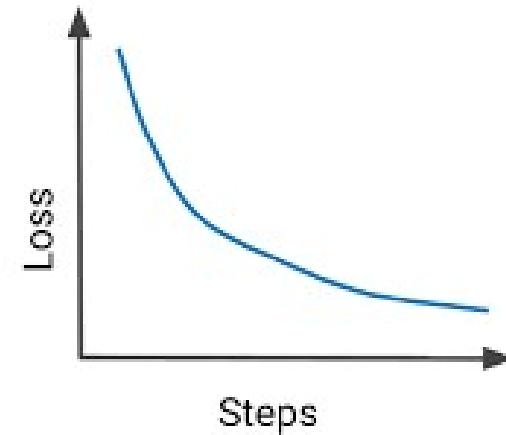
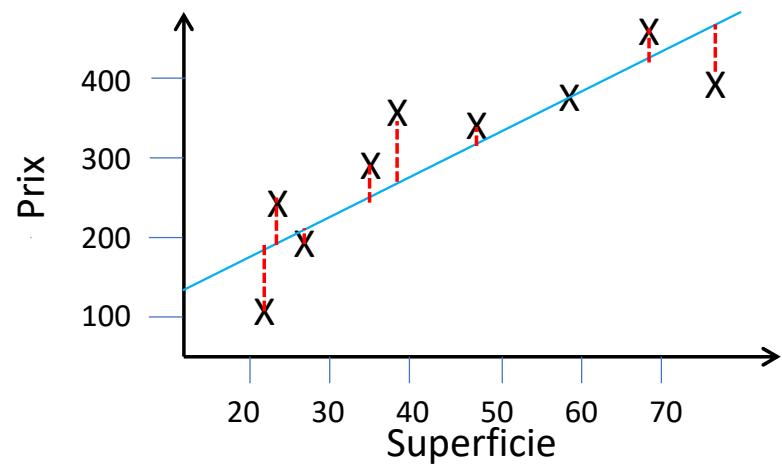
Problème de régression

Apprentissage du modèle en fonction des erreurs sur la base d'apprentissage



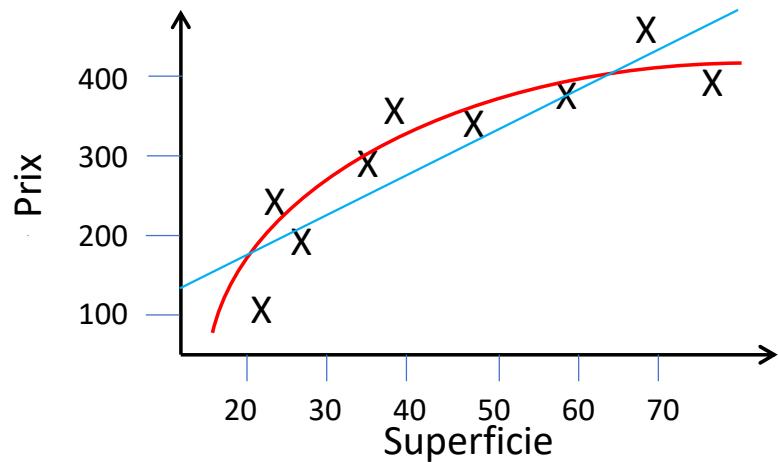
Problème de régression

Apprentissage du modèle en fonction des erreurs sur la base d'apprentissage



Problème de régression

Choisir un modèle

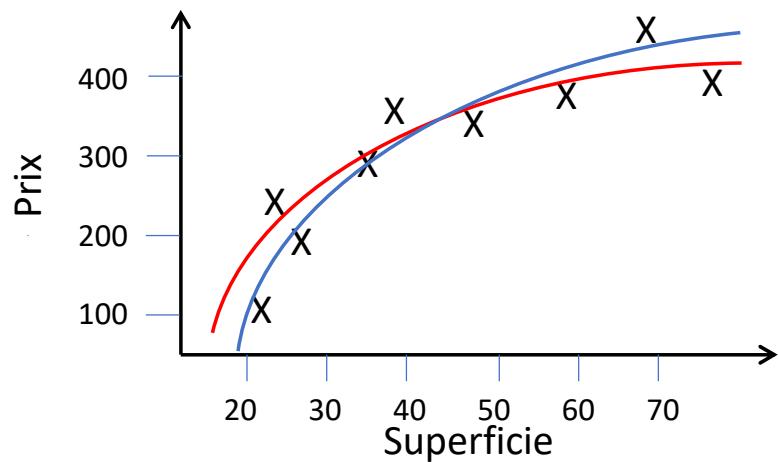


Modèle linéaire:

$$h(x) = \theta_0 + \theta_1 x$$

Problème de régression

Choisir un modèle



Modèle linéaire:

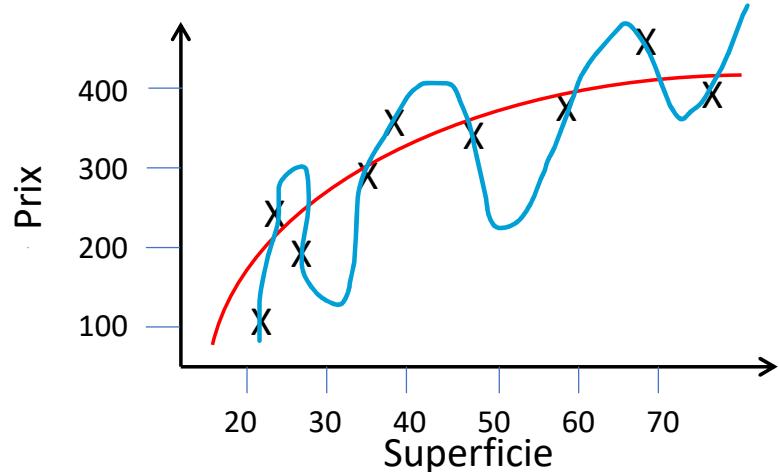
$$h(x) = \theta_0 + \theta_1 x$$

Polynôme de degré 2:

$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Problème de régression

Choisir un modèle



Modèle linéaire:

$$h(x) = \theta_0 + \theta_1 x$$

Polynôme de degré 2:

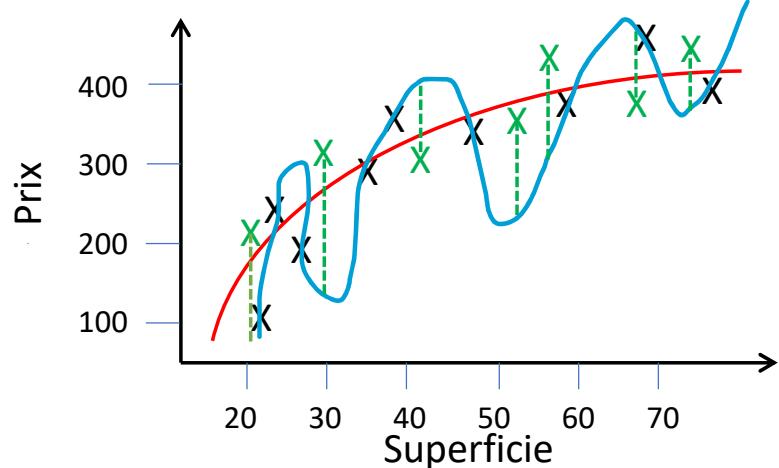
$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Polynôme de degré d:

$$h(x) = \theta_0 + \sum_{i=1}^d \theta_i x^i$$

Problème de régression

Choisir un autre modèle



Modèle linéaire:

$$h(x) = \theta_0 + \theta_1 x$$

Polynôme de degré 2:

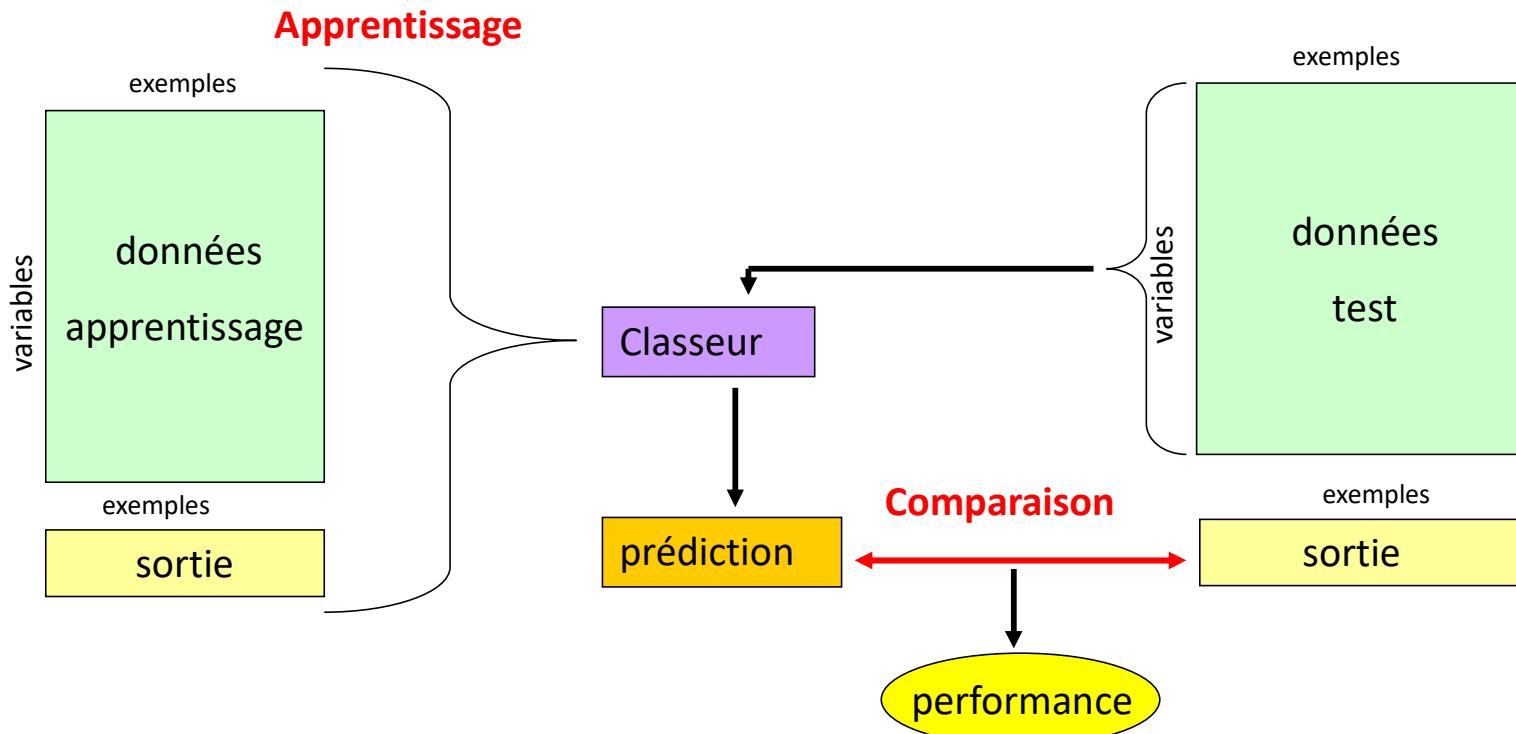
$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Polynôme de degré d:

$$h(x) = \theta_0 + \sum_{i=1}^d \theta_i x^i$$

Estimation de l'erreur de prédiction sur une base de test indépendante

Ensemble d'apprentissage et de test



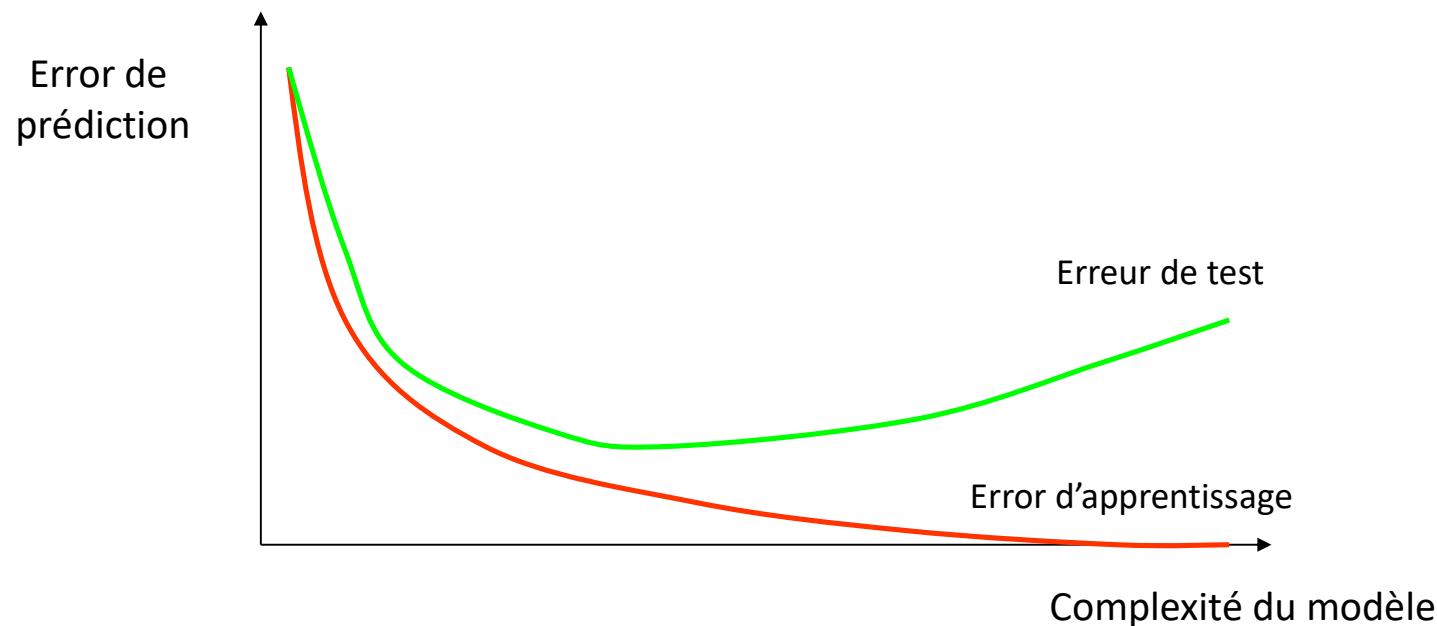
Sur-apprentissage

Comment le sur-apprentissage affecte la performance des classeurs ?



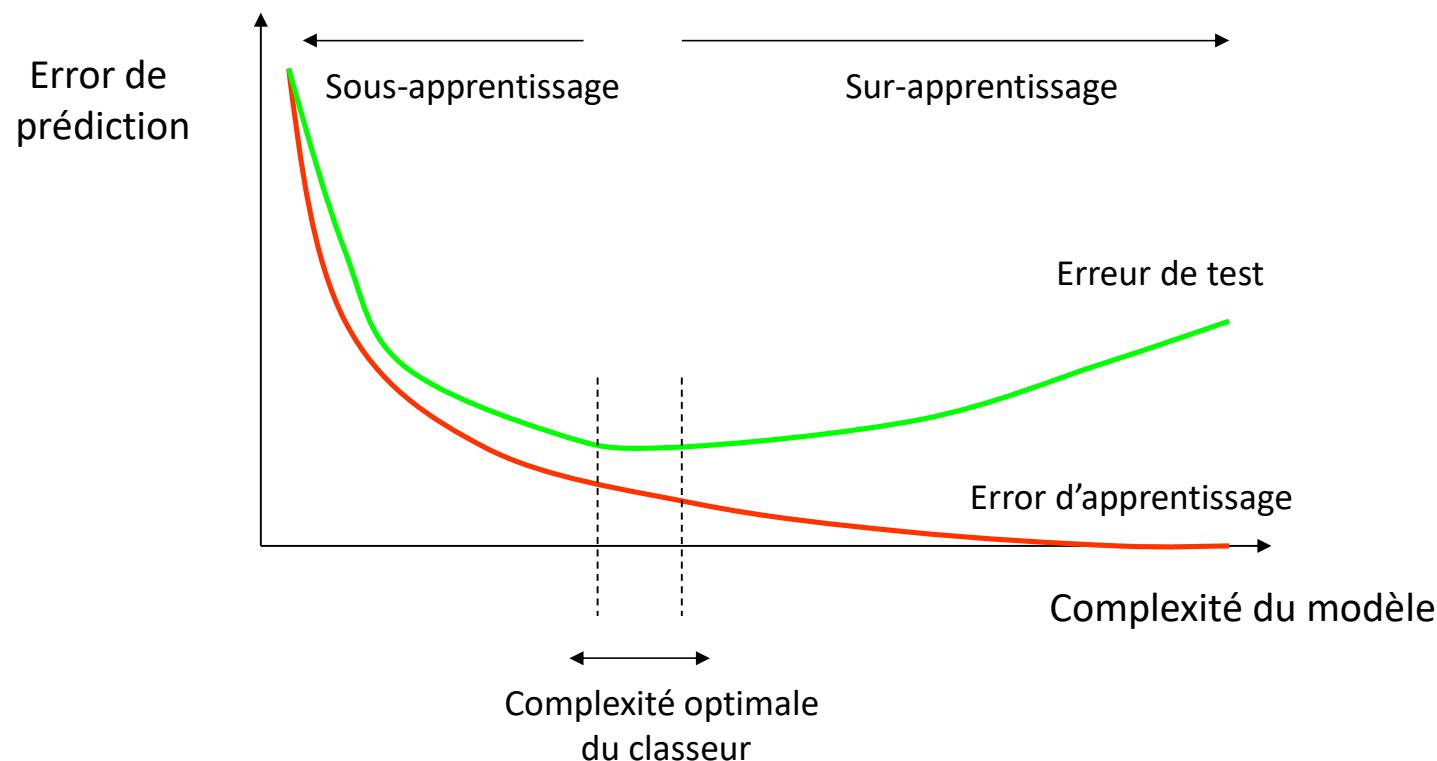
Sur-apprentissage

Comment le sur-apprentissage affecte la performance des classeurs ?



Sur-apprentissage

Comment le sur-apprentissage affecte la performance des classeurs ?



Réduire le sur-apprentissage

- Augmenter le nombre d'exemples
- Réduire le nombre de variables
- Limiter la complexité du classeur
- Méthode de régularisation

Régularisation

- Limiter la valeurs des poids pour éviter le sur-apprentissage

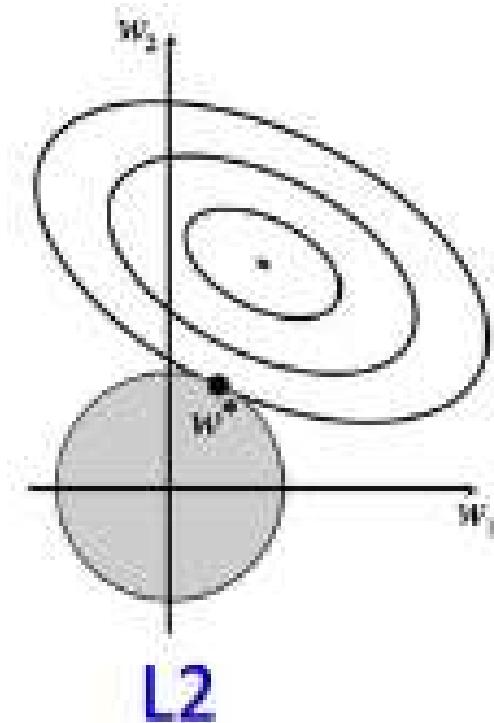
- Ajout d'un terme de régularisation à la fonction de cout

$$\hat{J}(\theta) = J(\theta) + \lambda\Omega(\theta)$$

- Régularisation L2

$$\Omega(\theta) = \frac{1}{2} \|\theta\|_2^2 = \sqrt{\sum_i \|\theta_i\|^2}$$

Ridge regression



Régularisation

- Régularisation L1

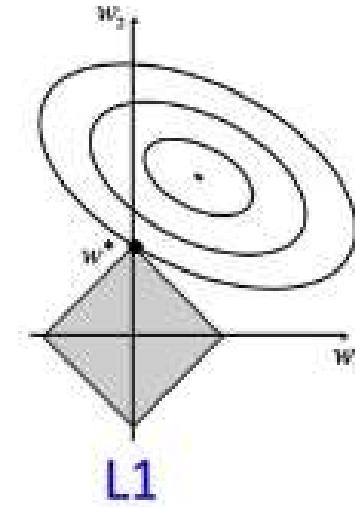
$$\Omega(W) = \|\theta\|^1 = \sum_i |\theta_i|$$

LASSO

- On peut combiner régularisation L1 et L2

$$\hat{J}(\theta) = J(\theta) + \lambda (\beta \Omega_{L1}(\theta) + (1 - \beta) \Omega_{L2}(\theta))$$

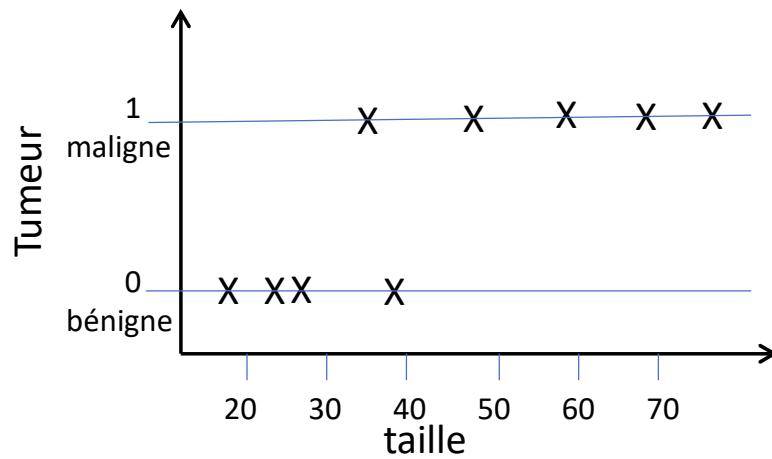
Elastic net



Problème de classification

Base d'apprentissage $S=\{X_i, Y_i\}$ de m exemples

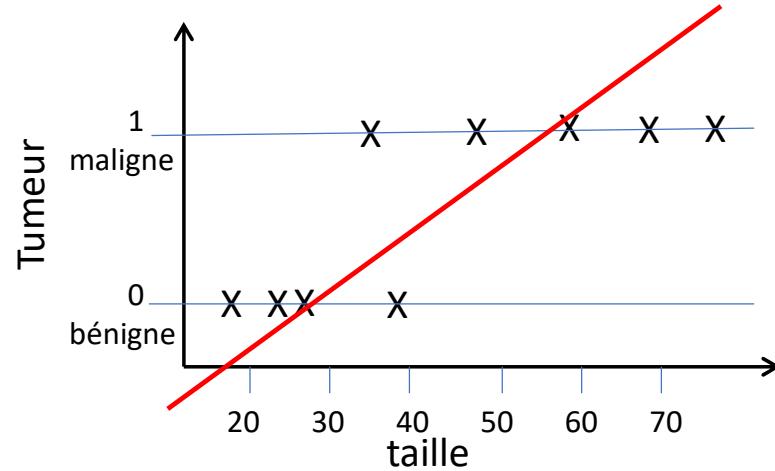
Y est une valeur discrete (une classe)



Taille	tumeur
18	0
23	0
27	0
36	1
39	0
48	1
60	1
68	1
77	1

0 bénigne
1 maligne

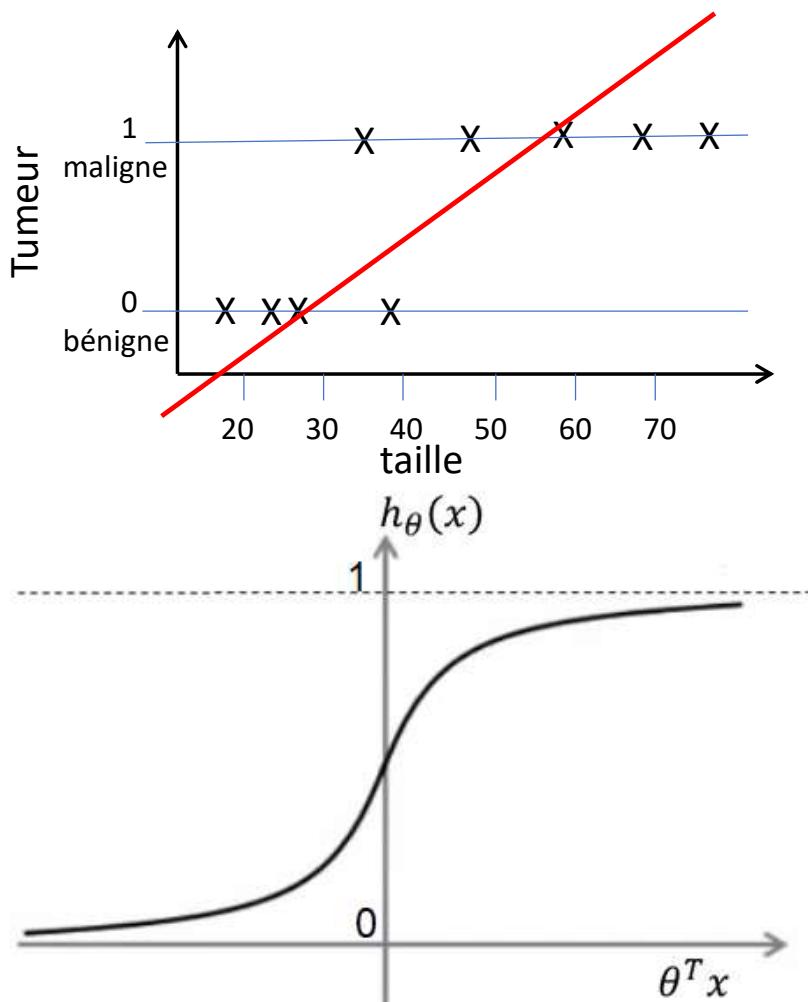
Problème de classification



Modèle linéaire:

$$h(x) = \theta_0 + \theta_1 x = \theta^T x$$

Problème de classification



Modèle linéaire:

$$h(x) = \theta_0 + \theta_1 x = \theta^T x$$

Régression logistique:

- On projette la sortie du modèle linéaire sur [0,1]
- On considère $h(x)$ comme la probabilité de x d'appartenir à la classe $y=1$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \approx p(y = 1|x)$$

- Prédiction dans la classe la plus probable

$$H(x) = \begin{cases} 0 & \text{si } h(x) < 0.5 \\ 1 & \text{si } h(x) \geq 0.5 \end{cases}$$

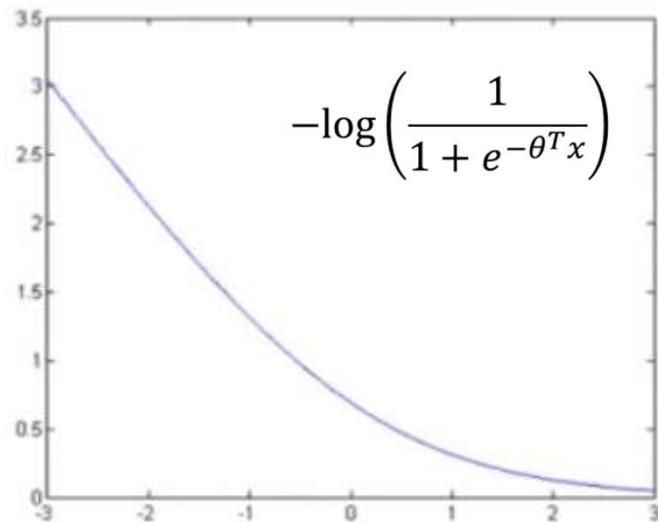
- Objectif :

$$\theta^* = \operatorname{argmin}_\theta J$$

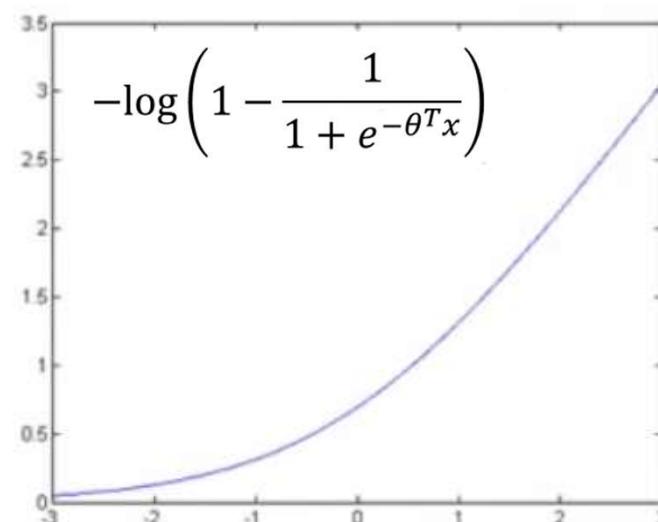
Fonction de cout

$$\begin{aligned}\text{Coût} &= -(y \log(h_\theta(x)) + (1 - y) \log(1 - h_\theta(x))) \\ &= -\left(y \log\left(\frac{1}{1+e^{-\theta^T x}}\right) + (1 - y) \log\left(1 - \frac{1}{1+e^{-\theta^T x}}\right)\right)\end{aligned}$$

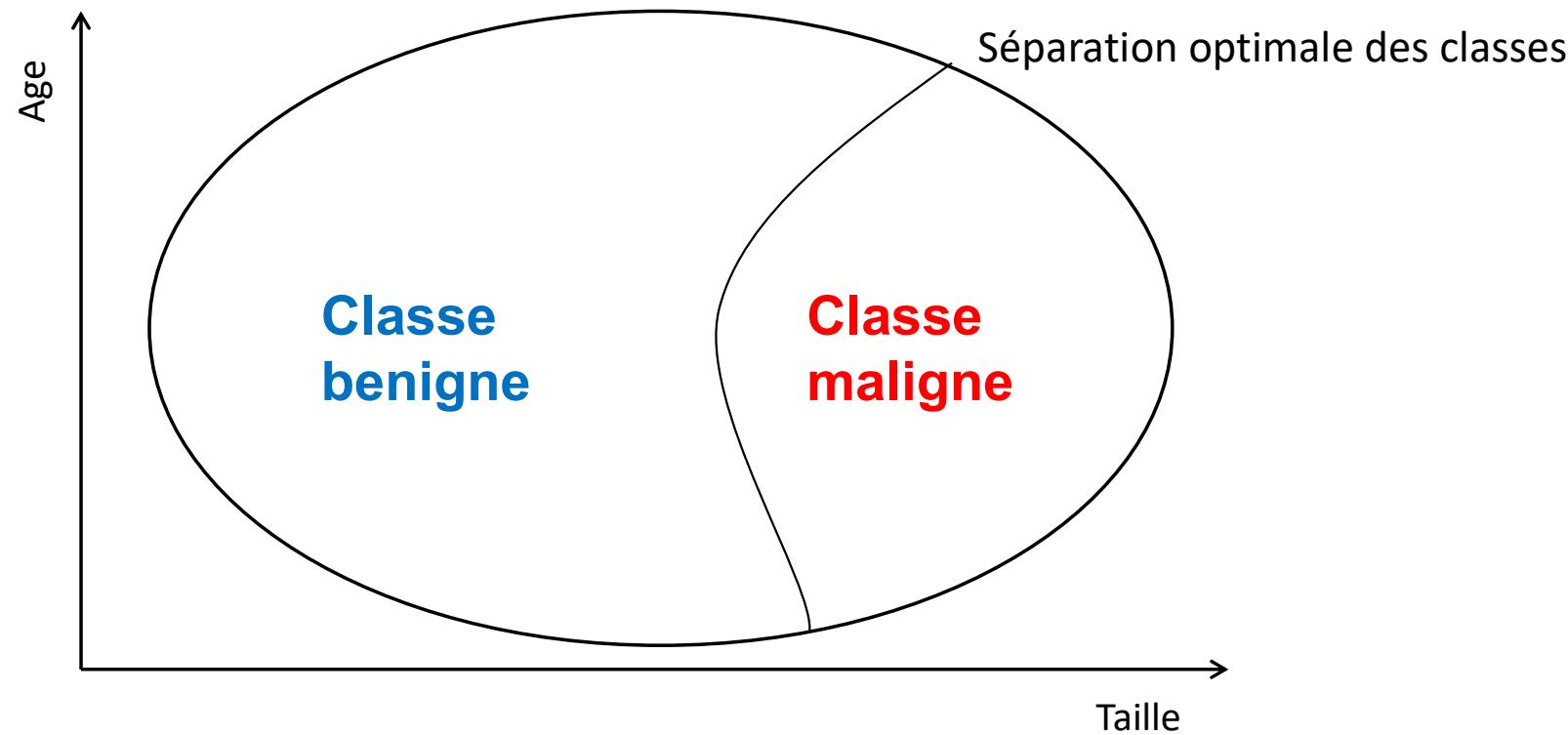
Si $y = 1$, on veut $\theta^T x \gg 0$



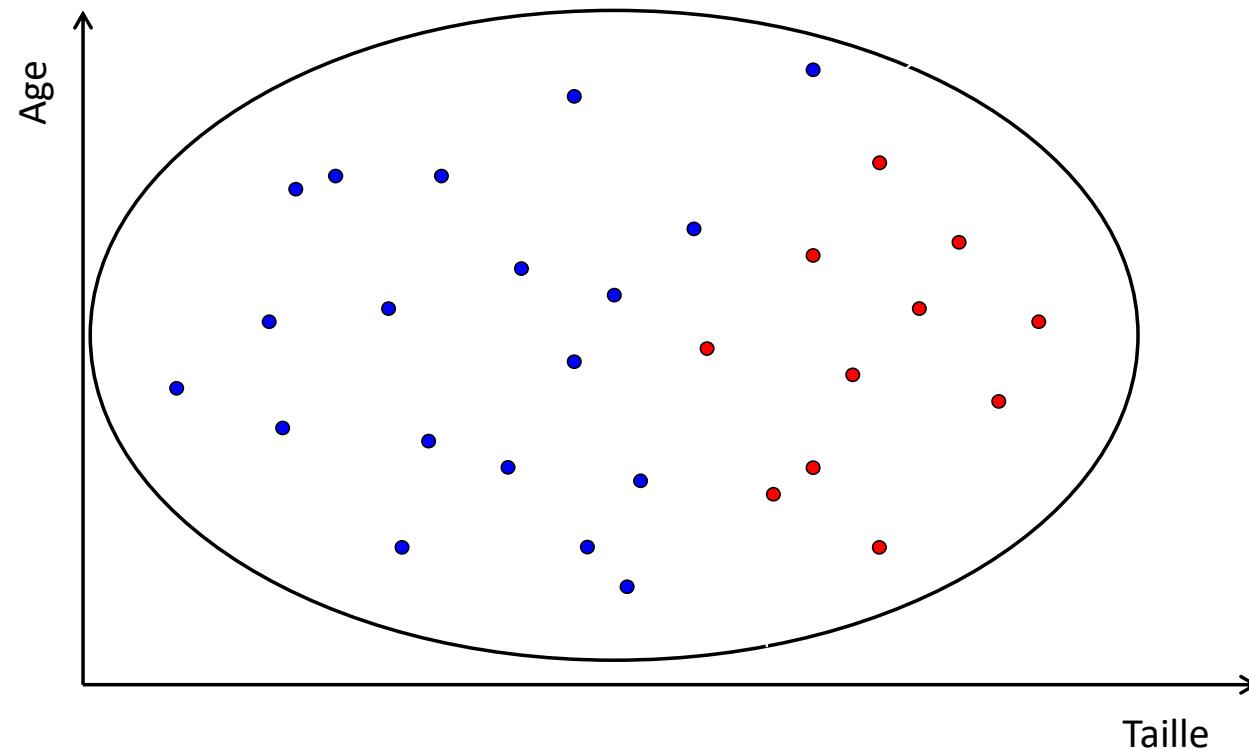
Si $y = 0$, on veut $\theta^T x \ll 0$



Problème de classification

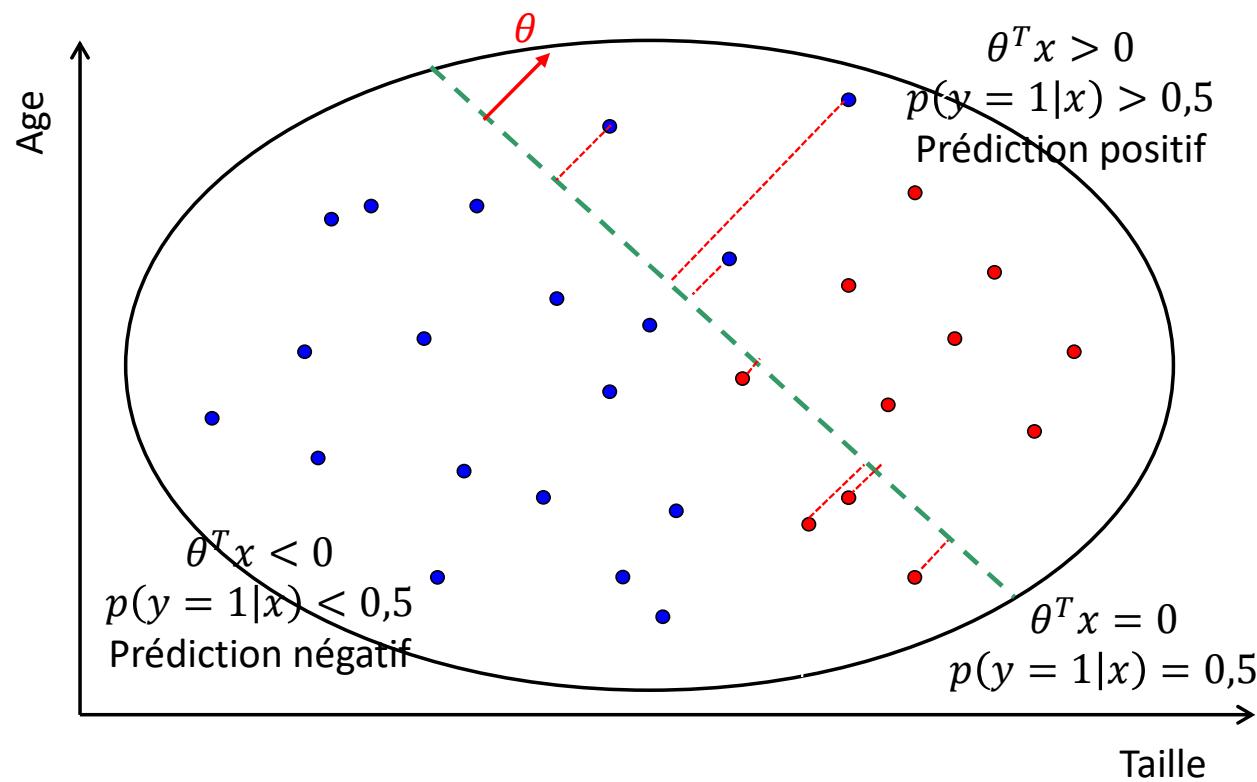


Base d'apprentissage

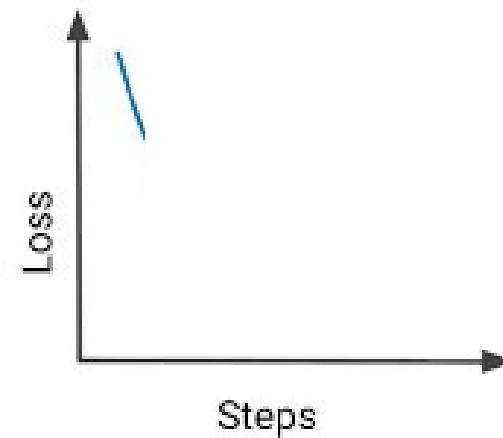


- Base d'apprentissage $S=\{X_i, Y_i\}$ de N exemples

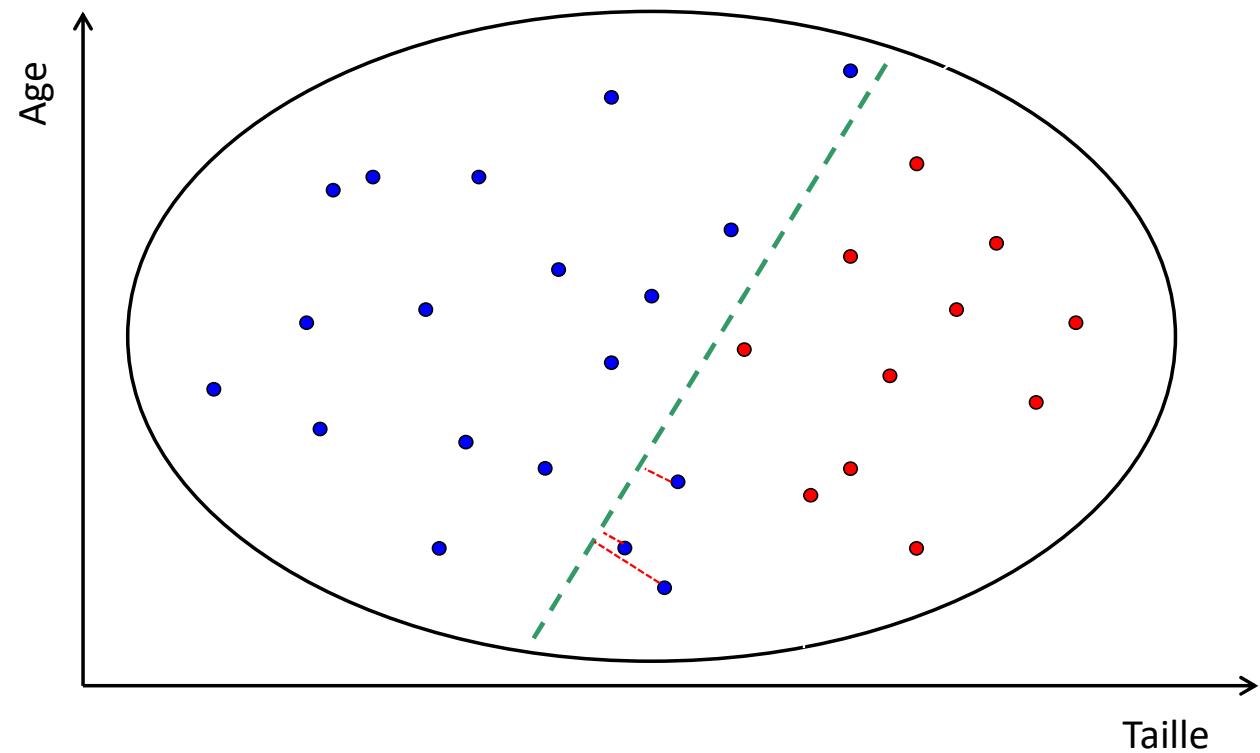
Construction d'un classeur



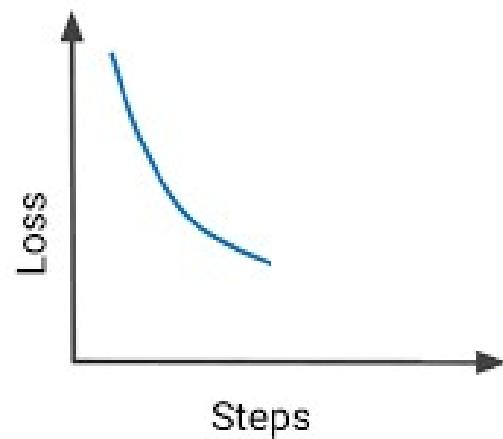
- Base d'apprentissage $S=\{X_i, Y_i\}$ de N exemples
- Définition d'un classeur
- Apprentissage du classeur en minimisant une fonction de cout



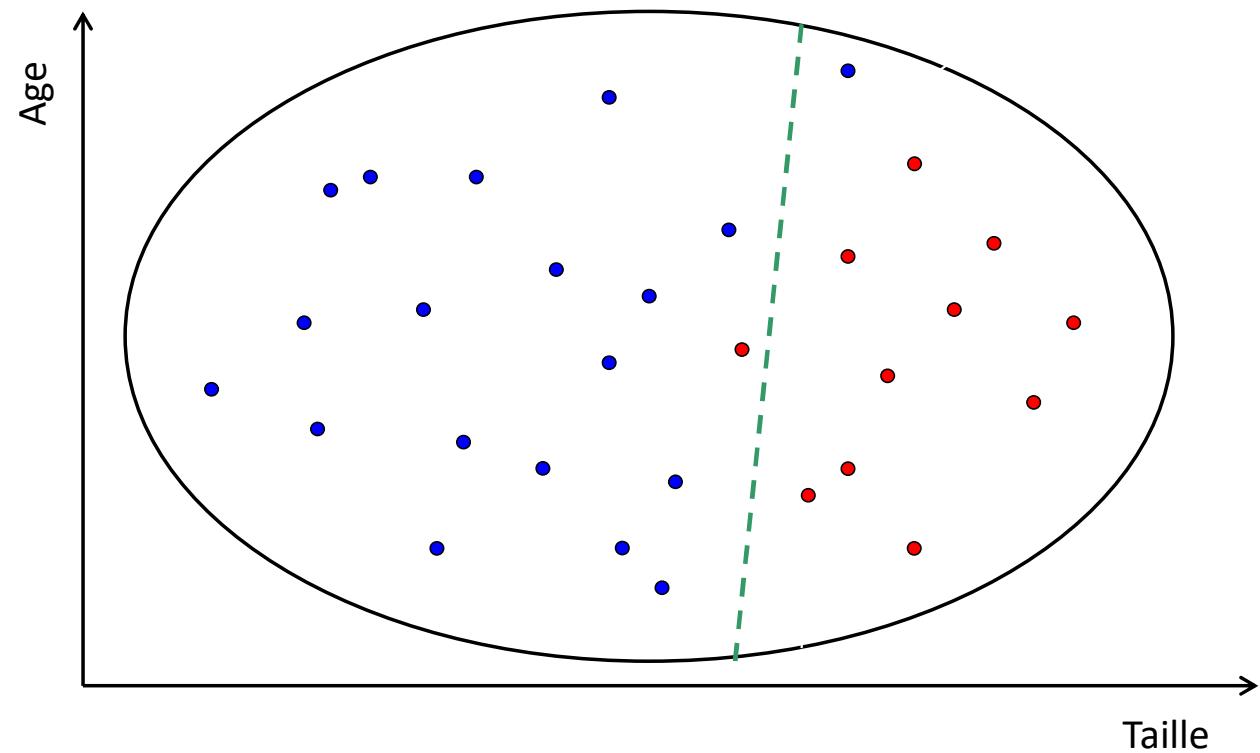
Construction d'un classeur



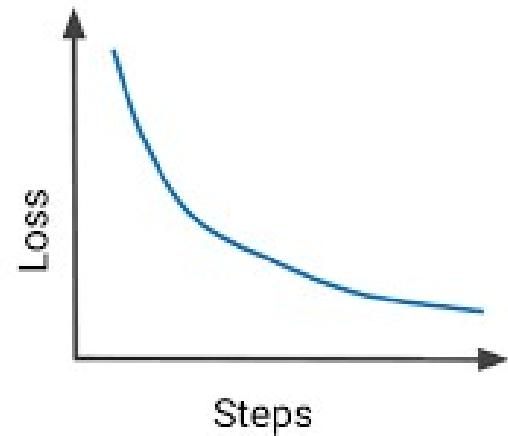
- Base d'apprentissage $S=\{X_i, Y_i\}$ de N exemples
- Définition d'un classeur
- Apprentissage du classeur en minimisant une fonction de cout



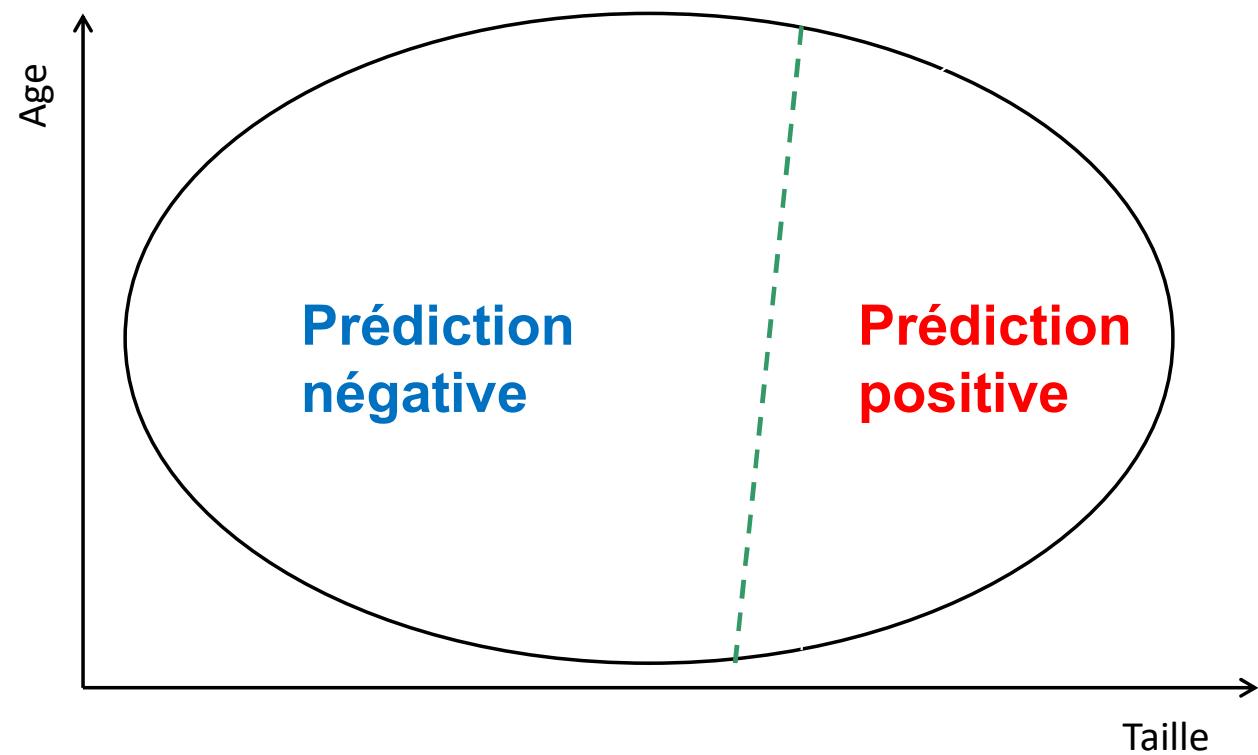
Construction d'un classeur



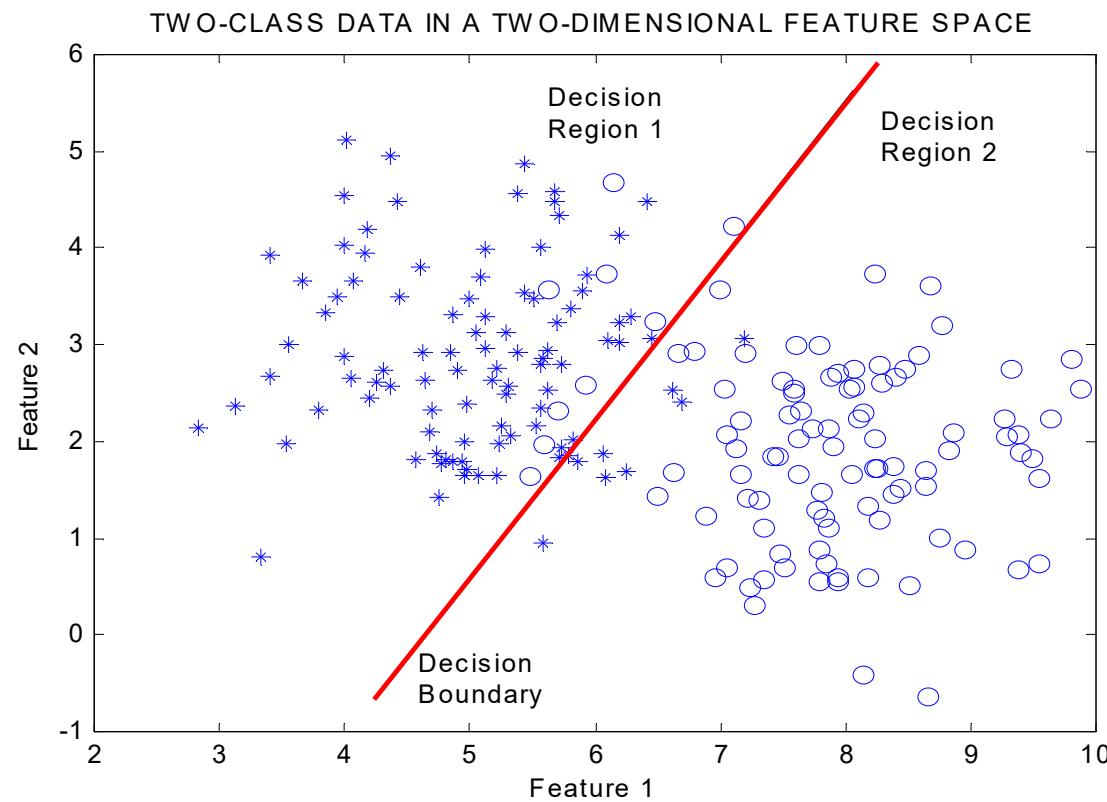
- Base d'apprentissage $S=\{X_i, Y_i\}$ de N exemples
- Définition d'un classeur
- Apprentissage du classeur en minimisant une fonction de cout



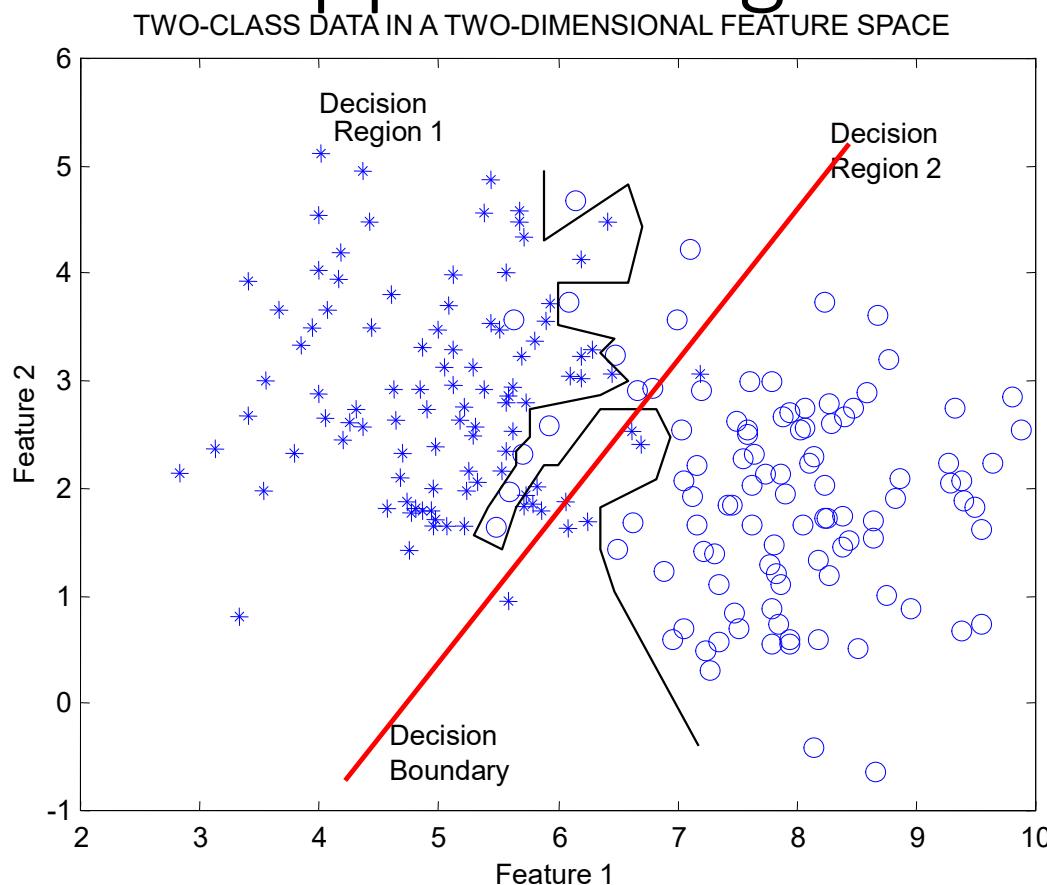
Construction d'un classeur



Exemple de Sur-apprentissage



Exemple de Sur-apprentissage



K Plus Proches Voisins (KPPV)

K Nearest Neighbors(KNN)

K-plus proche voisins

(K-nearest neighbors - KNN)

- **Apprendre par analogie**

Recherchant d'un ou des cas similaires déjà résolus

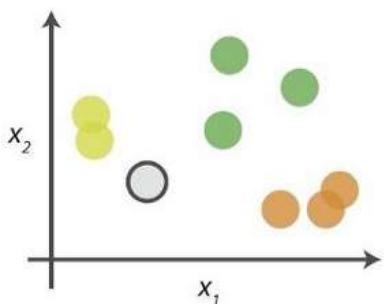
“Dis moi qui sont tes amis, et je te dirais qui tu es”

- **Pas d'apprentissage de modèle**

- Base d'apprentissage
- Fonction de distance
- Fonction d'agrégation

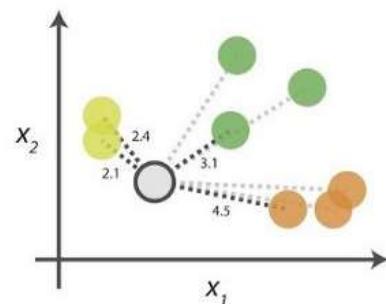
K-plus proche voisins

0. Look at the data



Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances



Start by calculating the distances between the grey point and all other points.

2. Find neighbours

Point	Distance	Nearest Neighbour
lime green	2.1	1st NN
green	2.4	2nd NN
orange	3.1	3rd NN
orange	4.5	4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

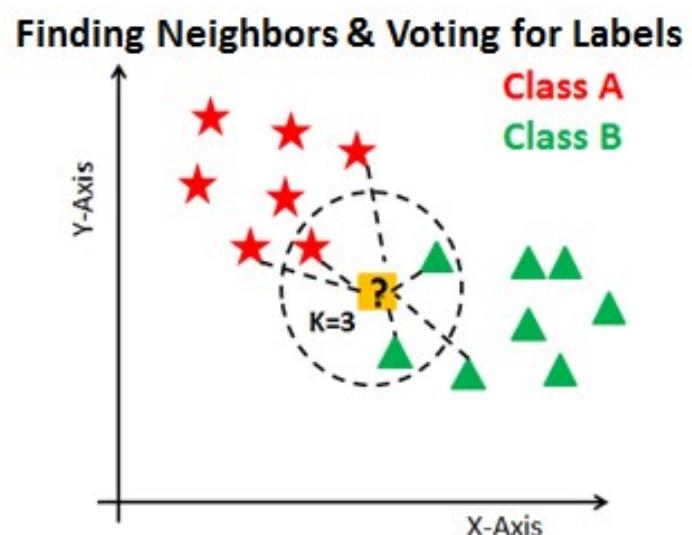
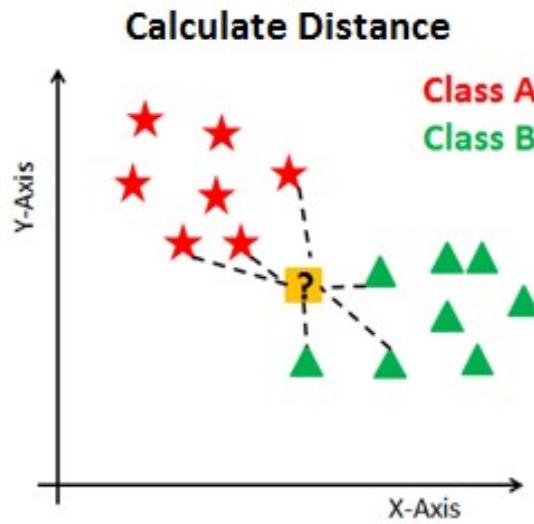
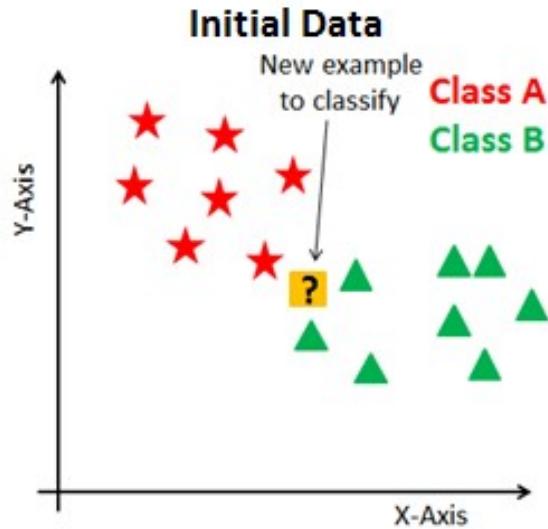
3. Vote on labels

Class	# of votes
lime green	2
green	1
orange	1

Class lime green wins the vote!
Point is therefore predicted to be of class lime green.

Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

[images from towardsdatascience.com]



Choix de la distance :

- Minkowski distance

$$d(i, j) = \sqrt[q]{|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \dots + |x_{ip} - x_{jp}|^q}$$

1st dimension 2nd dimension pth dimension

- Euclidean distance

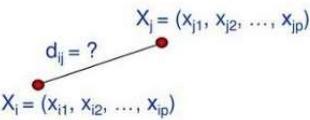
$$q = 2$$

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2}$$

- Manhattan distance

$$q = 1$$

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

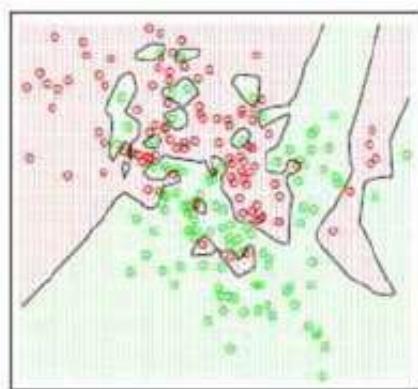
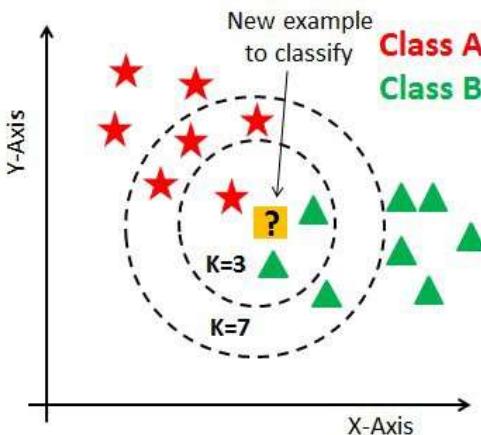


Fonction d'agrégation :

- Vote majoritaire (classification)
- Moyenne (régression)

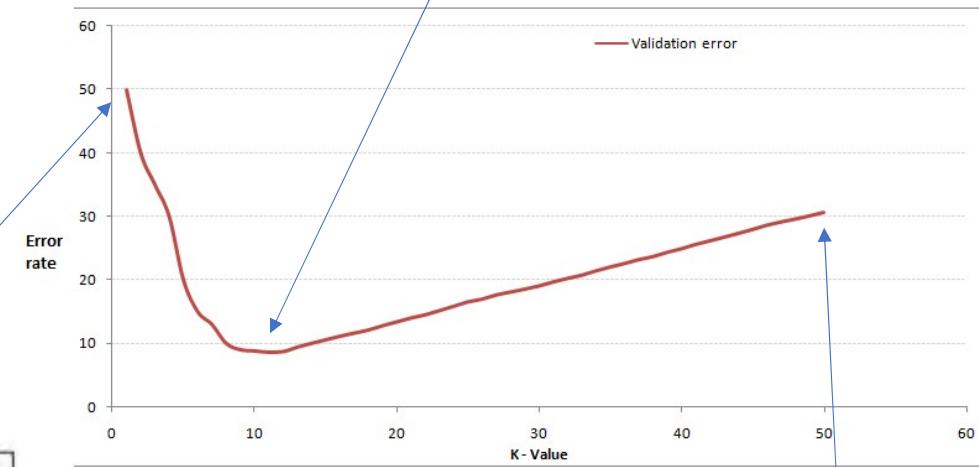
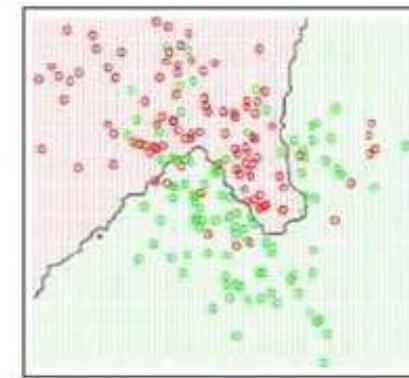
Comment choisir K ?

Les prédictions dépendent beaucoup de K



K plus grand

- Séparation plus lisse
- Variance réduite



K petit

- Séparation complexe
- Grande variance
- Risque de sur-apprentissage

K trop grand

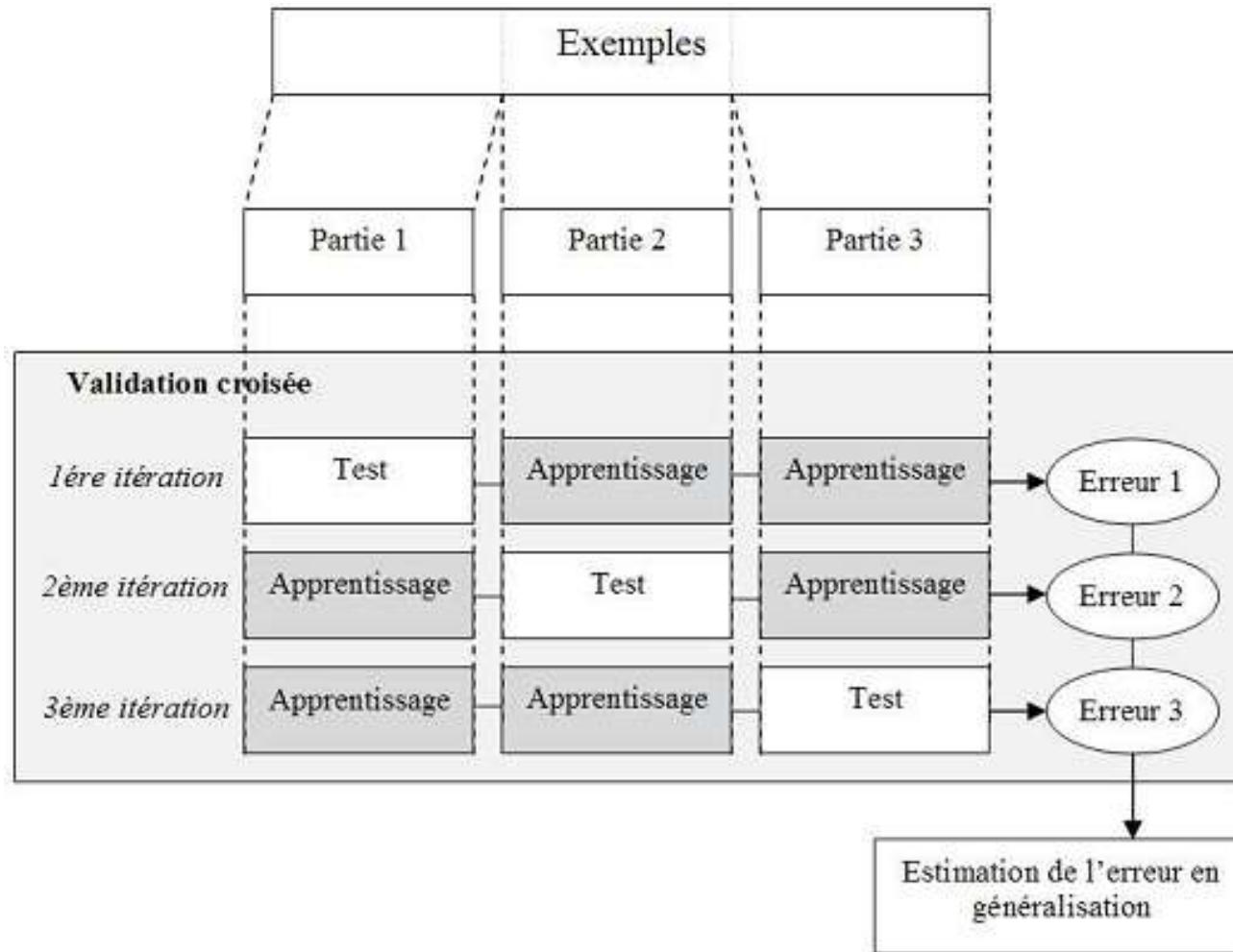
- Prédiction de la classe majoritaire

Optimisation des hyper-paramètres

On a besoin de trois ensembles d'exemples :

- Ensemble Apprentissage: Apprendre le classeur
- Ensemble Validation: Ajuster les hyper-paramètres
- Ensemble Test: Evaluer le classeur

Validation croisée

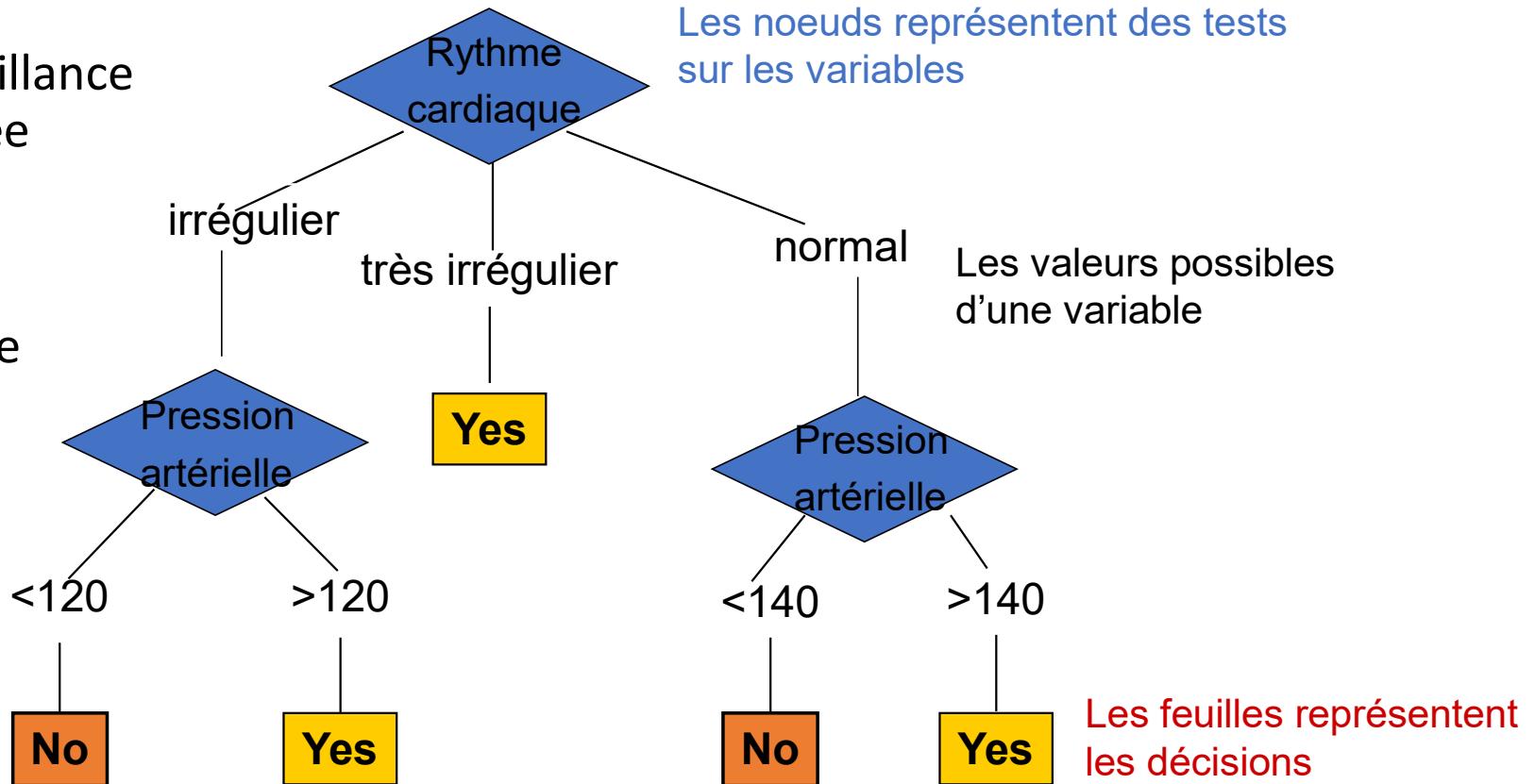


Arbres de décision

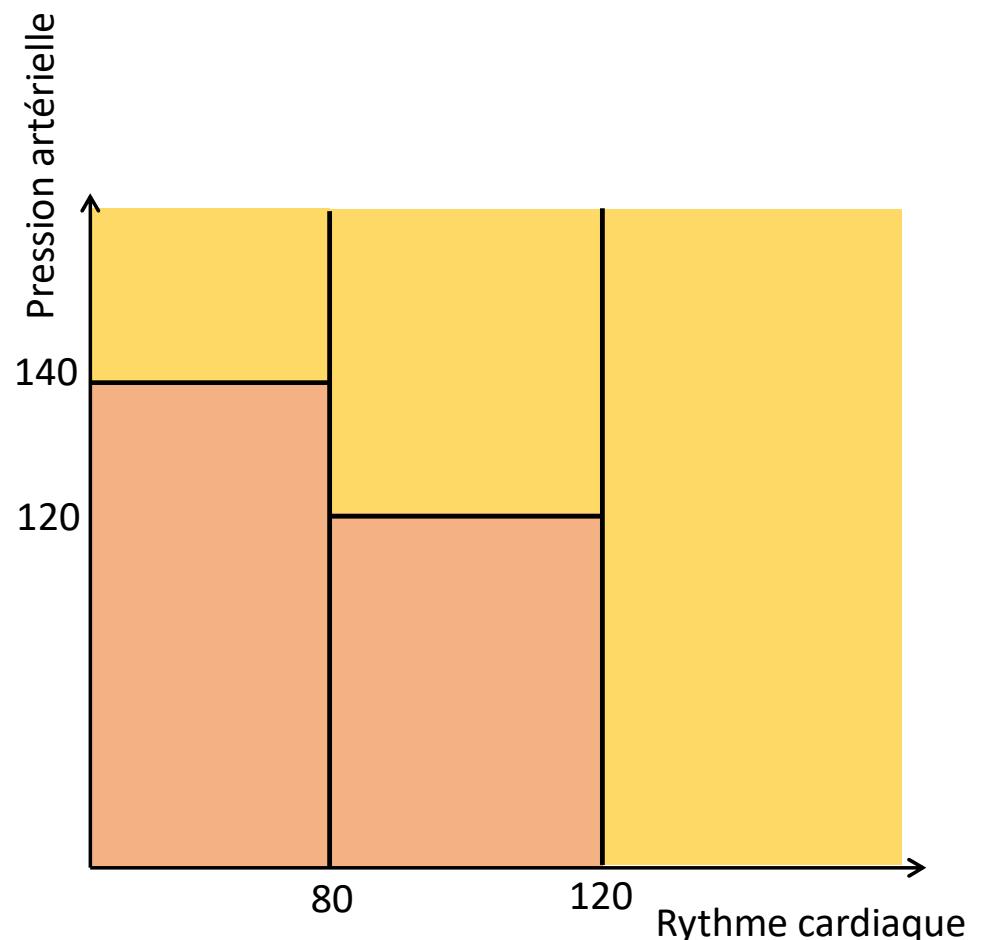
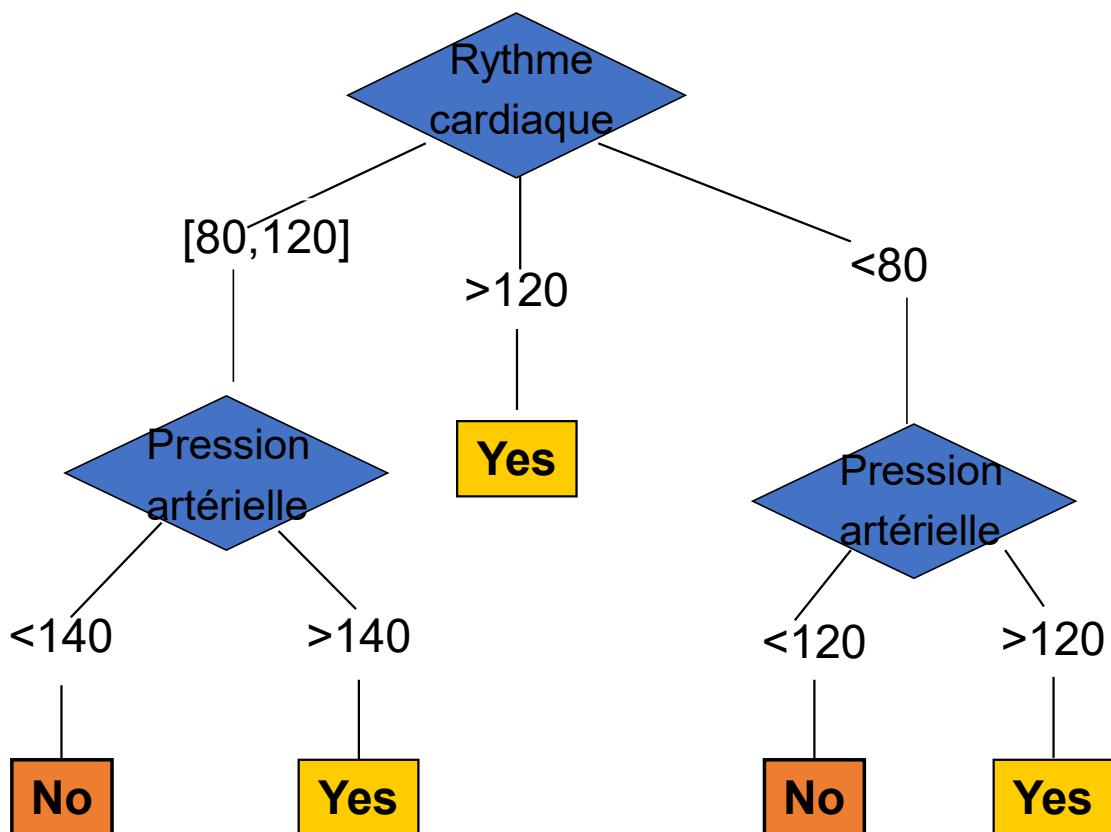
Arbres de décision

Prédire une surveillance médicale renforcée

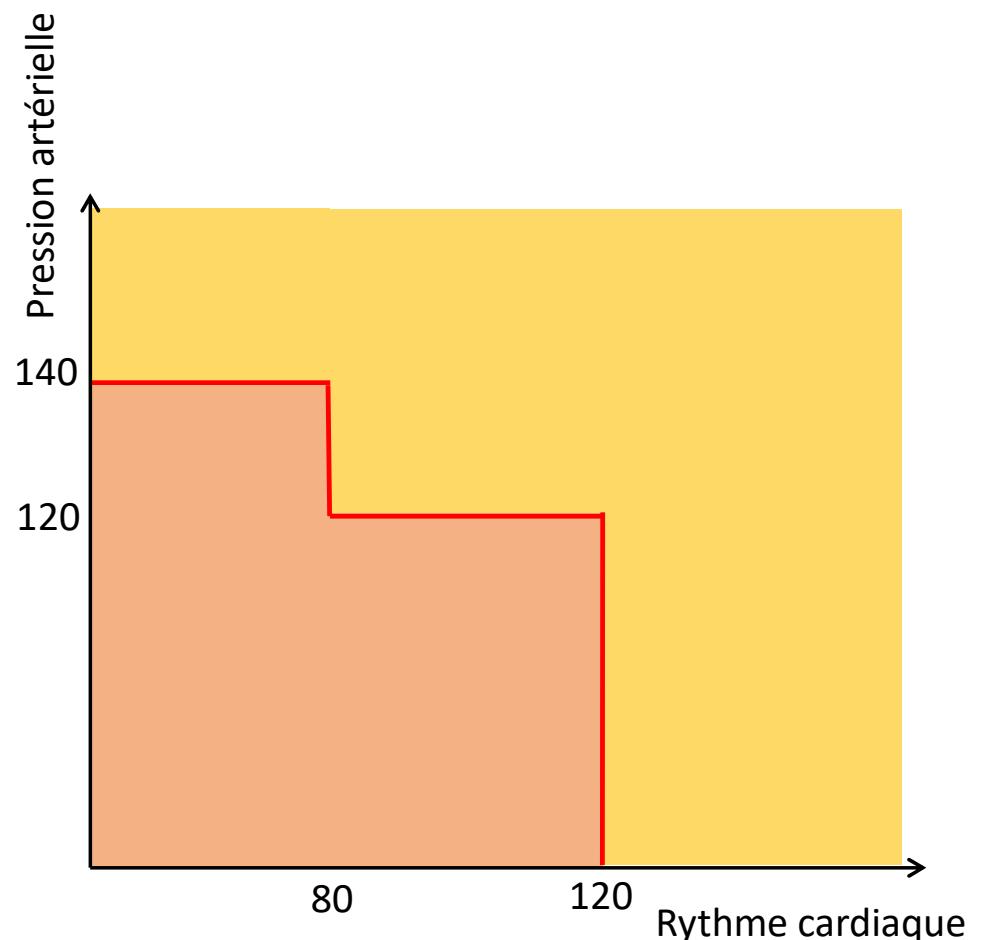
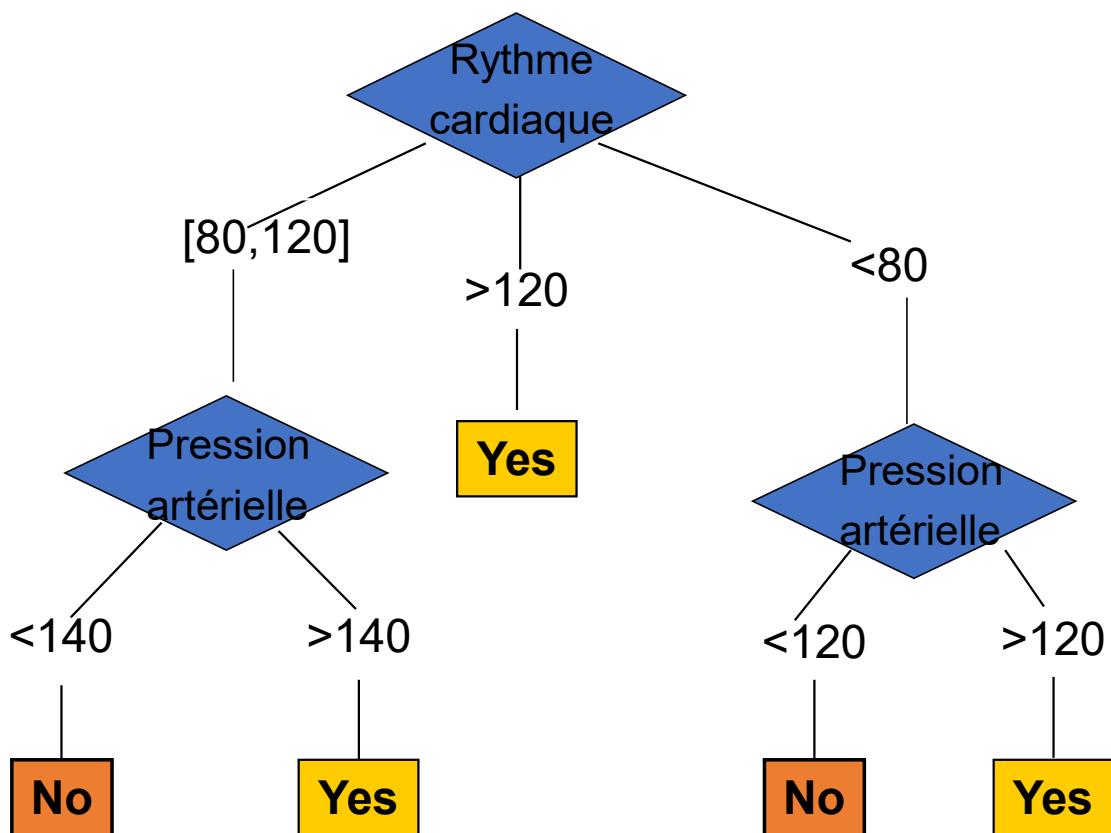
Modèle prédictif sous forme d'arbre



Arbres de décision



Arbres de décision



Construction de l' arbre de décision

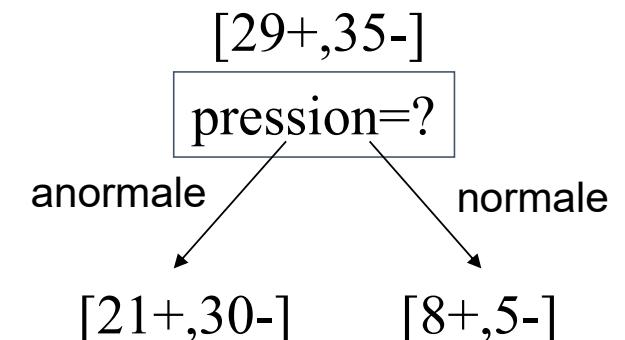
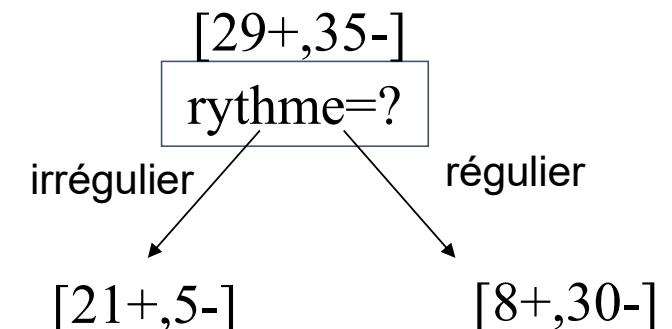
- 1) Choisir la “meilleure” variable
- 2) Diviser l’ensemble d’apprentissage suivant les valeurs de l’attribut choisi
- 3) Répéter les étapes 1 et 2 de manière récursive jusqu’à ce que tous les objets soient correctement classés.

Patient	Rythme cardiaque	Pression artérielle	Classe
1	irrégulier	normale	Malade
2	régulier	normale	En forme
3	irrégulier	anormale	Malade
4	irrégulier	normale	Malade
5	régulier	normale	En forme
6	régulier	anormale	Malade
7	régulier	normale	En forme
8	régulier	normale	En forme

Comment choisir la meilleure variable ?

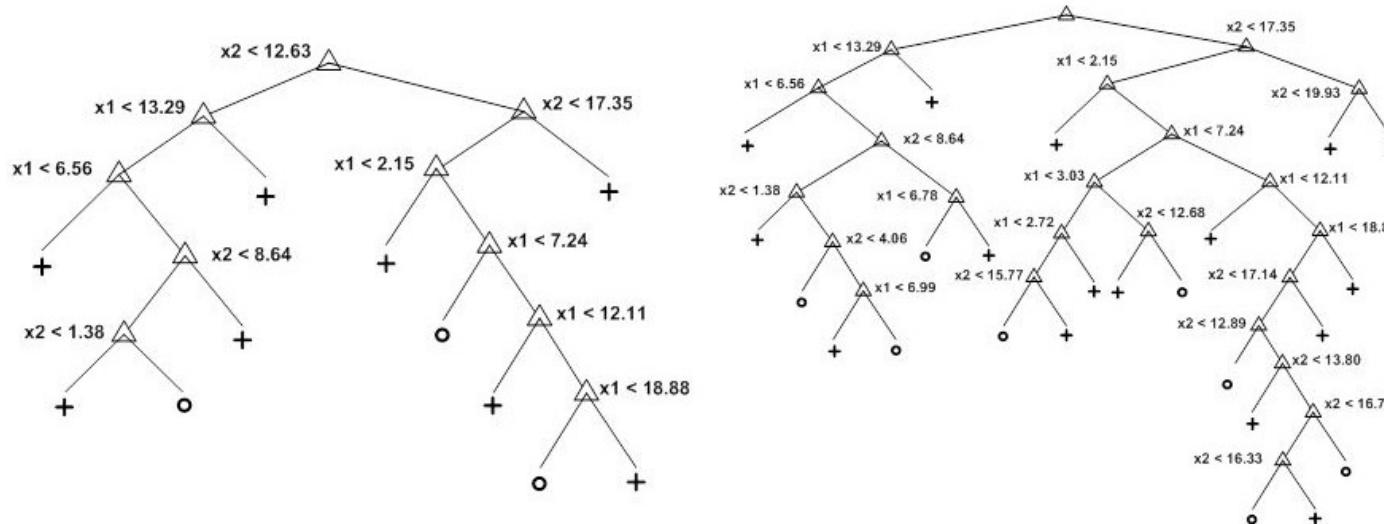
- Obtenir un petit arbre :
 - Maximiser la séparation des classes à chaque étape
Rendre les ensembles “successeurs” aussi pures que possible
- Mesure de pureté :
 - Indice de Gini
 - Gain d’information
 - Test Chi-square

Trouver une variable pour séparer 29 exemple + de 35 exemple -



Sur-apprentissage

- On multipliant les nœuds on risque de faire du sur-apprentissage
- Pallier au sur-apprentissage
 - Stopper la croissance de l'arbre quand la baisse de la fonction de cout devient faible
 - Construire l'arbre entier, puis élaguer les branches les moins importantes



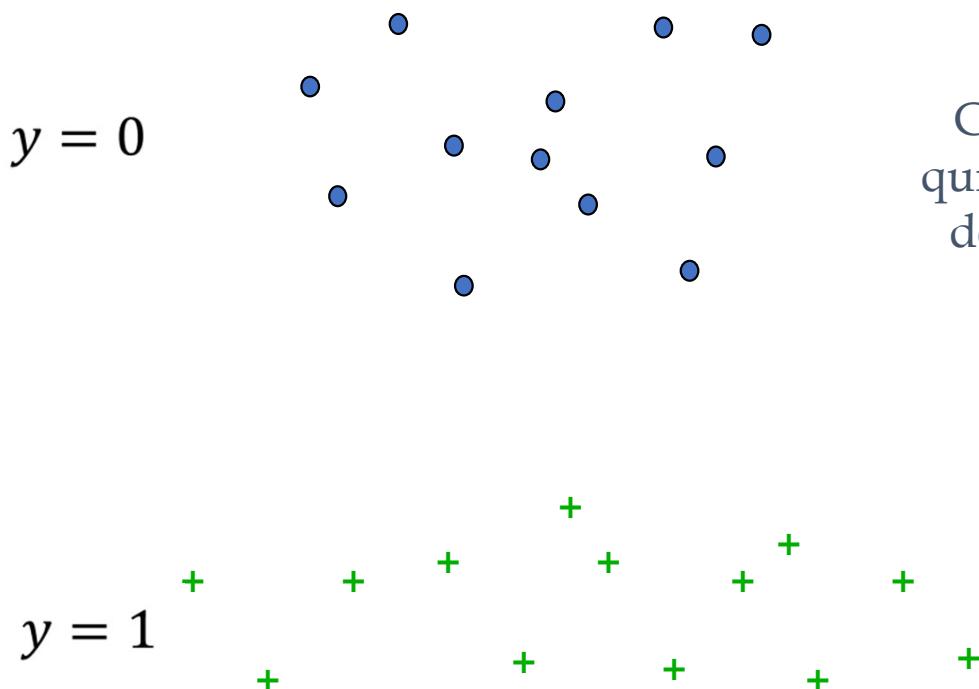
Avantages et limitations

- + Modèle facilement interprétable
- + Utilisable aussi pour la régression
- Sensible au bruit : modèle et performances peu robustes
- Performance moins bonne que les autres méthodes
 - L'arbre construit suivant des nœuds à un attribut approche par des frontières parallèles aux axes

Machine à vecteurs de support

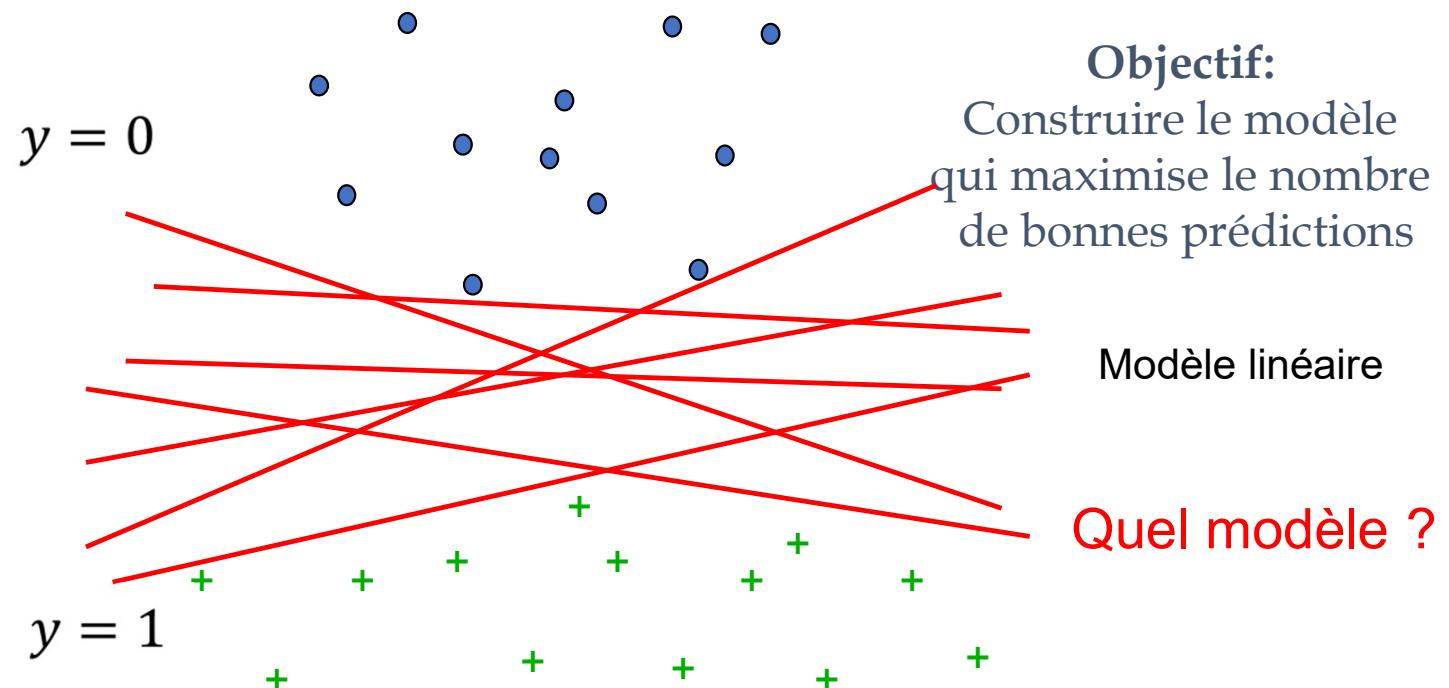
Support Vector Machine (SVM)

Problème binaire

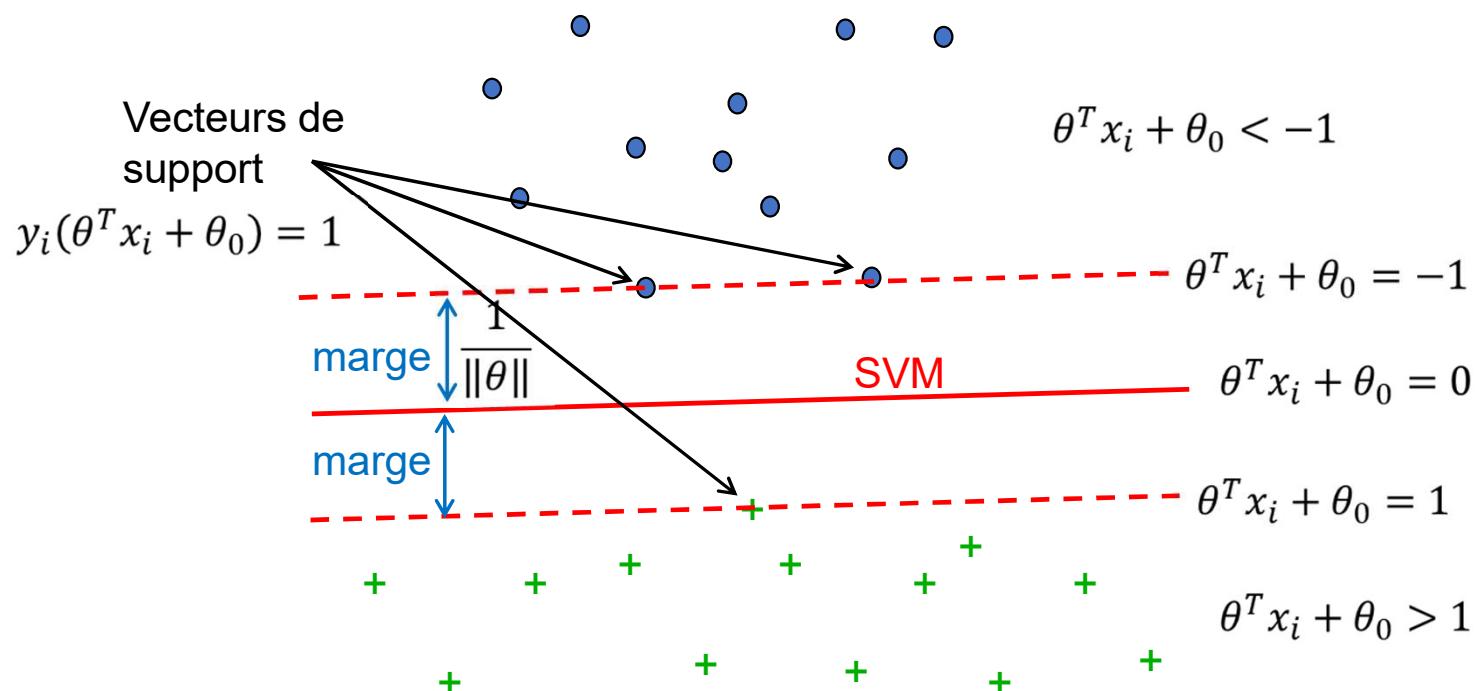


Objectif:
Construire le modèle
qui maximise le nombre
de bonnes prédictions

Problème binaire



SVM : Principe



Formulation SVM

- Soit un problème de classification à deux classes $y = \{-1, +1\}$
- Soit une base d'apprentissage contenant n exemples $\{(x_i, y_i)\} i = 1 \dots n$
- On cherche un modèle linéaire du type

$$h_{\theta}(x) = \theta^T x + \theta_0 \quad \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \quad x = \begin{bmatrix} x_{(1)} \\ \vdots \\ x_{(m)} \end{bmatrix}$$

- Problème de minimisation

$$\min_{\theta, \theta_0} \frac{1}{2} \theta^T \theta$$

s. c. $y_i(\theta^T x_i + \theta_0) \geq 1$

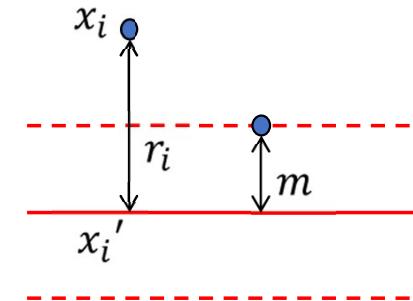
Formulation SVM

$$\min_{\theta, \theta_0} \frac{1}{2} \theta^T \theta$$

$$x_i' = x_i - y_i r_i \frac{\theta}{|\theta|} \quad r_i = y_i \frac{\theta^T x_i + \theta_0}{|\theta|}$$

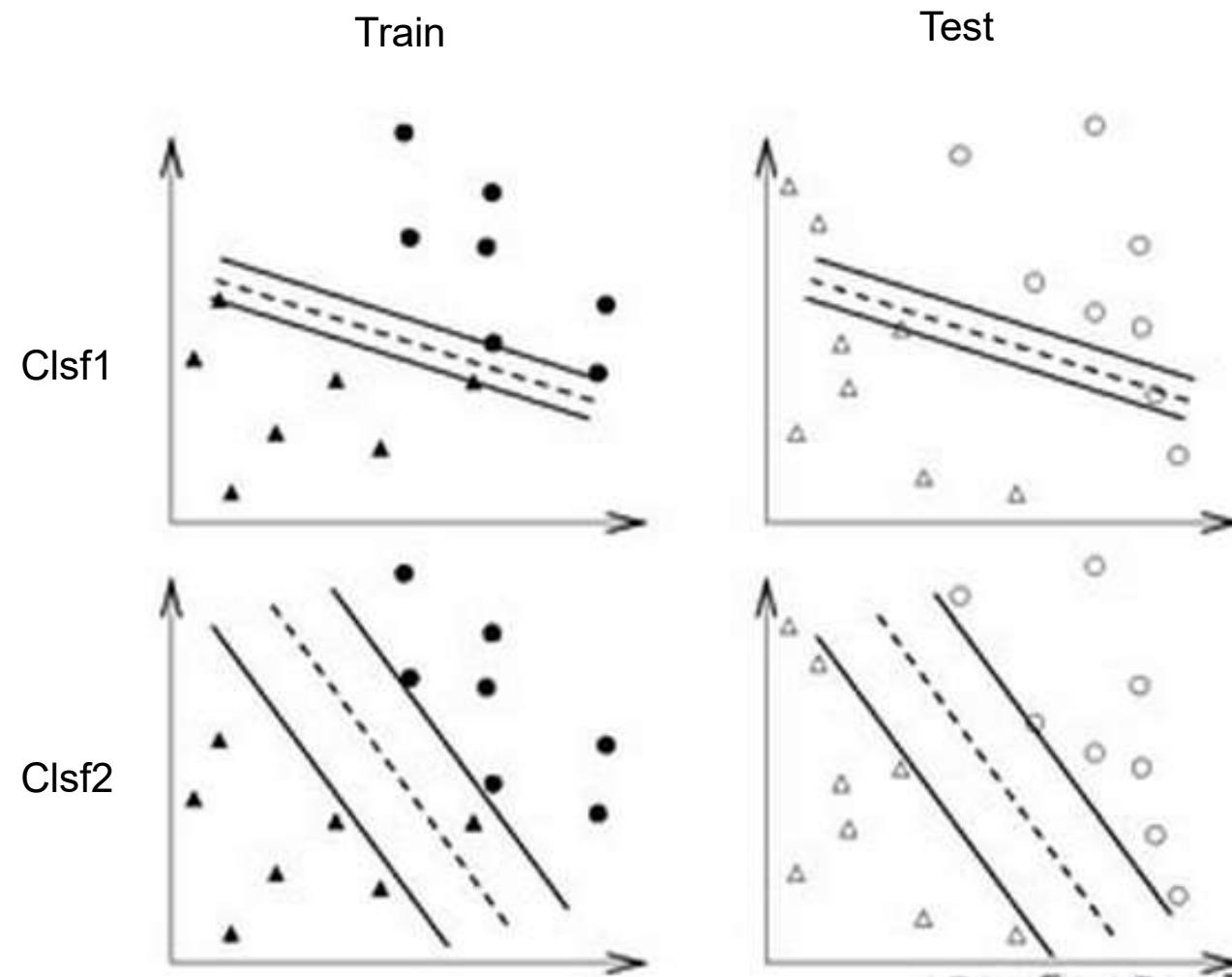
$$m = \frac{1}{|\theta|}$$

$$y_i(\theta^T x_i + \theta_0) \geq 1$$



y_i	$\theta^T x_i + \theta_0$	$y_i(\theta^T x_i + \theta_0)$
+1	+	+
+1	-	-
-1	+	-
-1	-	+

Problème des contraintes dures

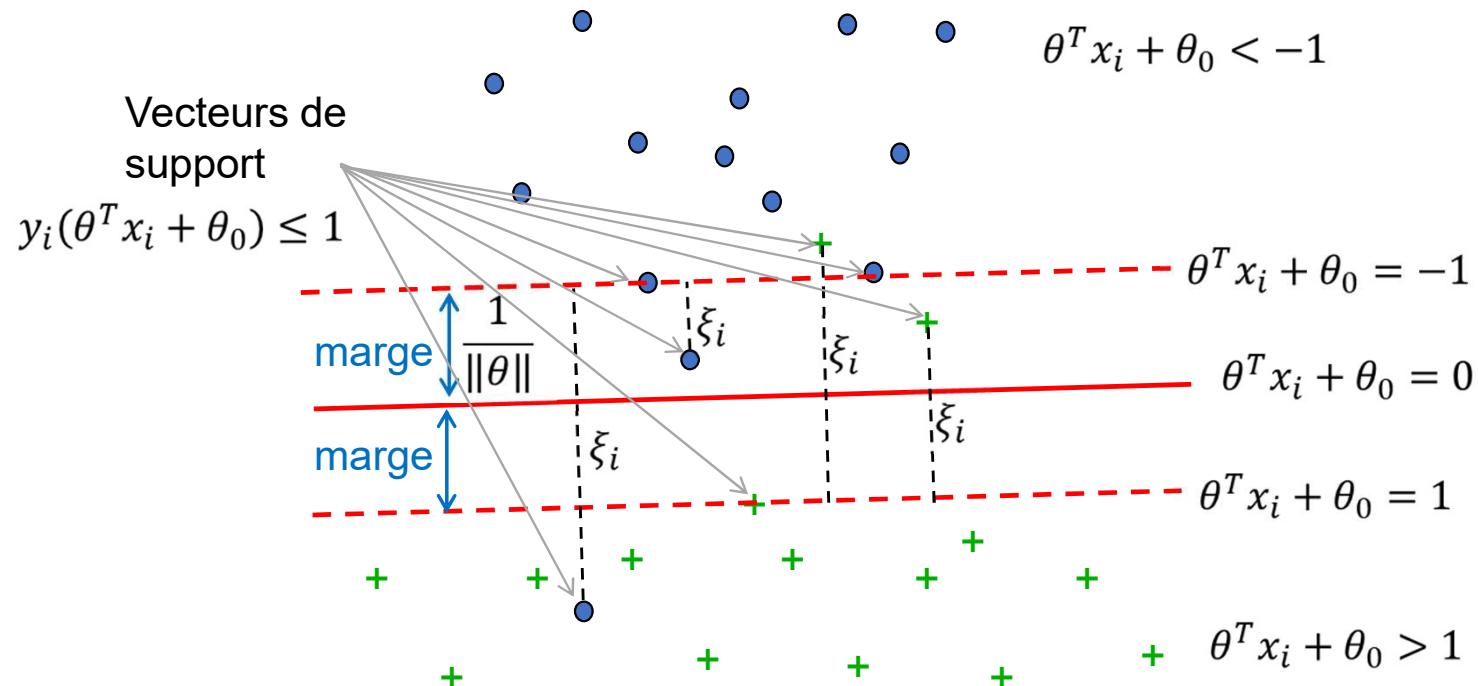


Variables de relâchement

- Il n'est pas toujours possible ou bénéfique de respecter toutes les contraintes
- Ajout de variables de relâchements pour assouplir les contraintes
- C contrôle le compromis en minimisation de la marge et respect des contraintes

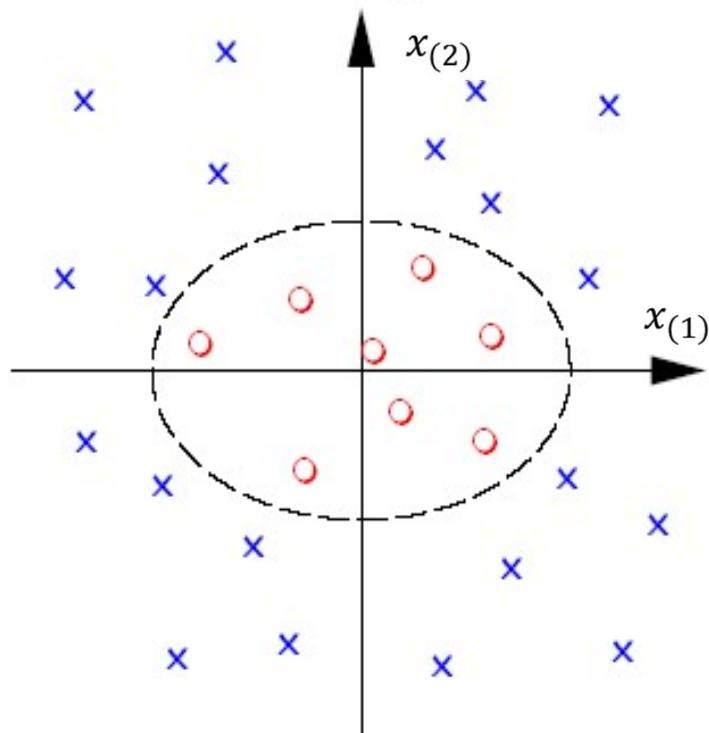
$$\begin{aligned} \min_{\theta, \theta_0, \xi_j} \quad & \frac{1}{2} \theta^T \theta + C \sum_{i=1}^n \xi_i \\ \text{s. c.} \quad & \begin{cases} y_i(\theta^T x_i + \theta_0) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \end{aligned}$$

Interprétation SVM



Non linéairement séparable

Problème non linéairement séparable

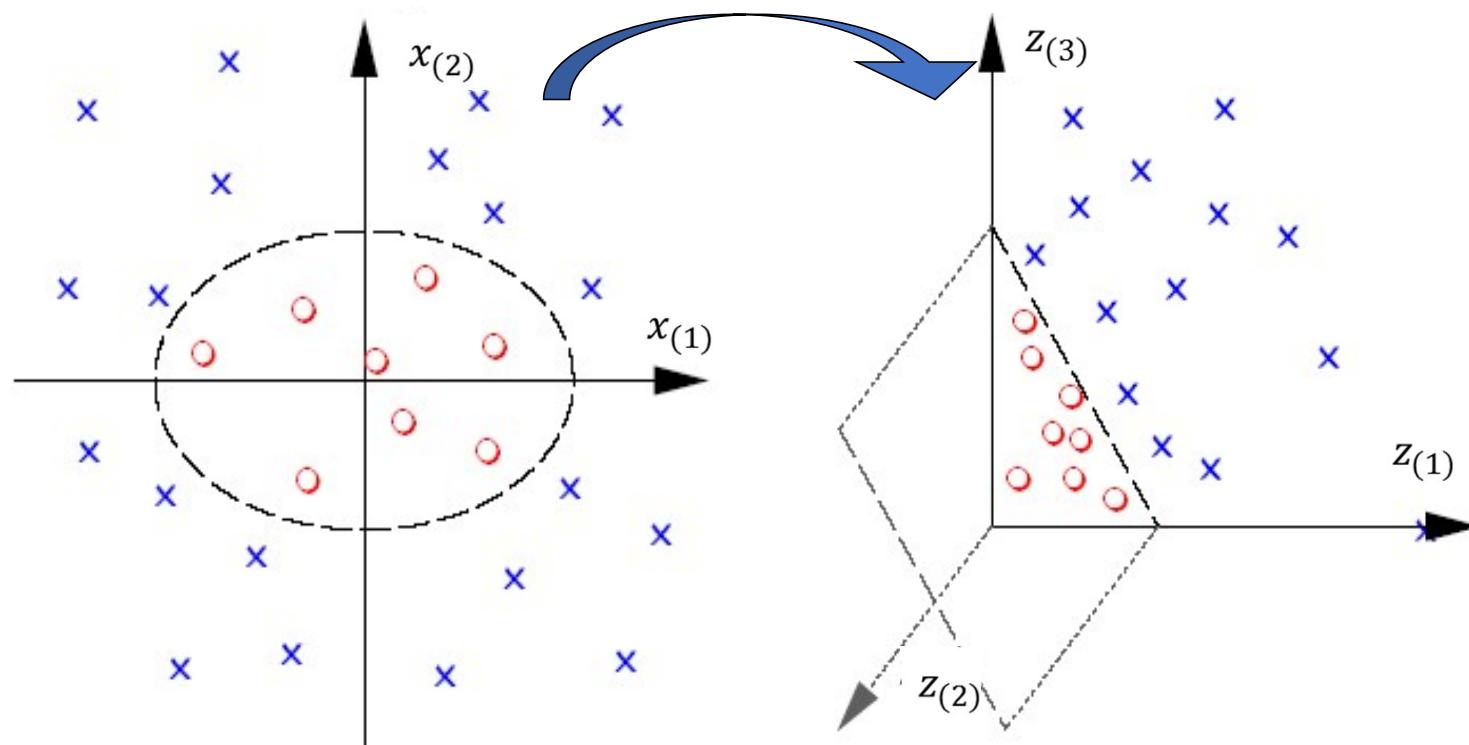


Projeter les données dans un espace de dimension supérieure dans lequel le problème est linéairement séparable

Projection du problème

$$\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(x_{(1)}, x_{(2)}) \rightarrow (z_{(1)}, z_{(2)}, z_{(3)}) = (x_{(1)}^2, \sqrt{2}x_{(1)}x_{(2)}, x_{(2)}^2)$$



Formulation standard

- Problème de minimisation avec contraintes :

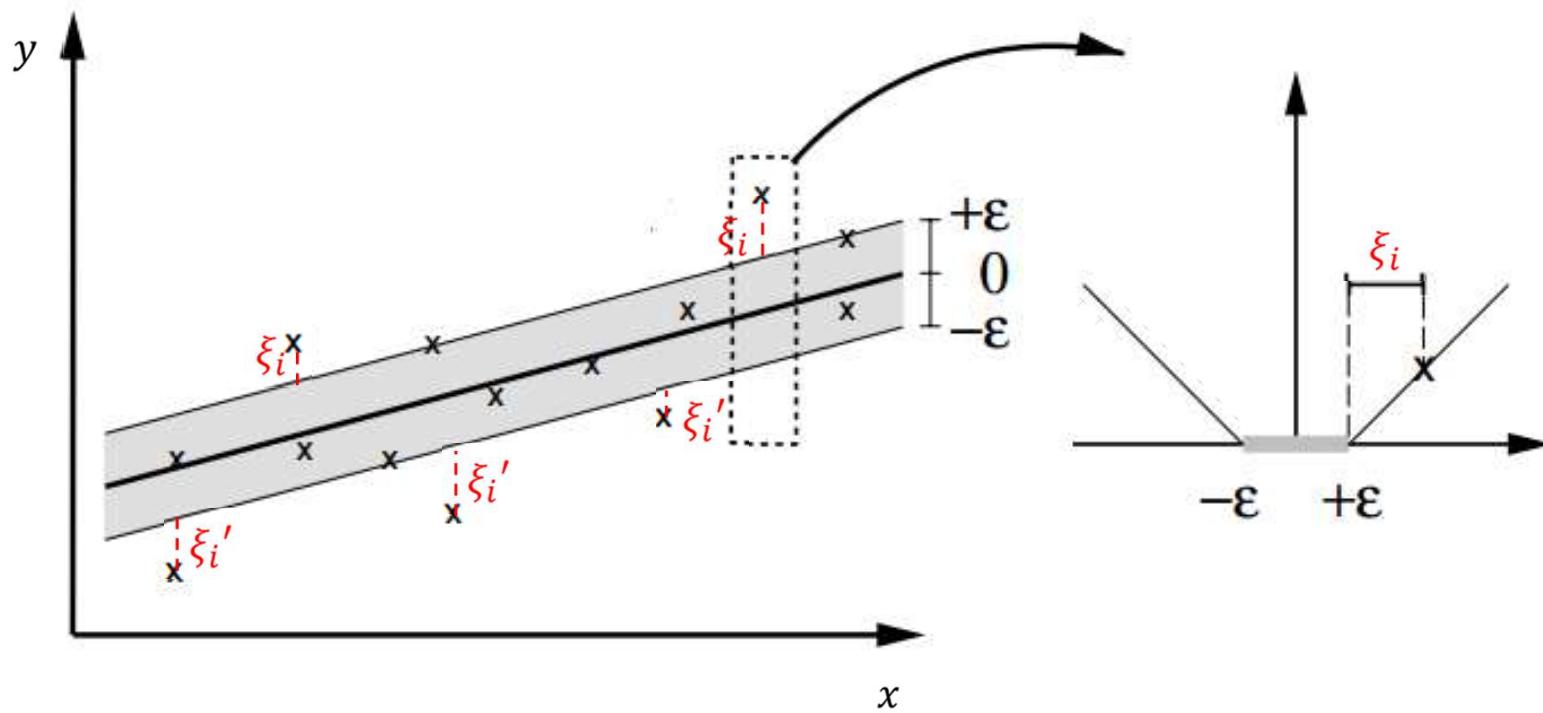
$$\begin{aligned} \min_{\theta, \theta_0, \xi_j} \quad & \frac{1}{2} \theta^T \theta + C \sum_{i=1}^n \xi_i \\ \text{s. c. } \quad & \begin{cases} y_i (\theta^T \phi(\mathbf{x}_i) + \theta_0) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \end{aligned}$$

- Fonction de décision :

$$h_\theta(x_j) = \sum_{i=1}^n \alpha_i y_i K(x_j, x_i) + \theta_0$$

$$G(x) = sign(h_\theta(x)) = sign\left(\sum_{i=1}^n \alpha_i y_i K(x_j, x_i) + \theta_0\right)$$

Support Vector Regression



Méthodes d'ensembles

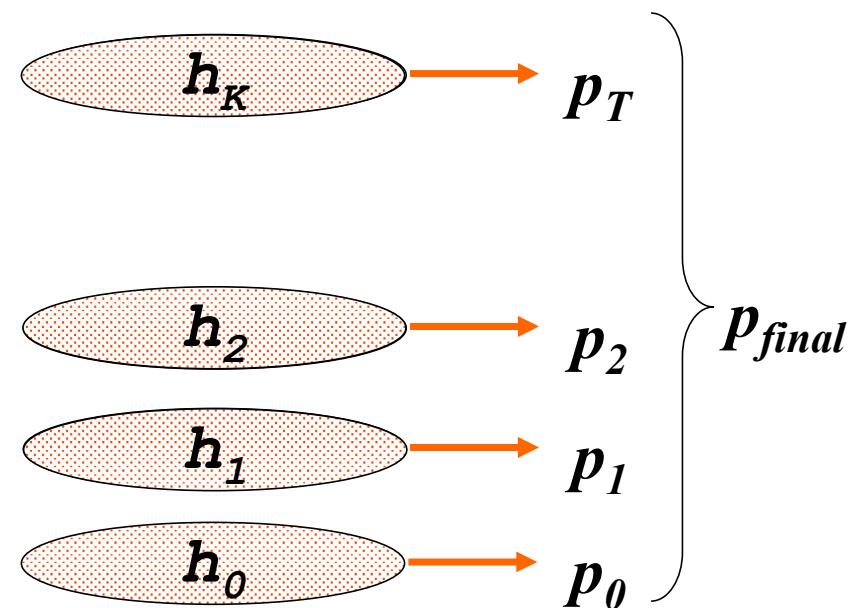
Forêt aléatoire

Boosting

Méthodes d'ensemble

Idée :

- Un ensemble d'avis est plus fiable qu'un avis individuel
- La majorité fait moins d'erreur qu'un individu
- Utiliser un ensemble de classeurs
- Agréger leur prédiction



Méthodes d'ensemble

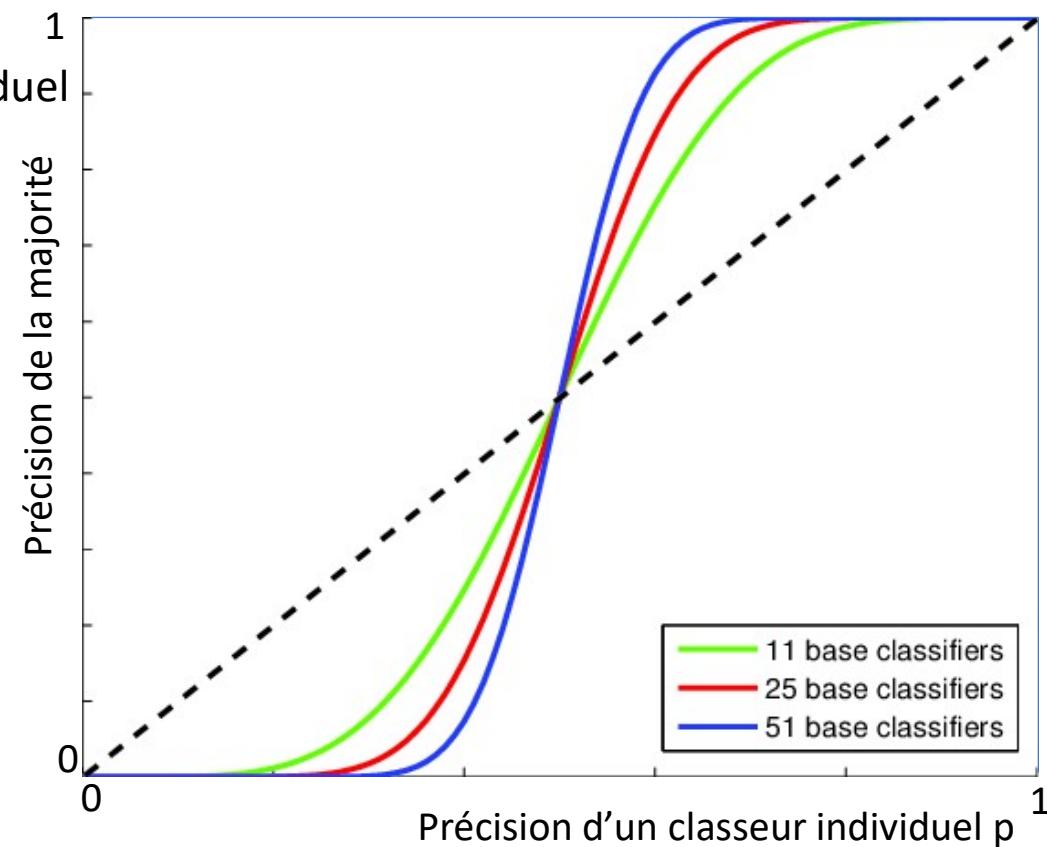
Idée :

- Un ensemble d'avis est plus fiable qu'un avis individuel
- La majorité fait moins d'erreur qu'un individu
- Utiliser un ensemble de classeurs
- Agréger leur prédiction

Conditions :

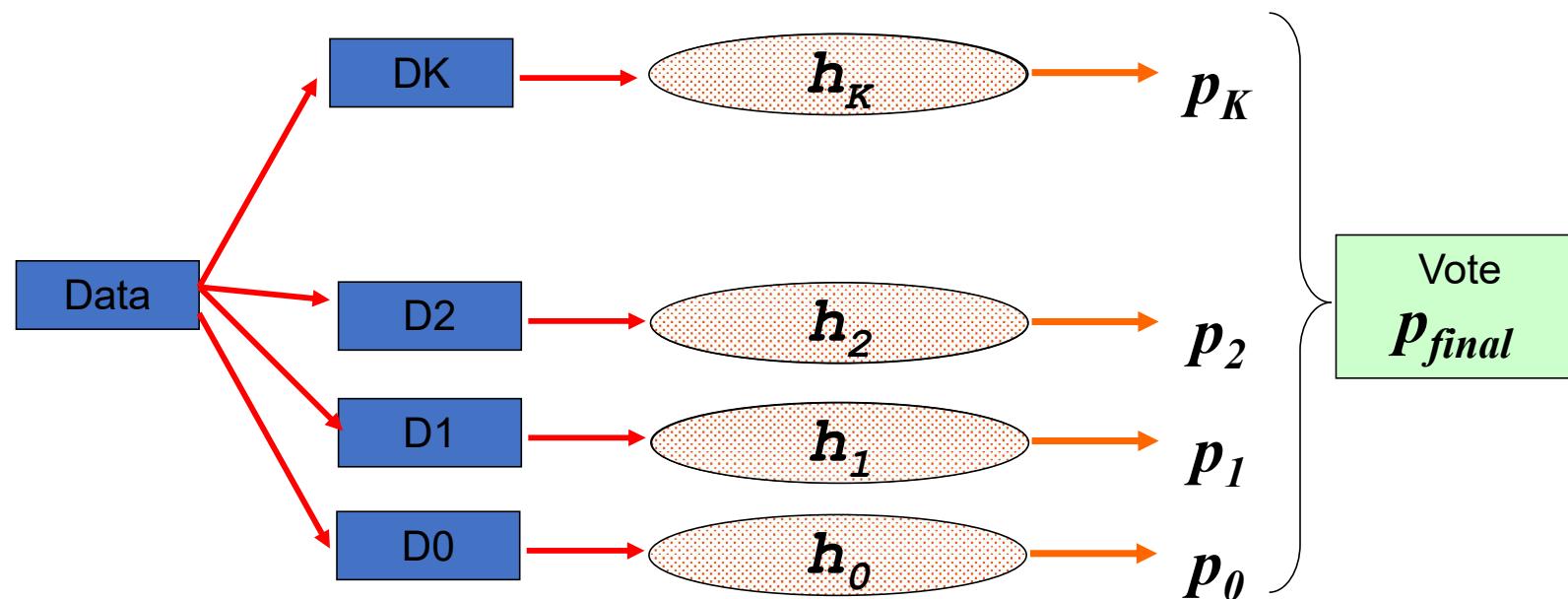
- Les classeurs doivent être meilleur que le hasard
- Les classeurs doivent être **indépendants**

Soit K classeurs indépendants de précision p

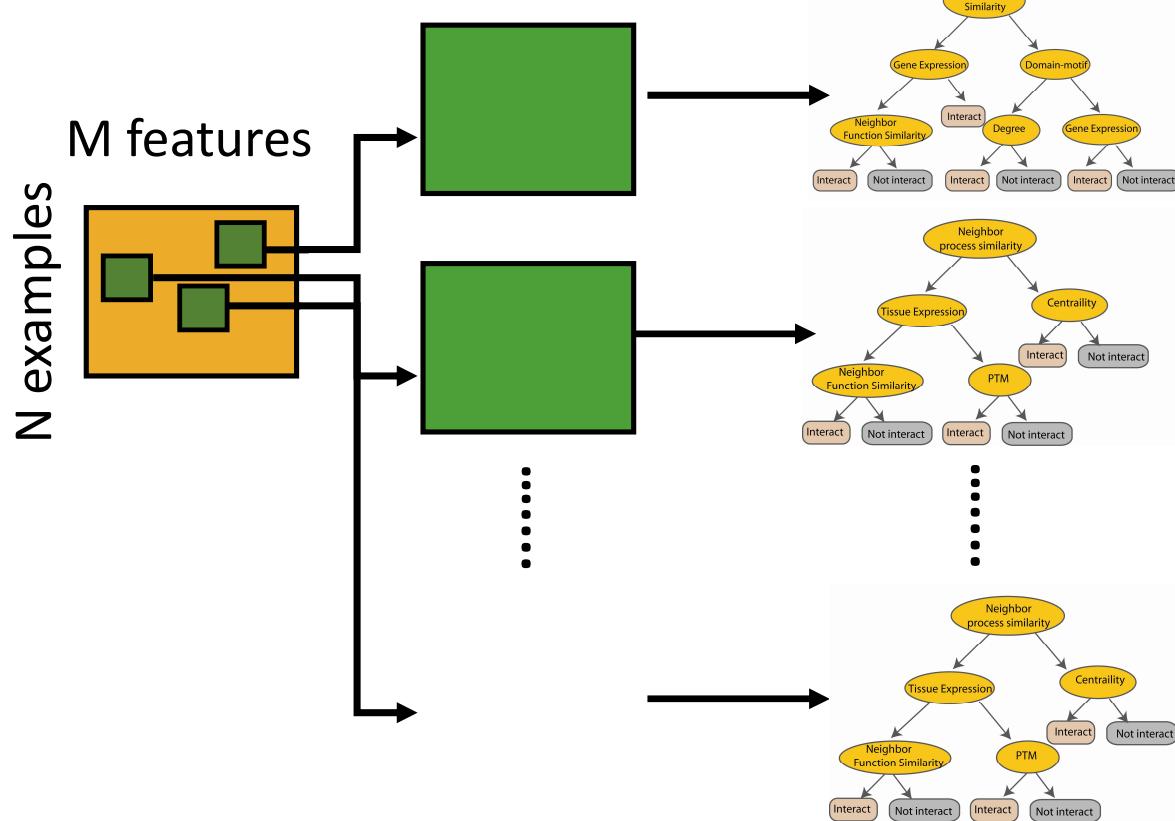


Méthodes d'ensemble

- Comment obtenir des classeurs indépendants ?
- Echantillonner une sous-ensemble d'exemples
- Projection aléatoire : sous ensemble de variables



Forêt aléatoire / random forest



Jeux de données contenant un ensemble d'exemples E_k et un ensemble de variables F_k

Pour $k = 1$ à K

- $E_k \leftarrow$ sous ensemble d'exemples
- $F_k \leftarrow$ sous ensemble d'exemples
- $h_k \leftarrow$ apprendre un modèle basé sur E_k et F_k

Classification :

$$h(x) = \text{majority}\{h_1(x), \dots, h_k(x)\}$$

Régression :

$$h(x) = \text{average}\{h_1(x), \dots, h_k(x)\}$$

[Image from Oznur Tastan Machine Learning 10601]

Boosting

- Les classeurs sont construits de manière séquentielle et dépendent des classeurs précédents
- Idée : On se concentre sur les exemples mal prédits par les classeurs précédents
 - Apprentissage avec une base d'apprentissage pondérés : **poids sur les exemples w_i**
 - Augmente le poids des exemples mal prédits
 - Diminue le poids des exemples bien prédits
- Calcule un vote majoritaire pondéré
 - Les classeurs les plus performants ont plus de poids
 - **poids sur les classeurs α_t**

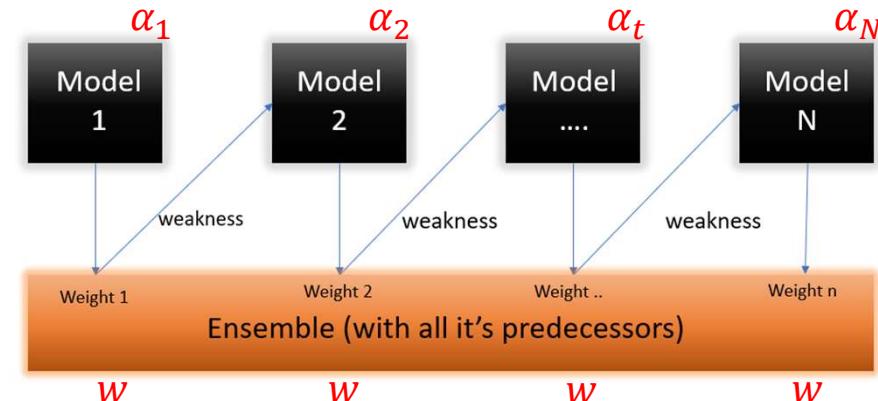
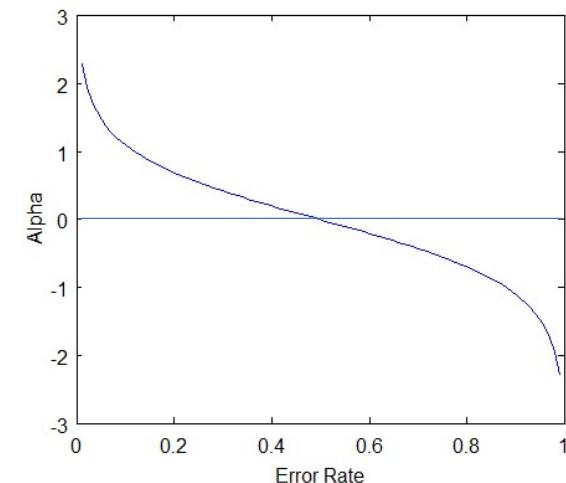


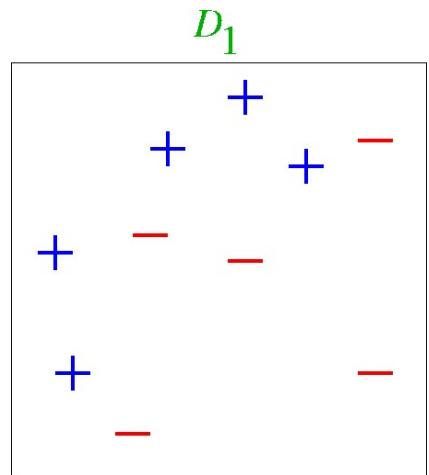
Image from analyticsvidhya.com

Boosting

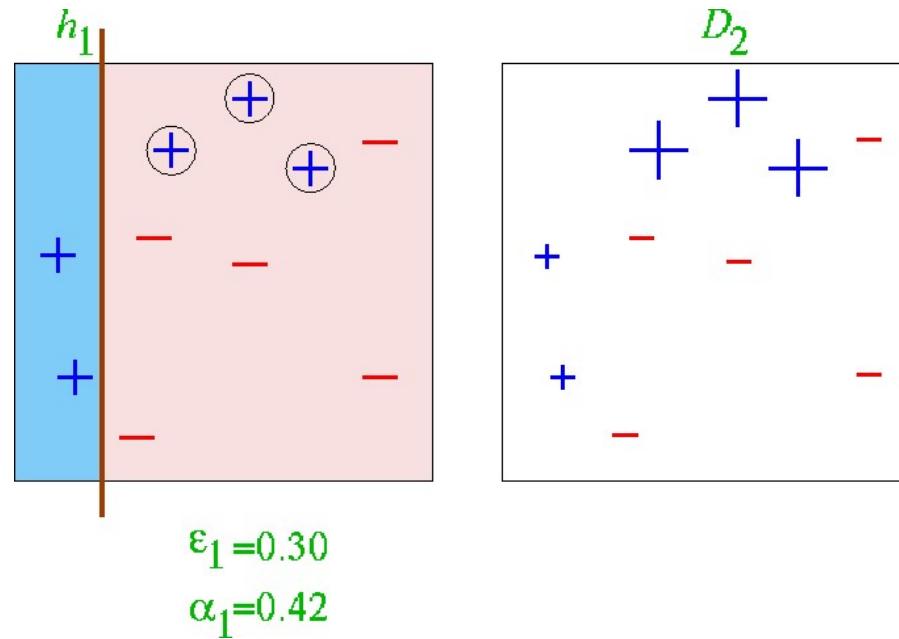
- Soit une base d'apprentissage $\{(x_i, y_i)\}_{i=1}^N$
- Initialisation des poids des exemples $w_i = \frac{1}{N}$
- Pour t de 1 à T :
 - h_t classeur minimisant la fonction de cout avec les poids w_i
 - Calcule de l'erreur de h_t : $\varepsilon_t = P_{w_i}(h_t(x_i) \neq y_i)$
 - Calcul des poids des classeurs $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$
 - Mise a jour des poids des exemples : $w_i := w_i \cdot \exp(-\alpha_t y_i h_t(x_i)) / z_t$
- Sortie : $H(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x_i))$



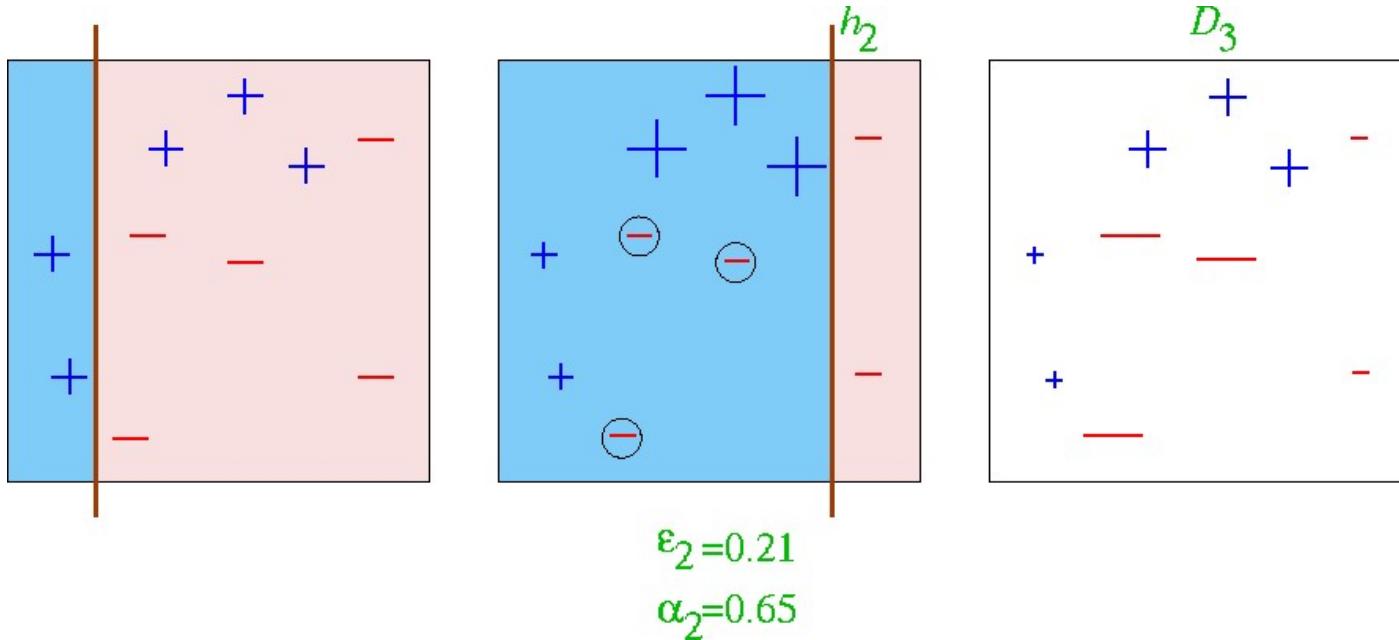
Exemple jouet



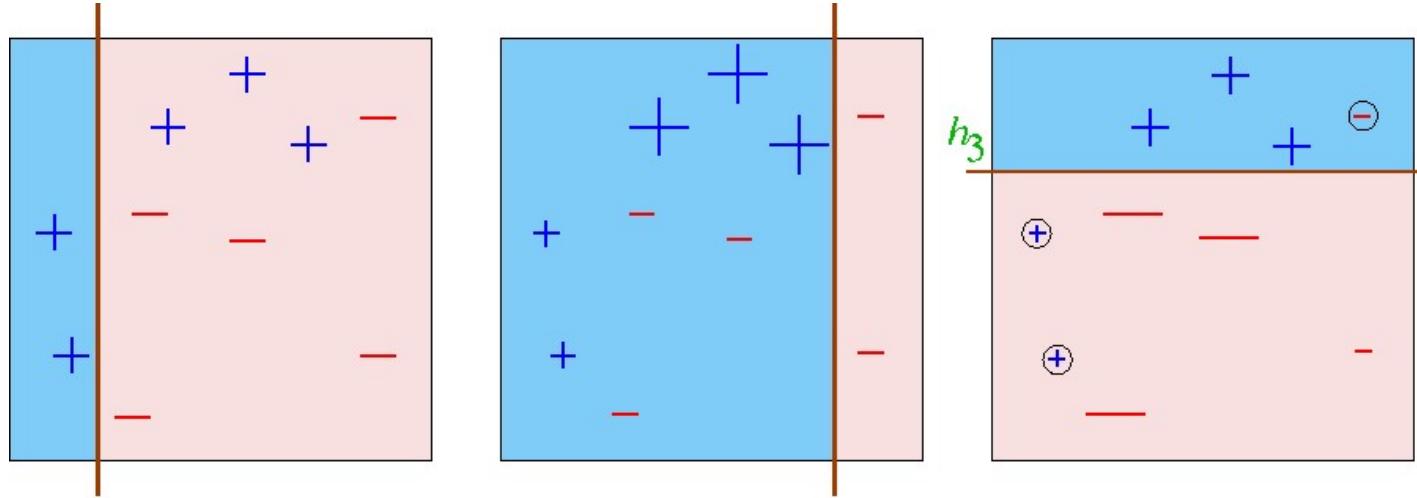
Étape 1



Étape 2



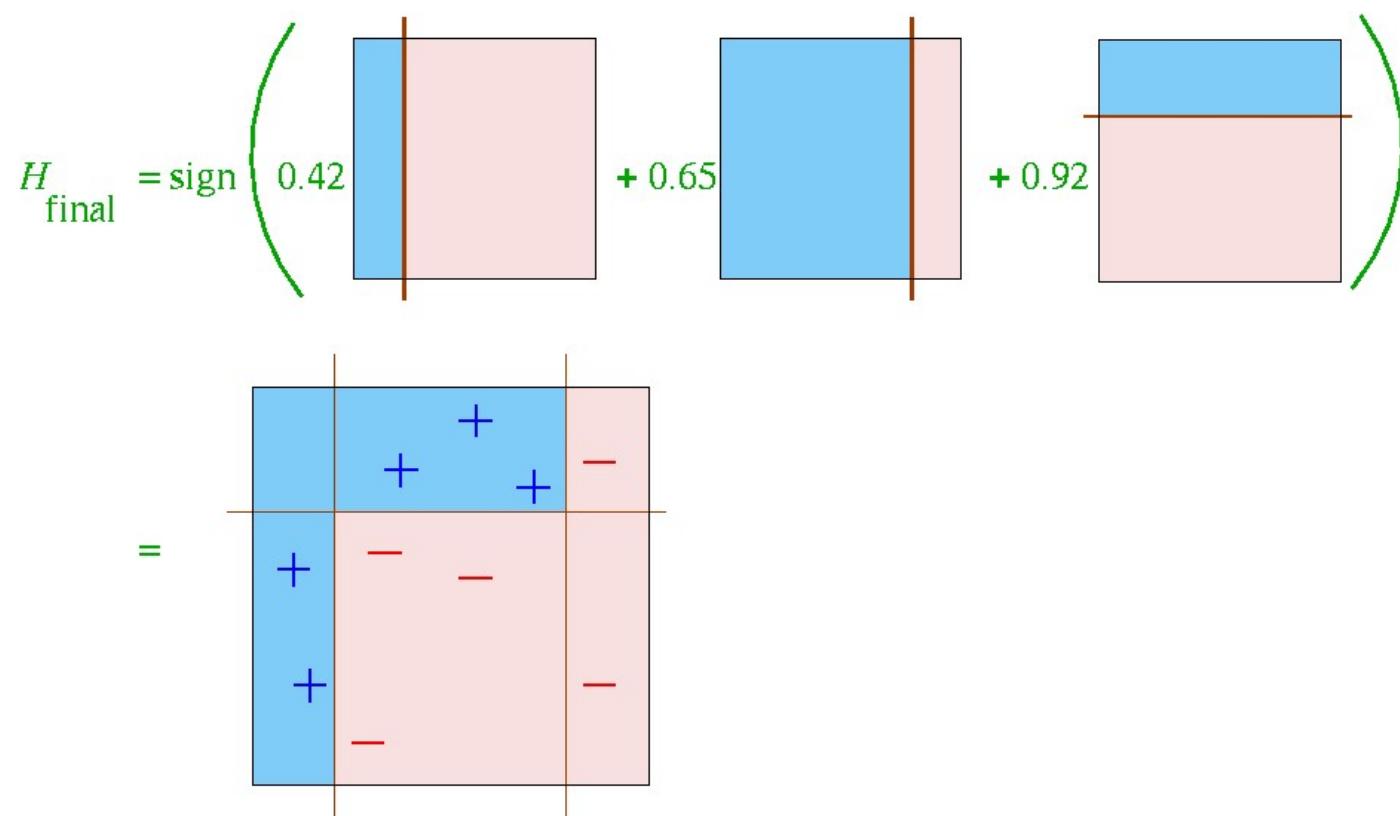
Étape 3



$$\varepsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

Hypothèse finale



Gradient Boosting

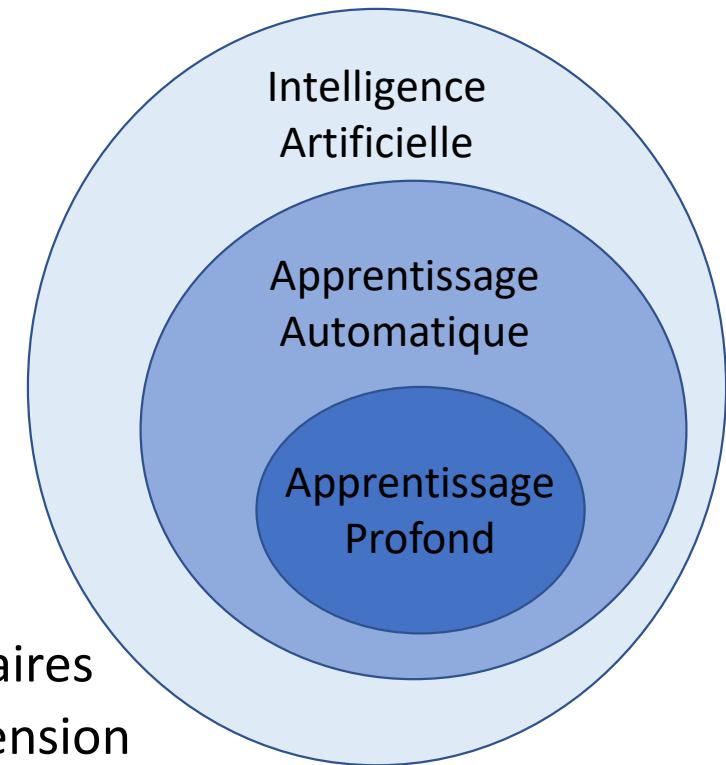
- Soit le modèle f_k qui agrège $\{h_0, \dots, h_k\}$
- Soit la fonction de cout : $L(f_k(x_i), y_i) = \frac{1}{2}(f_k(x_i) - y_i)^2$
- Le gradient négatif correspond au résidu
$$-\frac{\partial L(f_k(x_i), y_i)}{\partial f_k(x_i)} = y_i - f_k(x_i) = r_i$$
- Apprendre h_{k+1} avec le jeux de données $\{(x_i, r_i)\}_{i=1}^N$
- Mise a jour $f_{k+1}(x) \leftarrow f_k(x) + \alpha_{k+1}h_{k+1}(x)$
avec $\alpha_{k+1} = \operatorname{argmin}_\alpha \sum_{i=1}^N L(f_k(x_i) + \alpha_{k+1}h_{k+1}(x), y_i)$

Apprentissage Profond Deep Learning

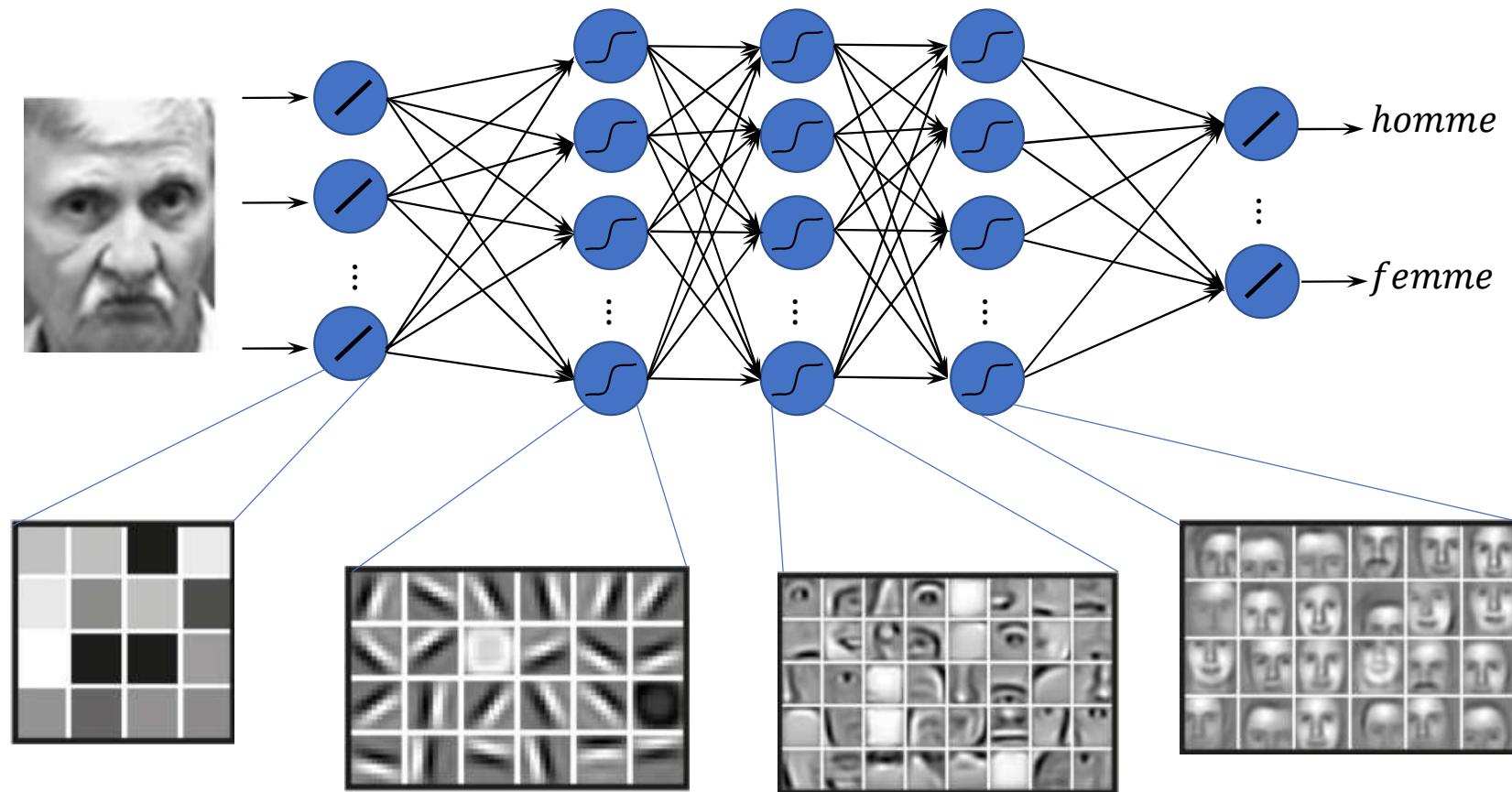
Perceptron multicouche (MLP)
Réseaux de convolution (CNN)
Réseaux récurrents (RNN, LSTM)

Apprentissage profond

- Sous domaine de l'apprentissage automatique (machine learning)
- Réseaux de neurones de grande taille
 - Composition de multiples transformations non linéaires
 - Découvrir des structures complexes en grande dimension
- Construction d'une **nouvelle représentation des données**
 - Plusieurs niveaux d'abstraction
 - Donner un sens aux données



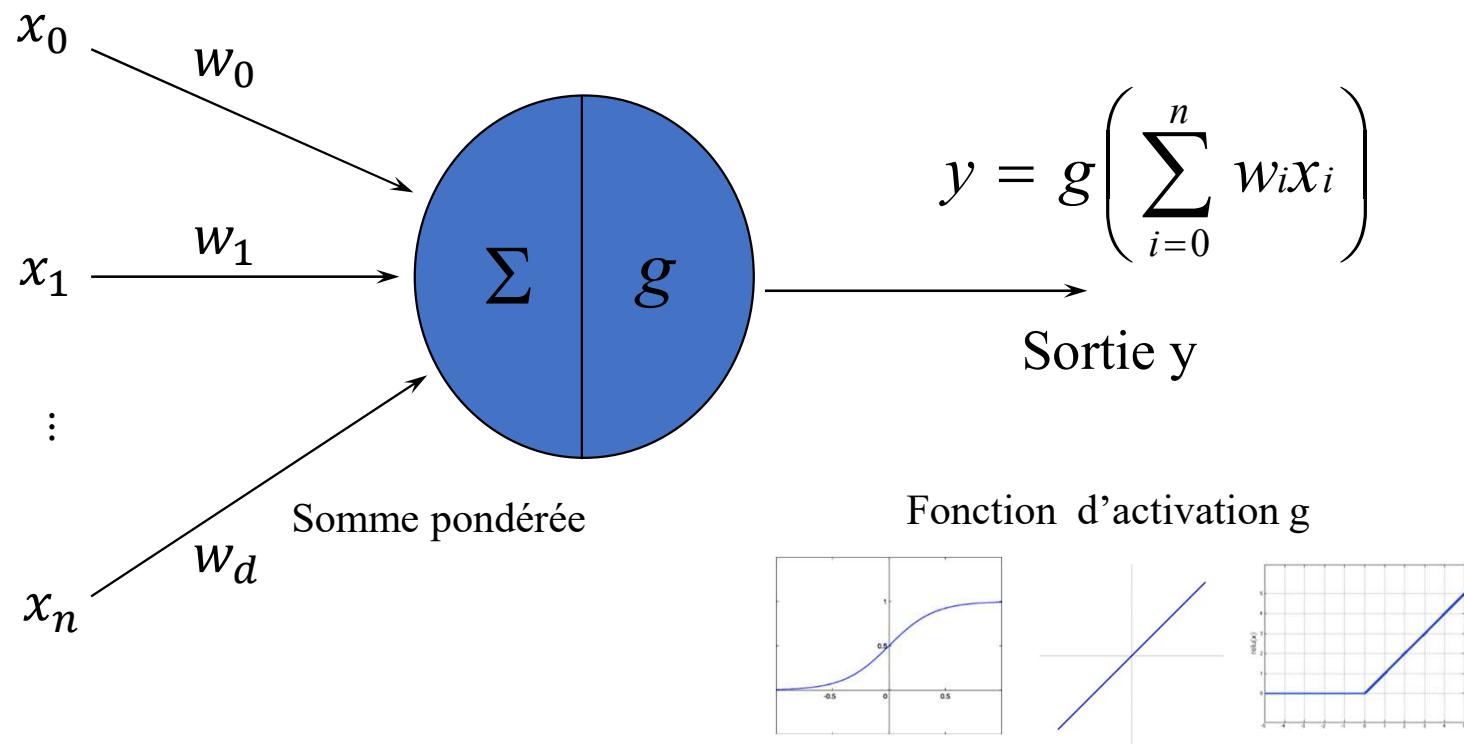
Apprentissage profond



Perceptron

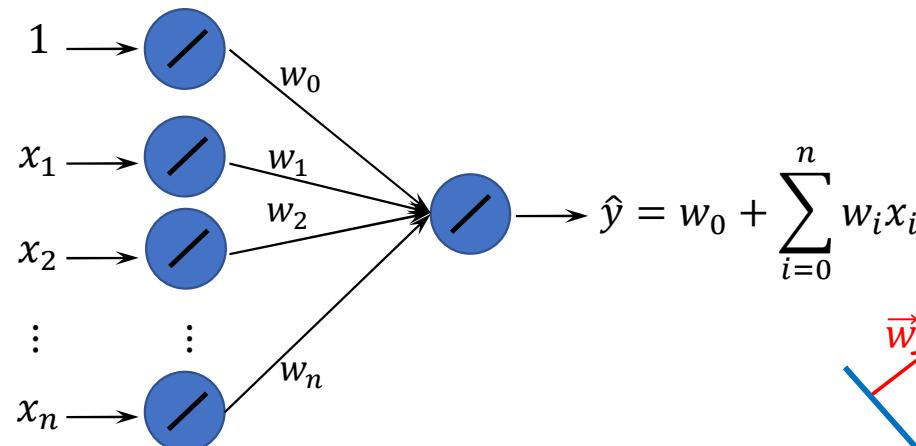
MLP (Fully connected)

Neurone formel

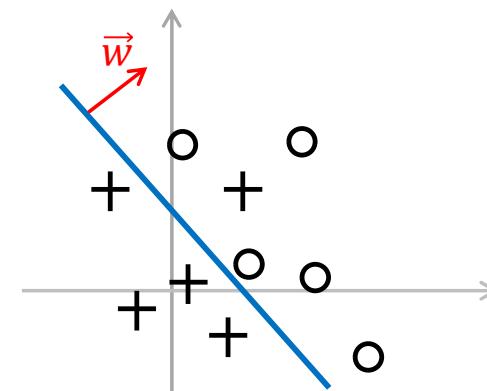


Perceptron

1ere génération de réseaux de neurones (50-60's): perceptron

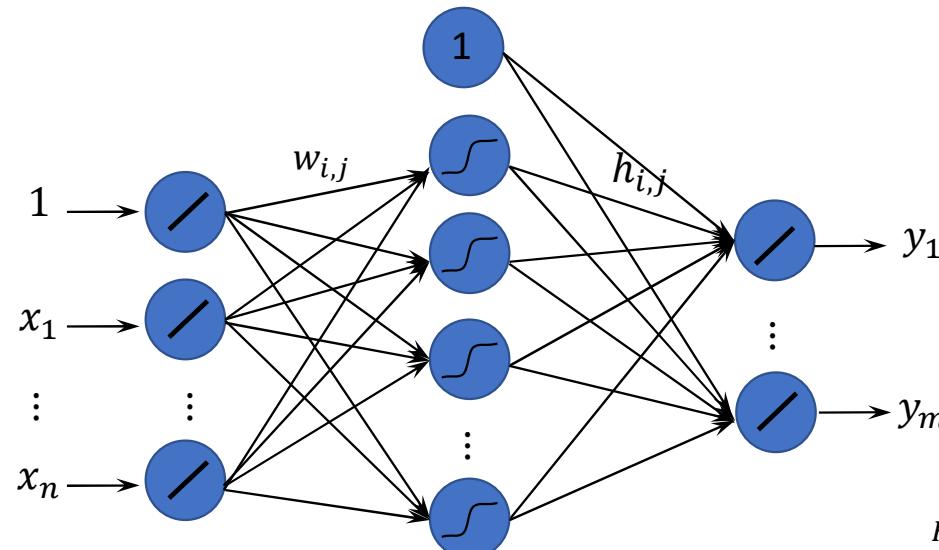


Limité a des modèles linaires [Rosenblatt 62]



Perceptron Multicouches

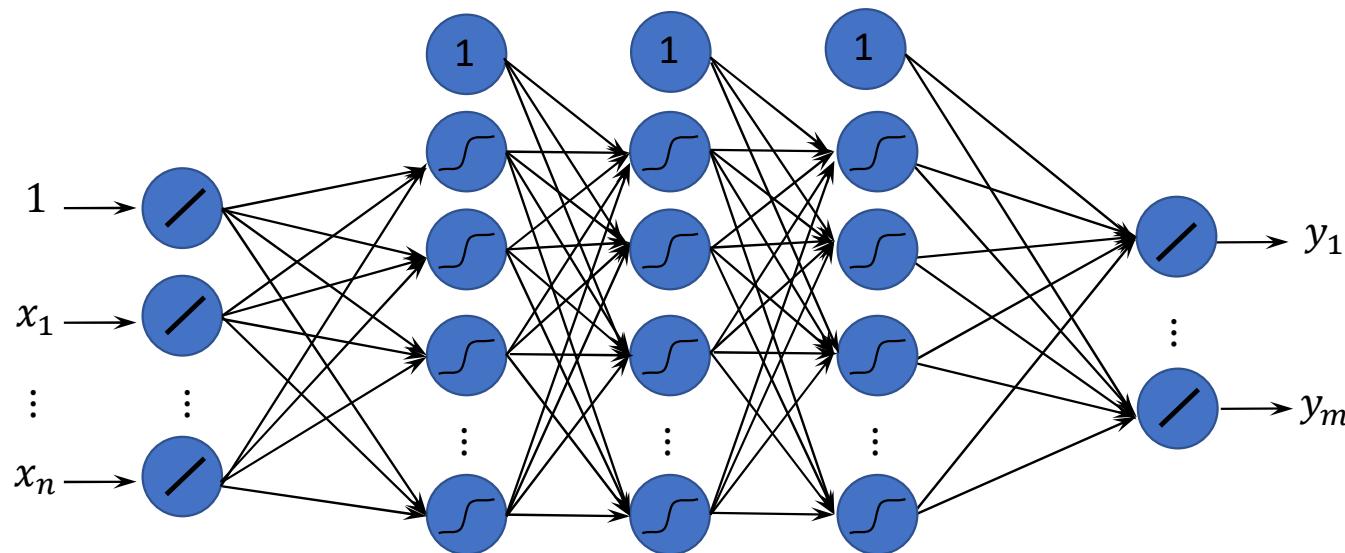
2^{ème} génération de réseaux de neurones (80's-90's) : perceptron multicouches – 1 couche caché



$$y_i = h_{i,0} + \sum_{j=1}^H h_{i,j} \cdot g\left(w_{j,0} \sum_{k=1}^n w_{jk} x_k\right)$$

Deep learning

3^{ème} génération de réseaux de neurones (2010's) : deep learning

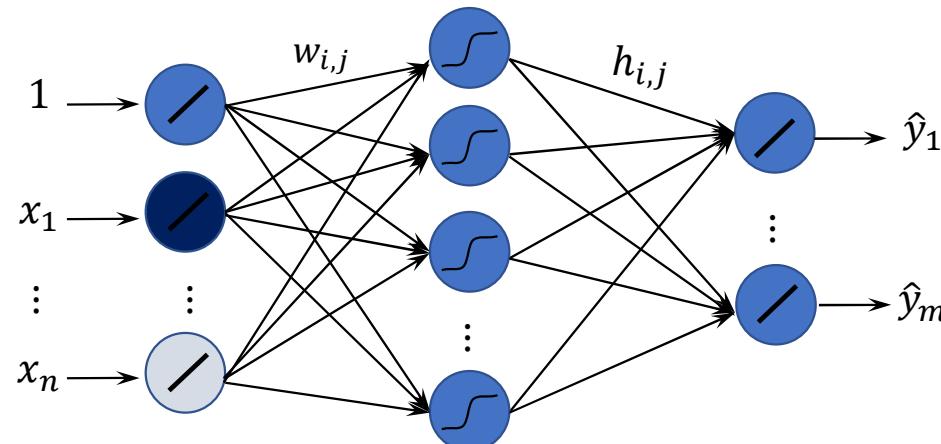


Propagation d'un exemple

Construction à l'aide d'un ensemble d'apprentissage

Propagation d'un exemple d'apprentissage (x, y)

$$x = (x_1, \dots, x_n), y = (y_1, \dots, y_m)$$

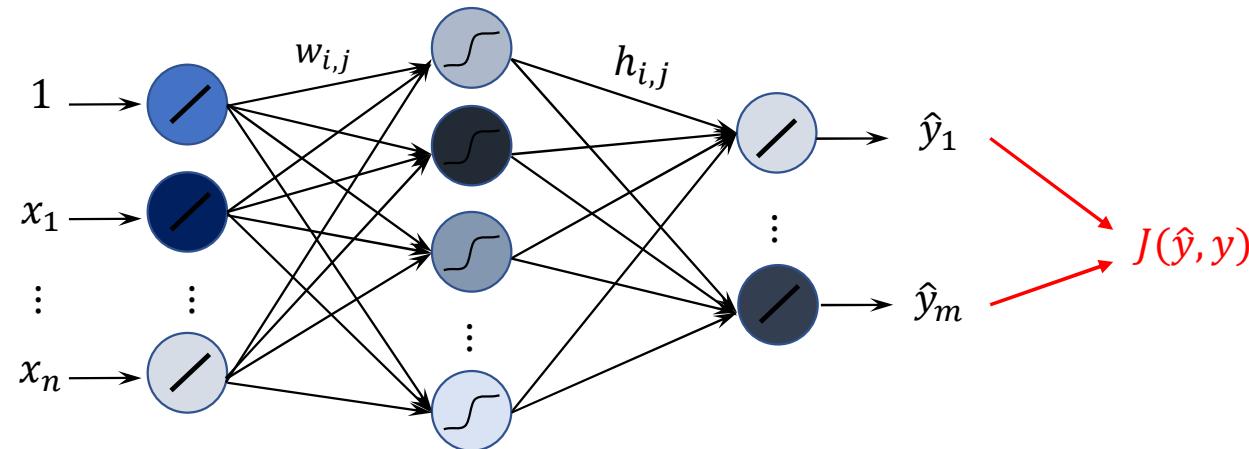


Propagation

Erreur de prédition

Construction à l'aide d'un ensemble d'apprentissage

Calcul d' erreur de prédition : fonction de cout J



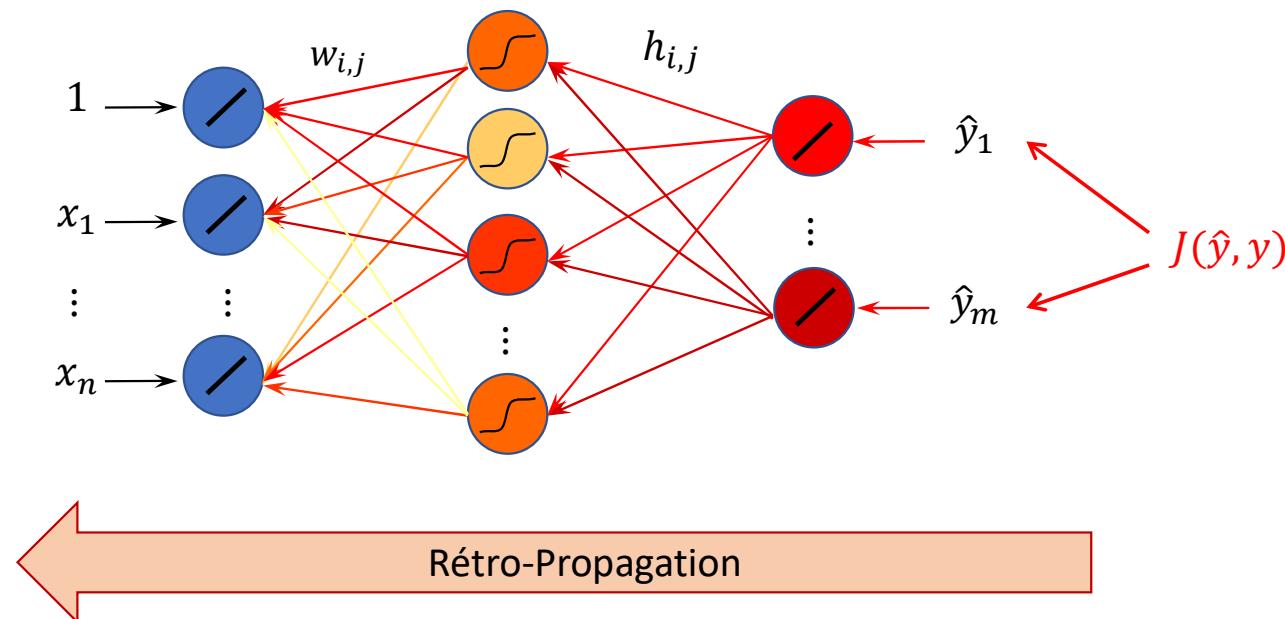
Rétropropagation de l'erreur

Construction à l'aide d'un ensemble d'apprentissage

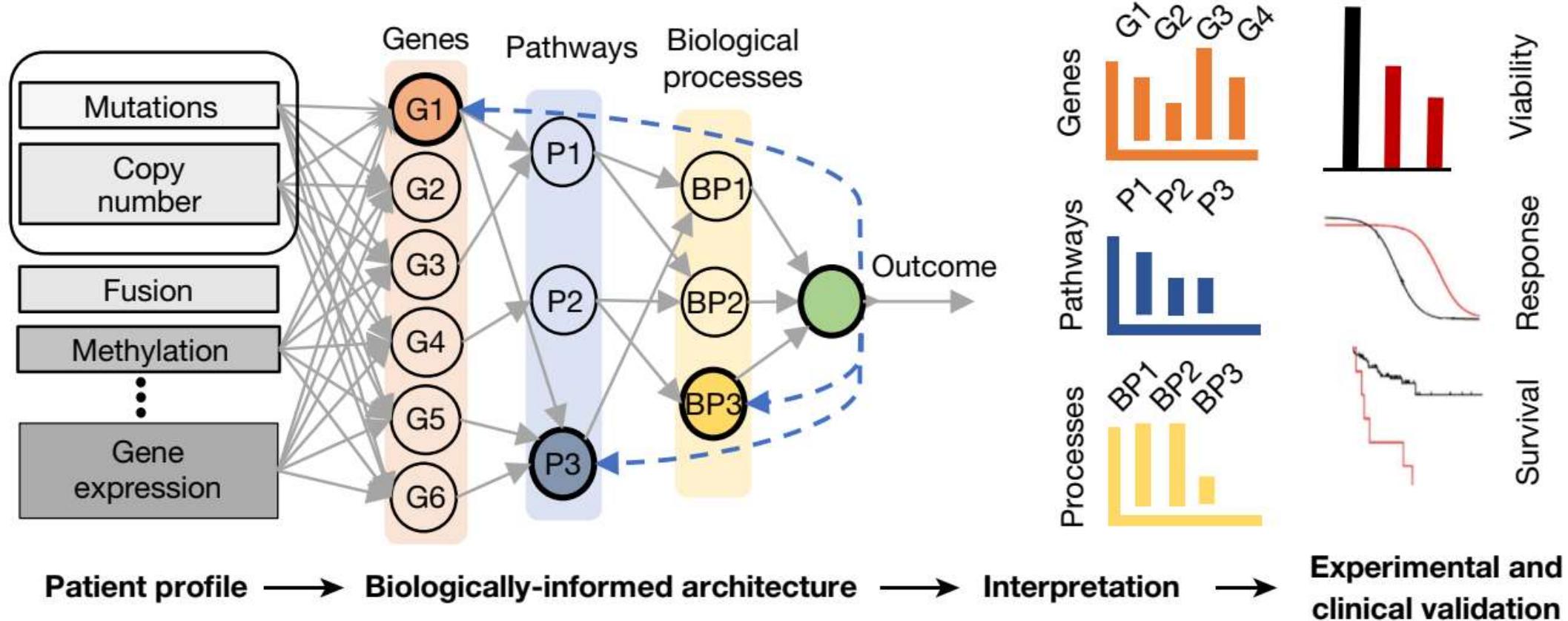
Calcul des gradients, mise à jour des poids

$$w'_{i,j} = w_{i,j} - \alpha \frac{\partial J}{\partial w_{i,j}}$$

$$h'_{i,j} = h_{i,j} - \alpha \frac{\partial J}{\partial h_{i,j}}$$



Elmarakeby, H. A., Hwang, J., Arafeh, R., Crowdus, J., Gang, S., Liu, D., ... & Van Allen, E. M. (2021). Biologically informed deep neural network for prostate cancer discovery. *Nature*, 598(7880), 348-352.



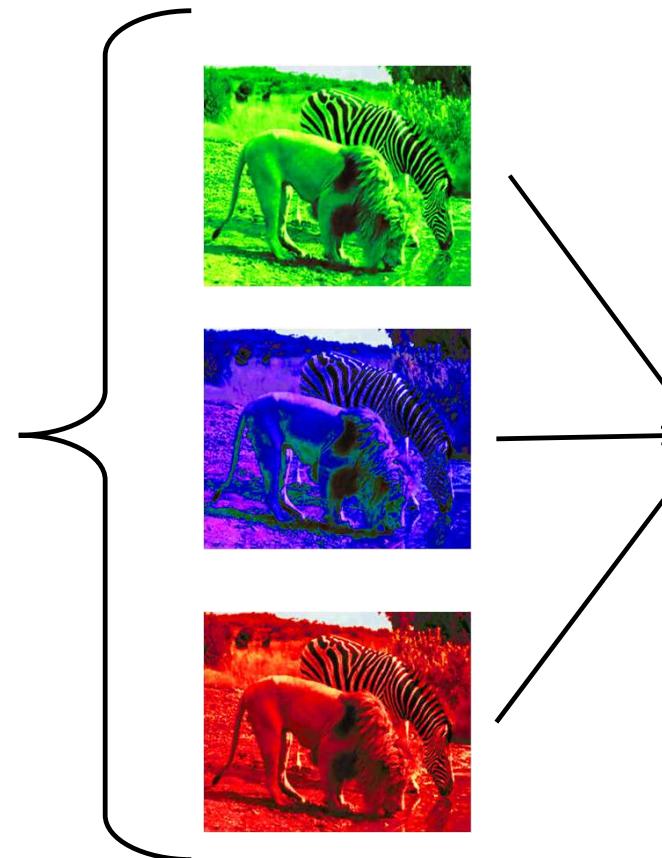
Réseaux de convolution

CNN (VGG, ResNet, Inception)

Représentation des données

Images :

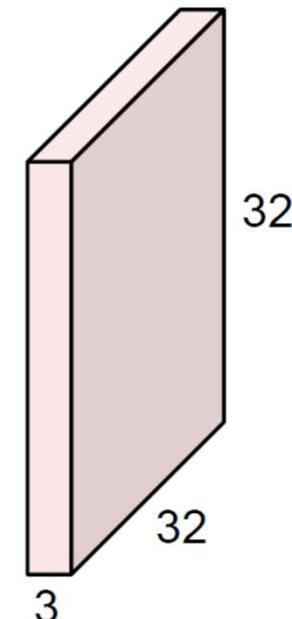
- Données comprenant plusieurs canaux
- Structures 2D des pixels



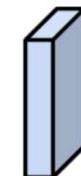
Couche de convolution

Préservation de la structure a l'aide de filtres :

- Utilise tous les canaux de l'image
- Représente une petite partie de l'image
- Glisse sur toute l'image



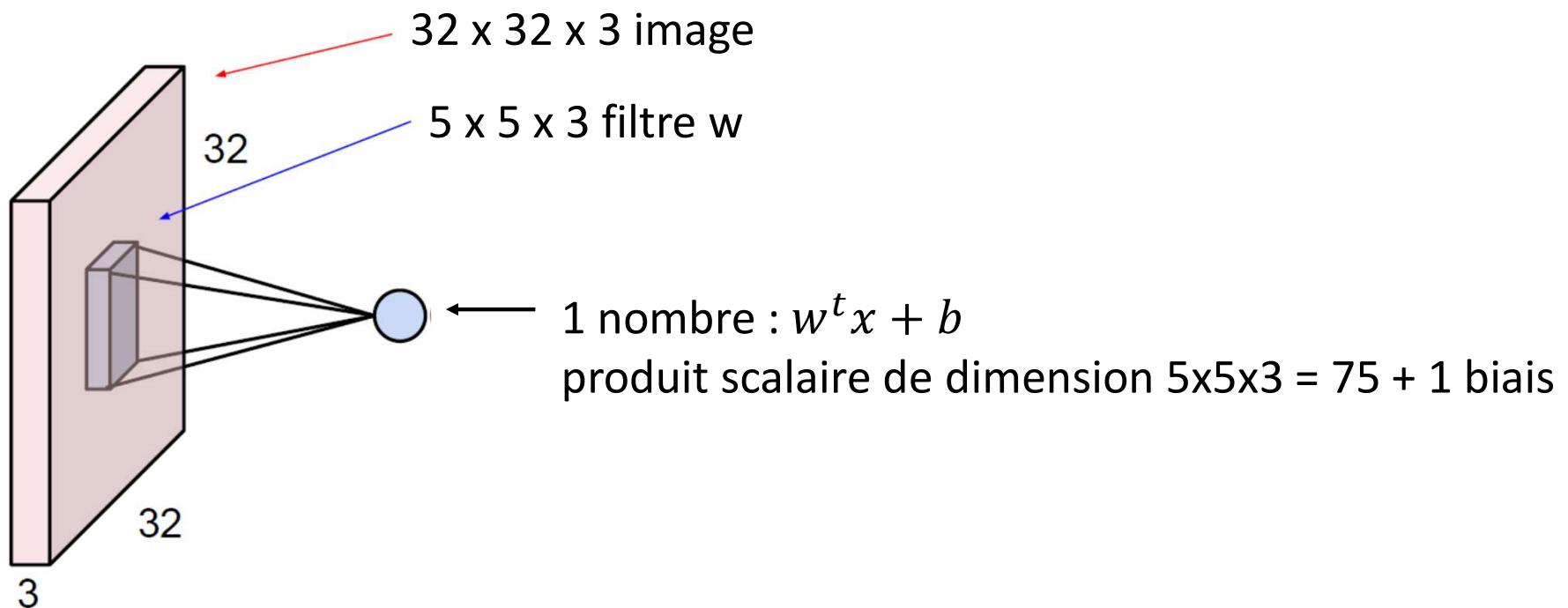
$32 \times 32 \times 3$ image



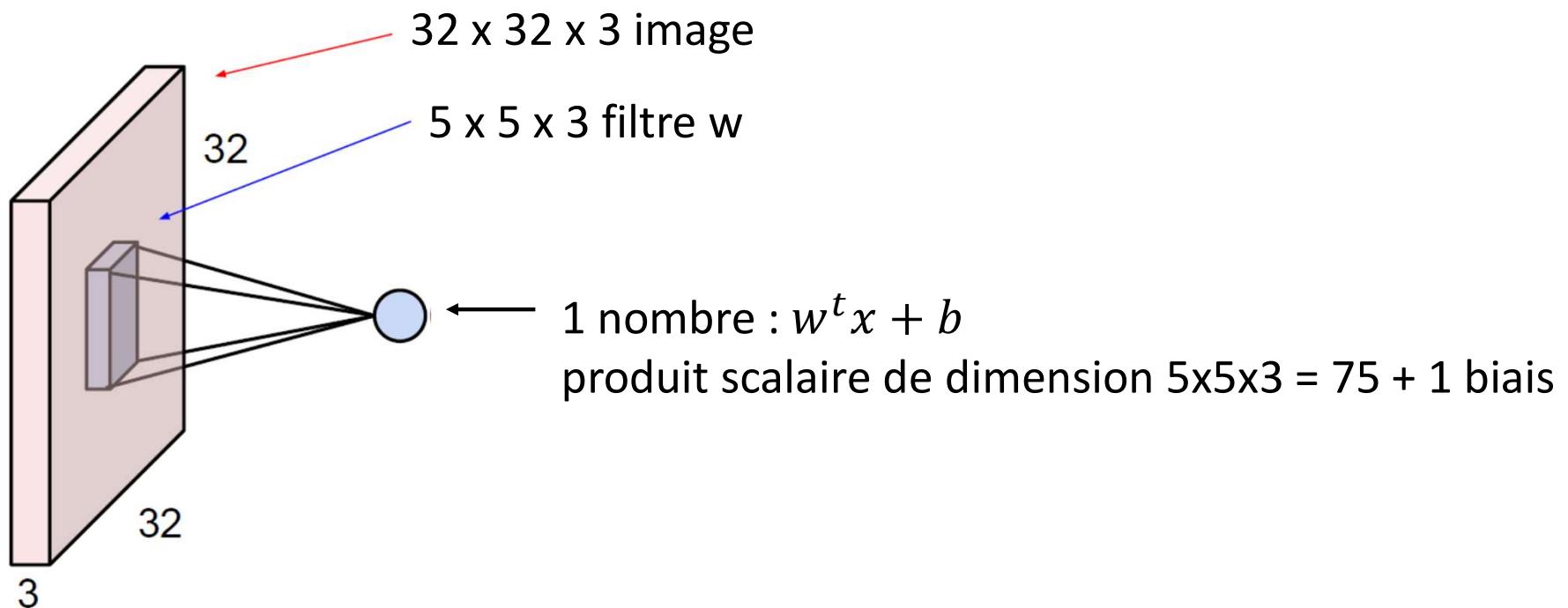
$5 \times 5 \times 3$ filtre

[Image from FeiFei course cs231n]

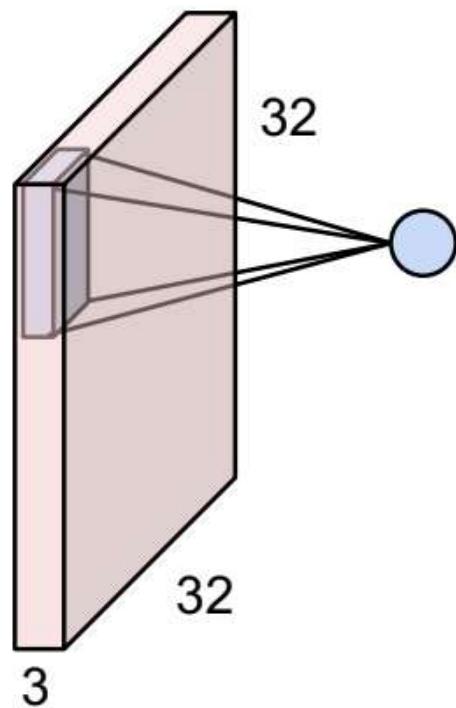
Couche de convolution



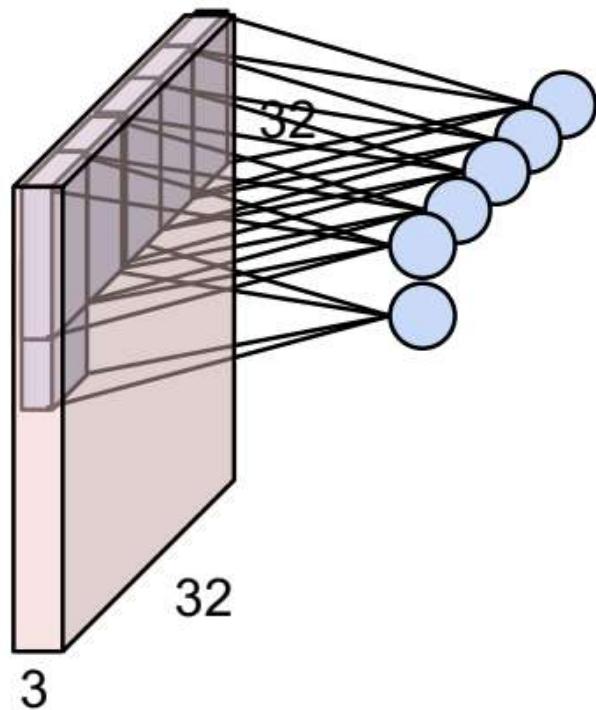
Couche de convolution



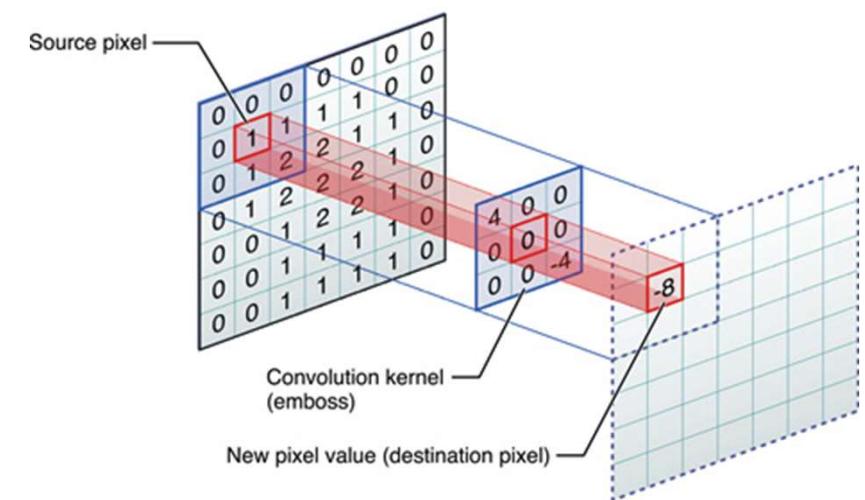
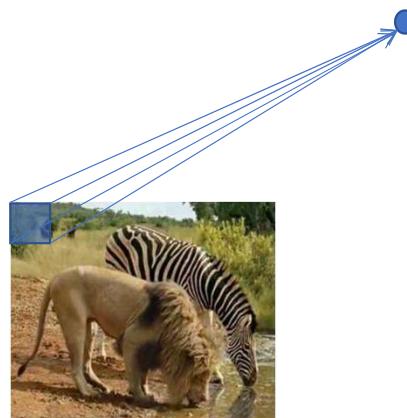
Couche de convolution



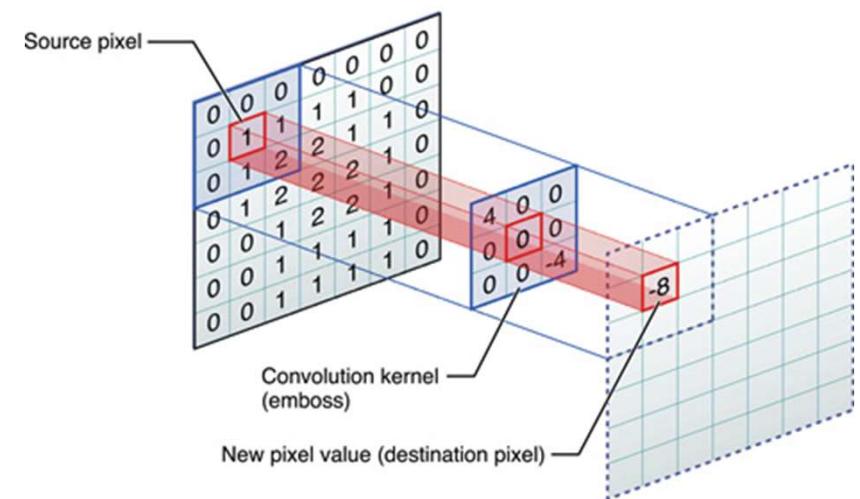
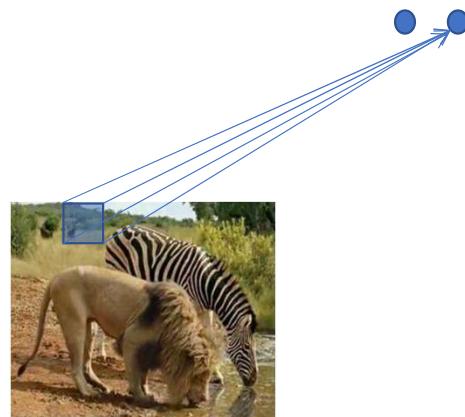
Couche de convolution



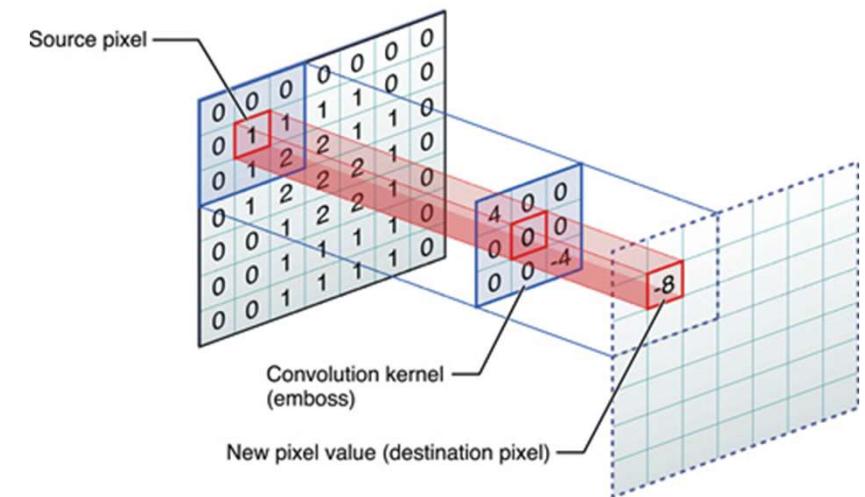
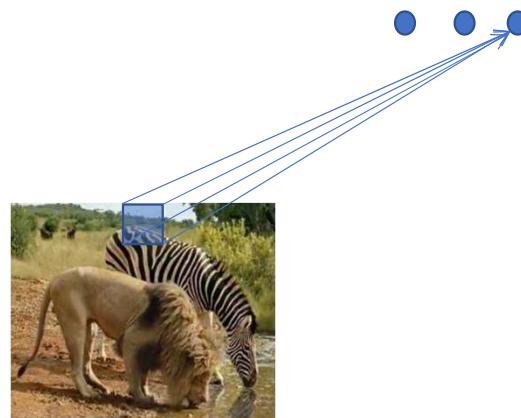
Couche de convolution



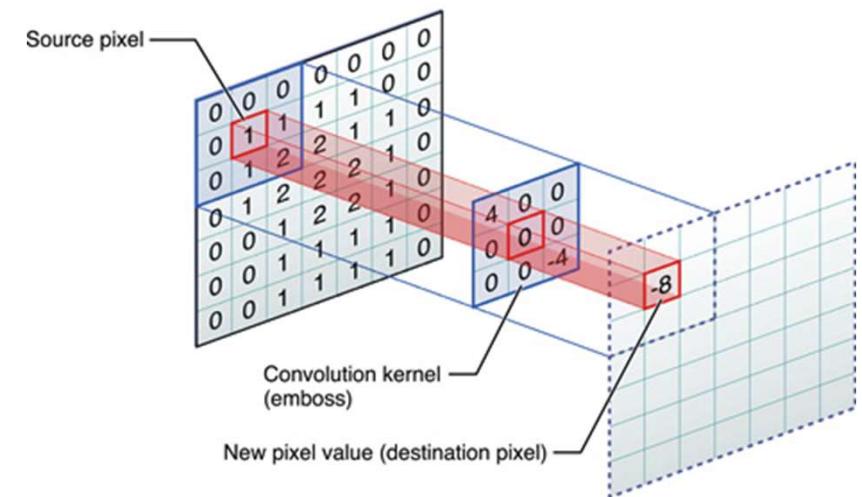
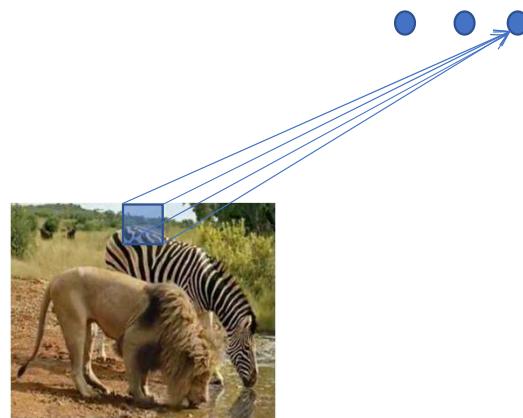
Couche de convolution



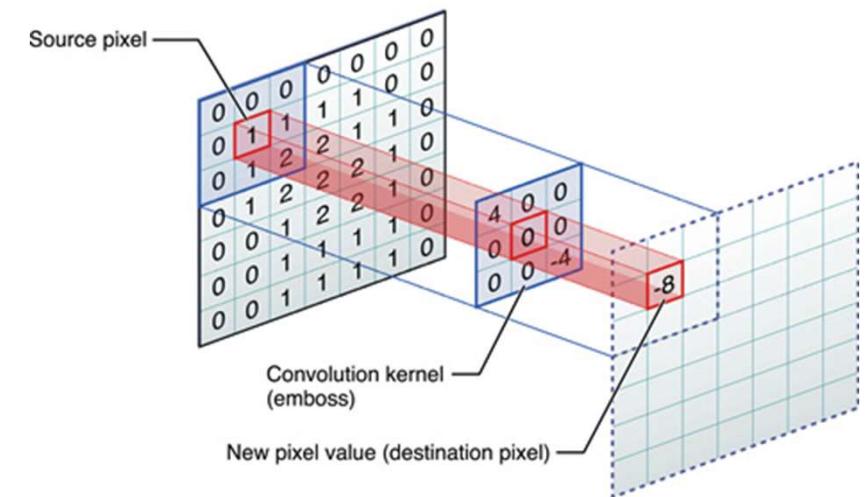
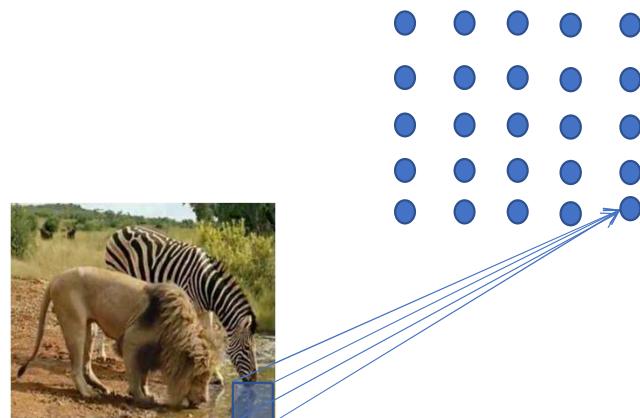
Couche de convolution



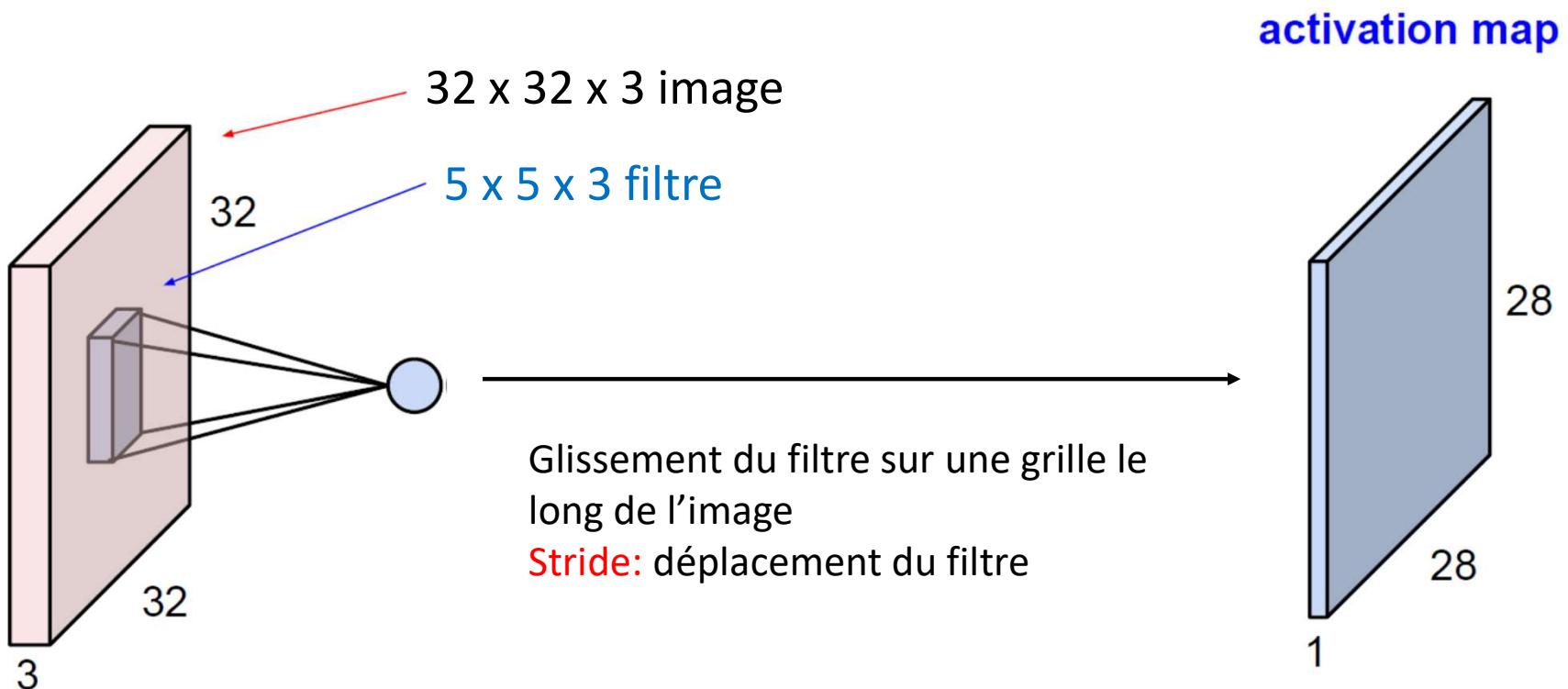
Couche de convolution



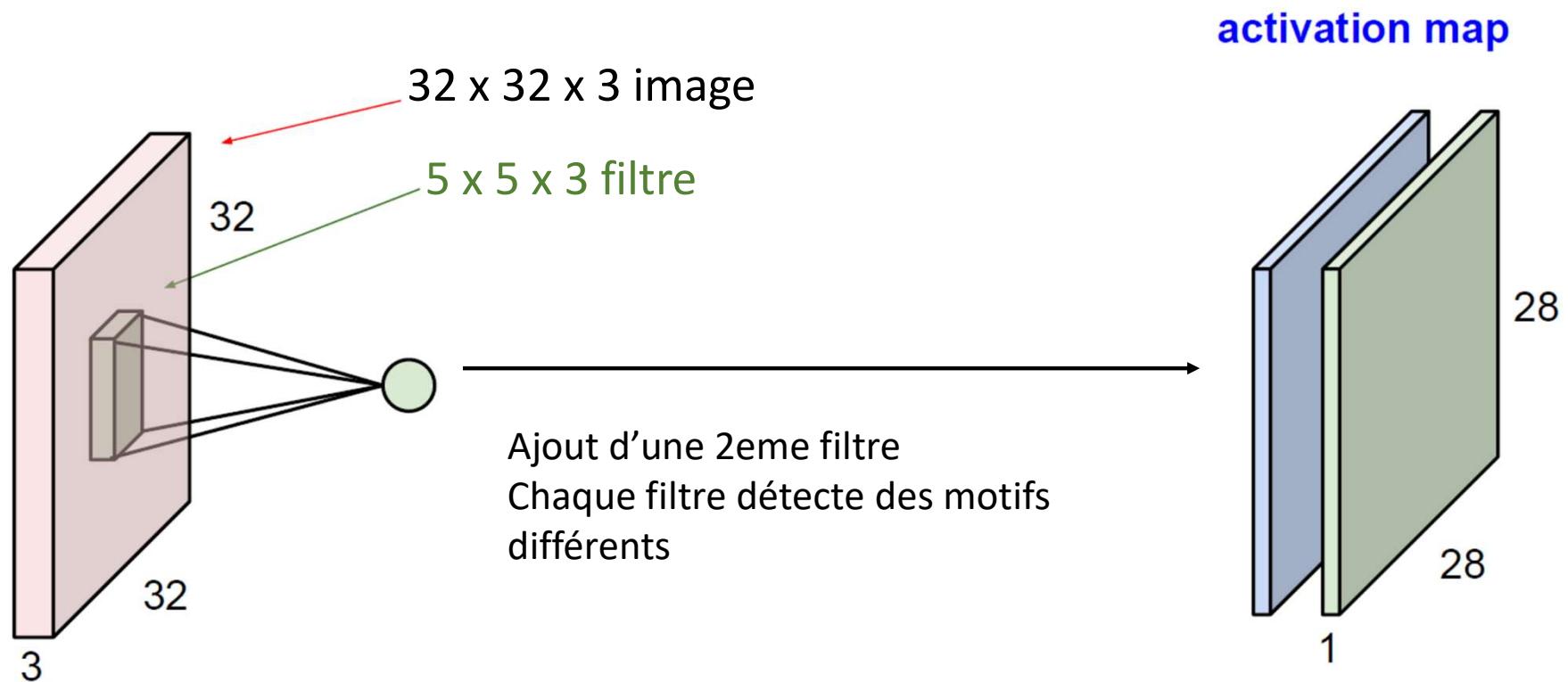
Couche de convolution



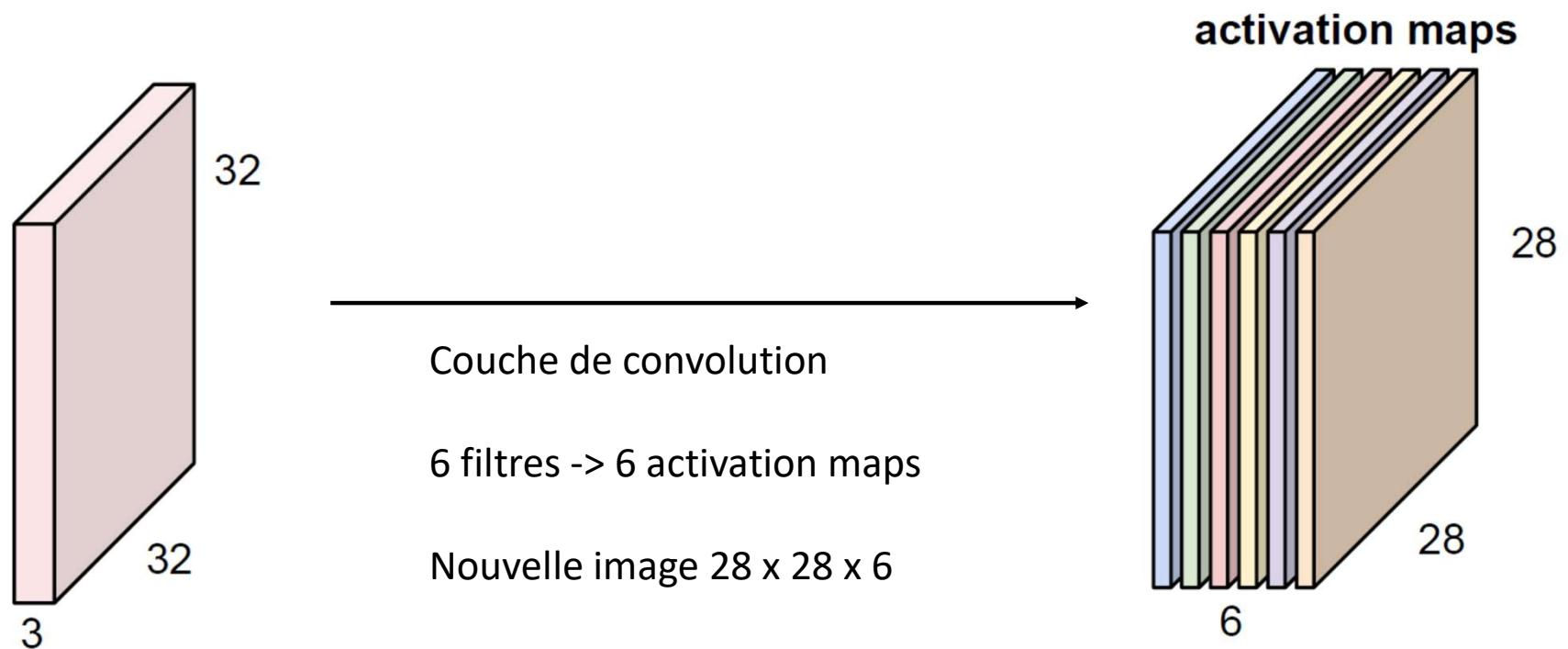
Couche de convolution



Couche de convolution

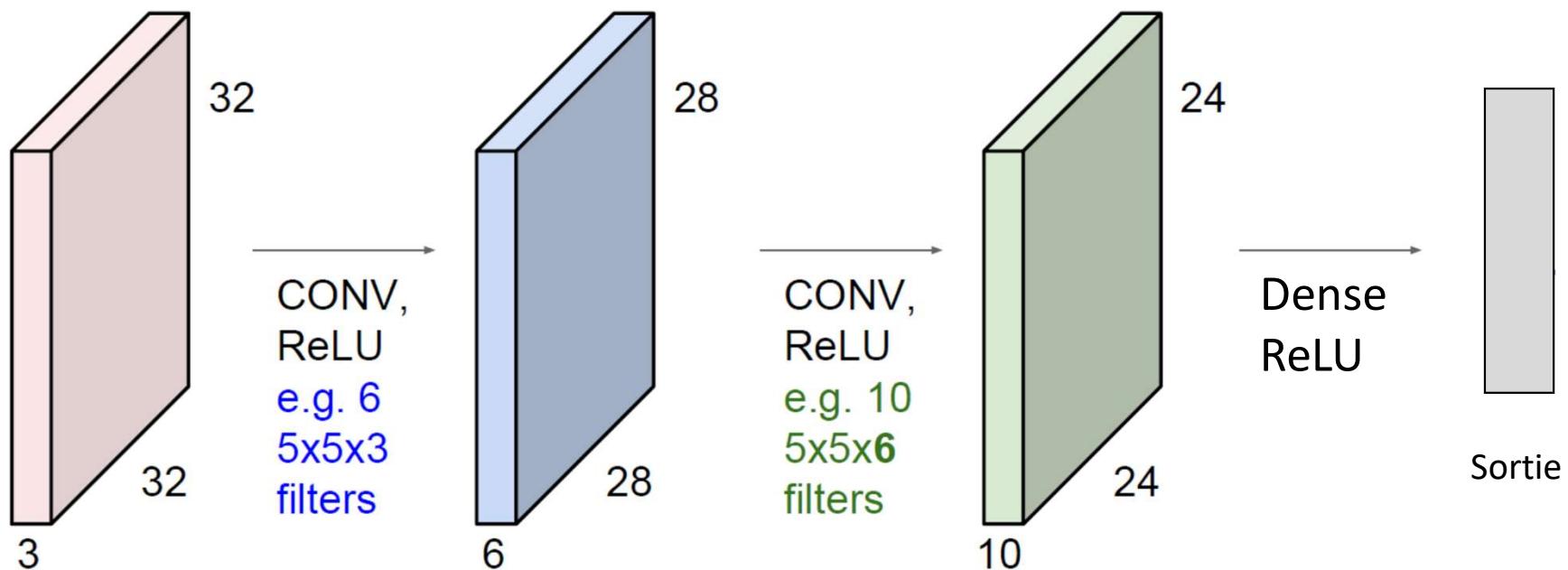


Couche de convolution



Réseau de convolution

CovNet : une séquence de couches de convolution entrecoupée de fonctions d'activation

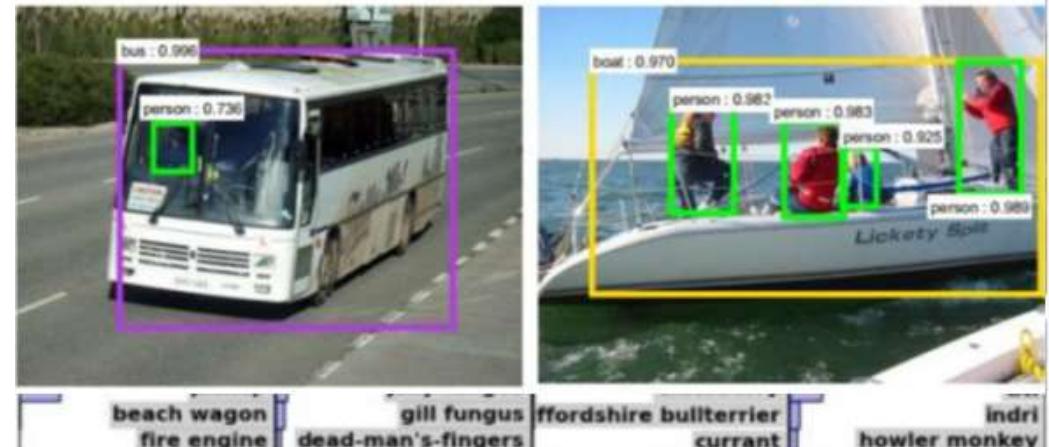


[Image from FeiFei course cs231n]

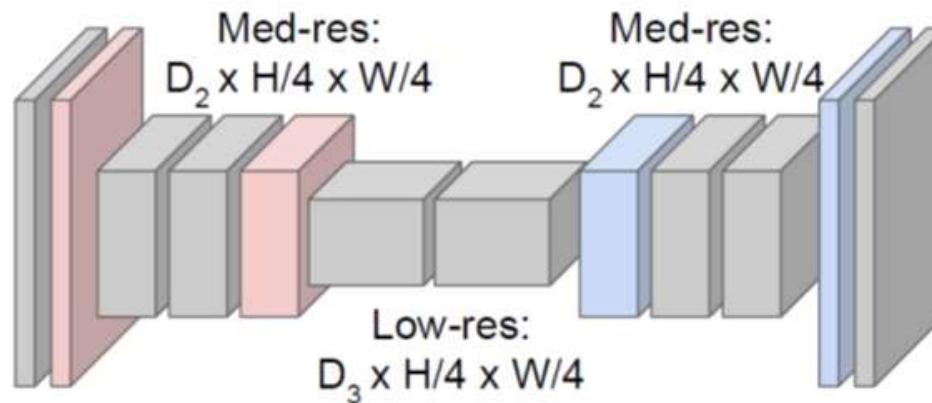
Classification



Localisation

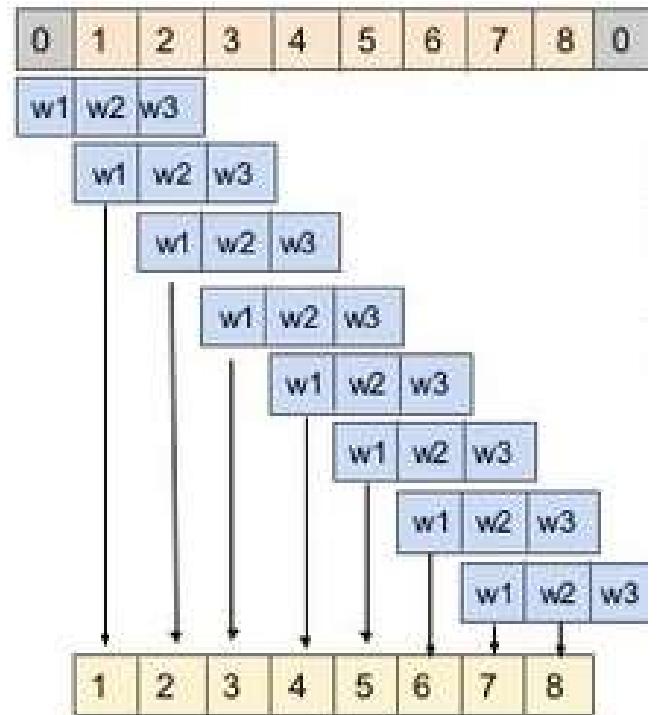
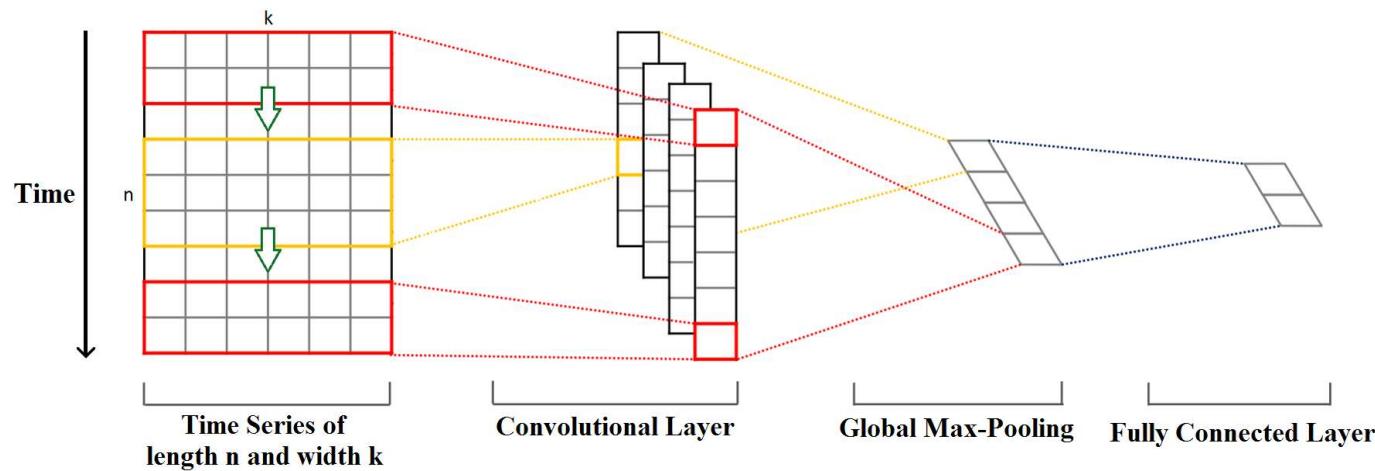


Segmentation



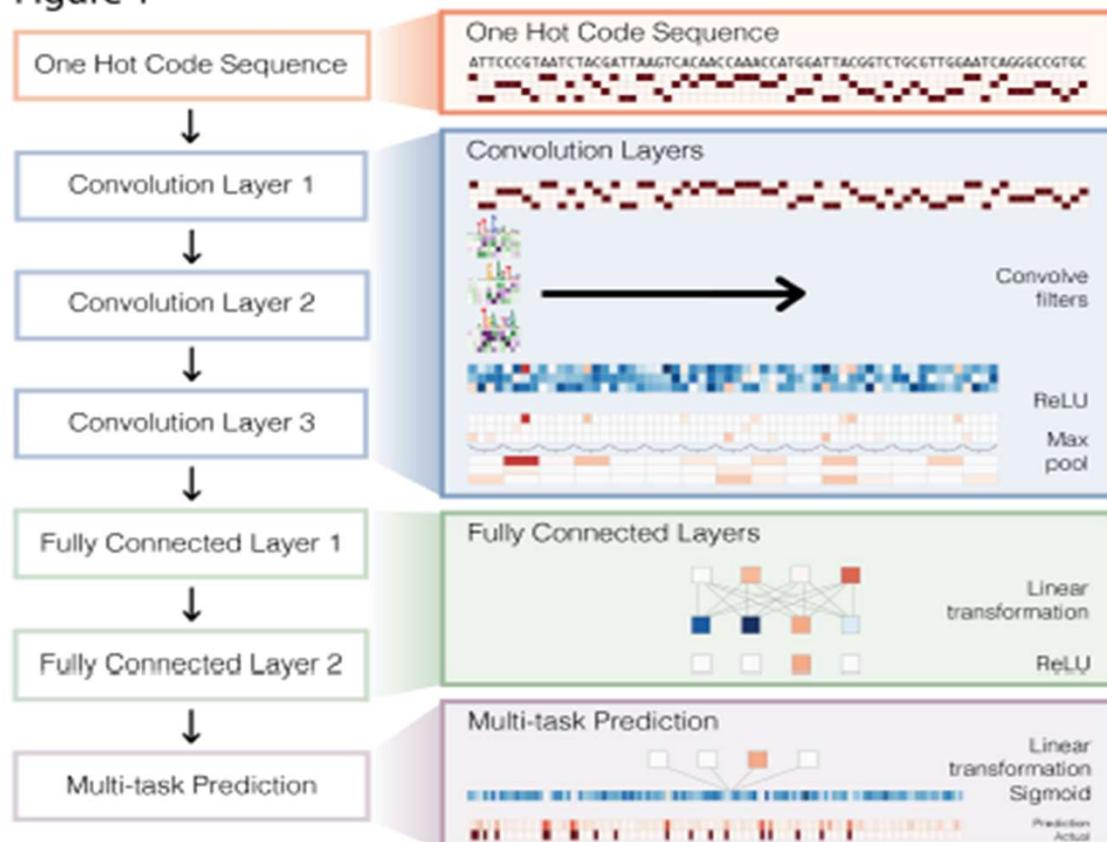
Données séquentielles avec des convolutions

- On peut traiter une séquence avec des convolution 1D

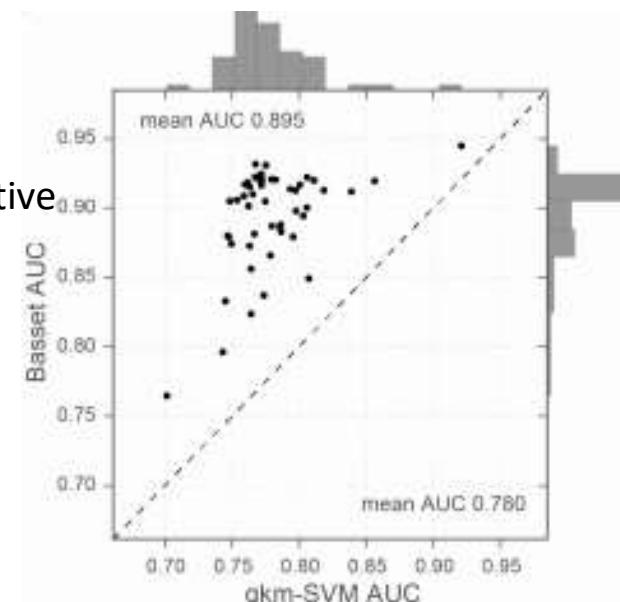


Prédiction de 164 facteurs de transcription et « sites de fixation »

Figure 1



Amélioration significative
des performances de
prédition



Mise en évidence dans les couches de convolution de
séquences connues

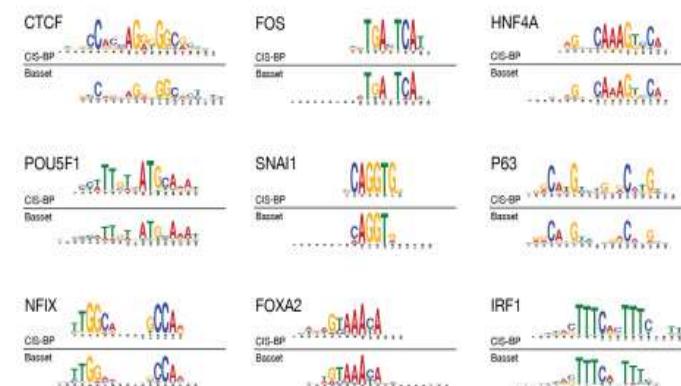


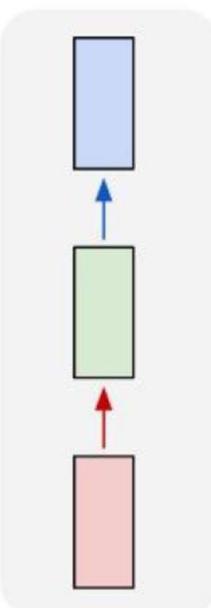
Figure 1 - Deep convolutional neural network for DNA sequence analysis

Réseaux récurrents et données séquentielles

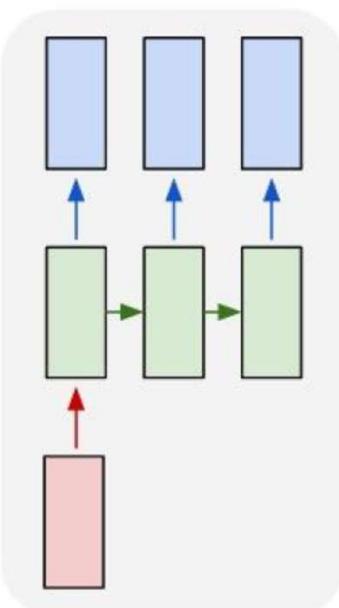
RNN, LSTM, GRU

Traitement de séquences

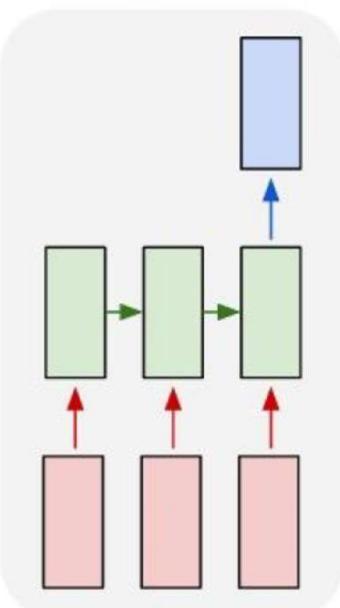
Elt -> Elt



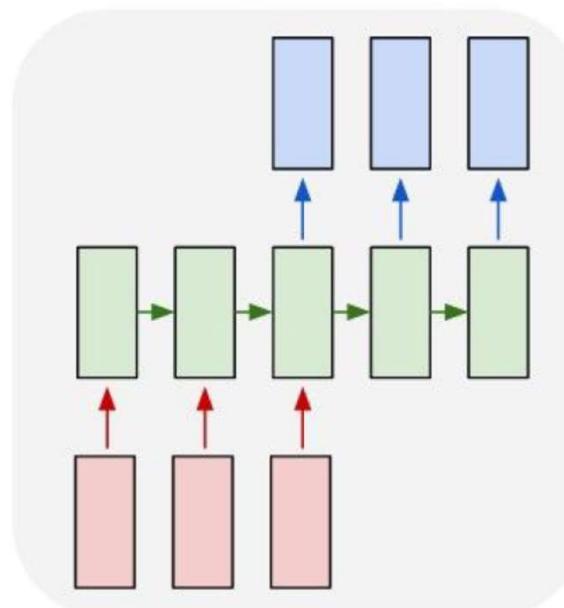
Elt -> Seq



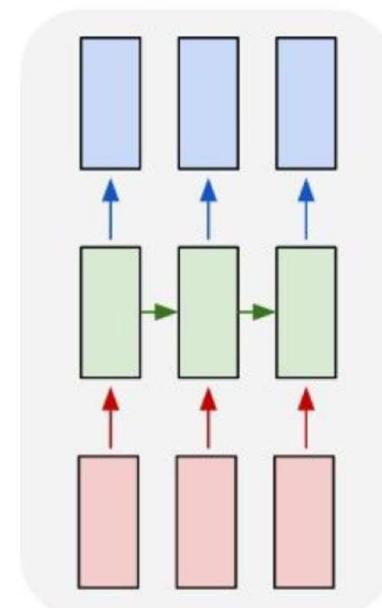
Seq -> Elt



Seq -> Seq



Seq -> Seq



Description d'images

Analyse de sentiments
Diagnostique génétique

Traduction automatique
Transformation vidéo

Détection d'anomalies
Classification de vidéo
(frame)

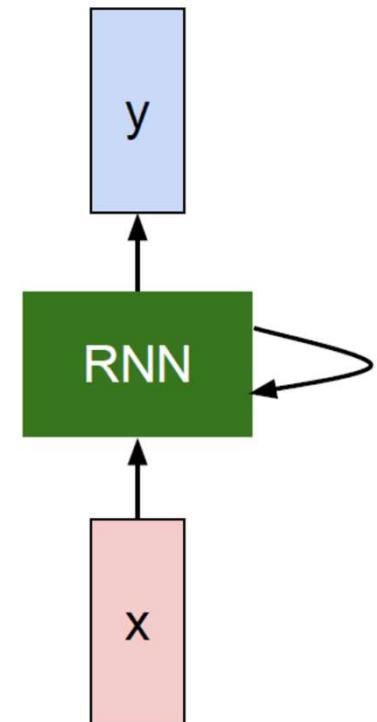
Réseaux récurrents

- Traitement des données séquentielles
- Réseaux classique : $h = f_W(x)$
- Réseaux récurrents :

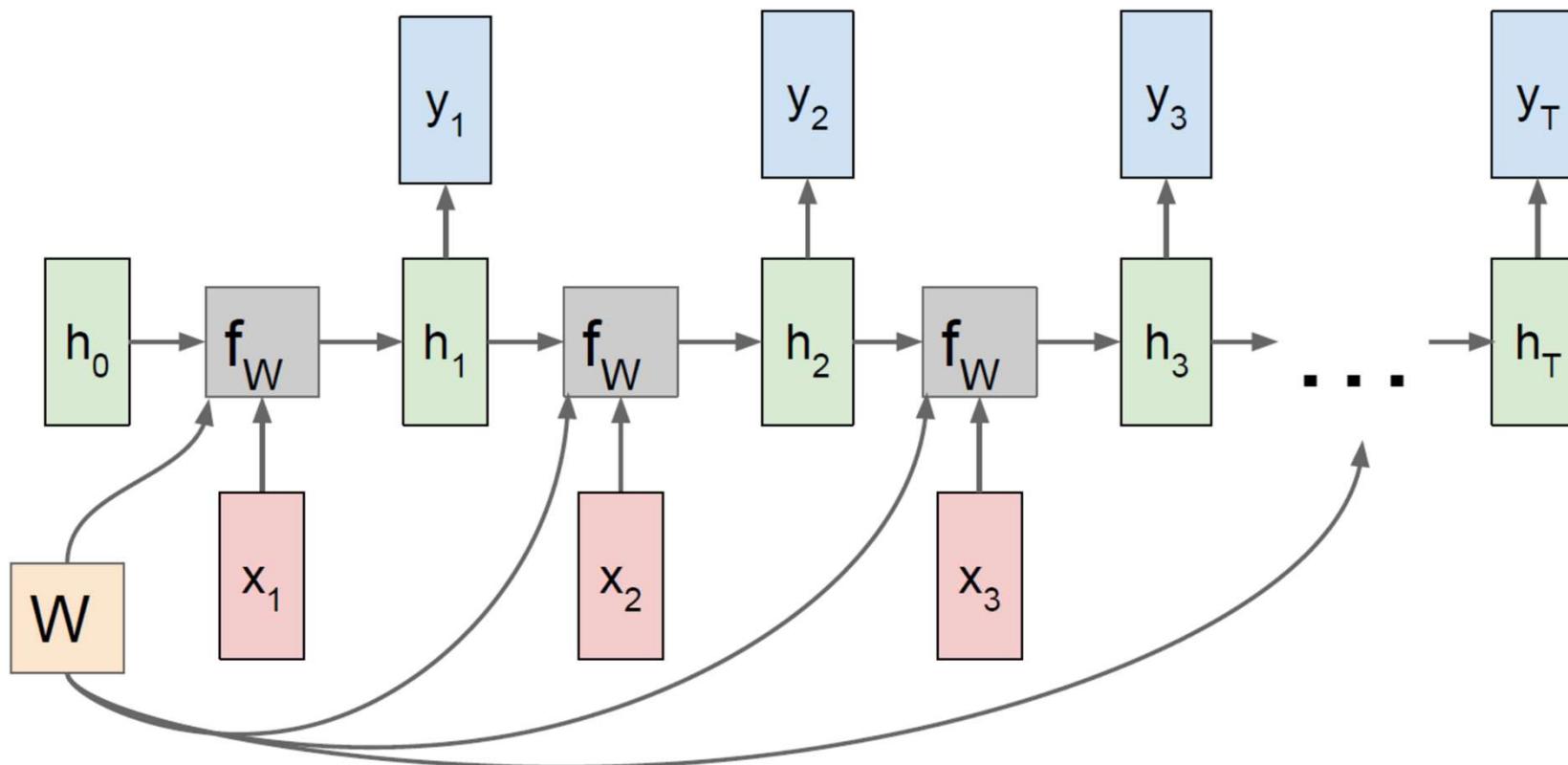
Relation entre les entrées x et les sorties y définie par un état caché h

Nouveau état	Fonction avec paramètres W	Etat précédent	Nouvelle entrée
h_t	$f_W(h_{t-1}, x_t)$		

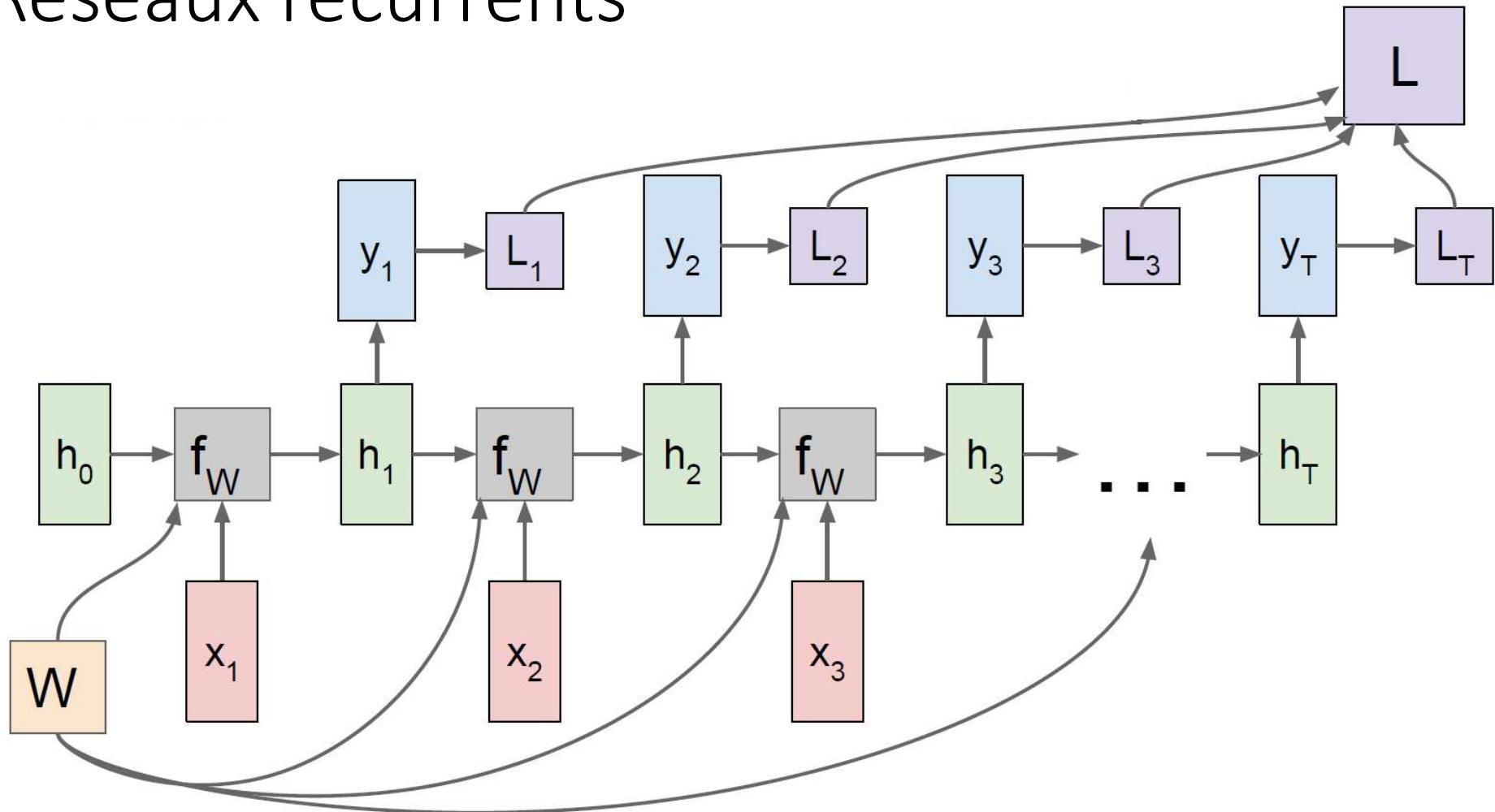
Même fonction et mêmes paramètres à chaque étape



Réseaux récurrents



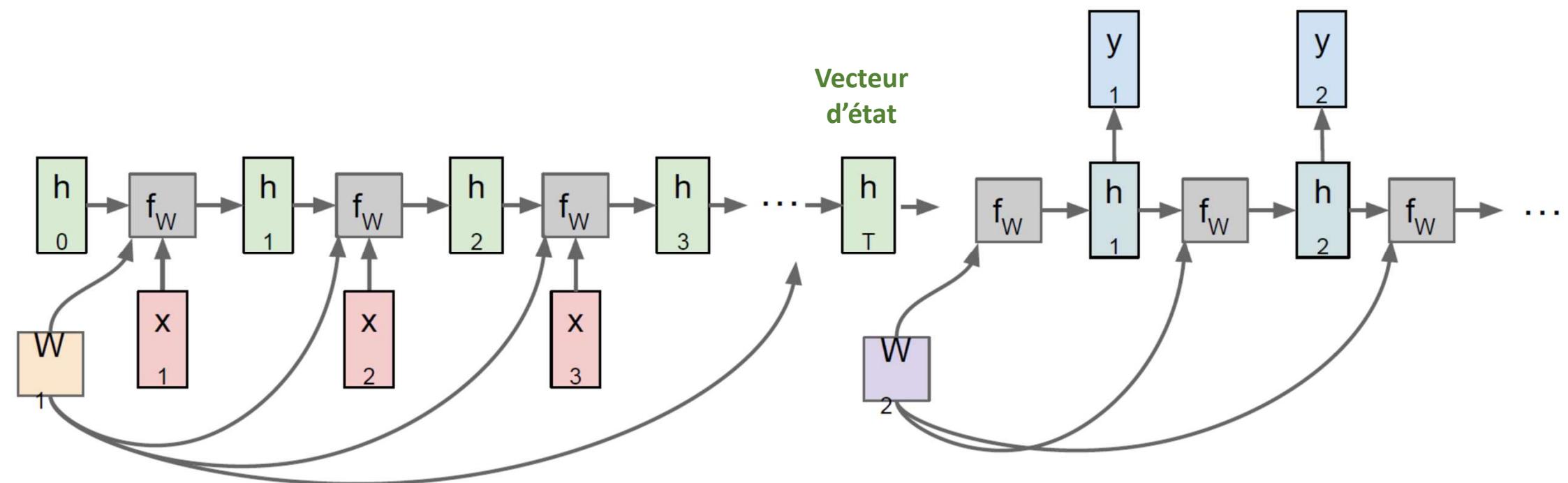
Réseaux récurrents



Modèle encodeur - décodeur

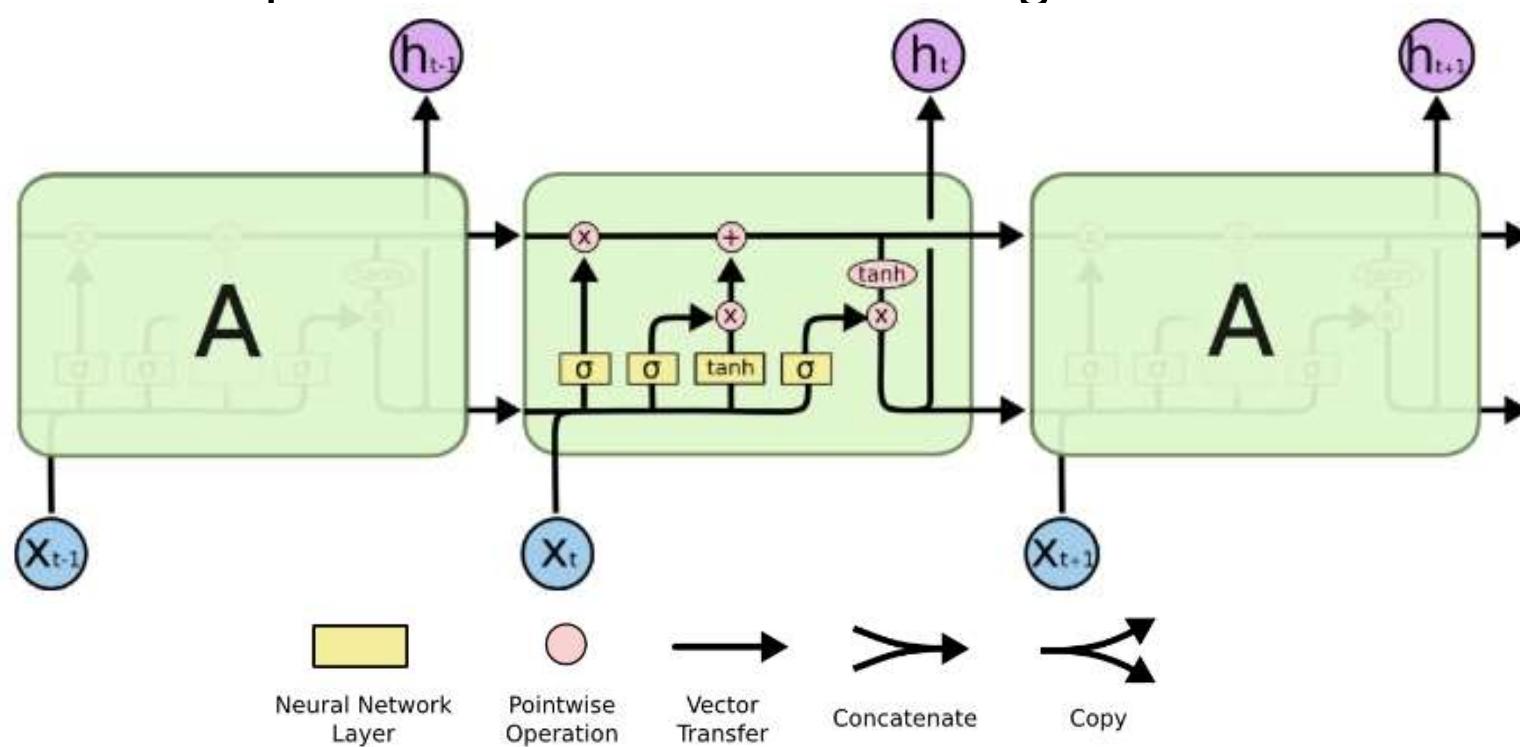
Encodeur : Encode les données d'entrée dans un vecteur d'état

Décodeur : Décode un vecteur d'état pour générer une prédiction



Réseaux LSTM

- LSTM : Long Short Term Memory
- Peuvent capturer de l'information à long terme



Frameworks

Code Python



```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

```
model_tree = DecisionTreeClassifier()
model_tree.fit(X_train,y_train)
y_pred = model_tree.predict(X_test)#Accuracy

model_boost = GradientBoostingRegressor(max_depth=1, n_estimators=1, learning_rate=0.5)
model_boost.fit(X_train, y_train)
y_pred_boost = model_boost.predict(X_test)
```

Code R :



```
model.svm <- svm(X_train, Y_train, kernel='radial', cost=10, type=C)
y.pred.svm <- predict(model.svm, X_test)
```



Code Keras :

```
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.optimizers import SGD

N, D, H = 64, 1000, 100

model = Sequential()
model.add(Dense(input_dim=D, output_dim=H))
model.add(Activation('relu'))
model.add(Dense(input_dim=H, output_dim=D))

optimizer = SGD(lr=1e-0)
model.compile(loss='mean_squared_error',
              optimizer=optimizer)

x = np.random.randn(N, D)
y = np.random.randn(N, D)
history = model.fit(x, y, nb_epoch=50,
                     batch_size=N, verbose=0)
```