Formation Machine Learning #1

Yannick Benezeth
Université de Bourgogne
Laboratoire ImViA

Plan

1. Contexte et concepts fondamentaux du Machine Learning
2. Transformation des données
   - Analyse en Composantes Principales
3. Modélisation des données
   - K plus proches voisins
   - Régression linéaire
   - Descente de gradient
   - Régression logistique
   - SVM
   - Decision tree – Random Forest
   - Réseaux de neurones (MLP, CNN, RNN, LSTM)

1. Contexte et concepts fondamentaux du Machine Learning

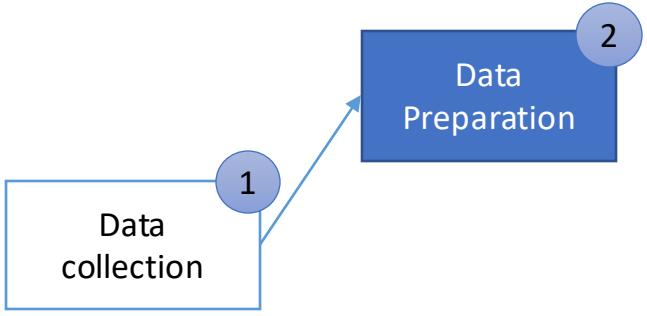
Les étapes classiques d'un projet de Machine Learning

1 Data collection

- Rassembler le plus de données possible (on triera plus tard)
- Multiplier les sources
- Souvent, c'est le point clé du process (ex : Google)

1. Contexte et concepts fondamentaux du Machine Learning

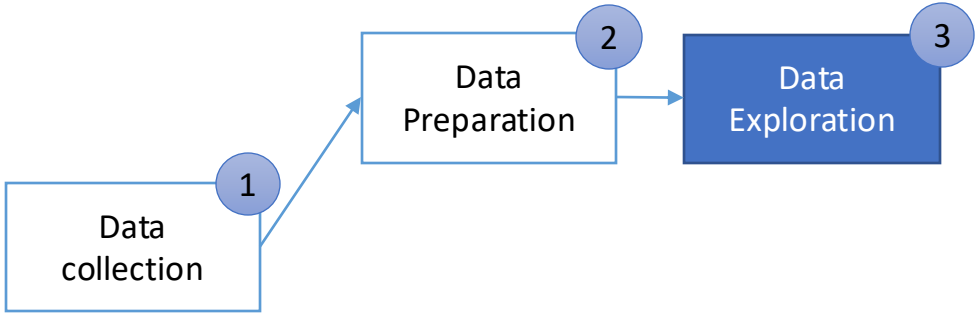
Les étapes classiques d'un projet de Machine Learning



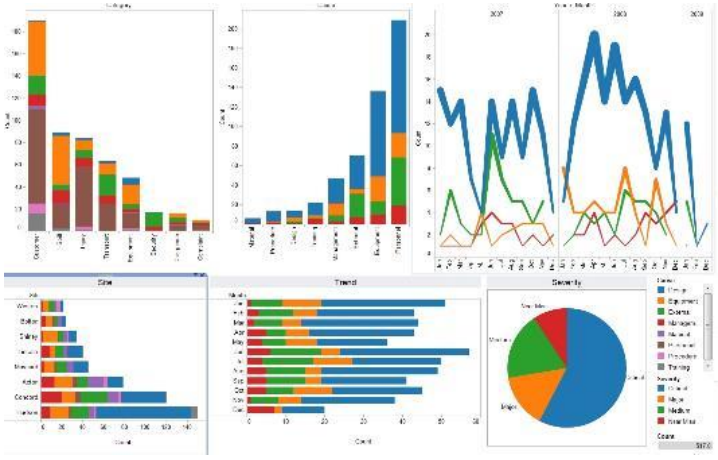
- Transformer ces données d'un format « brut », en un format plus approprié
- En général, on fait ça à la main (ou avec des scripts *custom*)
- Par exemple : mettre ses données dans le même format, filtrer, consolider (["Ouvriers", "Cadres"] vs ["O";"C"] vs [1;2]), standardiser, nettoyage des données (suppression des données incorrectes, incomplètes, mal formatées, …)…

1. Contexte et concepts fondamentaux du Machine Learning

Les étapes classiques d'un projet de Machine Learning

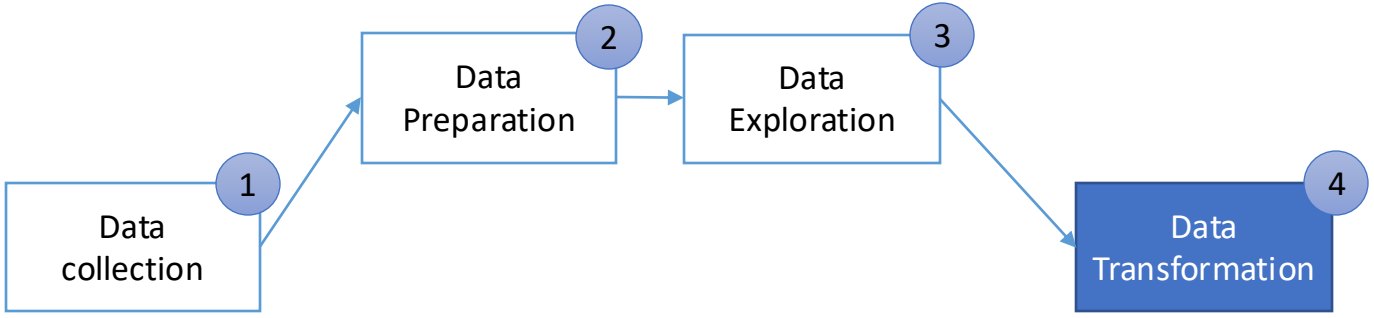


- Extraction de statistiques sur les données
- Visualisation des données par une représentation graphique



1. Contexte et concepts fondamentaux du Machine Learning

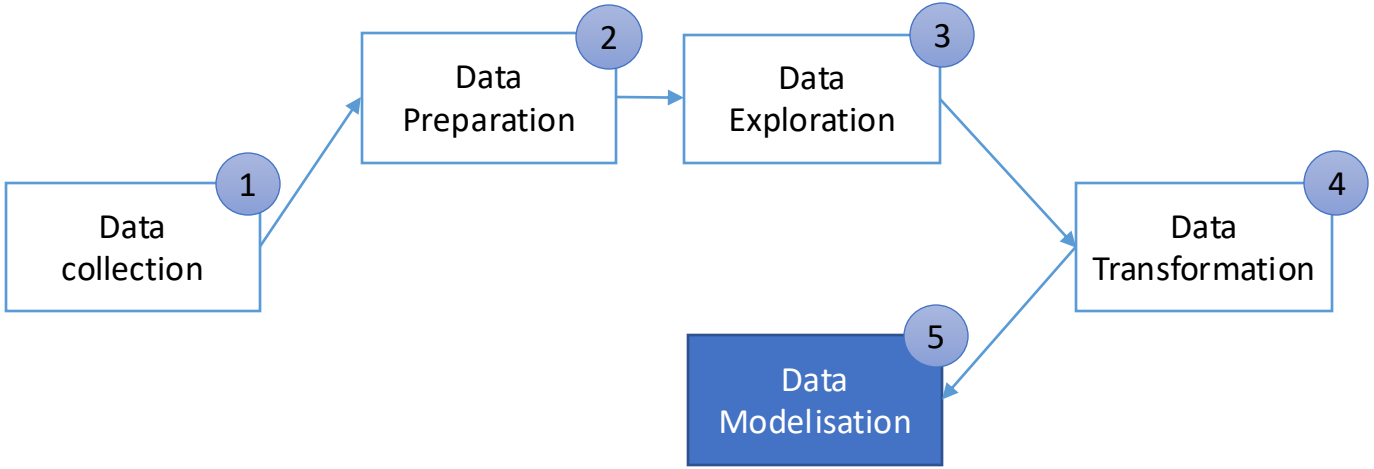
Les étapes classiques d'un projet de Machine Learning



– Préparation des données pour la modélisation (p. ex. réduire la dimension, normaliser, filtrer…)

1. Contexte et concepts fondamentaux du Machine Learning

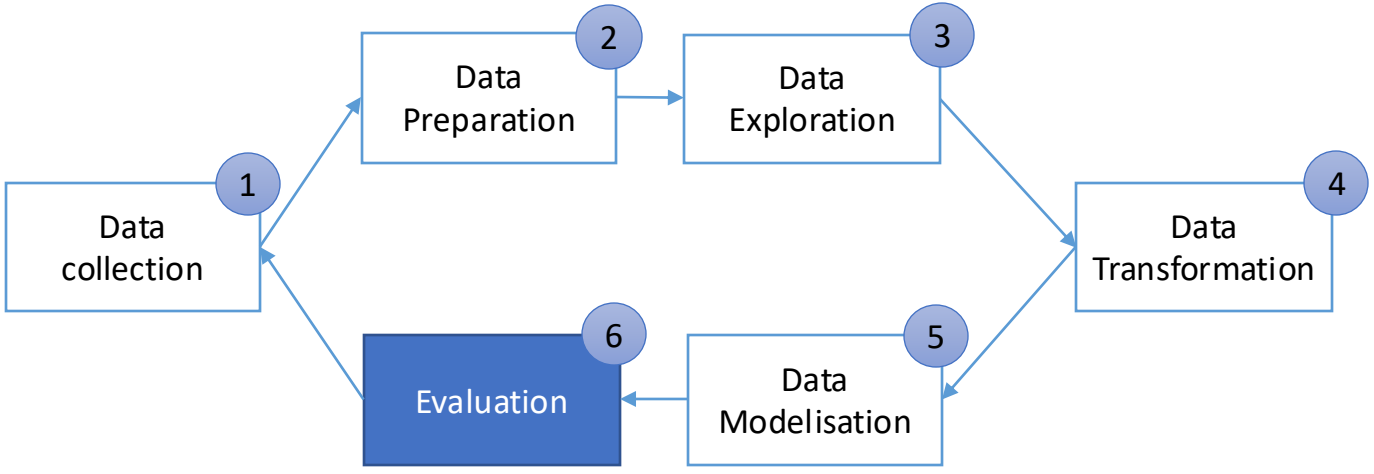
Les étapes classiques d'un projet de Machine Learning



– On utilise des algorithmes de *machine learning* pour apprendre des règles et des liens entre nos données.

1. Contexte et concepts fondamentaux du Machine Learning

Les étapes classiques d'un projet de Machine Learning



– Evaluation et bien souvent on recommence…

1. Contexte et concepts fondamentaux du Machine Learning

Le paradigme du *machine learning*

- Définition (Arthur Samuel - 1959) : étude de ce qui permet aux machines de réaliser des actions sans avoir été explicitement programmées pour cela.

- Sans machine learning : le programmeur apprend à un ordinateur à réaliser une tâche en lui donnant une liste détaillée d'instructions (*hard coding*)
- Avec machine learning
  - L'ordinateur va être capable de réaliser une tâche en examinant des données d'exemple.
  - Un même algorithme peut résoudre des problèmes différents si on lui donne des données différentes (reconnaitre un visage dans une image, reconnaitre un spam, prédire le prix d'une action…)
  - Il est possible de parcourir, analyser des milliers (voire plus) de lignes de données pour en déduire des liens et des informations pertinentes.

1. Contexte et concepts fondamentaux du Machine Learning

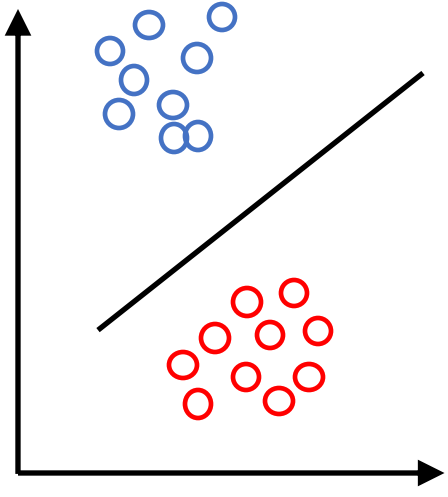
Utilisateur du *machine learning*

- Certains problèmes ne peuvent pas être résolus avec du *Machine Learning* (**ce n'est pas magique**)
- Certains problèmes ne doivent pas être résolus avec du *Machine Learning* (e.g. s'il n'est pas nécessaire d'analyser beaucoup de données, si on peut déduire des règles de classification...)
- A contrario, si j'ai beaucoup de données, des problèmes de passage à l'échelle, beaucoup de paramètres à régler, le *Machine Learning* est bien adapté.

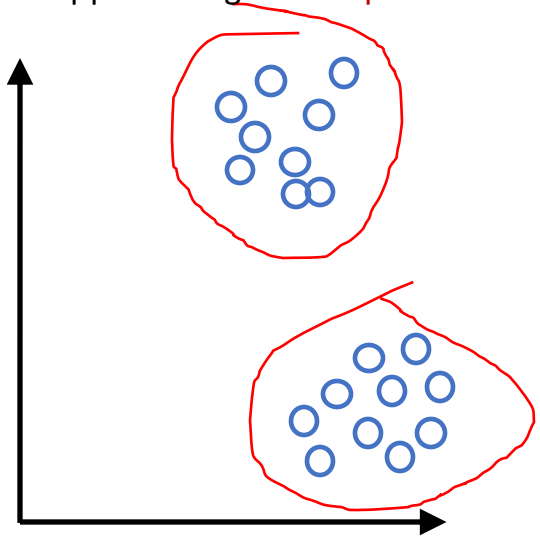
- Pas besoin d'être un expert en *Machine Learning* pour utiliser des algorithmes de *Machine Learning*
  - Il existe plein d'outils open source : Tensor Flow, Scikit-learn…
  - Il faut avoir les connaissances de base pour bien utiliser ces outils

1. Contexte et concepts fondamentaux du Machine Learning

Apprentissage supervisé



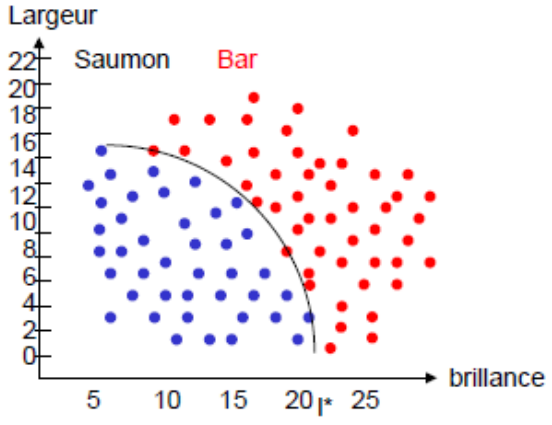
Apprentissage non supervisé



yannick.benezeth@u-bourgogne.fr

1. Contexte et concepts fondamentaux du Machine Learning

Classification



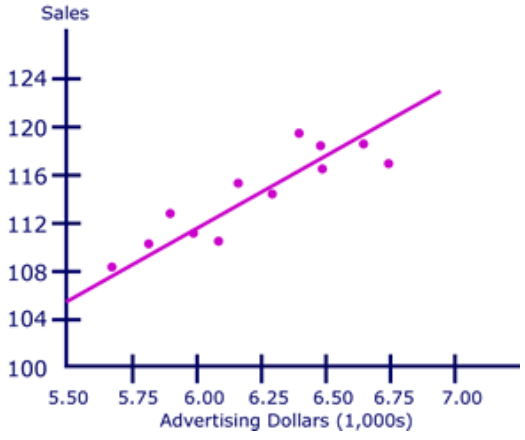
$y = f_\theta(x)$ avec $y \in \{0,1\}$, f le modèle, θ les paramètres du modèle et x les données en entrée.

Exemples

- Marquer les emails comme spam/non spam
- Reconnaitre un visage dans une image
- A partir d'un ensemble de symptômes, déterminer la maladie dont souffre une personne

1. Contexte et concepts fondamentaux du Machine Learning

Régression



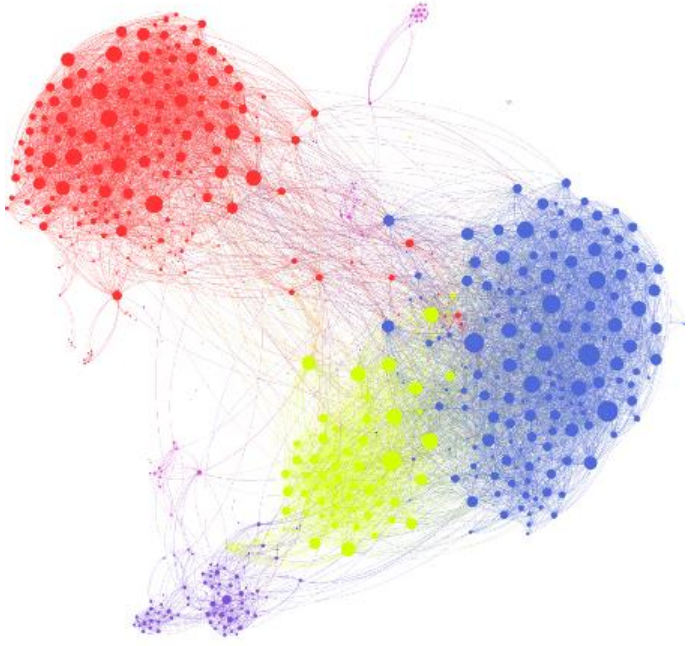
$y = f_\theta(x)$ avec $y \in R$, f le modèle, θ les paramètres du modèle et x les données en entrée.

Exemples :

- Calculer l'équation qui permet de prédire le prix d'une maison en fonction de sa superficie.
- Existe-t-il une corrélation entre le nombre d'heures passées à jouer aux jeux vidéo et les résultats d'un étudiant ?

1. Contexte et concepts fondamentaux du Machine Learning

Clustering



Exemples :

- Rassembler les personnes qui se ressemblent dans un réseau social.
- Chercher des caractéristiques communes de personnes souffrant de la même maladie.

1. Contexte et concepts fondamentaux du Machine Learning

Représentation des données

Habituellement les données sont représentées sous la forme de matrices (tableau 2D)

Features, caractéristiques,
dimensions, colonnes, variables, …

Samples
Exemples
Données

	Feature_0	Feature_1	Feature_2
Sample_0			
Sample_1			
Sample_2			
Sample_3			

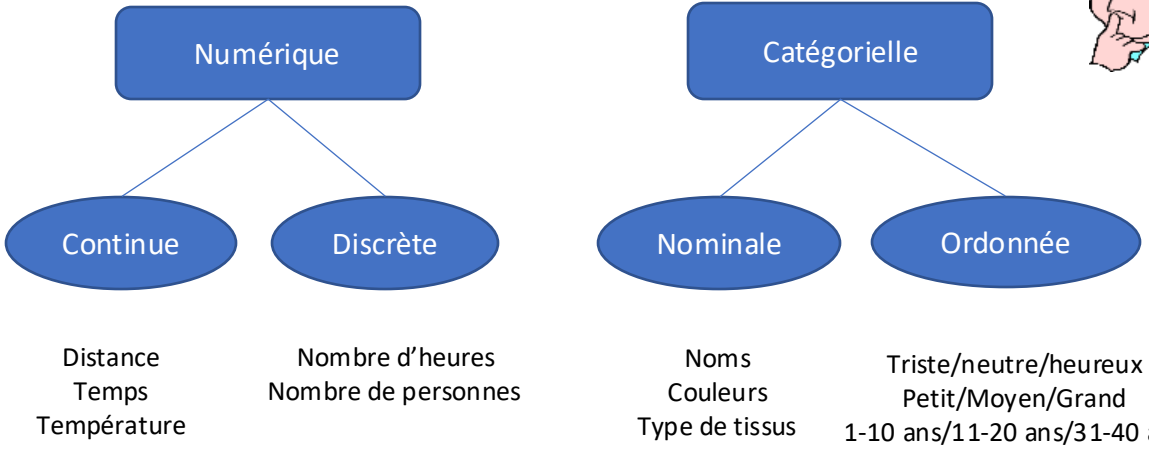
Prénom	**Taille (m)**	**Couleur**
Eric	1.75	Blond
Jean	1.90	Blond
Michel	1.63	Brun
Hélène	1.75	Brun

1. Contexte et concepts fondamentaux du Machine Learning

Représentation des données

Différentes categories de features

- Une note entre 0 et 20 ?
- Des espèces 1, 2 ou 3 ?



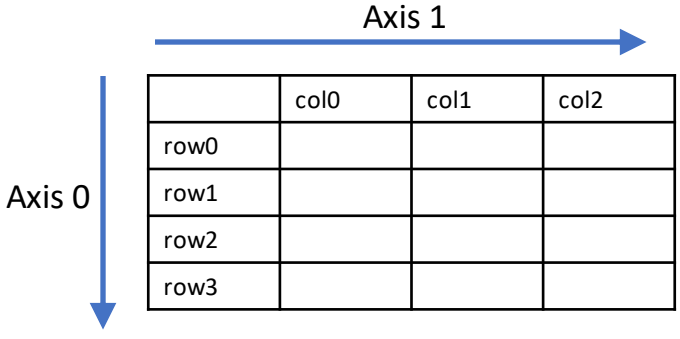
- Variables numériques
  - Je peux calculer une distance entre mes features
  - Les variables numériques discrètes peuvent être n'importe quelle variable numérique continue après quantification
- Variables catégorielles
  - Il y a un nombre spécifique de valeurs possibles
  - Je ne peux pas calculer de distance entre elles mais elles peuvent (ou pas) être ordonnées

1. Contexte et concepts fondamentaux du Machine Learning

Manipuler des données avec Pandas

Les dataframes et les series

- Avec *Pandas*, les données sont stockées dans des *DataFrame*



- Un *dataFrame* est une collection de *Series* (~ *NumPy Lists* mais avec un contenu homogène, d'un seul type)

Notebook 1

yannick.benezeth@u-bourgogne.fr

1. Contexte et concepts fondamentaux du Machine Learning

Représenter les features

Pour utiliser des algorithmes de *Machine Learning*, il faut formater mes données sous la forme de vecteurs de valeurs numériques.

Notebook 2

yannick.benezeth@u-bourgogne.fr

1. Contexte et concepts fondamentaux du Machine Learning

Visualiser les données

Notebook 3

yannick.benezeth@u-bourgogne.fr

1. Contexte et concepts fondamentaux du Machine Learning

Importance du partitionnement des données

Exemple

Je veux entrainer un algorithme de Machine Learning pour apprendre à prédire la couleur des yeux d'une personne en fonction de son âge.

$y = f(x)$ avec $y \in \{Bleu, Marron, Vert\}$ et x l'âge.

Je vais l'entraîner sur des exemples :

Âge	Couleur des yeux
25	Bleu
32	Marron
30	Marron
45	Vert

Pour évaluer le classifieur, je lui présente les 4 exemples ci-dessus.

$y = f(25)$? $y = f(32)$? $y = f(30)$? $y = f(45)$?

yannick.benezeth@u-bourgogne.fr

Quel est le problème ? Quel biais ? Quelle solution ?

1. Contexte et concepts fondamentaux du Machine Learning

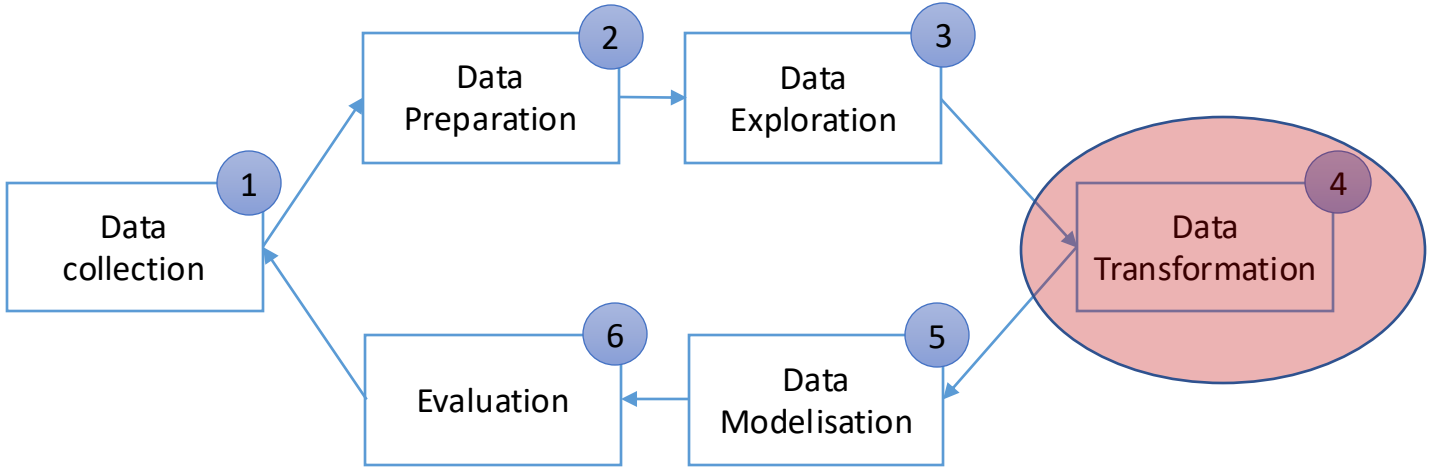
Importance du partitionnement des données

Base d'apprentissage

Base de test

Âge	Couleur des yeux
25	Bleu
32	Marron
30	Marron
45	Vert
50	Marron
40	Bleu
20	Marron
15	Marron
…	…
26	Vert
20	Bleu
18	Marron
19	Marron

2. Transformation des données



2. Transformation des données

La transformation des données est souvent appelée pré-traitement

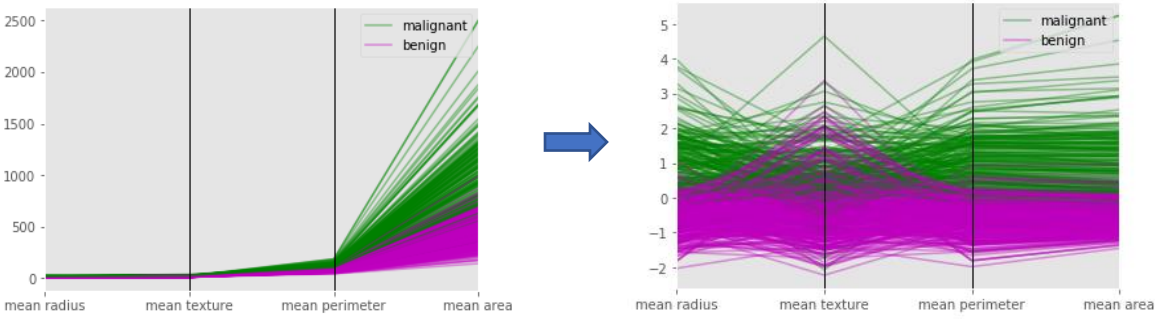
Les principales techniques

- Filtrage pour réduire le bruit (traitement du signal)
- La normalisation pour ne pas être impacté par des features avec des plages de valeurs très différentes

Normalisation centrée réduite

(Moyenne=0 et variance=1)

$$x' = \frac{x - \mu}{\sigma}$$

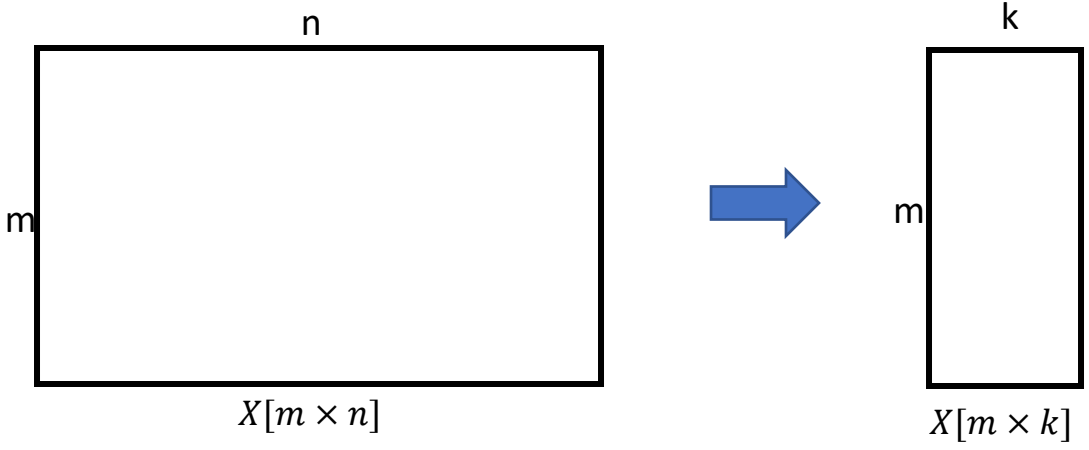


2. Transformation des données

La transformation des données est souvent appelée pré-traitement

Les principales techniques

- Sélection des features (p.ex. on garde les features qui contribuent le plus pour la classification)
- Réduction des dimensions (p. ex. avec *l'Analyse en Composantes Principales*)



2. Transformation des données

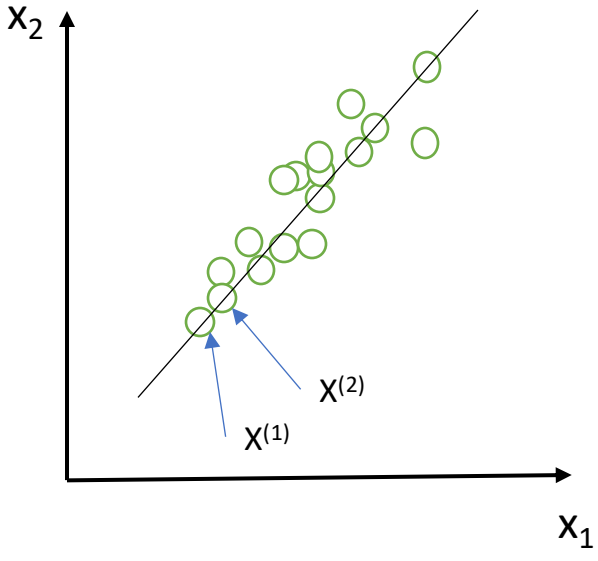
Objectifs de la réduction de dimension des données

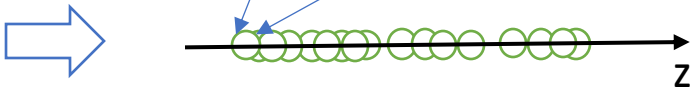
- Accélérer les traitements ultérieurs
- Améliore souvent les performances des algorithmes de Machine Learning (problème de *sur-apprentissage*, etc…)
- Permet de visualiser les données de dimension supérieure à 3

L'Analyse en Composantes Principales est très utilisée en Machine Learning

2. Transformation des données

Principe de la réduction de la dimension





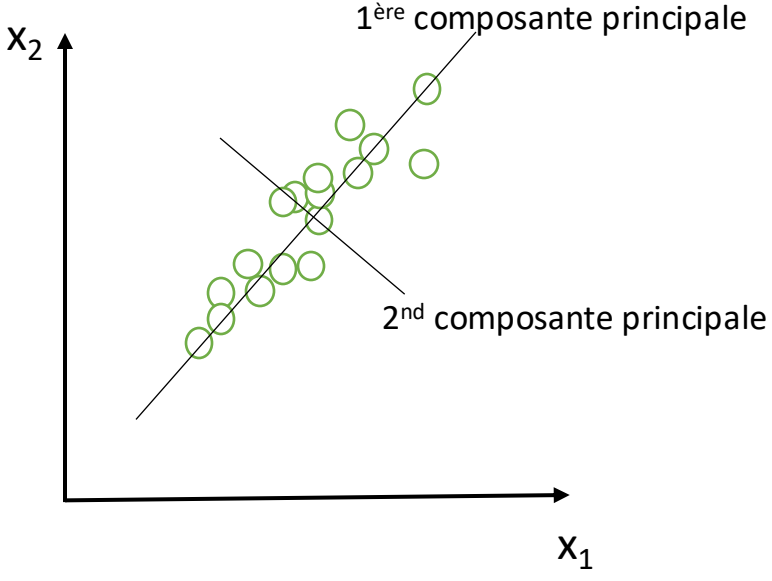
Réduction des données de 2D vers 1D

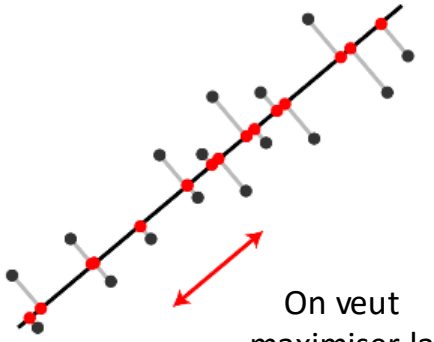
$X^{(1)} \in R^2 \rightarrow Z^{(1)} \in R$

$X^{(2)} \in R^2 \rightarrow Z^{(2)} \in R$

2. Transformation des données

Analyse en Composantes Principales (ACP)





2. Transformation des données

Analyse en Composantes Principales (ACP)

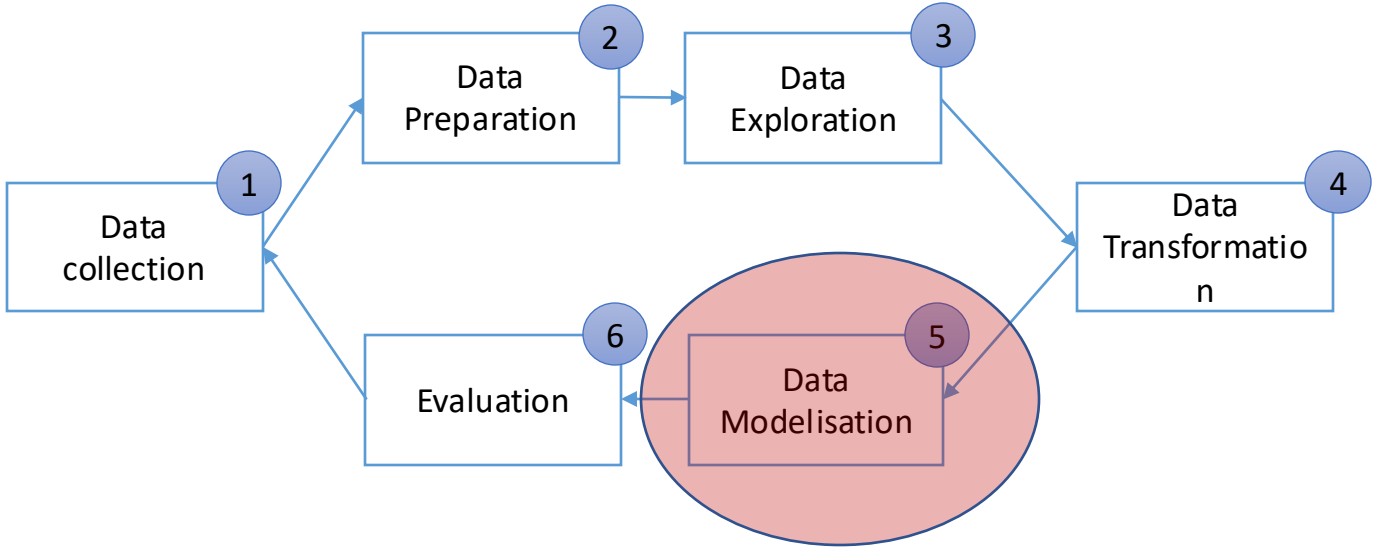
V1: Apply PCA manually and rconstruct the data

```
#1 Normalize the data
mu = X.mean(axis=0)
s = X.std(axis=0)
X_norm = (X-mu)/s

#2 Calculate the covariance
Sigma = np.cov(X_norm.T)

#3 Calculate the eigenvectors
U, S, V = np.linalg.svd(Sigma)

#4 Reduce dimension
Ureduced = U[:, 0:K]
Z = np.dot(X_norm,Ureduced)

#5 Reconstruct the data
X_rec  = np.dot(Z, Ureduced.T)
```

V2: Apply PCA with scikit-learn and reconstruct the data

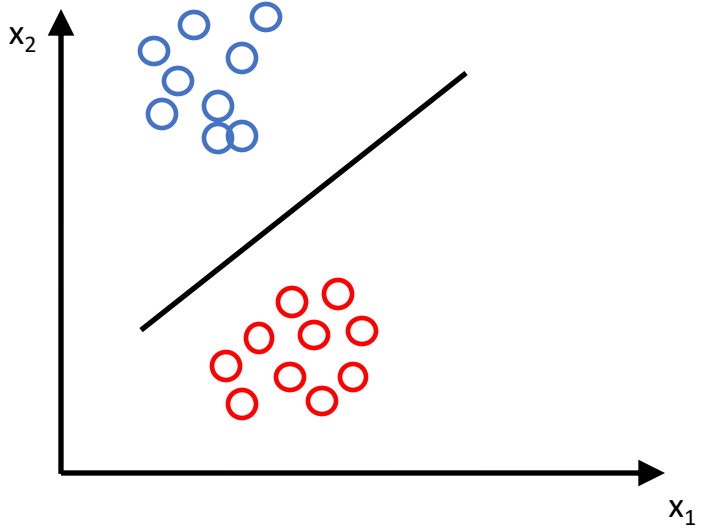
```
# Create a PCA object and specify the number of
# components to keep (k)
pca = PCA(n_components=K)

# Normalize and transform the data
Z = pca.fit_transform(X)

# Reconstruct the data
X_reconstructed = pca.inverse_transform(Z)
```

3. Modélisation des données



3. Modélisation des données

Apprentissage supervisé



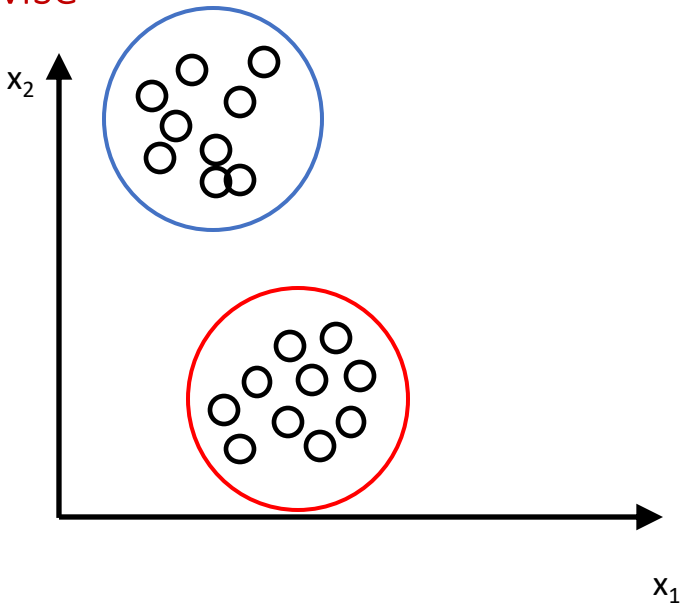
Ensemble d'apprentissage : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

Où j'ai un ensemble de m points labellisés, $x^{(i)} = \left(x_1^{(i)}, x_2^{(i)}\right)$ et $y^{(i)} = 0 \; ou \; 0$.

L'objectif de l'apprentissage supervisé est de trouver la frontière entre les classes.

3. Modélisation des données

Apprentissage non-supervisé



Ensemble d'apprentissage : $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

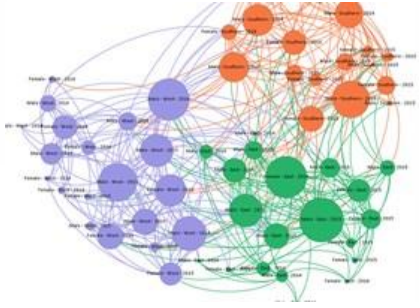
Où j'ai un ensemble de m points, $x^{(i)} = \left(x_1^{(i)}, x_2^{(i)}\right)$

L'objectif de l'apprentissage non-supervisé est de trouver la structure des données, par exemple en regroupant les points similaires (=clustering)

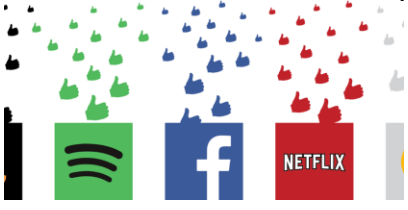
3. Modélisation des données

Applications du clustering

Marketing



Analyse de réseaux sociaux



Les systèmes de recommandations

Quelques algorithmes de Clustering :

K-means :
- Simple et rapide
- Sensible aux valeurs initiales et nécessite de connaître k

Clustering hiérarchique :
- Crée une hiérarchie de clusters, visualisation de la hiérarchie avec un dendrogramme
- Agglomératif ou divisif
- Complexe pour les grands jeux de données

Mean-Shift :
- Densité : Recherche les modes de la distribution de densité.
- Pas besoin de spécifier k
- Peut être lent pour les grands jeux de données.

Algorithmes de clustering basés sur les modèles (e.g. GMM) :
- Modélise les clusters comme des distributions de probabilité.
- Plus complexe à implémenter et à exécuter.

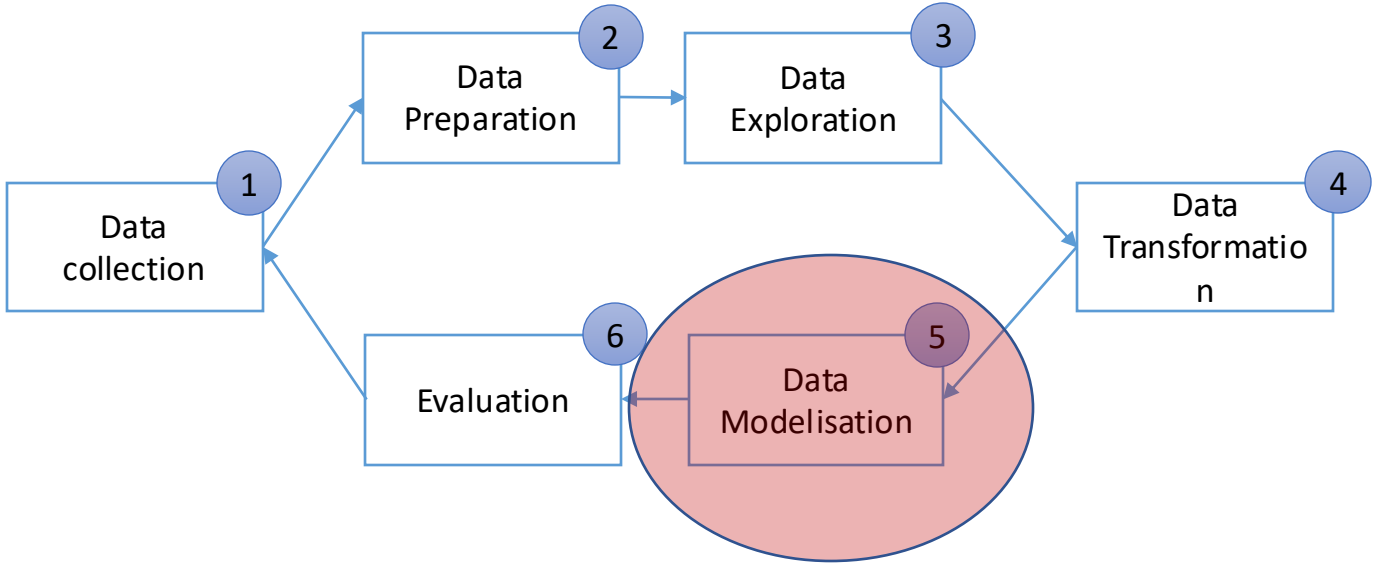
3. Modélisation des données

Algorithme des K-moyennes

Comment regrouper ces points en 2 classes ?

1. On initialise le centre des *K* classes au hasard
2. Chaque point est affecté à la classe dont le centre lui est le plus proche
3. On met à jour la moyenne de chaque classe
4. Retour à l'étape 2 jusqu'à convergence (la moyenne de chaque classe est stable

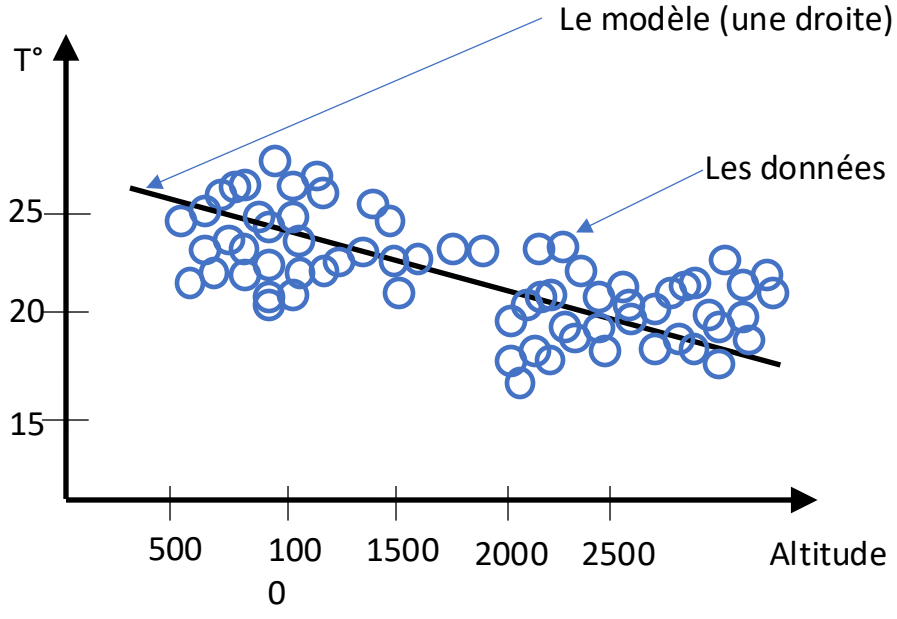
3. Modélisation des données



3. Modélisation des données

Régression linéaire

Exemple

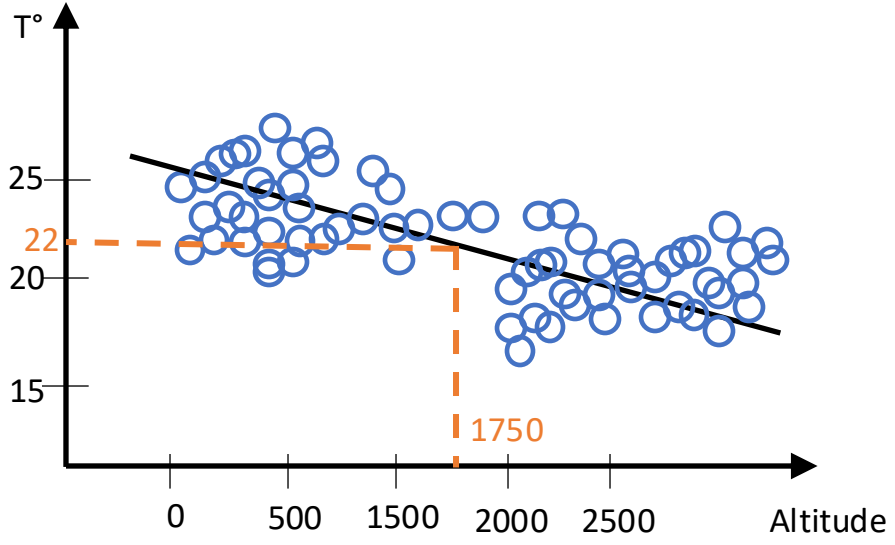


On modélise la relation entre les températures moyennes annuelles et l'altitude.

3. Modélisation des données

Régression linéaire

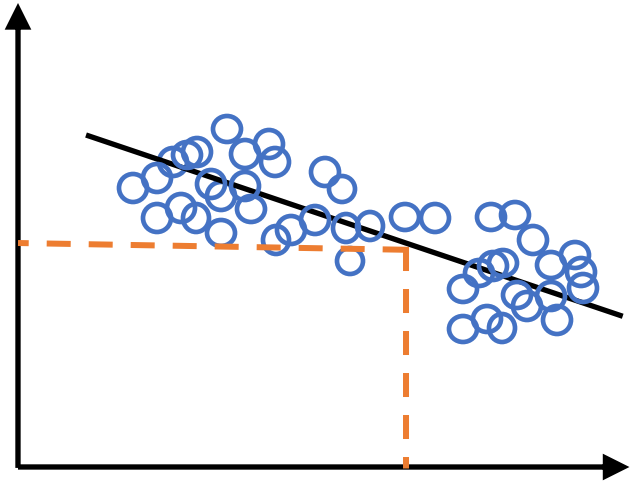
Exemple



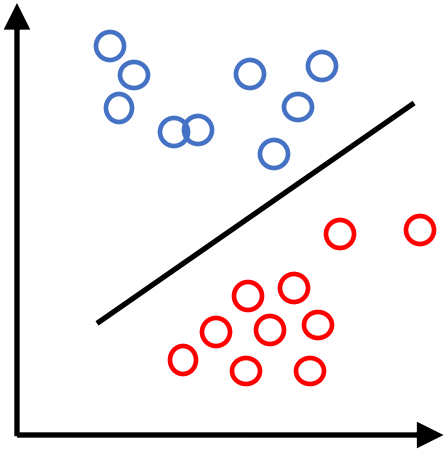
Je peux utiliser ce modèle pour prédire la température d'une ville en fonction de son altitude

3. Modélisation des données

Classification vs régression



Régression



Classification

Régression : prédire une valeur continue en fonction d'un exemple

Classification : prédire un label (valeur discrète) en fonction d'un exemple

3. Modélisation des données

Notation

Base de données des températures en fonction de l'altitude

x = variables d'entrée, caractéristiques

y = variables de sortie

$(x^{(i)}, y^{(i)})$ le $i^{ième}$ exemple

$x^{(1)}=1759$

$x^{(4)}=456$

$y^{(2)}=17.7$

Altitude en m (x)	Température en °C (y)
1759	19.0
2043	17.7
356	22.8
456	23.0
1098	20.8
2003	17.9
1650	19.5

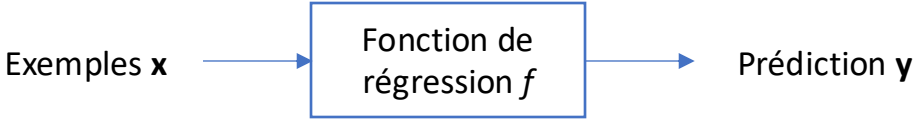
m le nombre d'exemples

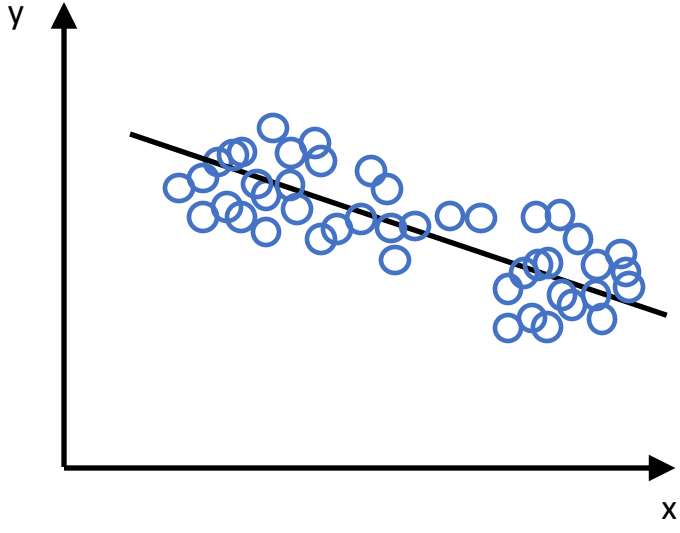
yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

Régression linéaire

Je vais chercher une fonction *f* permettant d'estimer les **y** à partir des **x**





En régression linéaire la fonction *f* est représentée par l'équation d'une droite :

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

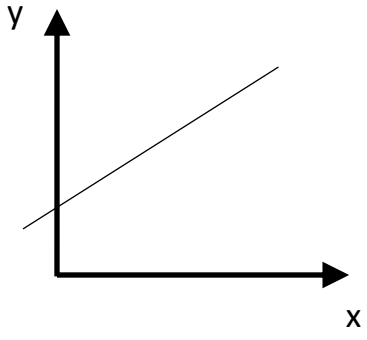
3. Modélisation des données

Paramètres de la régression linéaire

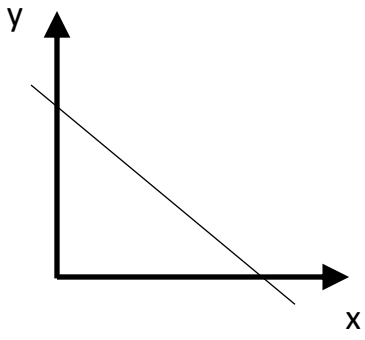
L'apprentissage consiste à déterminer les meilleurs θ_0 et θ_1 en fonction de nos données

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

Différents θ_0 et θ_1 donnent différents modèles f



$\theta_0 = 1$ et $\theta_1 = 1$



$\theta_0 = 3$ et $\theta_1 = -1$

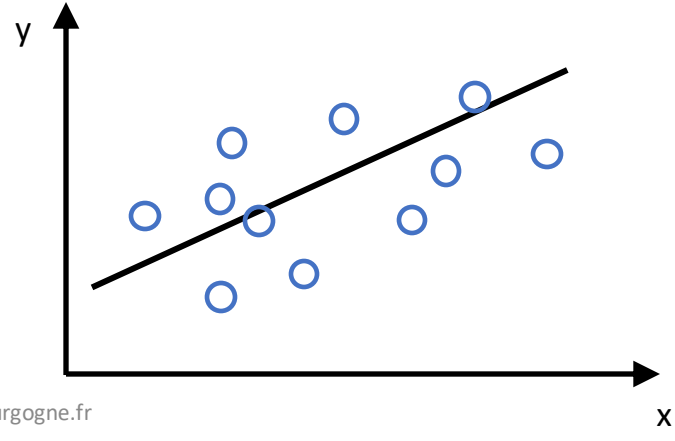
3. Modélisation des données

Fonction de coût

L'apprentissage consiste à déterminer les meilleurs θ_0 et θ_1 en fonction de nos données

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

L'apprentissage consiste à déterminer θ_0 et θ_1 de telle façon que *f(x)* soit proche des *y* pour nos exemples d'apprentissage *(x,y)*



Mathématiquement :

$$\min_{\theta_0,\theta_1} \frac{1}{m} \sum_{i=1}^{m} \left(f\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Je définis ma fonction de coût

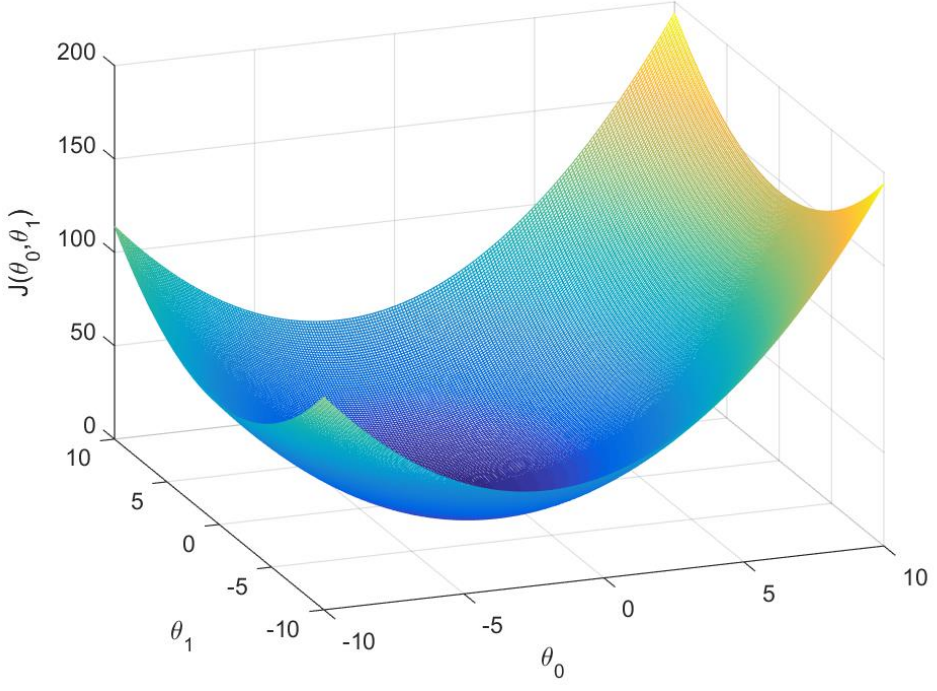
$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(f\left(x^{(i)}\right) - y^{(i)}\right)^2$$

appelée l'erreur quadratique moyenne

3. Modélisation des données

Fonction de coût

Représentation graphique d'une fonction de coût

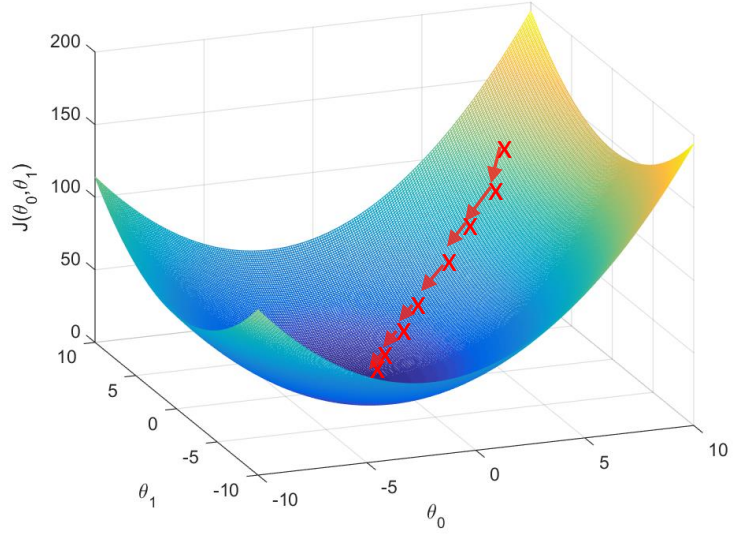


3. Modélisation des données

Algorithme de la descente de gradient

Entrée : Fonction $J(\theta_0, \theta_1)$

- Initialise θ_0 et θ_1
- Je change les valeurs de θ_0 et θ_1 pour réduire $J(\theta_0, \theta_1)$ jusqu'à convergence



yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

Algorithme de la descente de gradient

Faire jusqu'à convergence {

$\theta_j := \theta_j - \alpha \frac{\partial J(\theta_0,\theta_1)}{\partial \theta_j}$ (pour *j=0* et *j=1*)

}

α : learning rate (longueur du pas)

$\frac{\partial f(x,y)}{\partial x}$: dérivée partielle de f par rapport à x

Implémentation correcte :

$temp0 := \theta_0 - \alpha \frac{\partial J(\theta_0,\theta_1)}{\partial \theta_0}$

$temp1 := \theta_1 - \alpha \frac{\partial J(\theta_0,\theta_1)}{\partial \theta_1}$

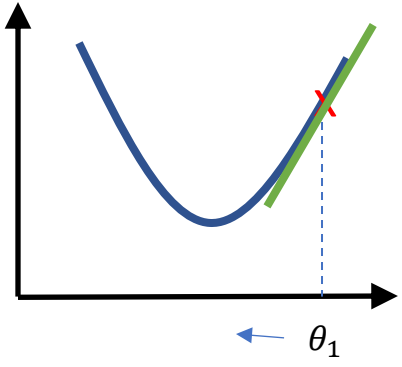
$\theta_0 := temp0$

$\theta_1 := temp1$

yannick.benezeth@u-bourgogne.fr

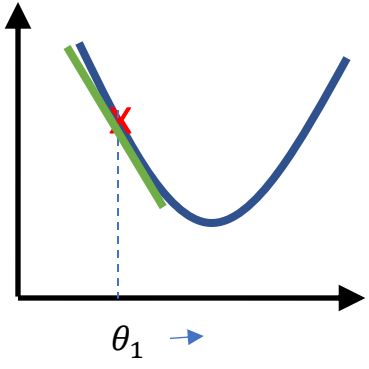
3. Modélisation des données

Algorithme de la descente de gradient : exemple 1D - $J(\theta_1)$



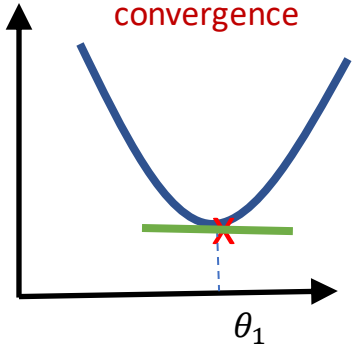
$$\theta_1 := \theta_1 - \alpha \frac{d\,J(\theta_1)}{d\theta_1}$$

$$\theta_1 := \theta_1 - \alpha(nb\ positif)$$



$$\theta_1 := \theta_1 - \alpha \frac{d\,J(\theta_1)}{d\theta_1}$$

$$\theta_1 := \theta_1 - \alpha(nb\ négatif)$$

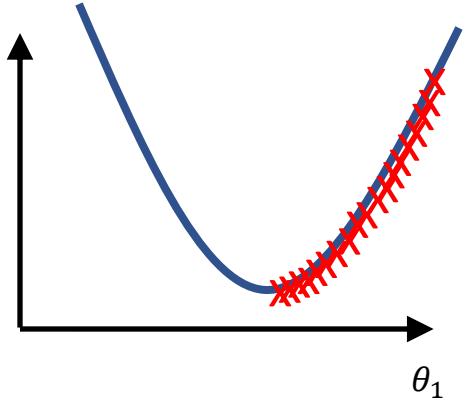


$$\theta_1 := \theta_1 - \alpha \frac{d\,J(\theta_1)}{d\theta_1}$$

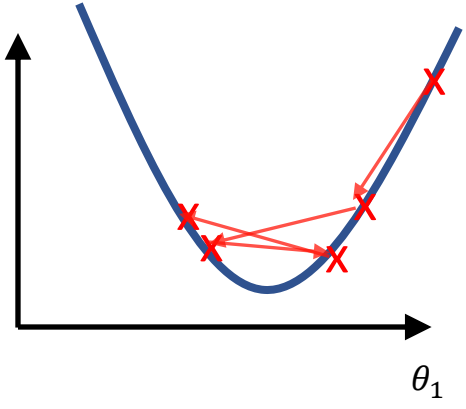
$$\theta_1 := \theta_1 - \alpha.0$$

3. Modélisation des données

Descente de gradient - Importance du paramètre α



Si α est trop petit, la descente de gradient sera très lente



Si α est trop grand, la descente de gradient peut manquer le minimum (voire ne jamais converger = diverger)

3. Modélisation des données

Algorithme de la descente de gradient

Faire jusqu'à convergence {

$\theta_j := \theta_j - \alpha \frac{\partial J(\theta_0,\theta_1)}{\partial \theta_j}$ (pour *j=0* et *j=1*)

}

On va utiliser la descente de gradient pour minimiser la fonction de coût de la régression linéaire $J(\theta_0, \theta_1)$

Modèle de régression linéaire

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

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(f\left(x^{(i)}\right) - y^{(i)} \right)^2$$

3. Modélisation des données

On calcule $\frac{\partial J(\theta_0,\theta_1)}{\partial \theta_j}$ pour j=0 et j=1

$$J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(f\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$\frac{\partial J(\theta_0,\theta_1)}{\partial \theta_0} = \frac{2}{m}\sum_{i=1}^{m}\left(f\left(x^{(i)}\right) - y^{(i)}\right)$$

$$\frac{\partial J(\theta_0,\theta_1)}{\partial \theta_1} = \frac{2}{m}\sum_{i=1}^{m}\left(f\left(x^{(i)}\right) - y^{(i)}\right).x^{(i)}$$

Algorithme

Faire jusqu'à convergence {

$$\theta_0 := \theta_0 - \alpha\left(\frac{2}{m}\sum_{i=1}^{m}\left(f\left(x^{(i)}\right) - y^{(i)}\right)\right)$$

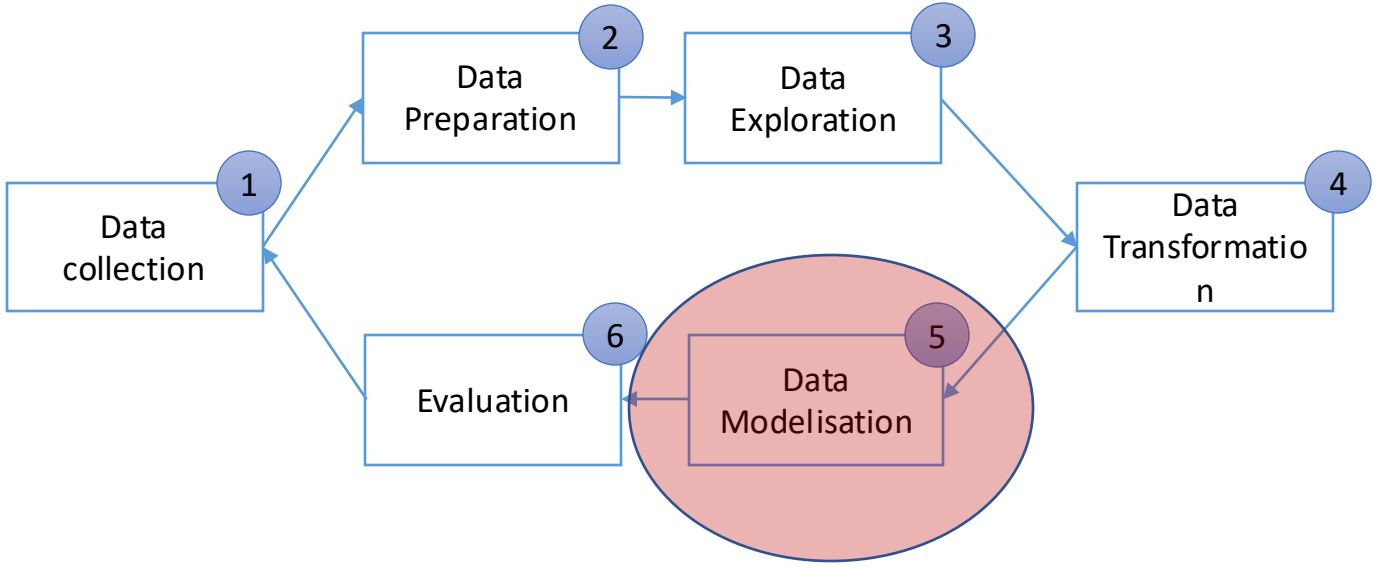
$$\theta_1 := \theta_1 - \alpha\left(\frac{2}{m}\sum_{i=1}^{m}\left(f\left(x^{(i)}\right) - y^{(i)}\right).x^{(i)}\right)$$

}

NB : mettre à jour θ_0 et θ_1 simultanément !

yannick.benezeth@u-bourgogne.fr

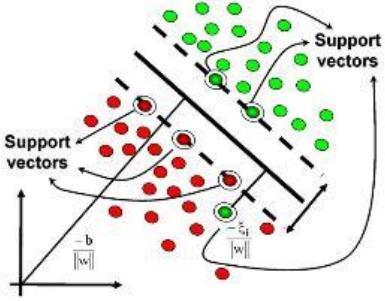
3. Modélisation des données

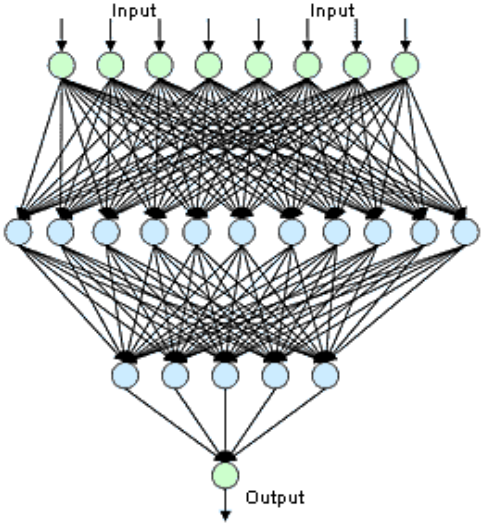


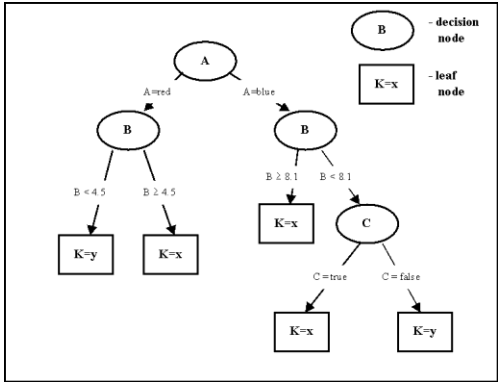
3. Modélisation des données

Les différentes techniques de classification

- Régression logistique
- K-NN
- Deep learning
- Réseaux de neurones
- Algorithme génétique
- Classification Bayesienne
- Machine à vecteurs de support
- Arbre de décision
- Adaboost
- …





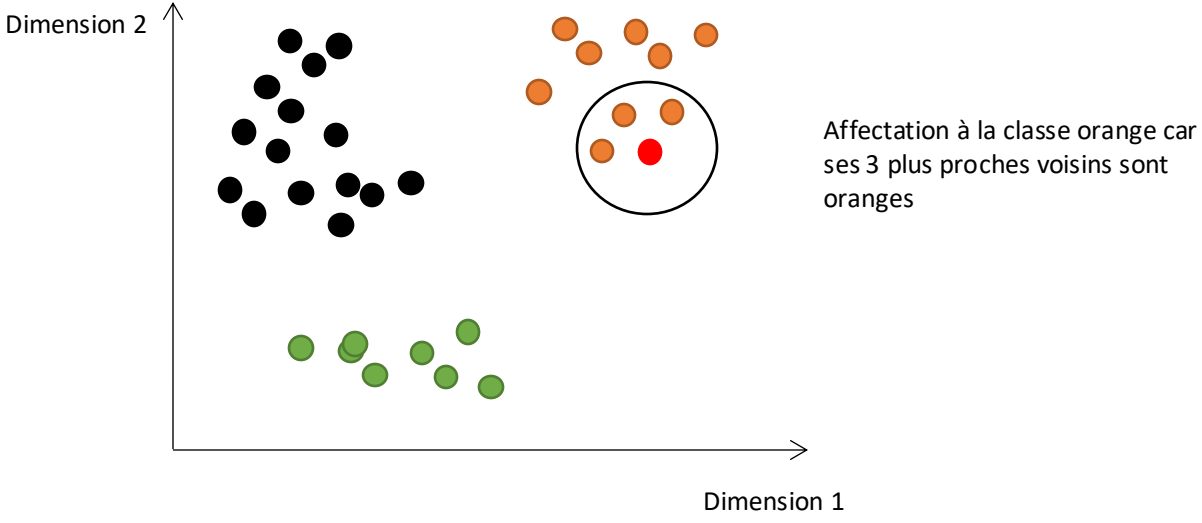


yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

k plus proches voisins (k-NN)

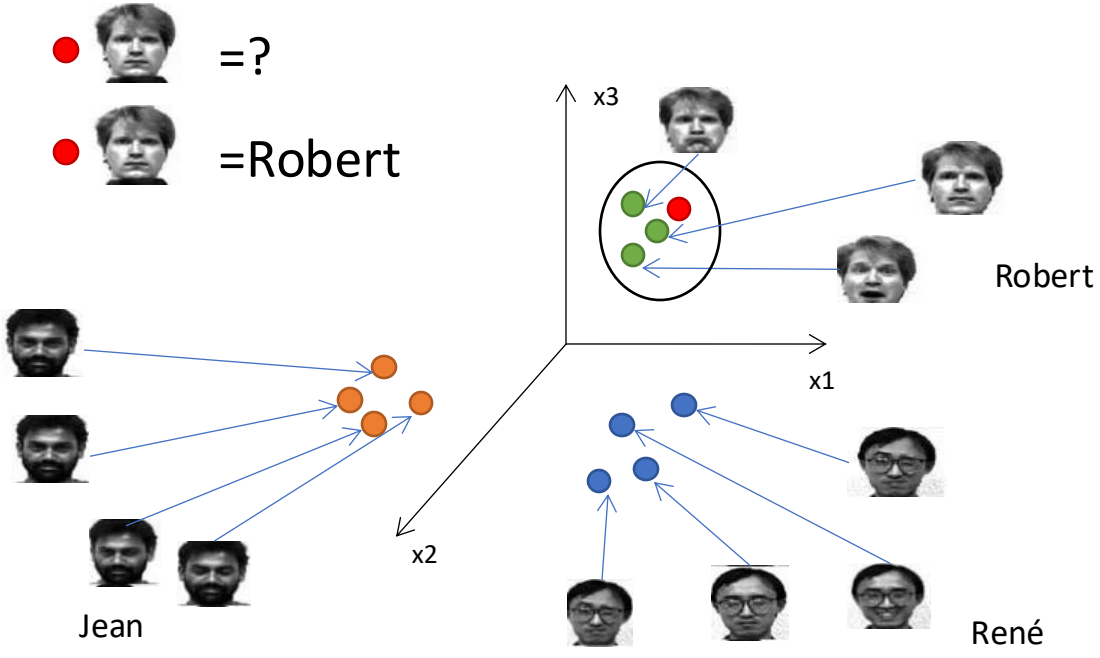
Pour estimer la classe d'un nouvel objet ● , on va prendre en compte la classe des k objets les plus proches.



3. Modélisation des données

k plus proches voisins (k-NN)

Pour estimer la classe d'un nouvel objet ● , on va prendre en compte la classe des k objets les plus proches.



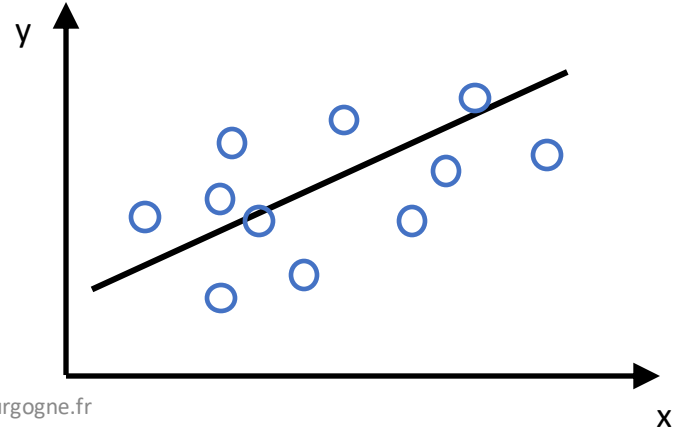
3. Modélisation des données

Rappel régression et notation

L'apprentissage de la régression linéaire consiste à déterminer les meilleurs θ_0 et θ_1 en fonction de nos données

$$f_\theta(x) = \theta_0 + \theta_1 x$$

L'apprentissage consiste à déterminer θ_0 et θ_1 de telle façon que $f_\theta(x)$ soit proche des *y* pour nos exemples d'apprentissage *(x,y)*



3. Modélisation des données

Rappel régression et notation

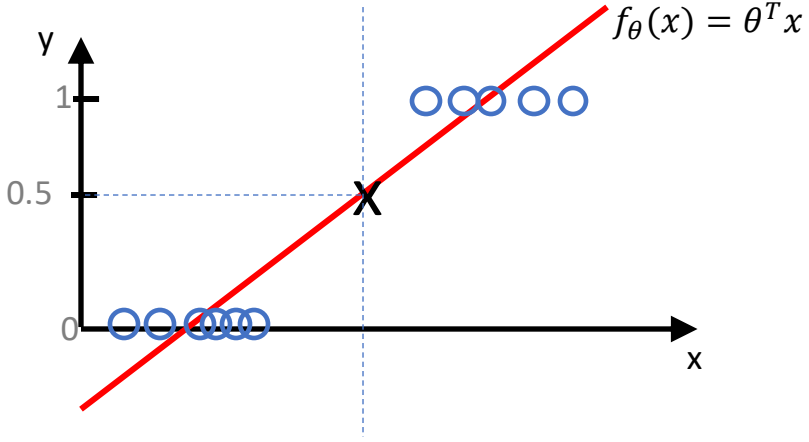
La fonction $f_\theta(x)$ est quelques fois écrite en notation matricielle

$$f_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$f_\theta(x) = \theta^T x = [\theta_0\ \theta_1\ \theta_2] \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

3. Modélisation des données

Peut-on utiliser la régression linéaire pour classifier des données ?

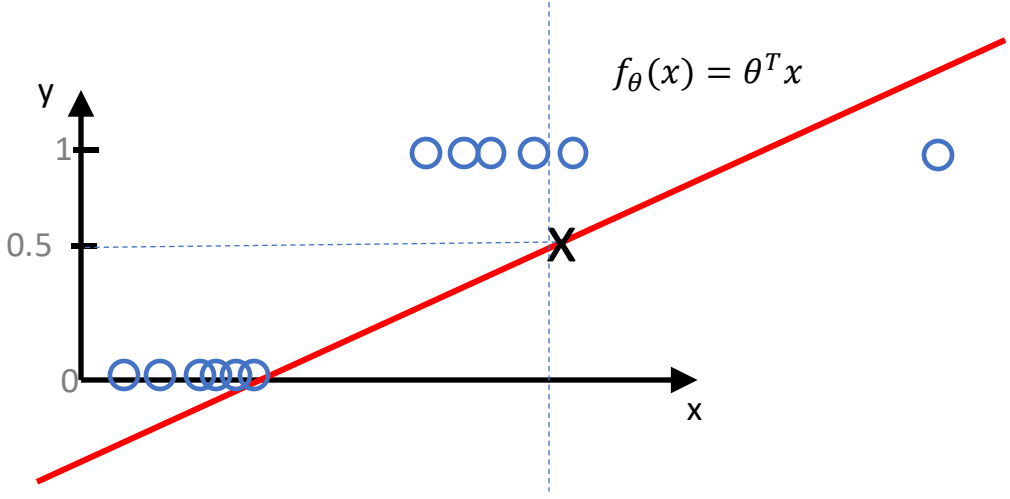


Si $f_\theta(x) \geq 0.5$, alors je prédis y=1

Si $f_\theta(x) < 0.5$, alors je prédis y=0

3. Modélisation des données

Peut-on utiliser la régression linéaire pour classifier des données ?



Bof !

Si $f_\theta(x) \geq 0.5$, alors je prédis y=1

Si $f_\theta(x) < 0.5$, alors je prédis y=0

3. Modélisation des données

Peut-on utiliser la régression linéaire pour classifier des données ?

Classification $y \in \{0,1\}$ mais $f_\theta(x)$ peut être >1 et <0

Régression logistique, on va utiliser une fonction $f_\theta(x)$ telle que $0 \leq f_\theta(x) \leq 1$

Malgré son nom trompeur, la régression logistique est utilisée pour la classification

yannick.benezeth@u-bourgogne.fr

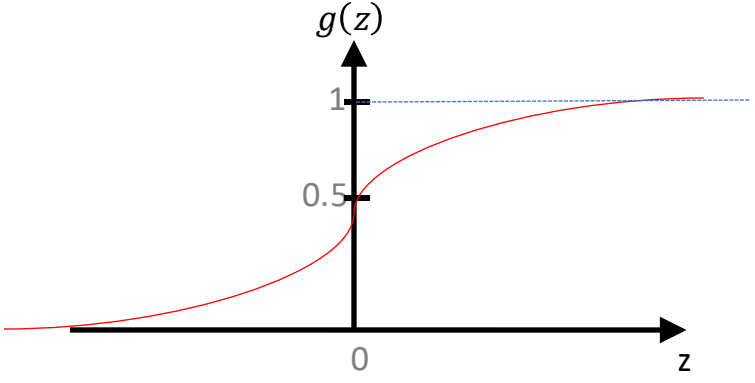
3. Modélisation des données

Fonction logistique

On veut une fonction $f_\theta(x)$ telle que $0 \leq f_\theta(x) \leq 1$

$f_\theta(x) = g(\theta^T x)$ où $g(z) = \frac{1}{1+e^{-z}}$

Ce qui donne $f_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

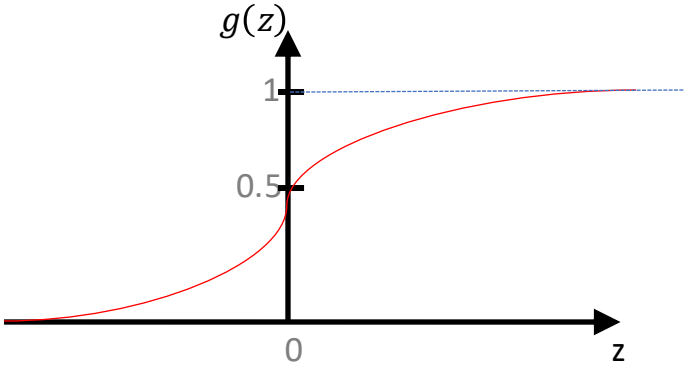


$g(z)$ est appelée la fonction logistique ou sigmoïde

3. Modélisation des données

Frontière de décision

$$f_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$



Si $f_\theta(x) \geq 0.5$, alors je prédis y=1

Si $f_\theta(x) < 0.5$, alors je prédis y=0

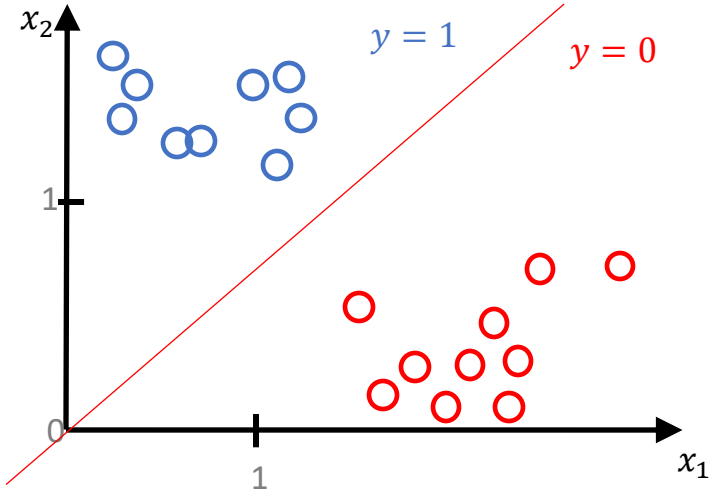
Or $f_\theta(x) \geq 0.5$ lorsque $\theta^T x \geq 0$ et $f_\theta(x) < 0.5$ lorsque $\theta^T x < 0$

Donc

Si $\theta^T x \geq 0$, alors je prédis y=1

Si $\theta^T x < 0$, alors je prédis y=0

3. Modélisation des données



Frontière de décision

$$f_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$f_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

Si je choisis $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$,

je prédis $y = 1$ si $\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \geq 0$ soit si $-x_1 + x_2 \geq 0$

Comment déterminer les paramètres θ ?

3. Modélisation des données

Déterminer les paramètres θ : Notation

Ensemble d'apprentissage : $\{(x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)\}$

x = variables d'entrée, caractéristiques, $x = [x_0, x_1, \dots, x_n]$ avec $x_0 = 1$

y = variables de sortie, $y \in \{0,1\}$

m exemples, dimension des données = n

3. Modélisation des données

Déterminer les paramètres θ

On va suivre la même démarche que pour la régression linéaire :

- Je définis une fonction de coût

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(f_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- Je vais calculer $\min_\theta J(\theta)$ avec la descente de gradient (j'obtiens θ tel que $J(\theta)$ est minimum)
- Avec un nouveau x, je fais la prédiction et calcule $f_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

$$f_\theta(x) = P(y = 1|x;\theta)$$

En pratique $J(\theta)$ est difficile à dériver → je vais utiliser une approximation de $J(\theta)$.

3. Modélisation des données

Déterminer les paramètres θ

On va « simplifier » la fonction de coût $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(f_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$ par :

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log\left(f_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - f_\theta\left(x^{(i)}\right)\right)\right]$$

On appelle cette fonction de coût : Binary Cross Entropy (BCE) ou aussi log loss.

On applique l'algorithme de la descente de gradient

Faire jusqu'à convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

}

Grâce à la simplification de J, $\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(f\left(x_j^{(i)}\right) - y^{(i)}\right).x_j^{(i)}$

3. Modélisation des données

Déterminer les paramètres θ

On applique l'algorithme de la descente de gradient

Faire jusqu'à convergence {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(f_\theta \left(x_j^{(i)} \right) - y^{(i)} \right) . x_j^{(i)}$$

}

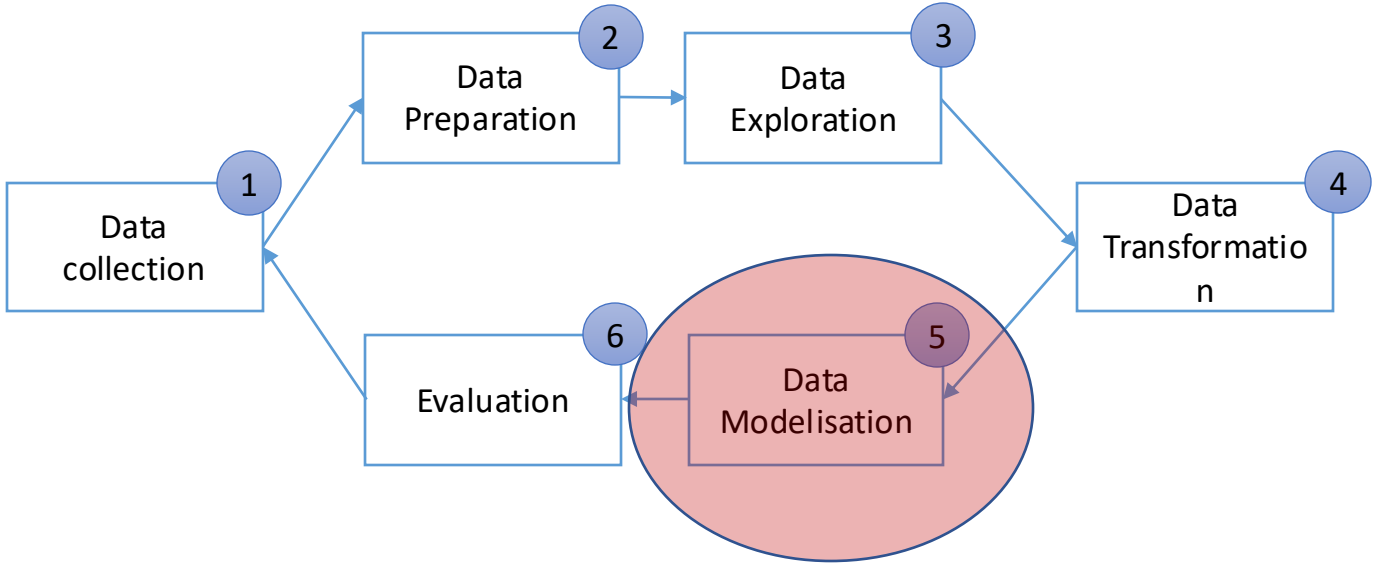
Procédure identique à la régression linéaire !

$f_\theta(x) = \theta^T x$ (régression linéaire)

$f_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ (régression logistique)

yannick.benezeth@u-bourgogne.fr

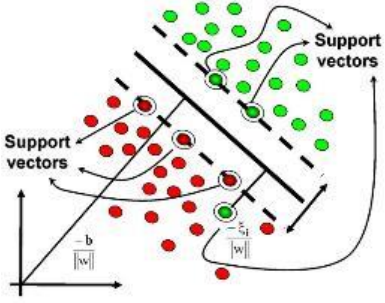
3. Modélisation des données

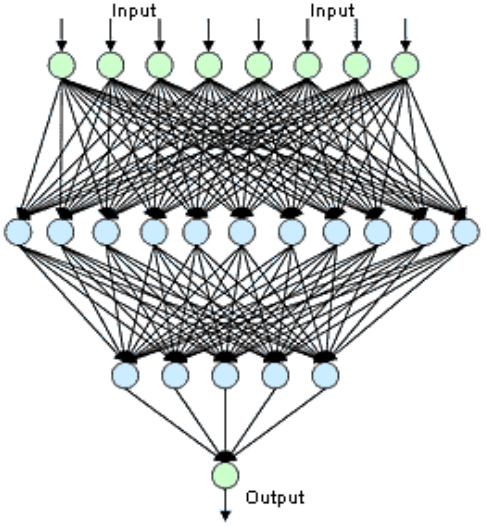


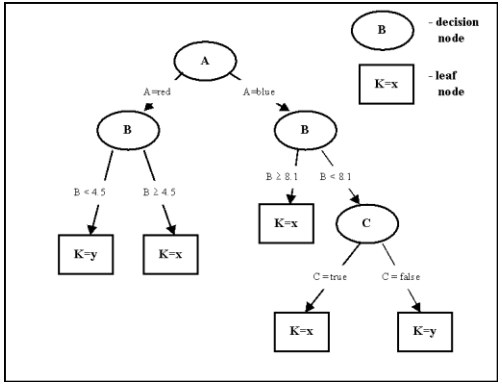
3. Modélisation des données

Les différentes techniques de classification

- Régression logistique
- K-NN
- Deep learning
- Réseaux de neurones
- Algorithme génétique
- Classification Bayesienne
- Machine à vecteurs de support
- Arbre de décision
- Adaboost
- …



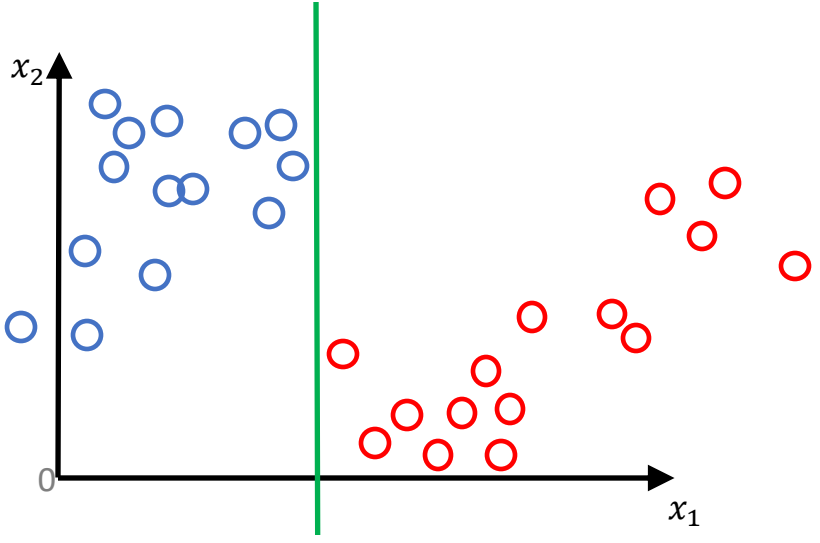




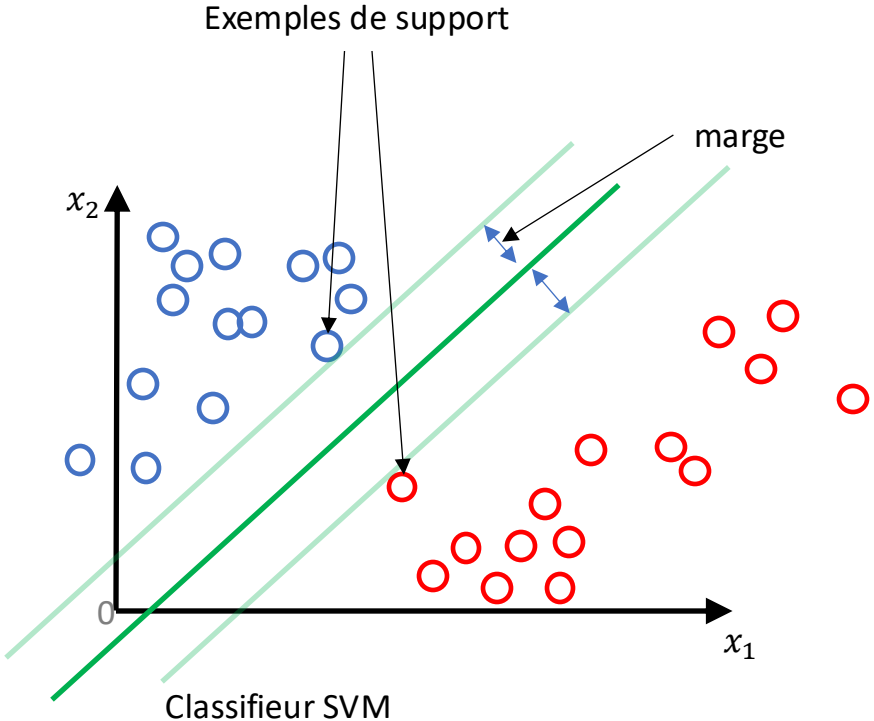
yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

SVM – intuition



Classifieur linéaire (e.g. régression logistique)

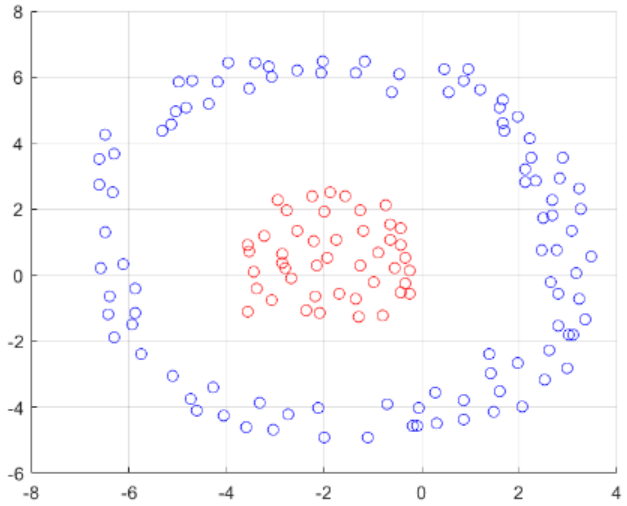


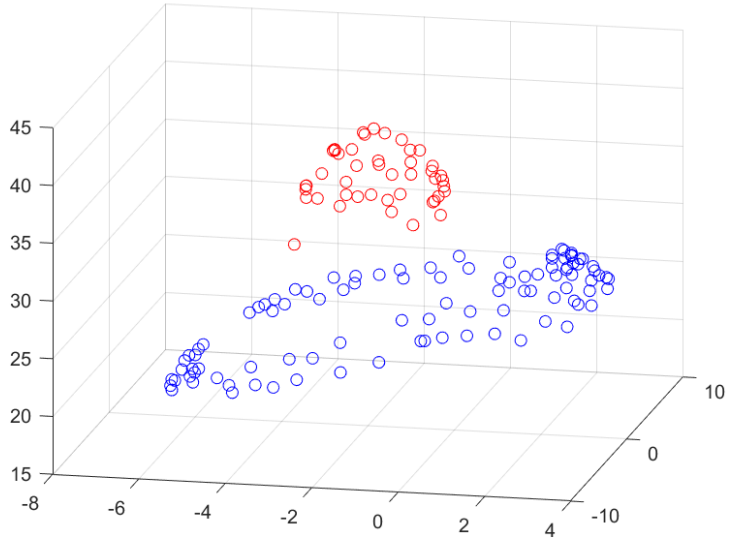
Classifieur SVM

- Le classifieur SVM (support vector machine) va chercher la droite qui sépare les 2 classes mais qui va aussi maximiser la marge entre le modèle et les *exemples de support*.
- SVM appelé aussi classifieur à vastes marges (Large Margin Classifier)

3. Modélisation des données

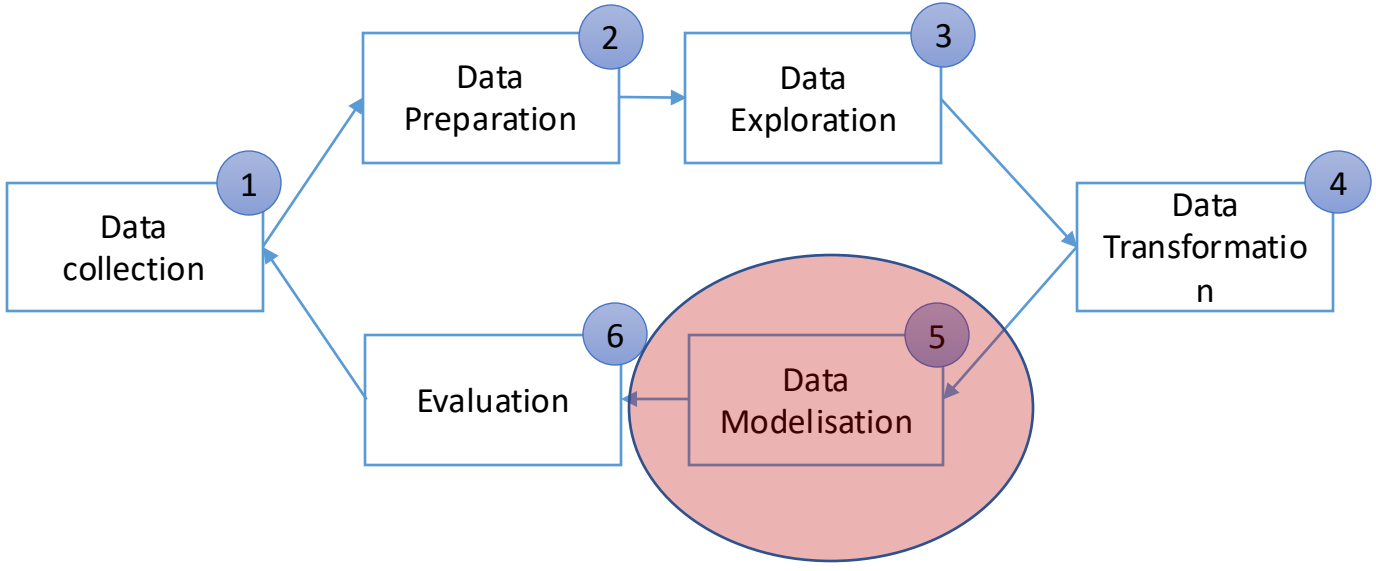
SVM – kernel trick





Si je ne peux pas trouver de séparateur linéaire de mes données dans mon espace de caractéristiques, je projette les données dans un espace de plus grande dimension dans lequel mes données seront séparables.
La projection est faite avec un kernel (=noyau).
Kernel linéaire = pas de kernel. Le kernel le plus courant est le kernel gaussien (=RBF).

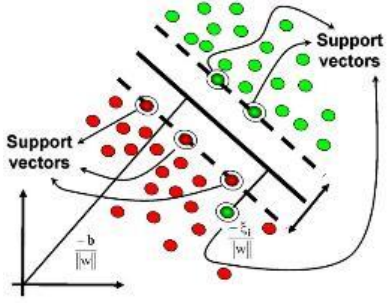
3. Modélisation des données

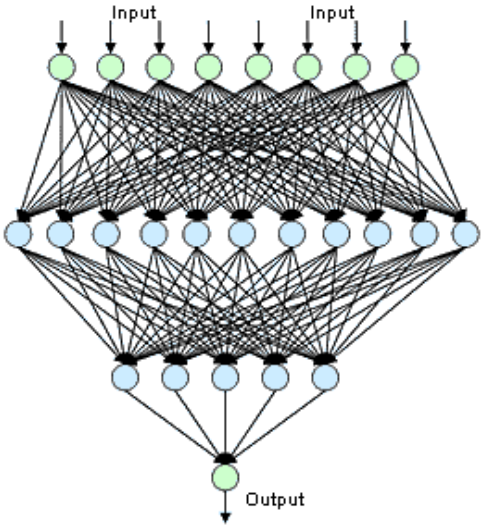


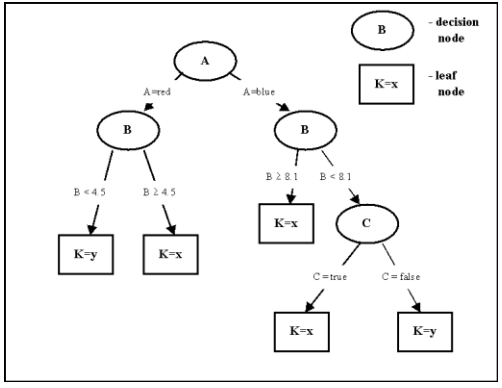
3. Modélisation des données

Les différentes techniques de classification

- Régression logistique
- K-NN
- Deep learning
- Réseaux de neurones
- Algorithme génétique
- Classification Bayesienne
- Machine à vecteurs de support
- Arbre de décision
- Random Forest
- Adaboost
- …







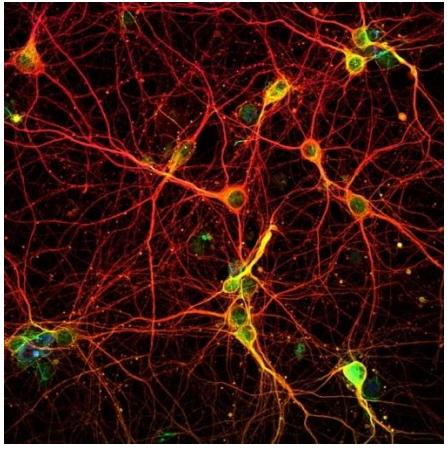
yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

Les réseaux de neurones artificiels

Un réseau de neurones est une imitation simpliste du fonctionnement du cerveau

- Le cerveau est composé de plus de 100 milliards de neurones. Chaque neurone est connecté à + de 10k neurones
- Un neurone reçoit un signal, le traite et le propage (ou pas)



yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

Les réseaux de neurones artificiels

Modèle mathématique d'un neurone

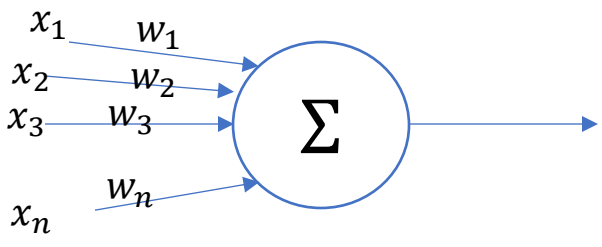
Un neurone est connecté à *n* sources

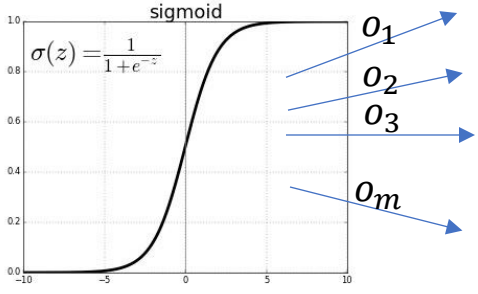
Il calcule une somme pondérée :

$$h = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n$$

Passe par une fonction d'activation

Envoie la sortie de son calcul à m neurones.



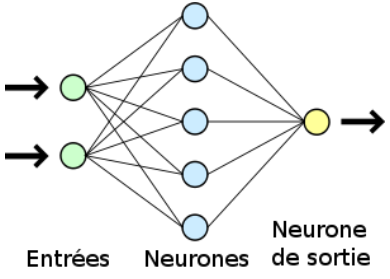


3. Modélisation des données

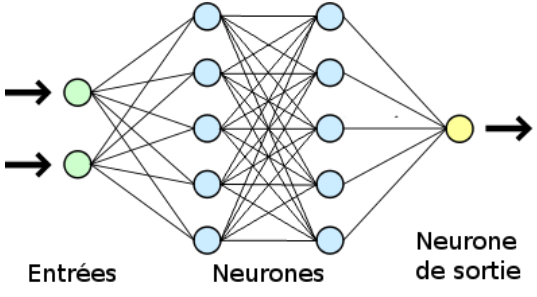
Les réseaux de neurones artificiels

Architectures classiques d'un réseau de neurones

le Perceptron simple (avec une couche cachée)



le Perceptron multicouches

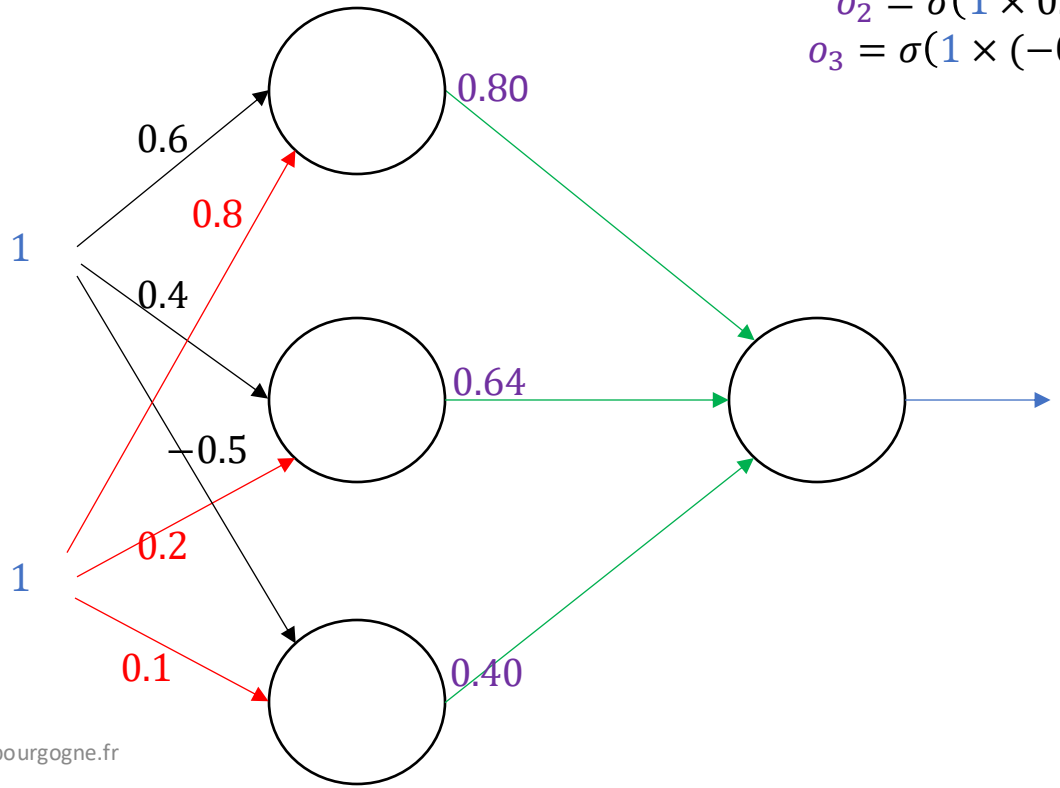


3. Modélisation des données

Les réseaux de neurones artificiels

Exemple avec 1 couche cachée

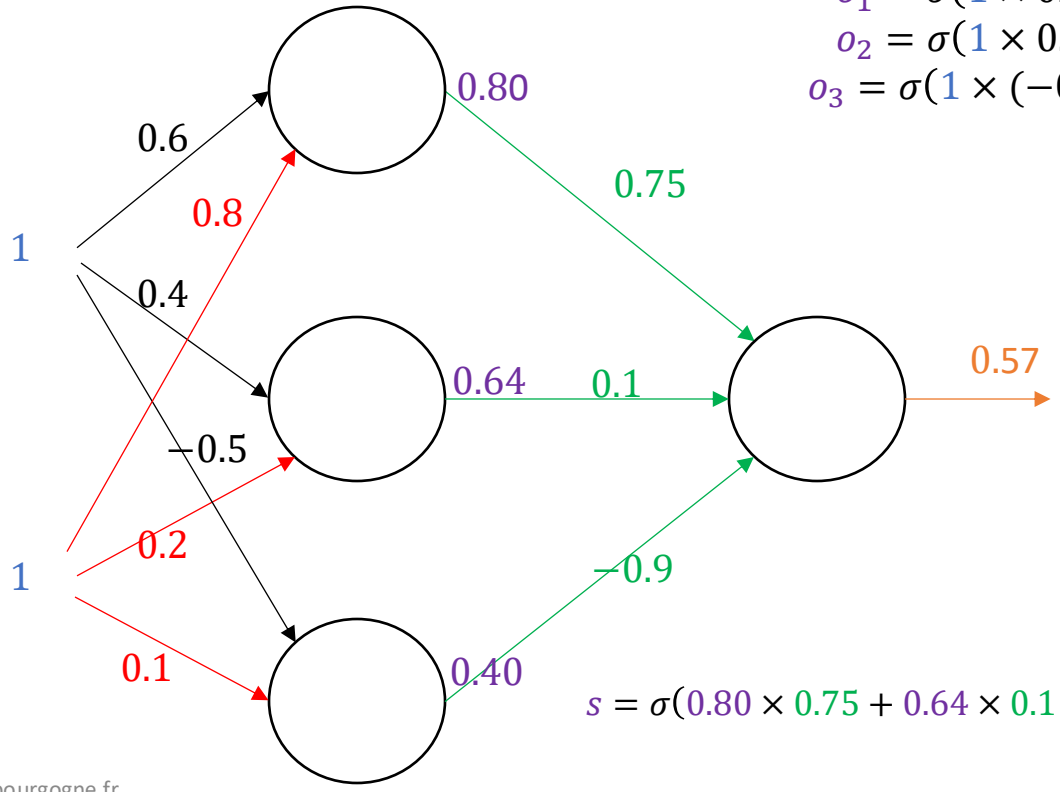
$$o_1 = \sigma(1 \times 0.6 + 1 \times 0.8) = 0.80$$
$$o_2 = \sigma(1 \times 0.4 + 1 \times 0.2) = 0.64$$
$$o_3 = \sigma(1 \times (-0.5) + 1 \times 0.1) = 0.40$$



3. Modélisation des données

Les réseaux de neurones artificiels

Exemple avec 1 couche cachée



$o_1 = \sigma(1 \times 0.6 + 1 \times 0.8) = 0.80$

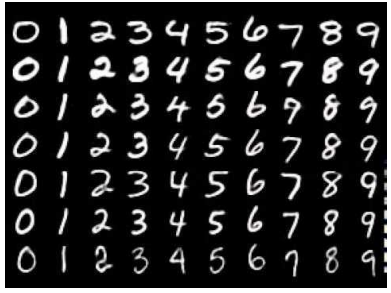
$o_2 = \sigma(1 \times 0.4 + 1 \times 0.2) = 0.64$

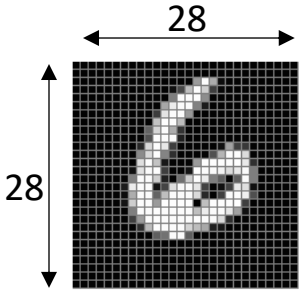
$o_3 = \sigma(1 \times (-0.5) + 1 \times 0.1) = 0.40$

$s = \sigma(0.80 \times 0.75 + 0.64 \times 0.1 + 0.40 \times (-0.9)) = 0.57$

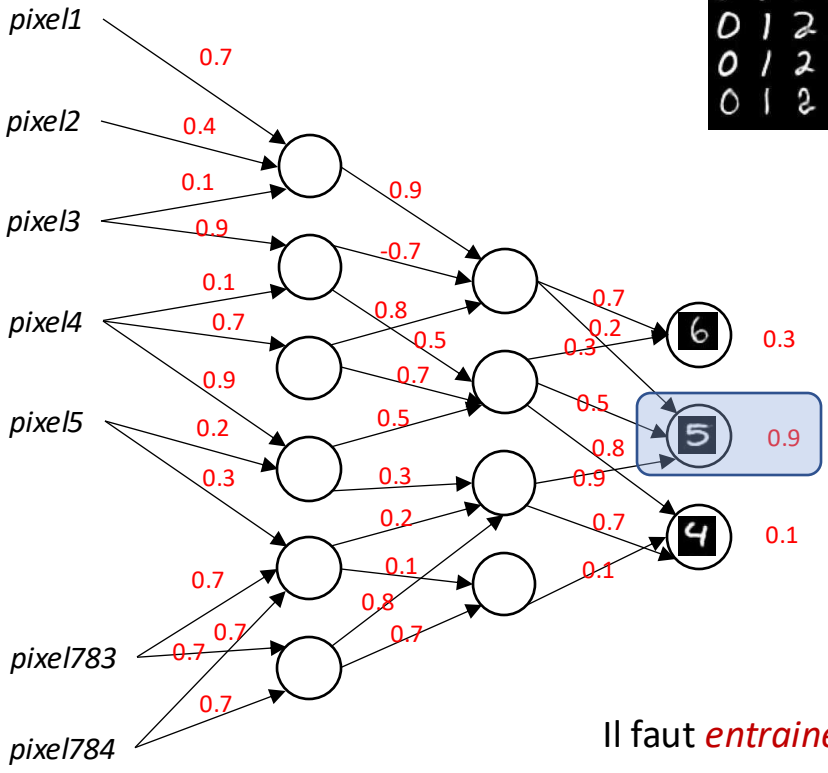
3. Modélisation des données

Les réseaux de neurones artificiels





$28 \times 28 = 784\ pixels$

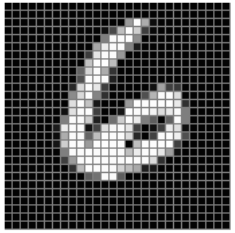


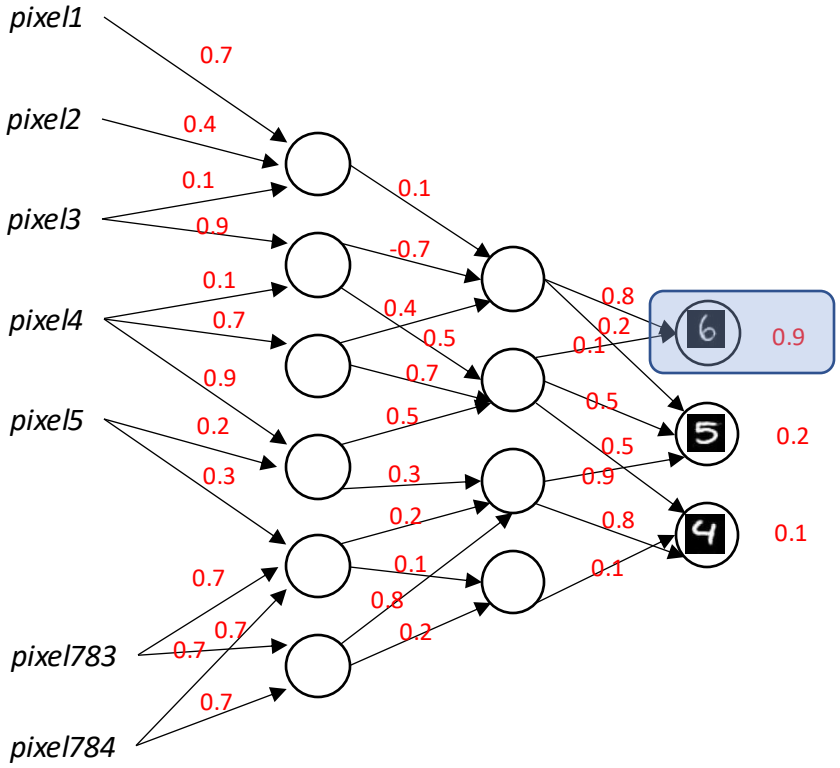
Il faut *entrainer* notre réseau!

3. Modélisation des données

Les réseaux de neurones artificiels

Entrainement : on va presenter des exemples d'images et on va ajuster les poids pour que le reseau se trompe le moins possible sur les exemples d'entrainement



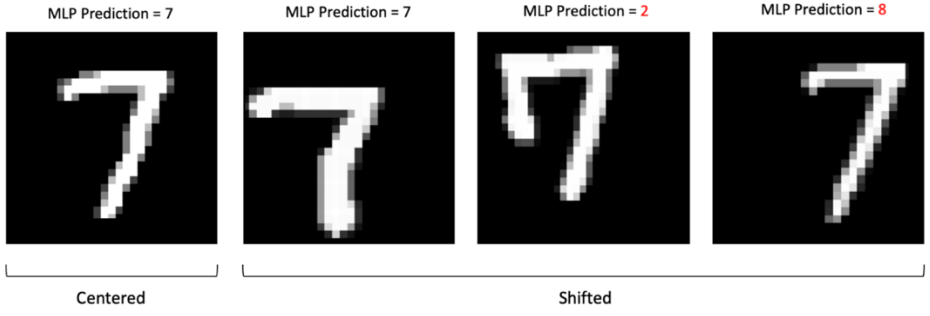


Pour aller plus loin : https://www.youtube.com/watch?v=aircAruvnKk&t=970s&ab_channel=3Blue1Brown (super chaîne!)

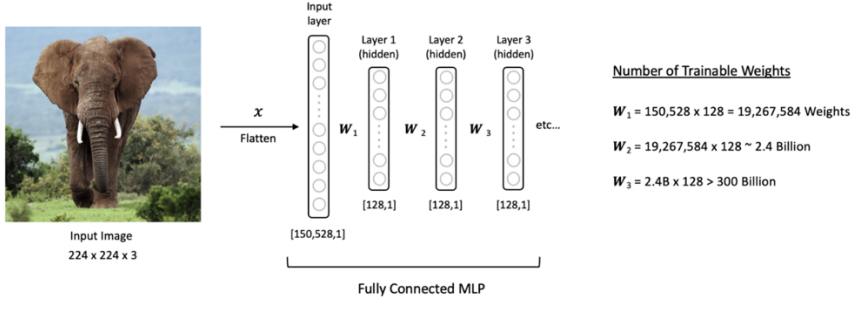
3. Modélisation des données

Limites des MLP pour classification d'images

Pas robuste aux translations



Beaucoup de paramètres → risque d'overfitting



Number of Trainable Weights

W_1 = 150,528 x 128 = 19,267,584 Weights

W_2 = 19,267,584 x 128 ~ 2.4 Billion

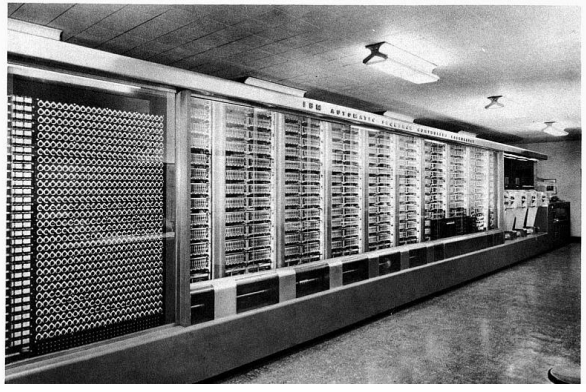
W_3 = 2.4B x 128 > 300 Billion

3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Un peu d'histoire

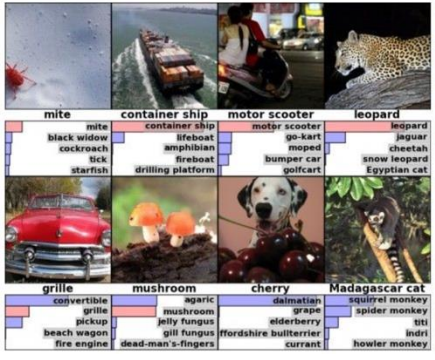
- 1957 : première implémentation du Perceptron
- 1986 : algorithme de back-propagation
- 1998 : 1er réseau convolutionnel (LeCun et al.)
- 2012 : AlexNet remporte challenge ImageNet
- Et depuis, les CNN sont partout…



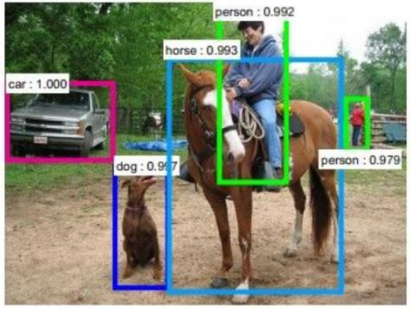
3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Classification



Detection



Segmentation

Voiture autonome

Estimation de la pose

Image captioning

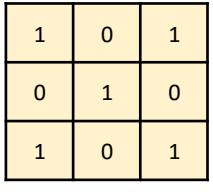
A white teddy bear sitting in the grass

yannick.benezeth@u-bourgogne.fr

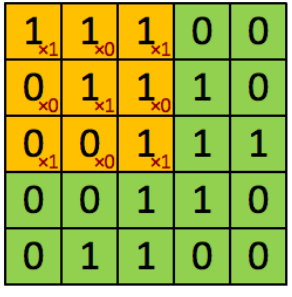
3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Les CNN sont composés de couches de convolution, d'activation, de pooling et une couche *totalement connectée*.



Filtre 3x3



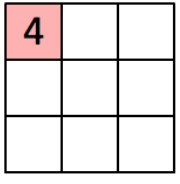


Image convoluée

3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Les CNN sont composés de couches de convolution, d'activation, de pooling et une couche *totalement connectée*.

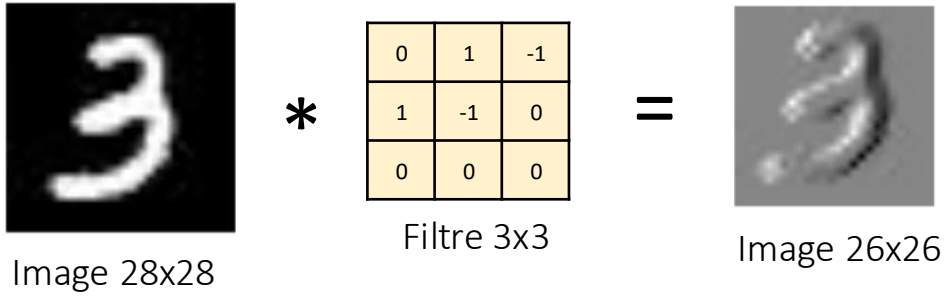


Image 28x28 — Filtre 3x3 — Image 26x26

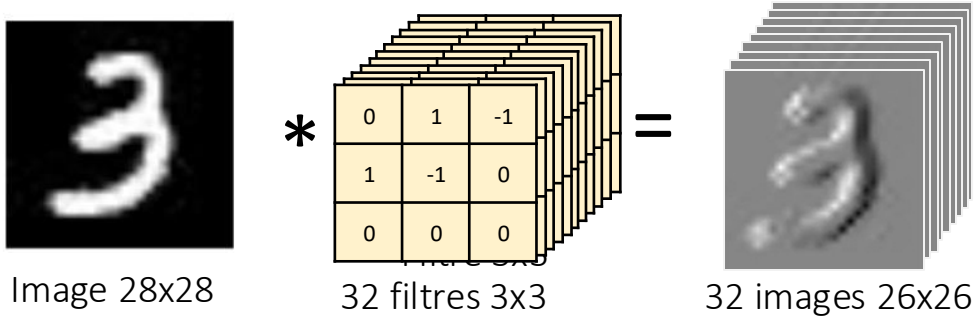


Image 28x28 — 32 filtres 3x3 — 32 images 26x26

yannick.benezeth@u-bourgogne.fr

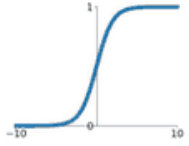
3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Les CNN sont composés de couches de convolution, d'activation, de pooling et une couche *totalement connectée*.

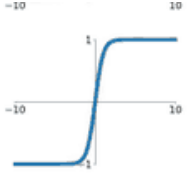
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



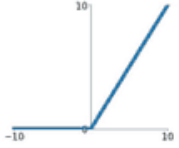
tanh

$\tanh(x)$



ReLU

$\max(0, x)$



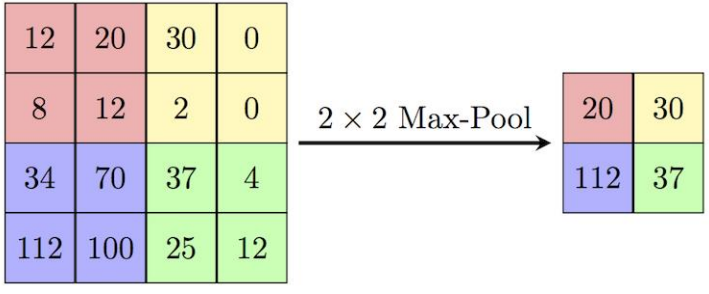
Nécessaire car : $A_1 \times (A_2 \times X) = (A_1 \times A_2) \times X = A \times X$

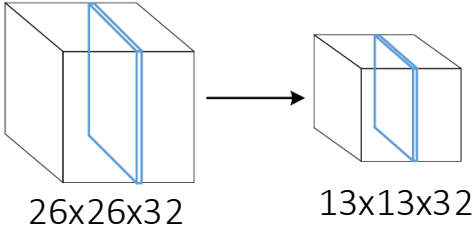
3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Les CNN sont composés de couches de convolution, d'activation, de pooling et une couche *totalement connectée*.

Permet de **réduire la dimension spatiale** des données (max-pooling, avg-pooling…).





yannick.benezeth@u-bourgogne.fr

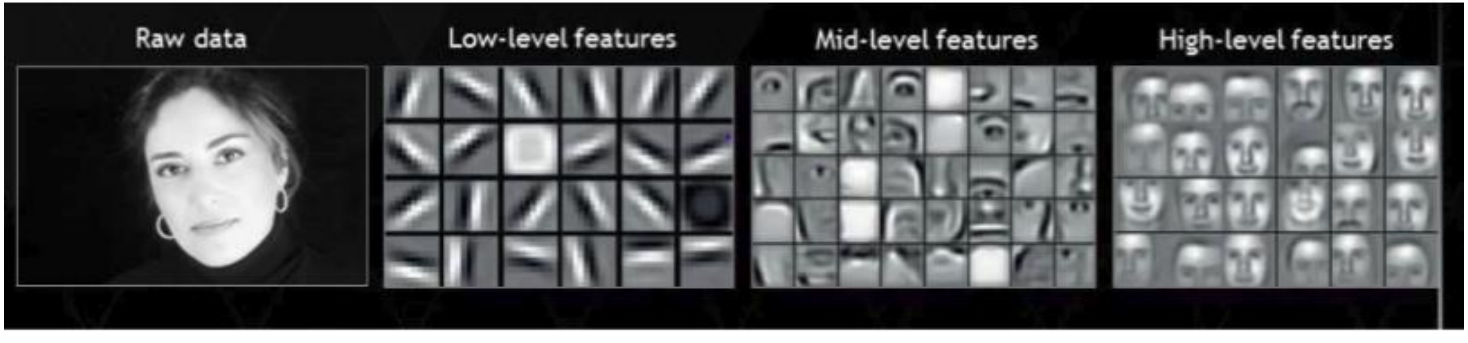
3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Les CNN sont composés de couches de convolution, d'activation, de pooling et une couche *totalement connectée*.

La combinaison de couches successives de convolution, d'activation et de pooling permet d'extraire des descripteurs de haut-niveau.

Ensuite, on utilise en général des couches totalement connectées pour faire une classification (~Perceptron multicouches)



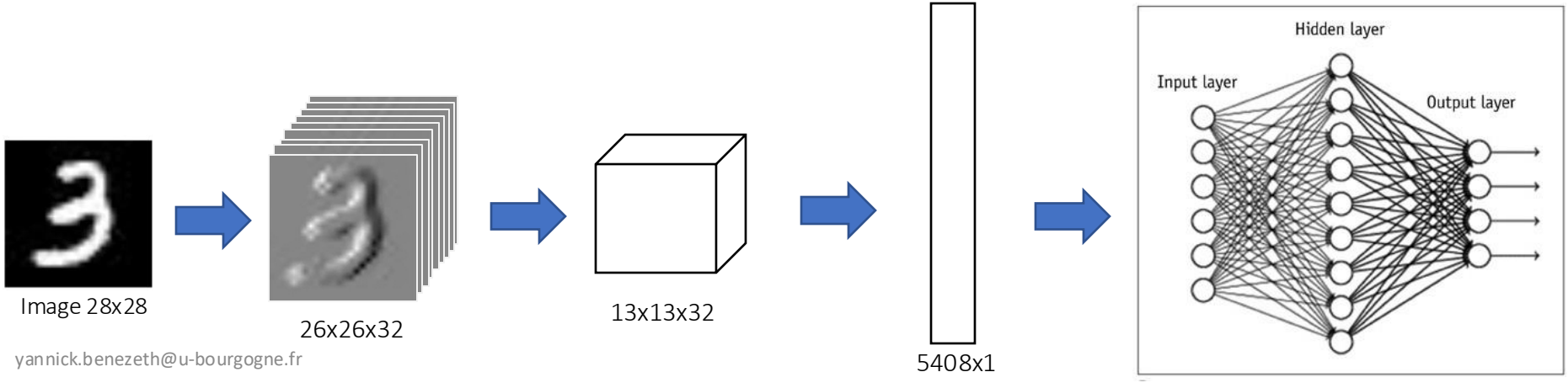
3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Les CNN sont composés de couches de convolution, d'activation, de pooling et une couche *totalement connectée*.

La combinaison de couches successives de convolution, d'activation et de pooling permet d'extraire des descripteurs de haut-niveau.

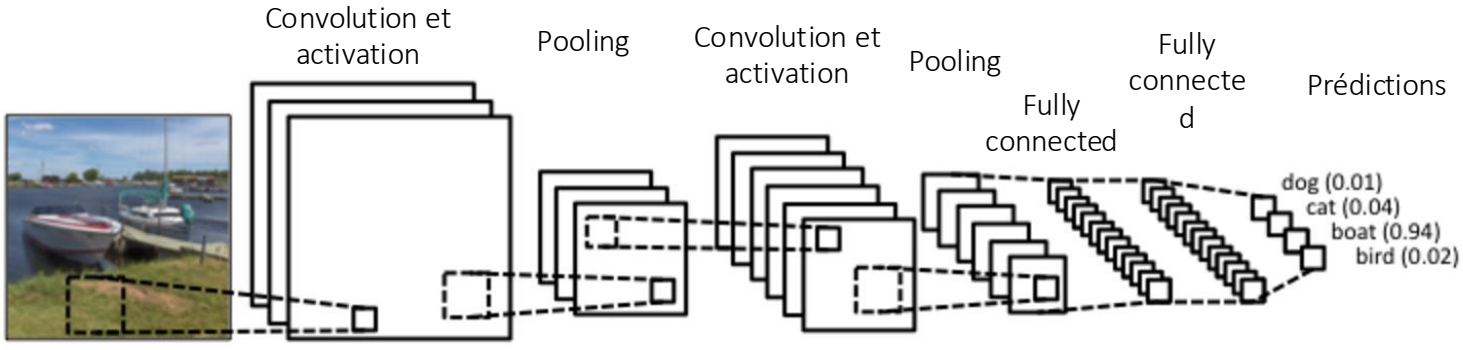
Ensuite, on utilise en général des couches totalement connectées pour faire une classification (~Perceptron multicouches)



3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Exemples d'architecture

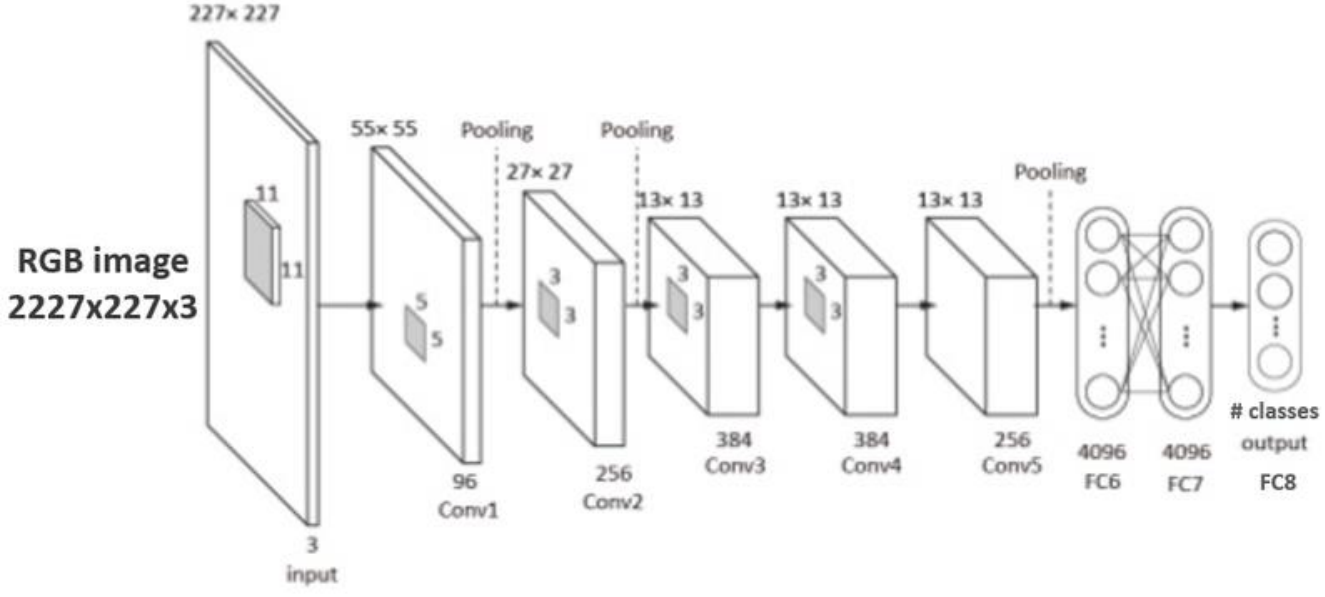


Architecture classique

3. Modélisation des données

Les réseaux de neurones convolutionnels (CNN)

Exemples d'architecture

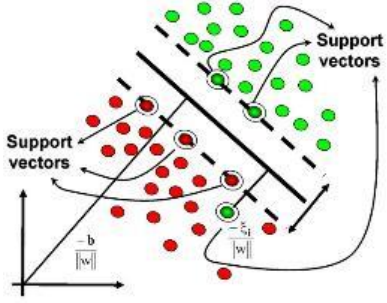


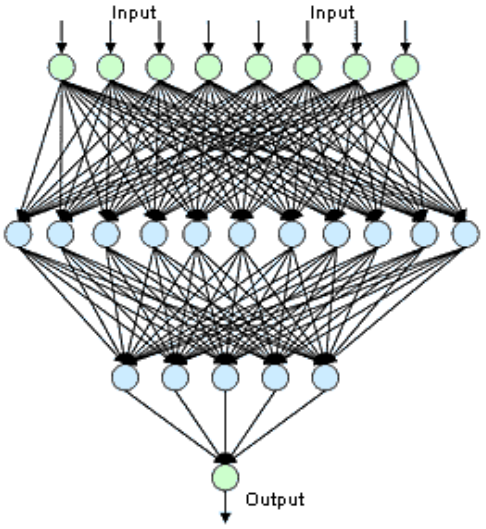
AlexNet 2012

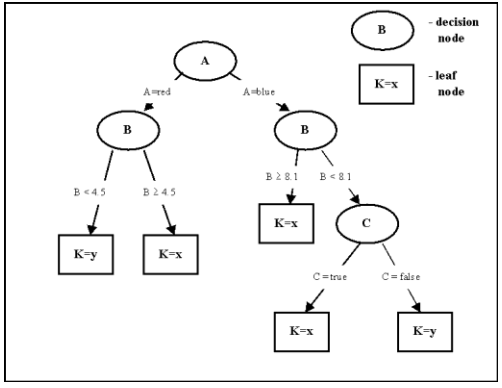
3. Modélisation des données

Les différentes techniques de classification

- Régression logistique
- K-NN
- Deep learning
- Réseaux de neurones
- Réseaux de neurones récurrents
- Algorithme génétique
- Classification Bayesienne
- Machine à vecteurs de support
- Arbre de décision
- Random Forest
- Adaboost
- …







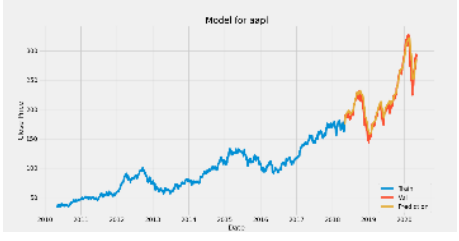
yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

Examples of sequence modeling

A sequence can be composed of numerical values, musical notes, letters, words, images…

In sequence modeling, the order on the observations must be preserved when training models and making predictions.

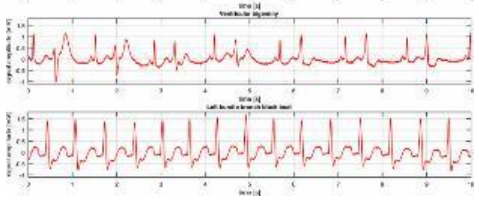


Stock price prediction

Image captioning

Machine translation

Music generation

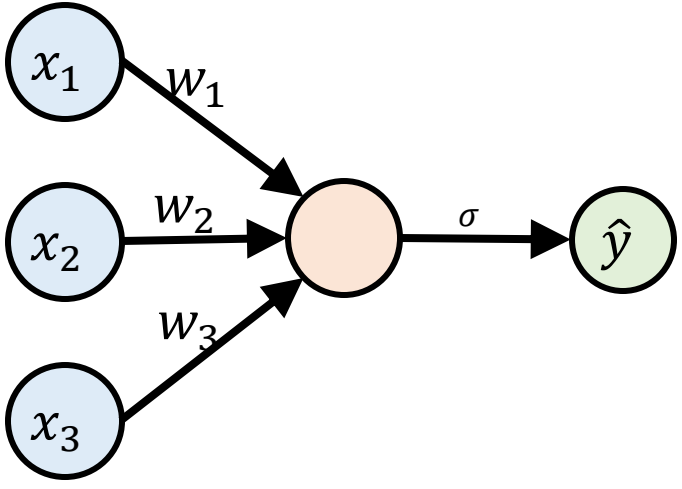


ECG signal classification

Activity recognition in videos

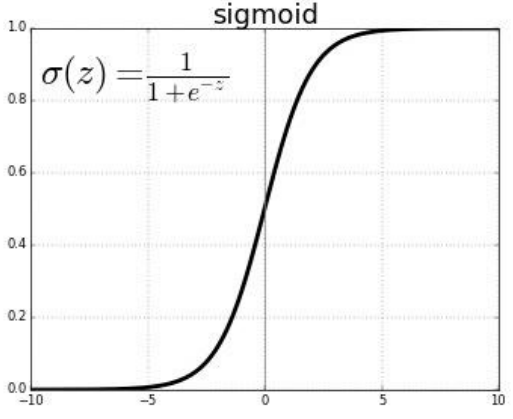
3. Modélisation des données

Modeling data sequence with a perceptron



$$\hat{y} = \sum_{j=1}^{3} \sigma(w_j x_j + b)$$

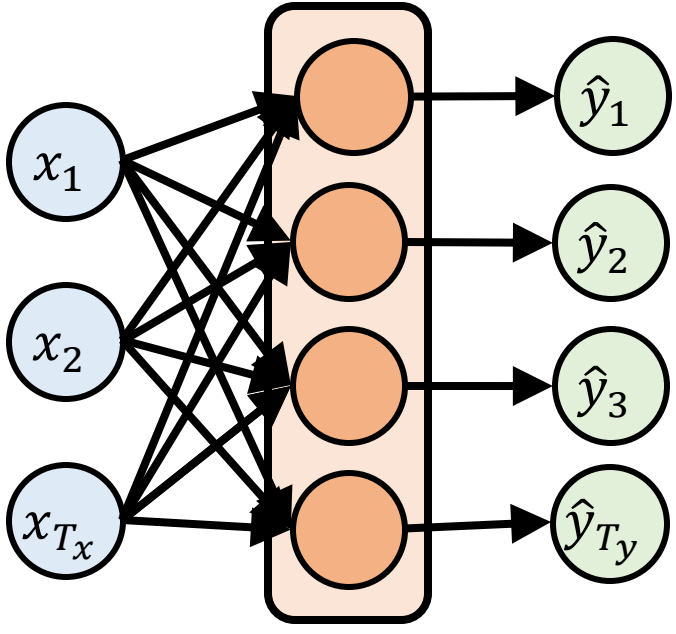
With σ an activation function



x_1, x_2, x_3 are 3 values of a temporal sequence.

3. Modélisation des données

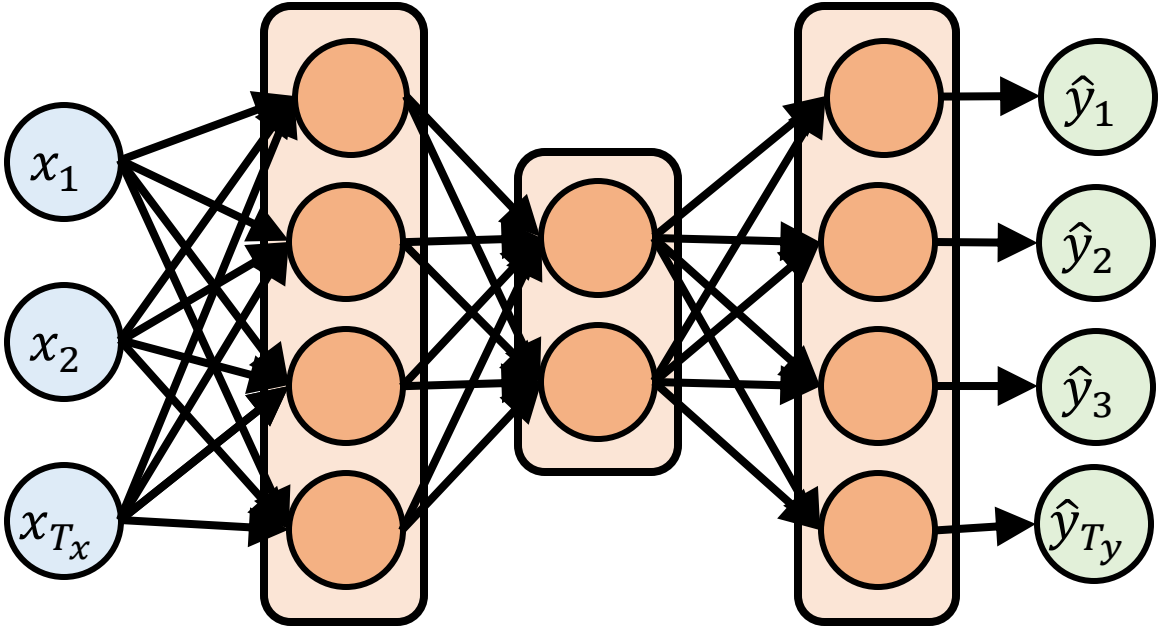
Modeling data sequence with a layer of perceptrons



T_x is the length of the input sequence
T_y is the length of the output sequence

3. Modélisation des données

Modeling data sequence with a multilayer perceptron



Limitations

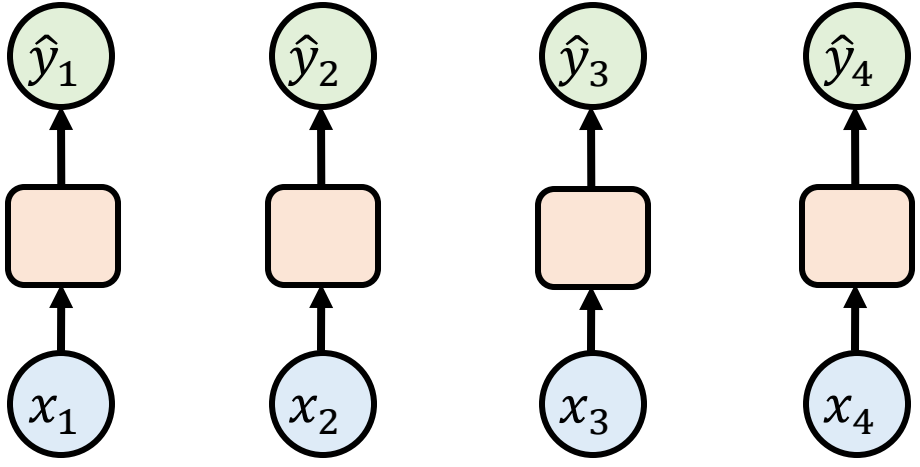
Unaware of temporal structure

Fixed sized inputs/outputs

Time steps are modeled as input features, meaning that network has no explicit handling or understanding of the temporal structure or order between observations.

3. Modélisation des données

Modeling sequence of data

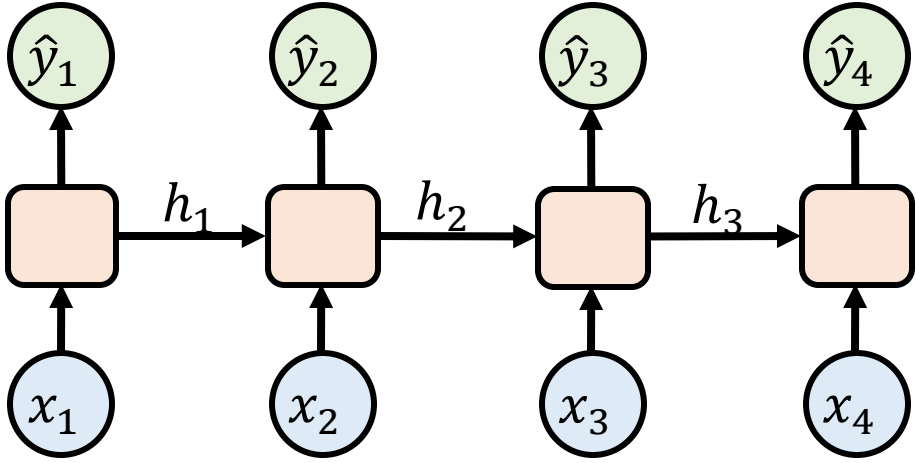


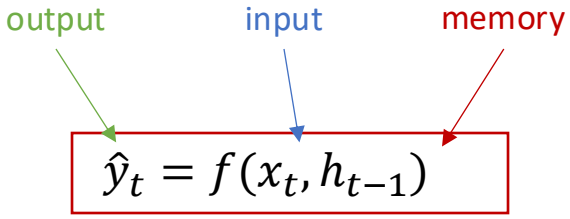
$\hat{y}_t = f(x_t)$
The prediction at time t is given from the observed data x_t at time t.

Same model with different data.
Previous info are just ignored, the structure of the temporal data is not considered.
Not suited to model temporal data. It's likely that y4 depends on x1 and x2 !

3. Modélisation des données

Modeling sequence of data



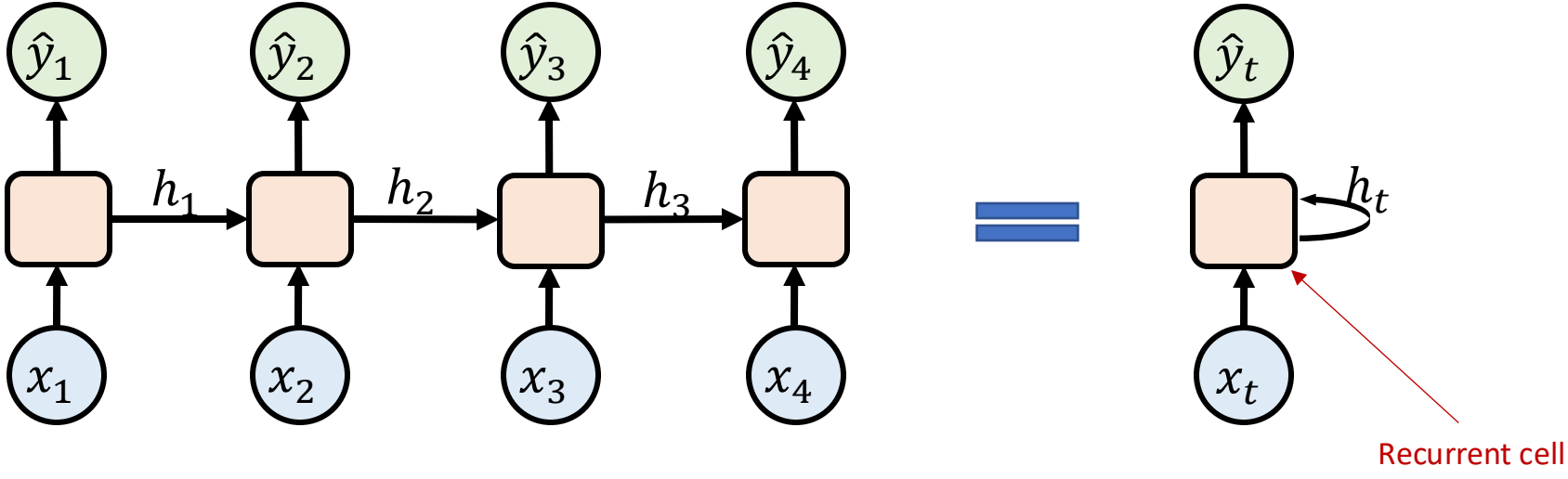


The prediction at time *t* is given from the observed data x_t at time *t* and the *hidden states* at previous timestamp h_{t-1}.

The computation that the network is doing is related with previous observations.
The information is passed forward.

3. Modélisation des données

Modeling sequence of data



output input memory

$$\hat{y}_t = f(x_t, h_{t-1})$$

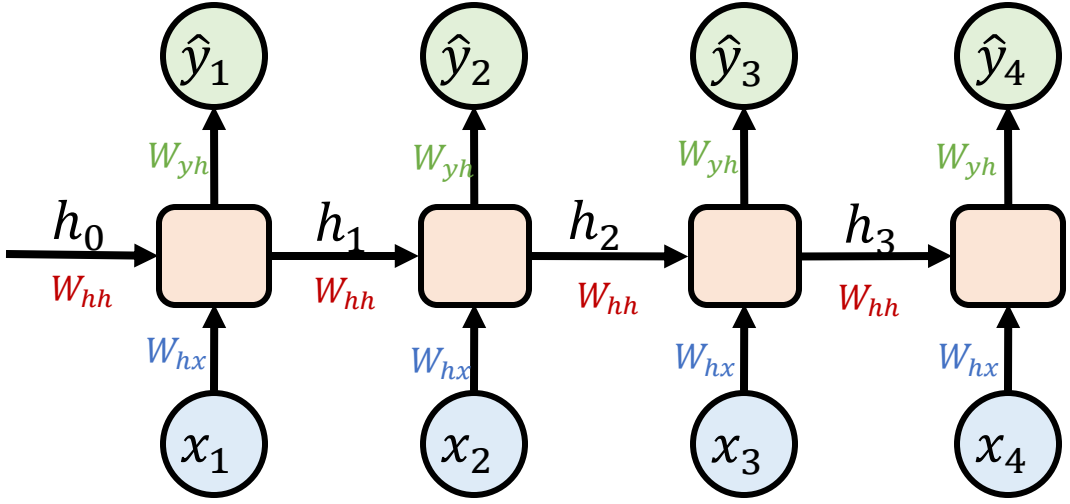
Recurrence relation

$x_t \in \mathbb{R}^m$

$\hat{y}_t \in \mathbb{R}^n$

3. Modélisation des données

Equations for the forward propagation



$$h_1 = g_1(W_{hh}h_0 + W_{hx}x_1 + b_h)$$

$$\hat{y}_1 = g_2(W_{yh}h_1 + b_y)$$

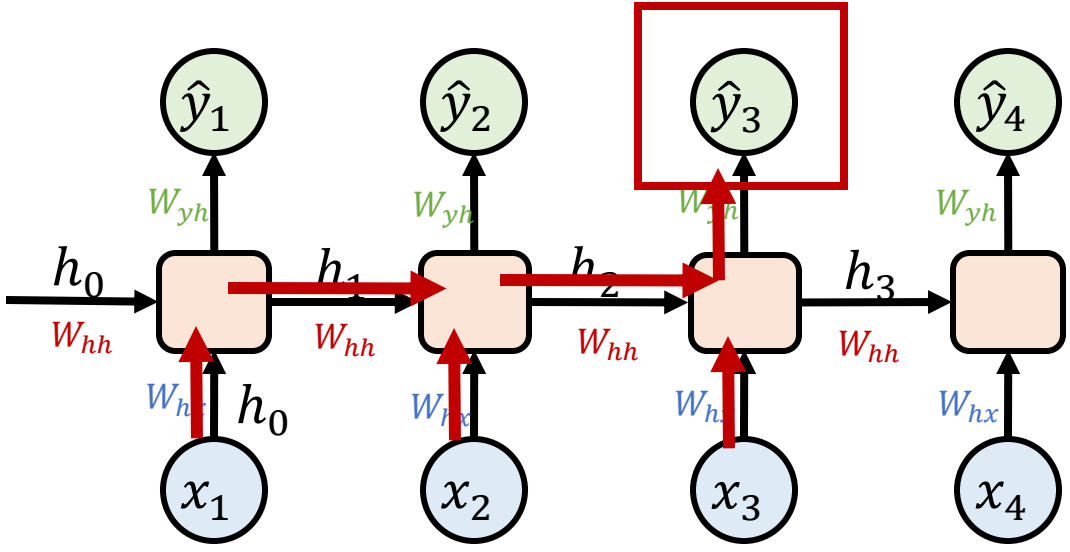
The same weights W_{hh}, W_{hx} and W_{yh} are re-used at every time steps

We add h_0 for the first timestep (usually a vector of 0).
g_1 and g_2 are two activation functions. Often, we use *tanh* or *ReLU* for g_1 .
g_2 depends on the application:

- *sigmoid* for binary classification,
- *softmax* for multiple classes,
- *linear* activation function for regression.

3. Modélisation des données

Equations for the forward propagation



$$h_1 = g_1(W_{hh}h_0 + W_{hx}x_1 + b_h)$$

$$\hat{y}_1 = g_2(W_{yh}h_1 + b_y)$$

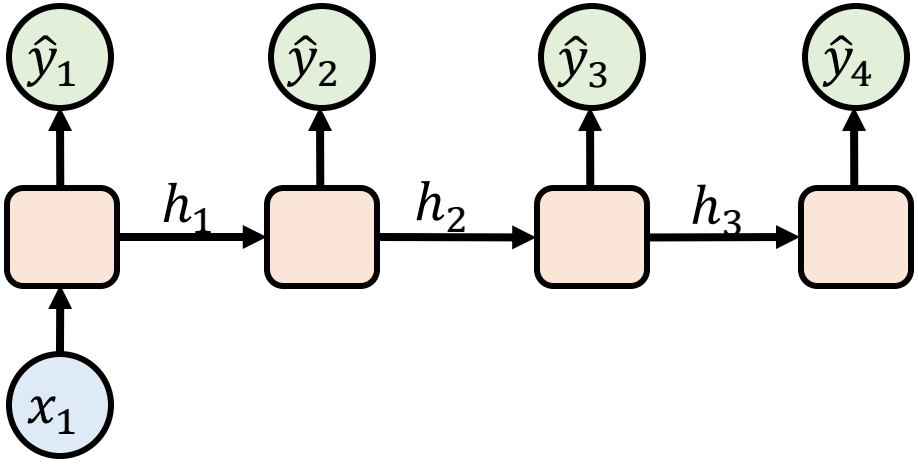
The same weights W_{hh}, W_{hx} and W_{yh} are re-used at every time steps

To estimate $\hat{y}_3$ we use x_1, x_2 and x_3.
We can note that we don't use future timestamps, even if it can be useful in some cases.

3. Modélisation des données

Different kind of models

One-to-Many



Examples

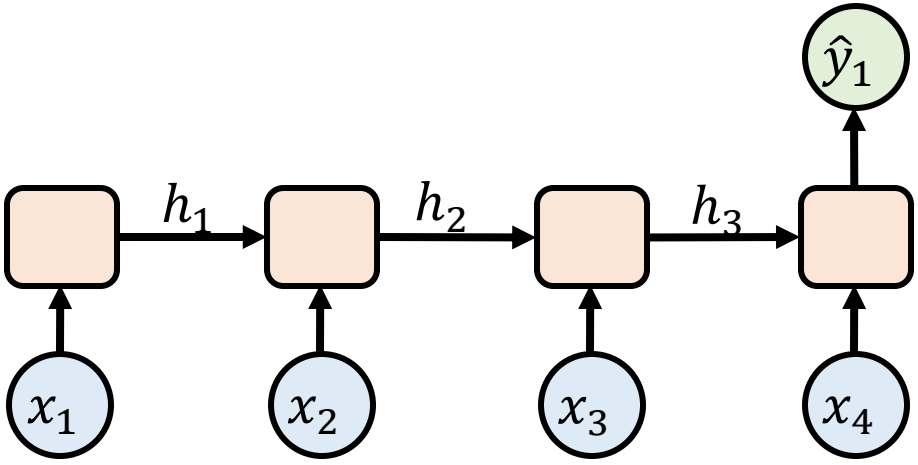
Predicting a sequence of words from a single image.
Forecasting a series of observations from a single event.

Internal state is accumulated as each value in the output sequence is produced.

3. Modélisation des données

Different kind of models

Many-to-One



Examples

Forecasting the next real value in a time series given a sequence of input observations.

Predicting the classification label for an input sequence of words (e.g. sentiment analysis).

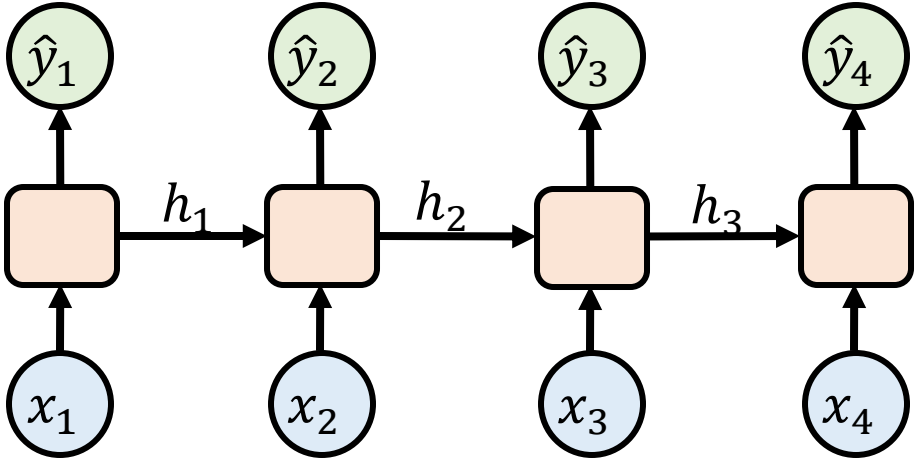
A many-to-one model produces one output $\hat{y}_t$ value after receiving multiple input values x_t, x_{t-1}, …
Internal state is accumulated with each input value before a final output value is produced

3. Modélisation des données

Different kind of models

Many-to-Many

With $T_x = T_y$



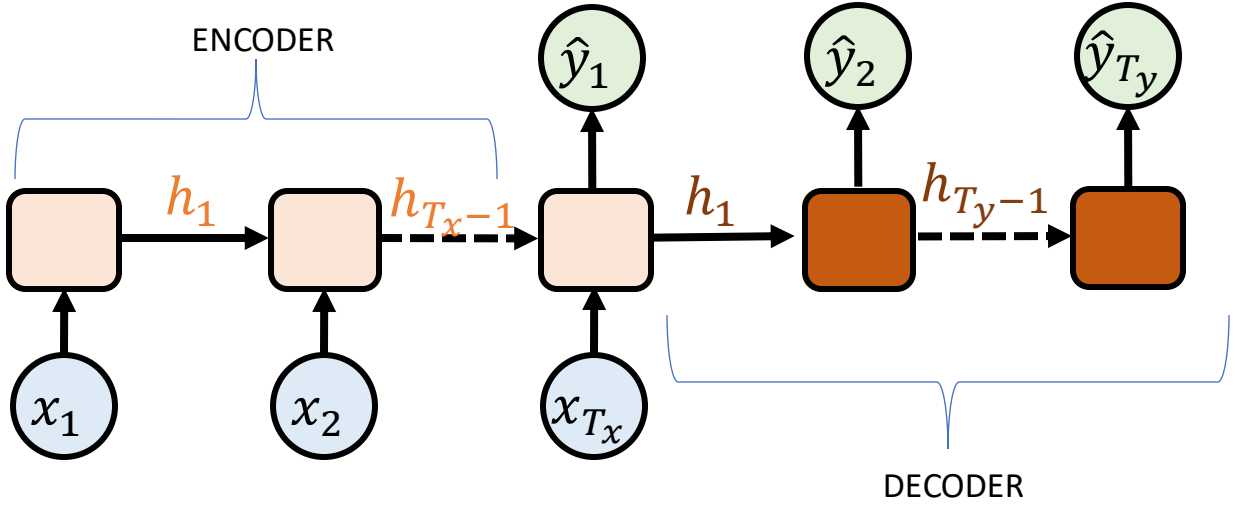
Examples

Frame level video classification or sample level ECG signal classification…

3. Modélisation des données

Different kind of models

Many-to-Many



Often called sequence-to-sequence (seq2seq)

As with the many-to-one case, the hidden state is accumulated until the 1st output is created, but in this case multiple time steps are output.

The number of input time steps does not have to match the number of outputs.

This model is appropriate for sequence predictions, where multiple input timesteps are required to predict a sequence of output time steps

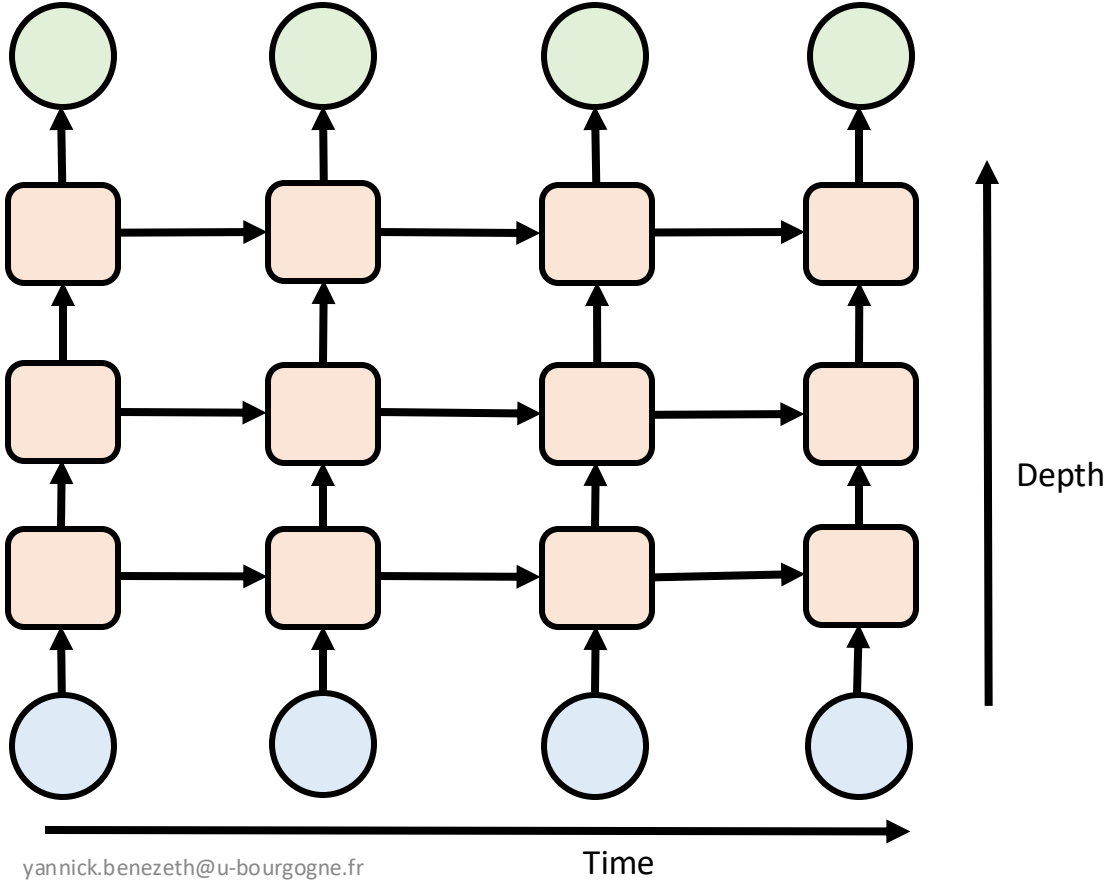
Examples

Summarize a document of words into a shorter sequence of words.

Classify a sequence of audio data into a sequence of words.

3. Modélisation des données

Multilayer RNN

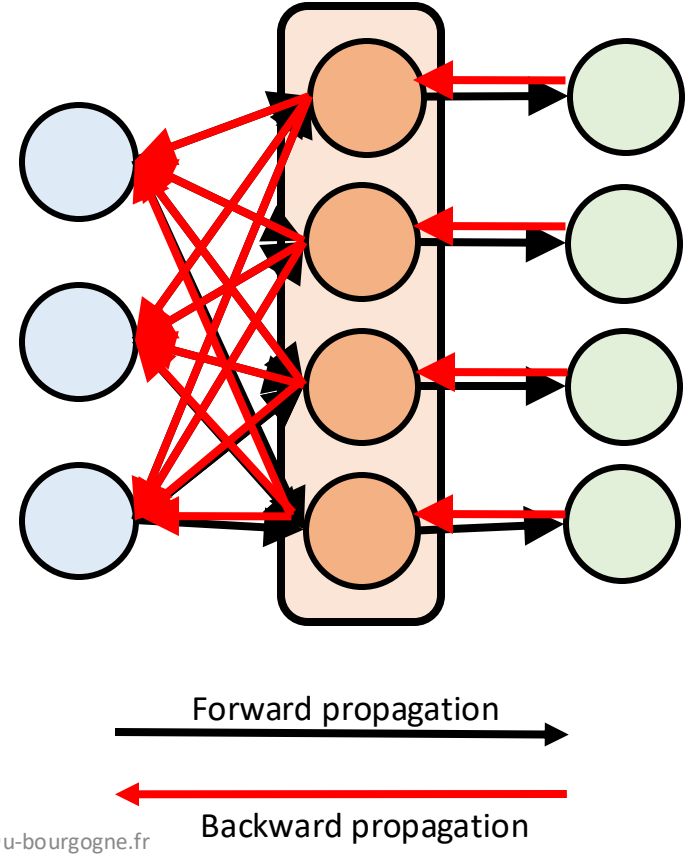


RNNs process sequential data by generating a sequence of hidden states. Each layer takes the previous layer's hidden states as input.

RNNs rarely exceed 1-3 layers.

3. Modélisation des données

Backpropagation

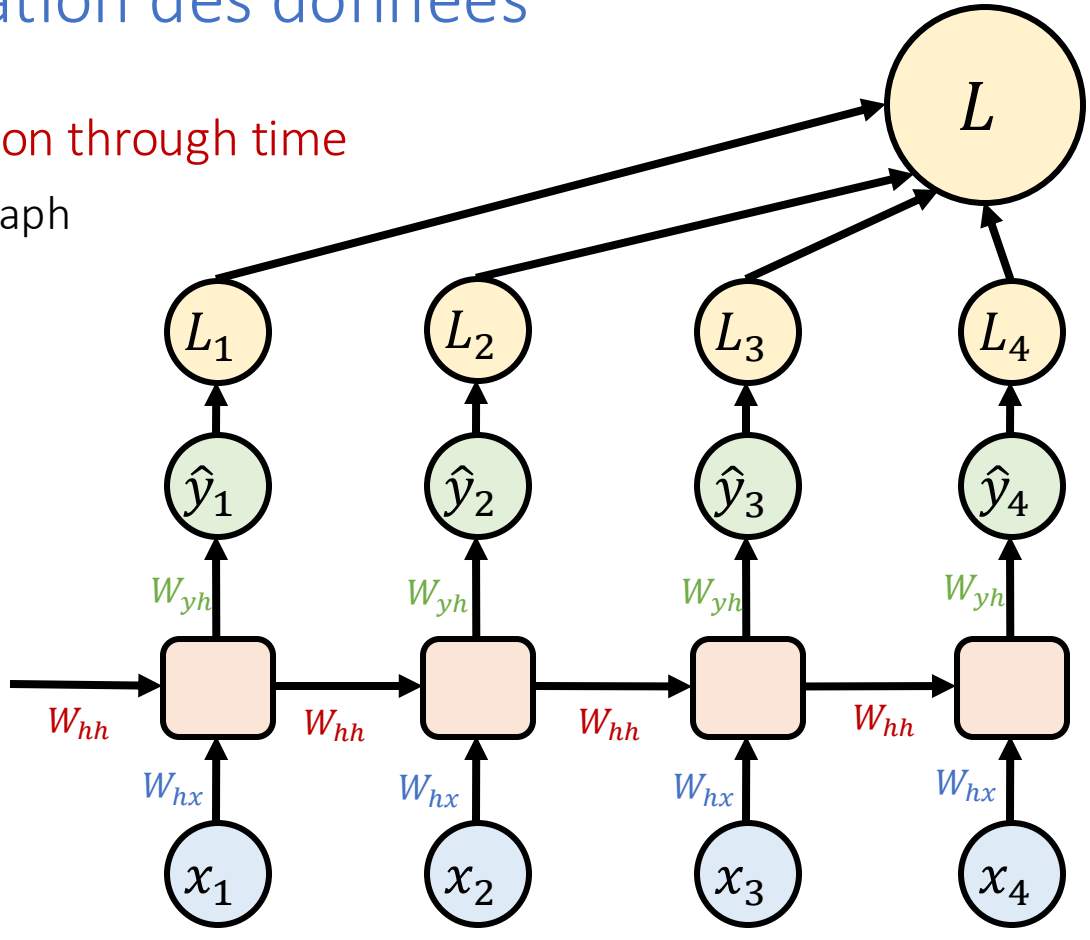


Backpropagation computes the gradient of the loss function with respect to the weights of the network.
Adjust parameters to minimize the loss.

3. Modélisation des données

Backpropagation through time

Computation graph



At each time step

$L_t(\hat{y}_t, y_t)$

Then we can calculate the total loss

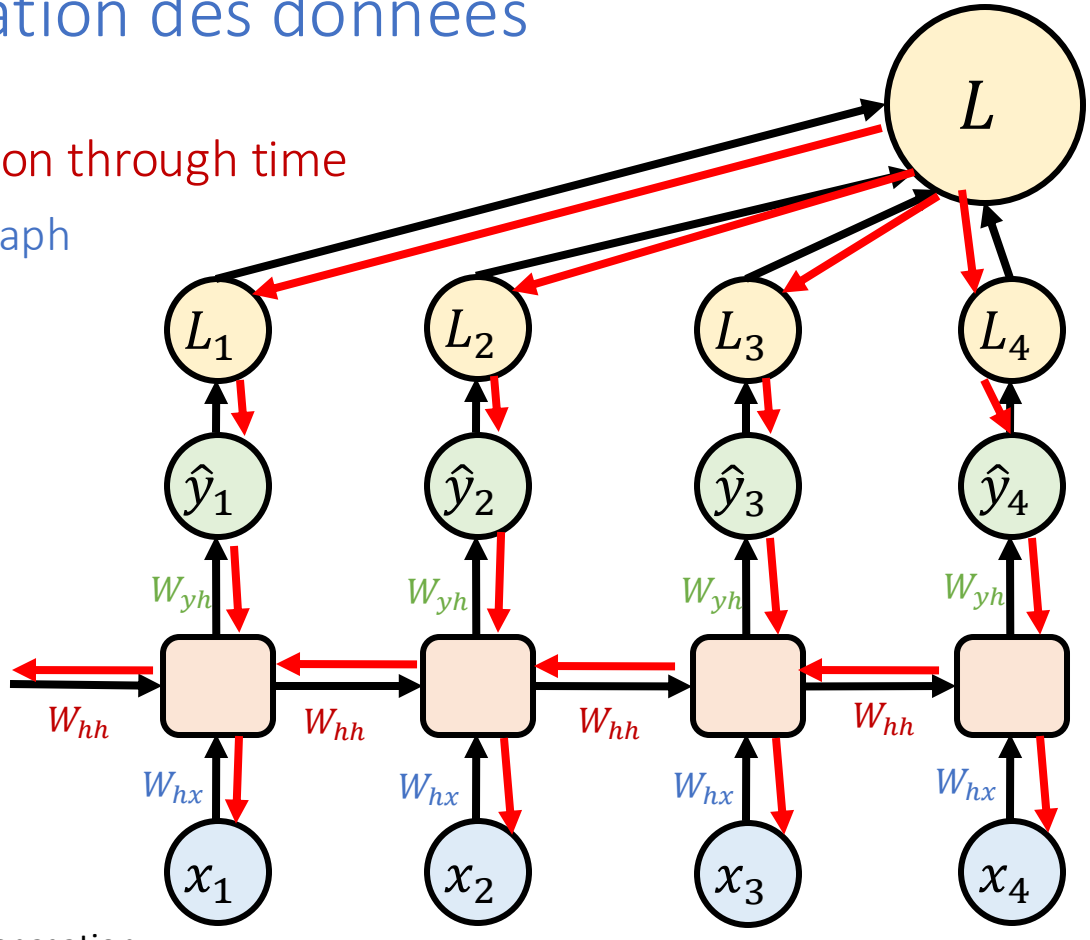
$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L_t(\hat{y}_t, y_t)$$

Forward propagation

3. Modélisation des données

Backpropagation through time

Computation graph



At each time step

$L_t(\hat{y}_t, y_t)$

Then we can calculate the total loss

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L_t(\hat{y}_t, y_t)$$

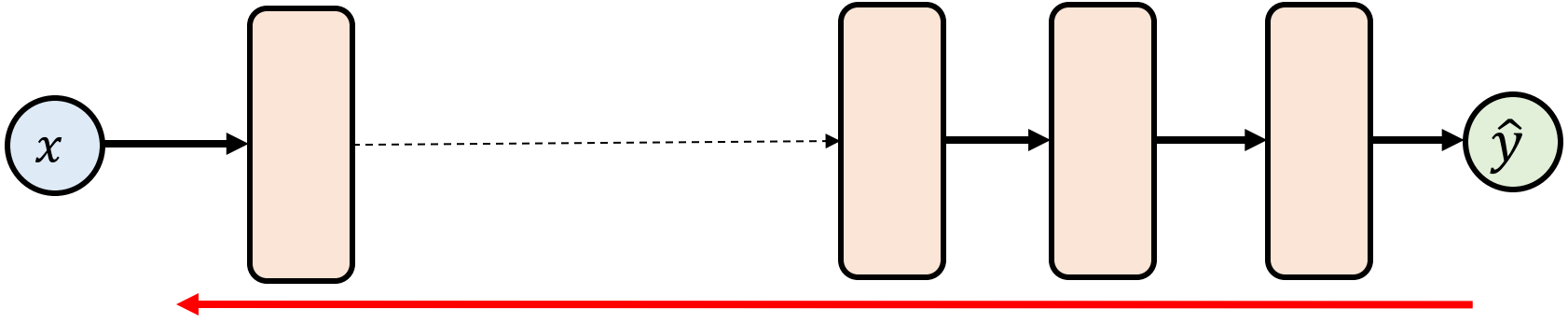
Backward propagation

Forward propagation

yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

Vanishing gradients with Deep Neural Networks



Deep neural networks use backpropagation to update weights based on the error at the output. This involves calculating gradients and propagating them backwards through the network.

However, in very deep networks, these gradients can diminish significantly as they travel through many layers, making it difficult to update the weights of earlier layers.

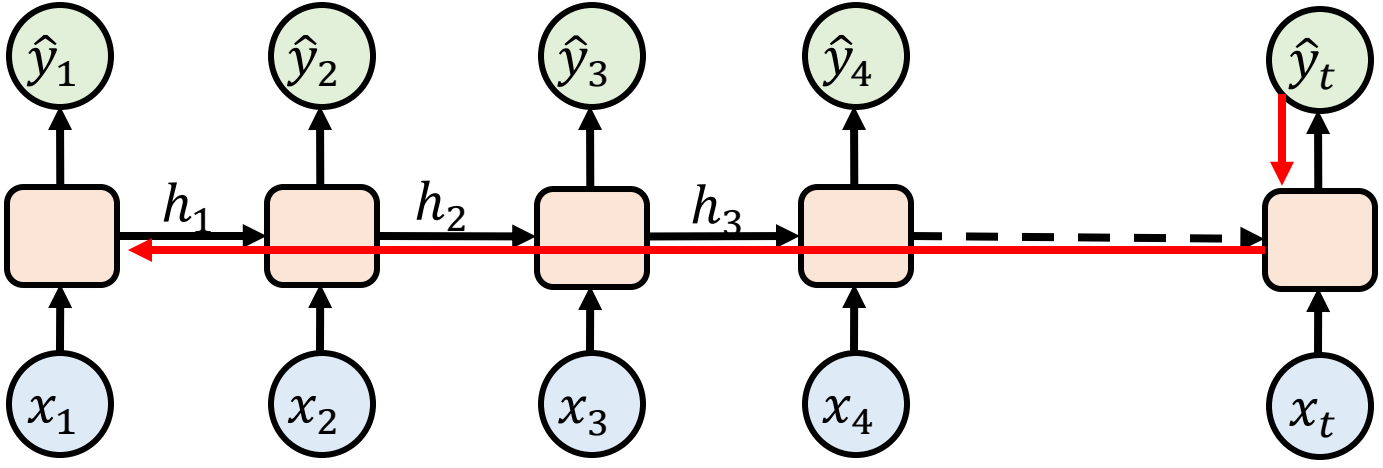
This is known as the vanishing gradient problem.

3. Modélisation des données

Vanishing gradients with Recurrent Neural Networks

Similar problems with RNN.

It’s difficult to capture long-term dependencies with regular RNNs.

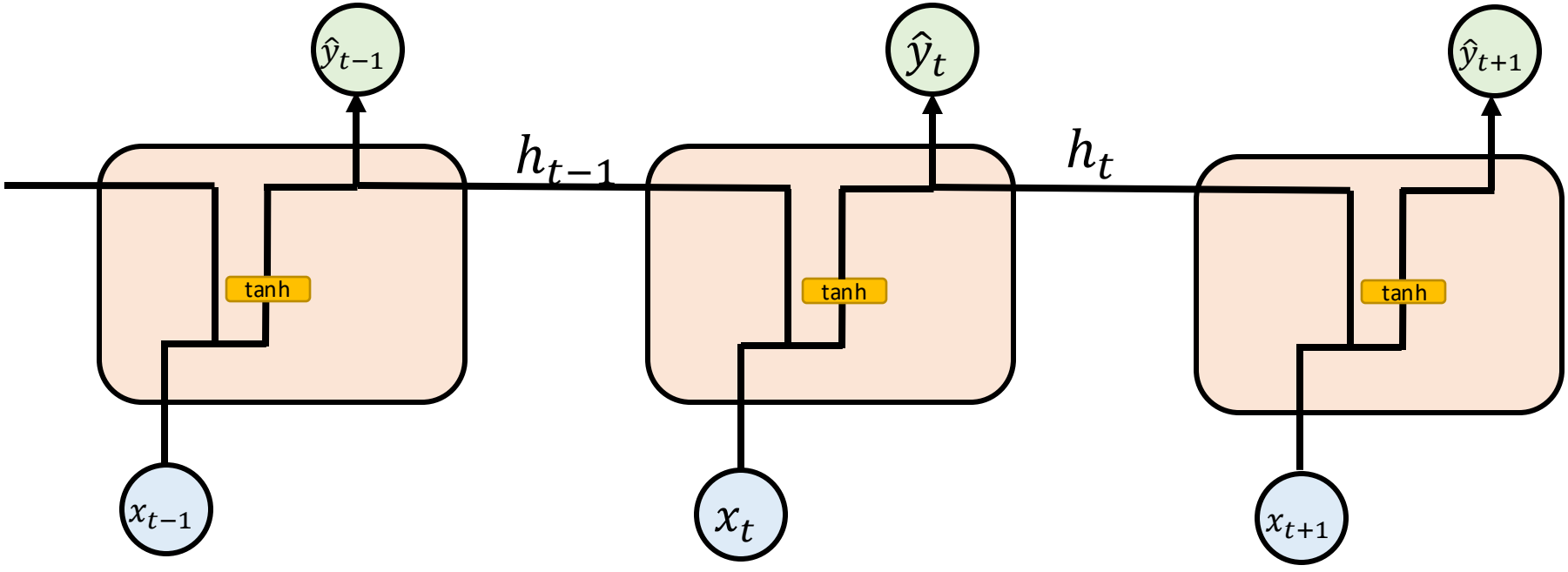


For an RNN, we also have the vanishing gradient problem.

It's hard for errors at the end of a sequence to influence the beginning.

3. Modélisation des données

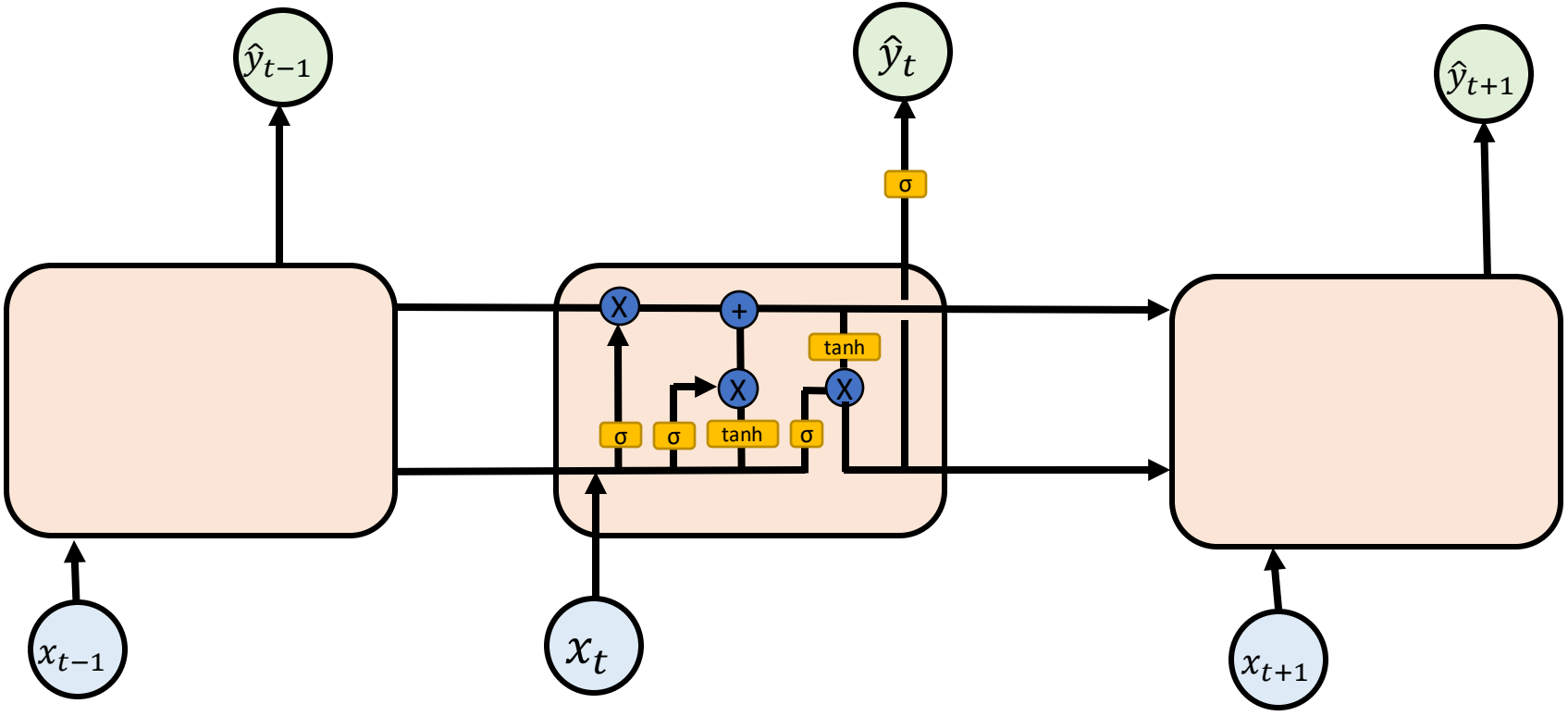
Standard RNN diagram



$$h_t = tanh(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$$
$$\hat{y}_t = g(W_{yh}h_t + b_y)$$

3. Modélisation des données

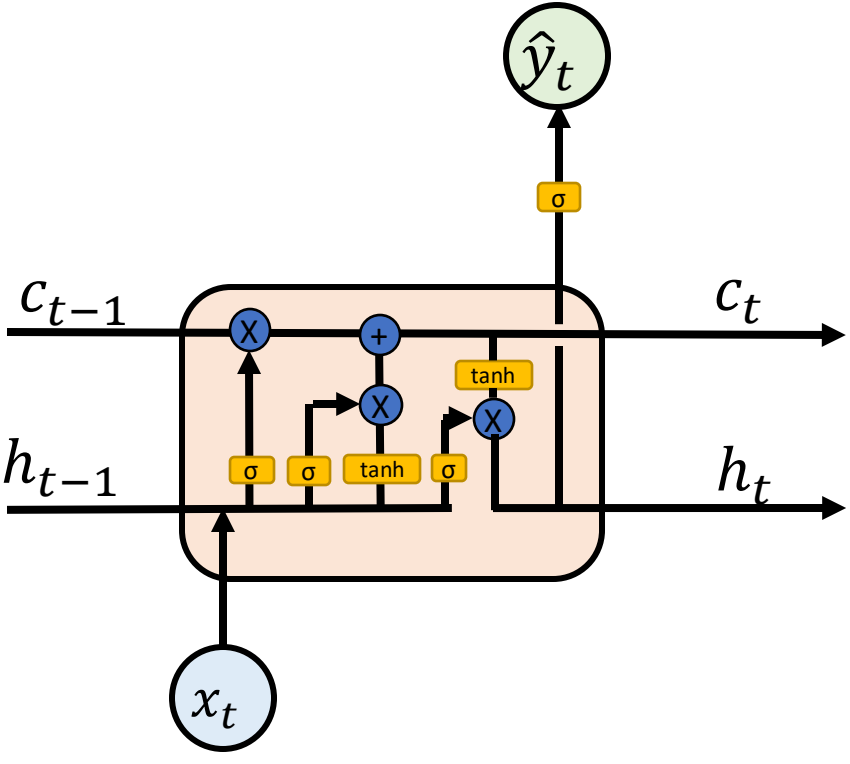
LSTM – Long Short Term Memory diagram

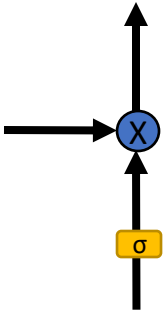


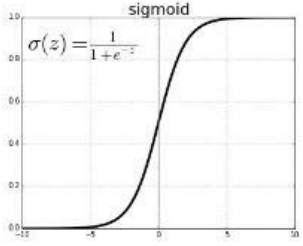
LSTM have the same chain-like structure.

3. Modélisation des données

LSTM – Long Short Term Memory diagram







Gates Γ are key component of LSTM. They optionally let the information flow.
Based on the sigmoid (most of the time, the output of the sigmoid is 0 or 1) and element-wise multiplication (X)

0 -> no information passes,

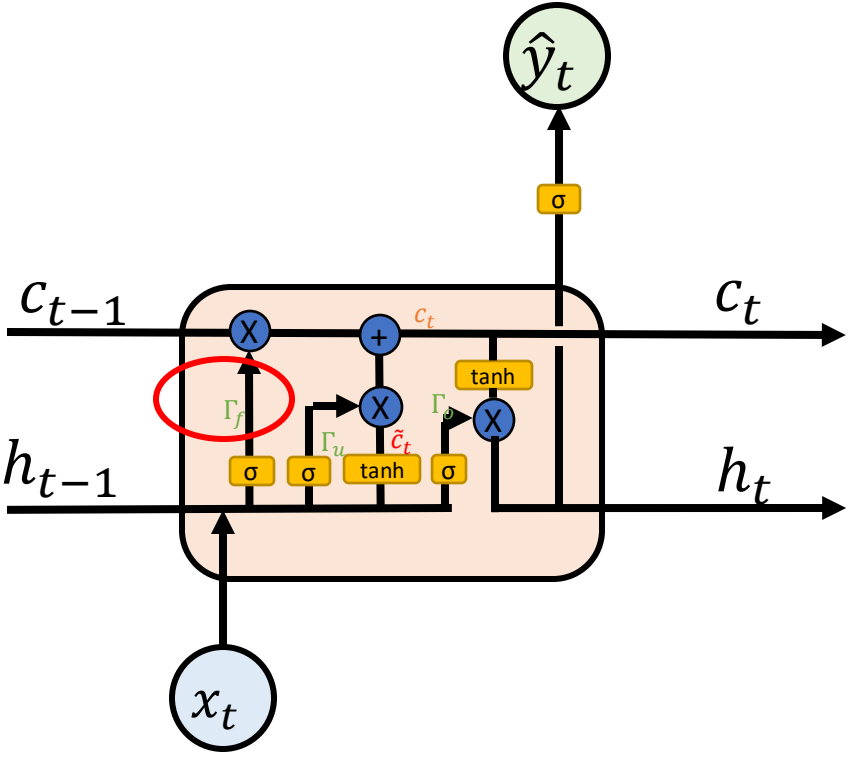
1 -> everything passes.

Then in addition of h, we called the *hidden state*, there is a separate *cell state* that works along with h.

yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

LSTM – Long Short Term Memory diagram



LSTM has 3 gates

1. Forget

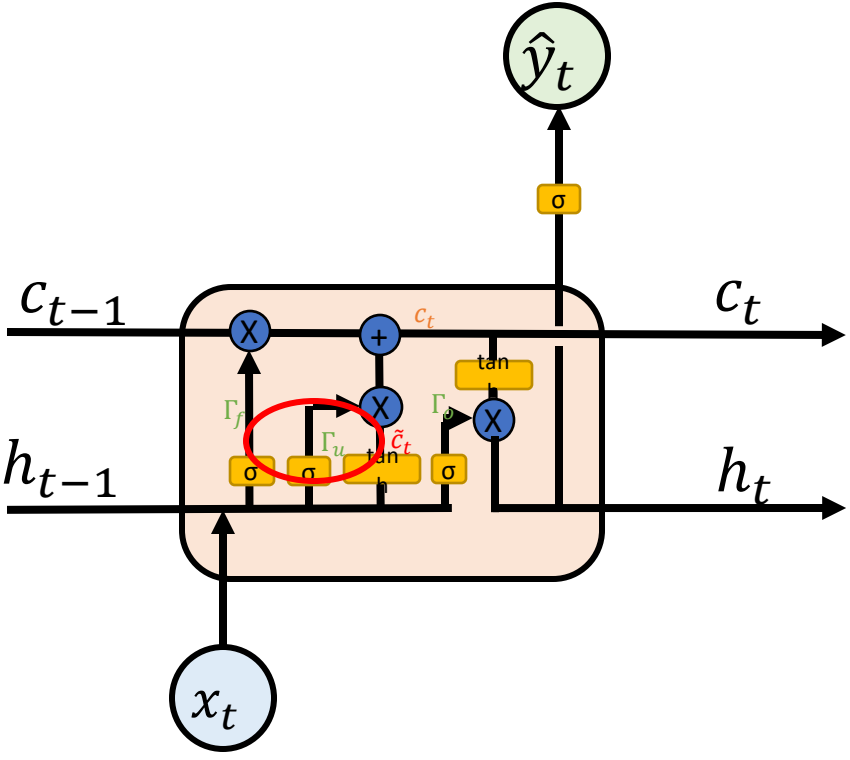
$$\Gamma_f = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f)$$

The forget gate decides what information we forget from the *cell state* c_{t-1}. It inputs h_{t-1} and x_t, and outputs numbers between 0 and 1. These numbers are multiplied by the *cell state* c_{t-1}.

A 1 means “completely keep this” while a 0 represents “completely get rid of this.”

3. Modélisation des données

LSTM – Long Short Term Memory diagram



LSTM has 3 gates

2. Update

$$\Gamma_u = \sigma(W_{uh}h_{t-1} + W_{ux}x_t + b_u)$$

$$\tilde{c}_t = \tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c)$$

$$c_t = \Gamma_u * \tilde{c}_t + \Gamma_f * c_{t-1}$$

The update gate Γ_u decides which values are updated.

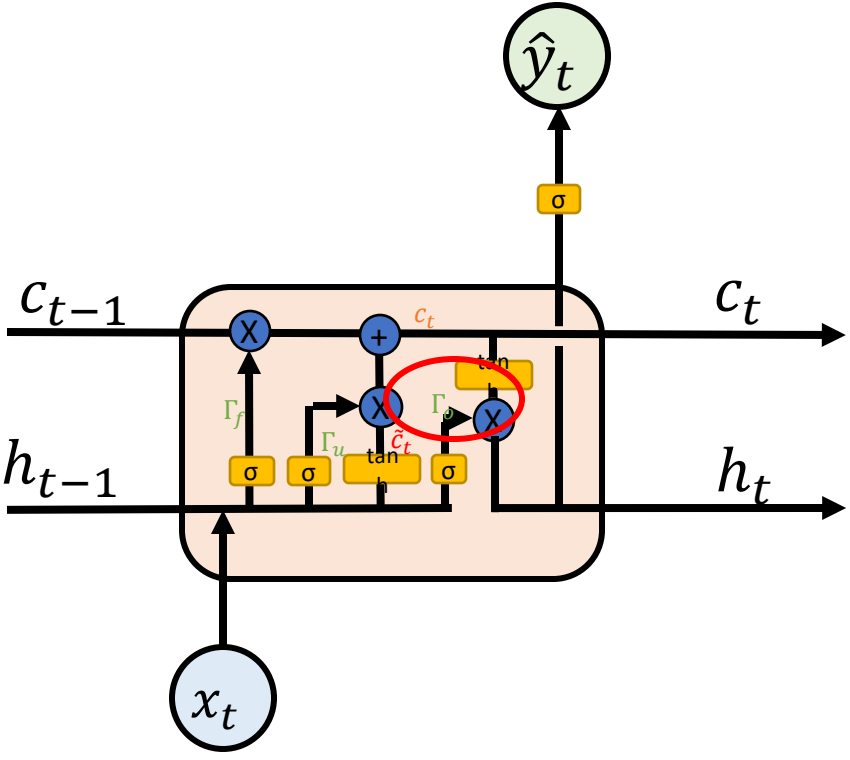
A *tanh* function creates new candidate cell state $\tilde{c}_t$.

We combine $\tilde{c}_t$ and c_{t-1} to obtain the new cell state.

yannick.benezeth@u-bourgogne.fr

3. Modélisation des données

LSTM – Long Short Term Memory diagram



LSTM has 3 gates

3. Output

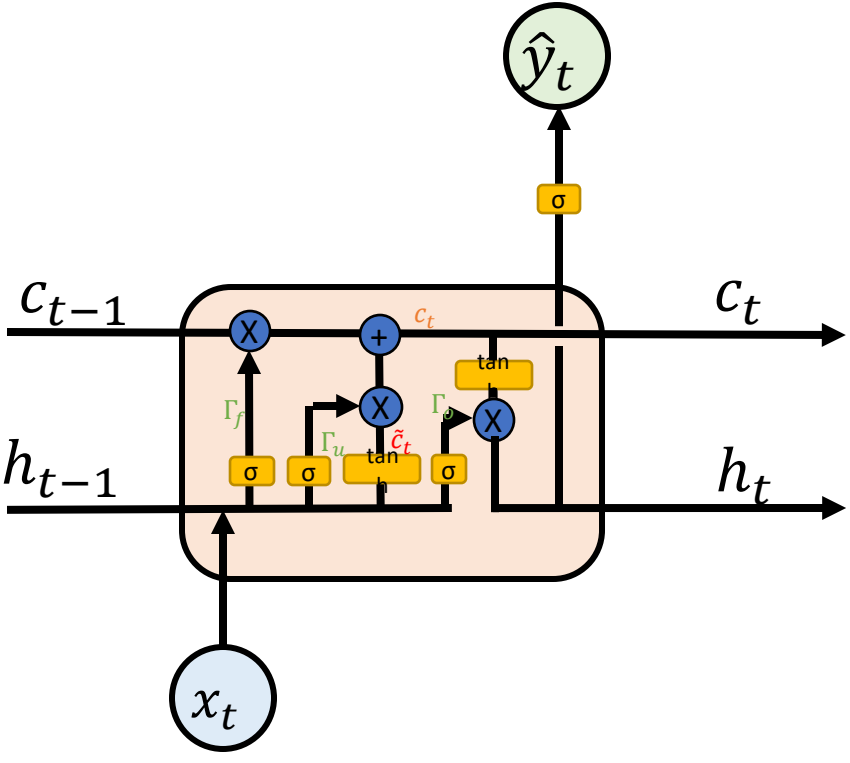
$$\Gamma_o = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o)$$

$$h_t = \tanh(c_t) * \Gamma_o$$

The output gate decides what we output (i.e. what parts of the cell state).

3. Modélisation des données

LSTM – Long Short Term Memory diagram



LSTM has 3 gates

1. Forget

$$\Gamma_f = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f)$$

2. Update

$$\Gamma_u = \sigma(W_{uh}h_{t-1} + W_{ux}x_t + b_u)$$

$$\tilde{c}_t = \tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c)$$

$$c_t = \Gamma_u * \tilde{c}_t + \Gamma_f * c_{t-1}$$

3. Output

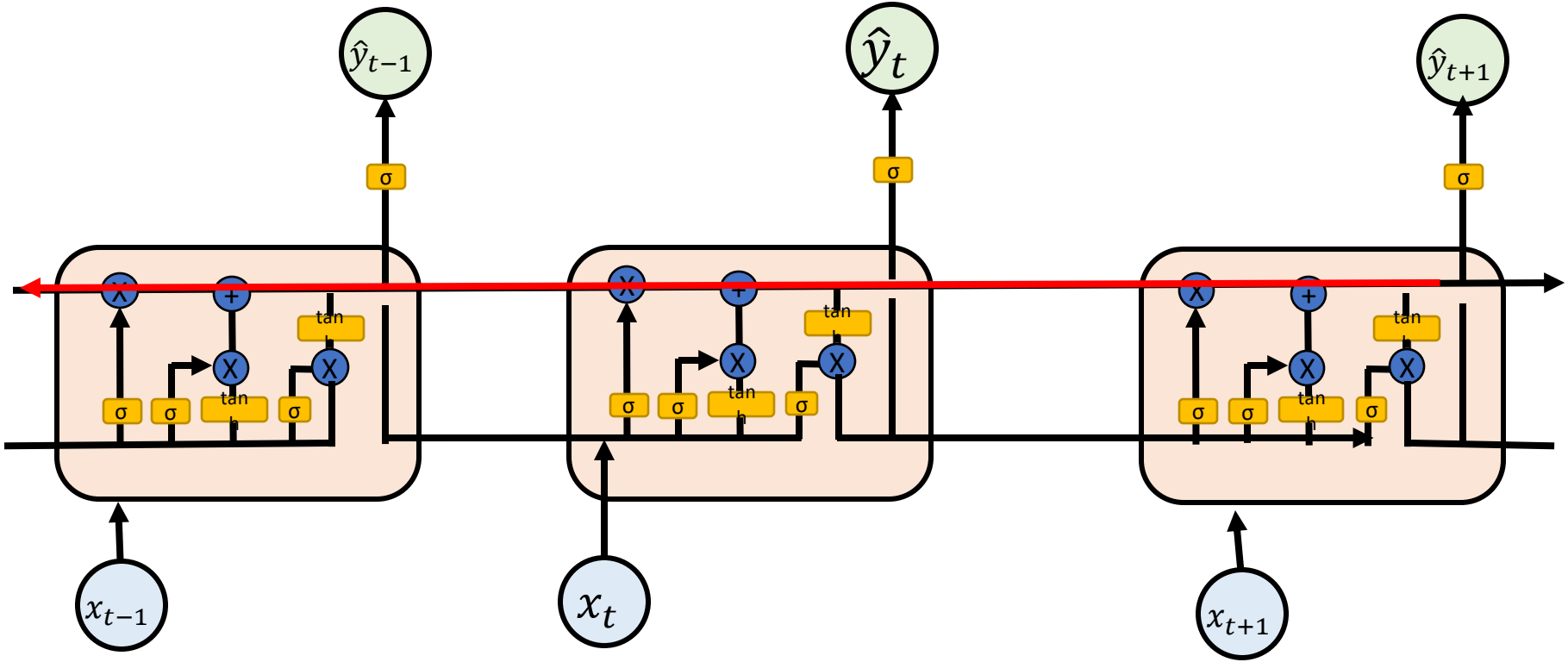
$$\Gamma_o = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o)$$

$$h_t = \tanh(c_t) * \Gamma_o$$

3. Modélisation des données

LSTM – Long Short Term Memory diagram

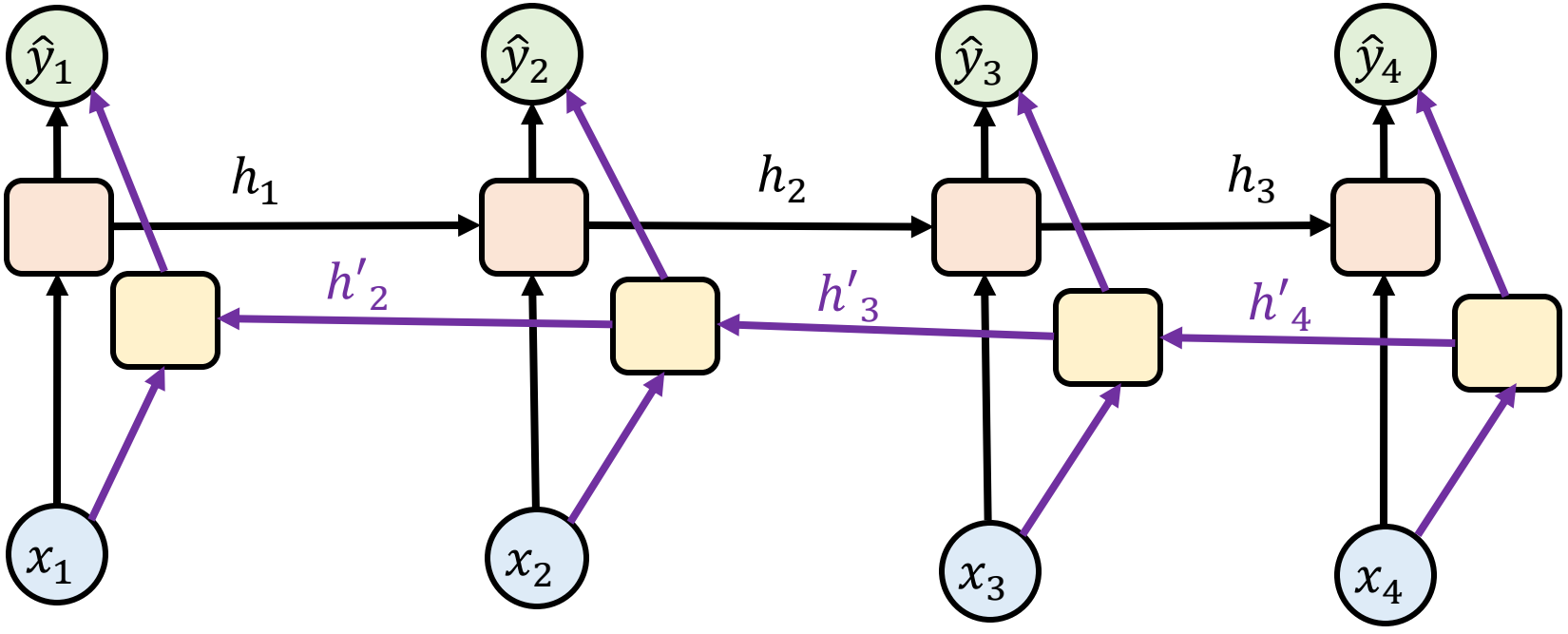
Uninterrupted gradient flow



3. Modélisation des données

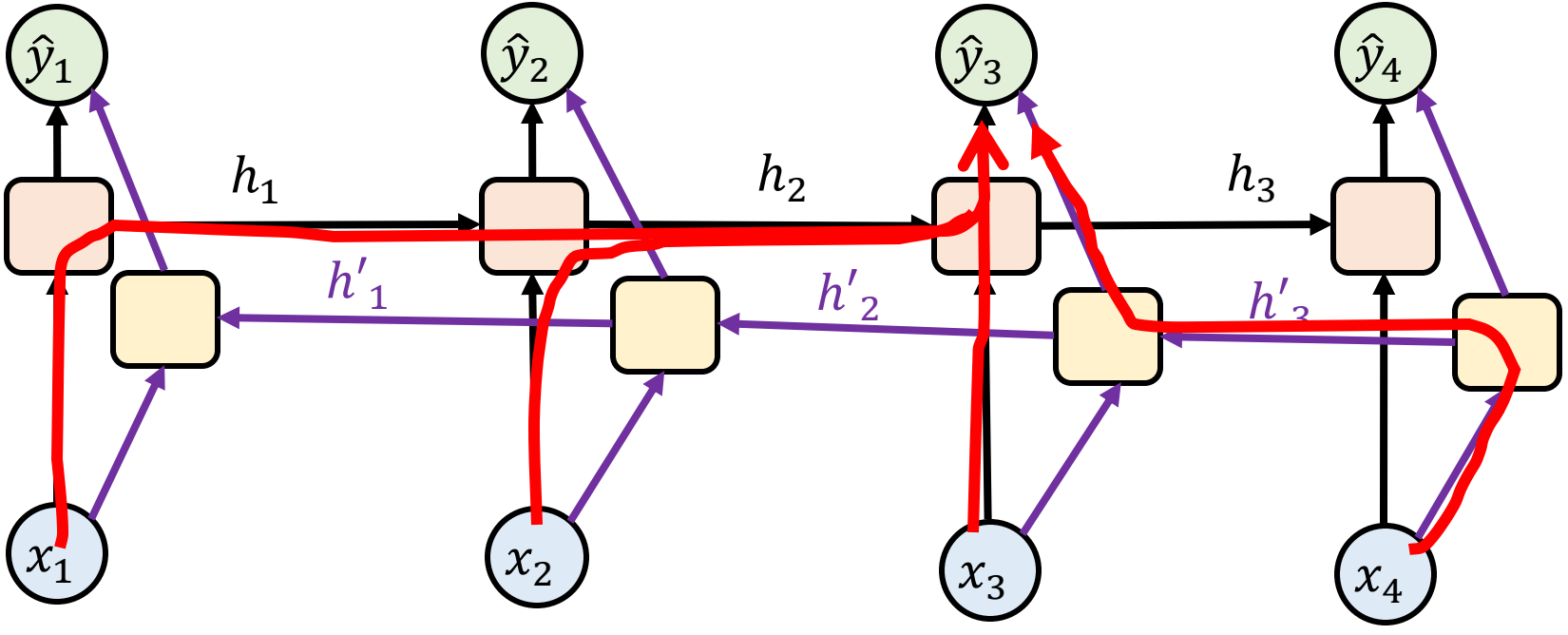
$$\hat{y}_t = g(W_1 h_t + W_2 h'_t + b_y)$$

Bidirectional RNN (BRNN)



3. Modélisation des données

Bidirectional RNN (BRNN)



3. Modélisation des données

Define the model with Keras

First, we import the main packages

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

Neural networks are defined in Keras as a sequence of layers.

```
model = keras.Sequential()
model.add(layers.LSTM(2))
model.add(layers.Dense(1))
```

← One LSTM layer with 2 memory cells followed by a dense output layer.

The shape of the input layer must define the size of the input data. Input must be in 3D comprised of samples, time steps, and features in that order.

You can specify the *input_shape* argument that expects a tuple containing the number of time steps and the number of features.

```
model = keras.Sequential()
model.add(layers.LSTM(5, input_shape=(2,1)))
model.add(layers.Dense(1))
```

← 2 timesteps and 1 feature for a univariate sequence with two observations per row