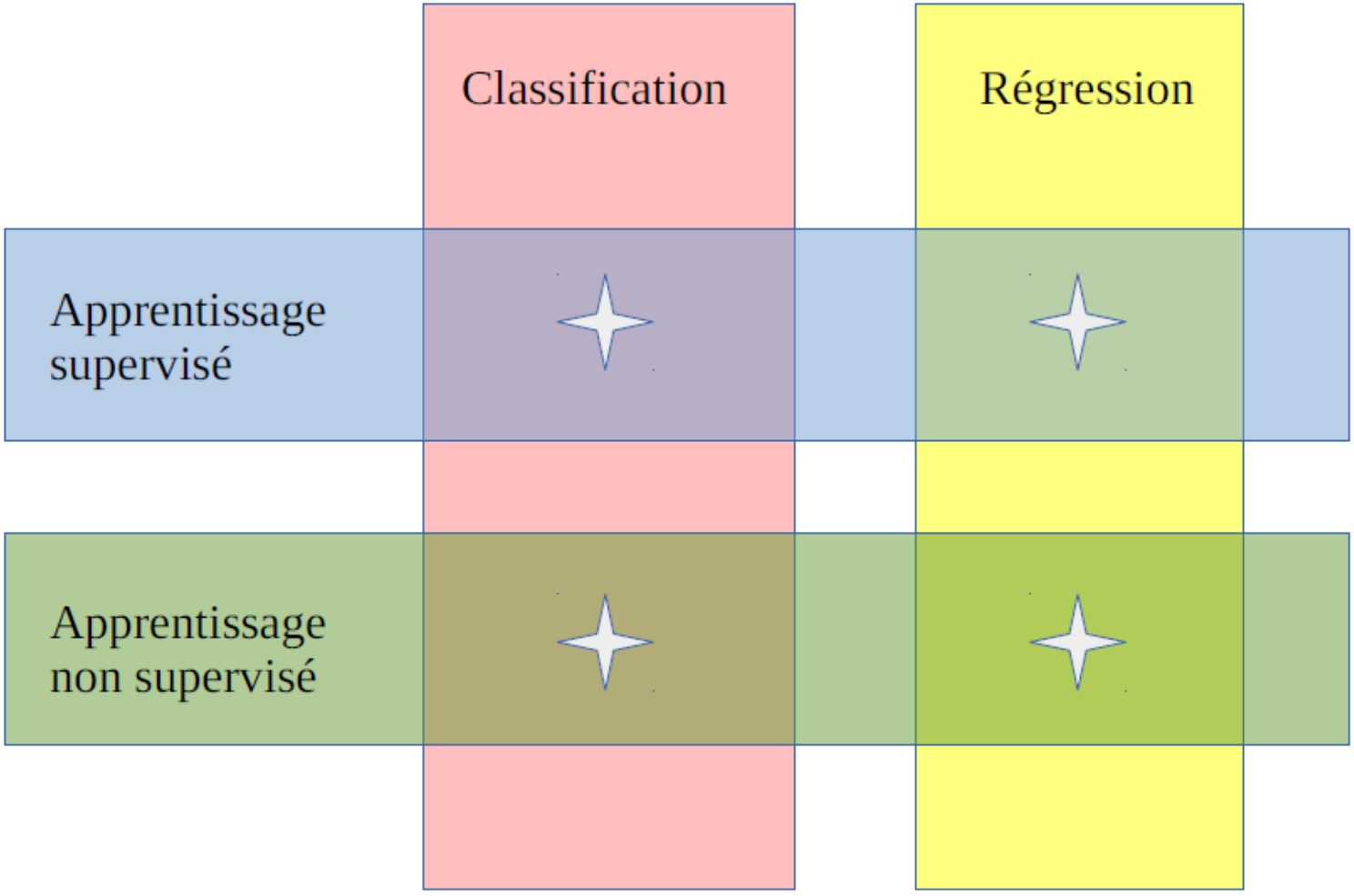
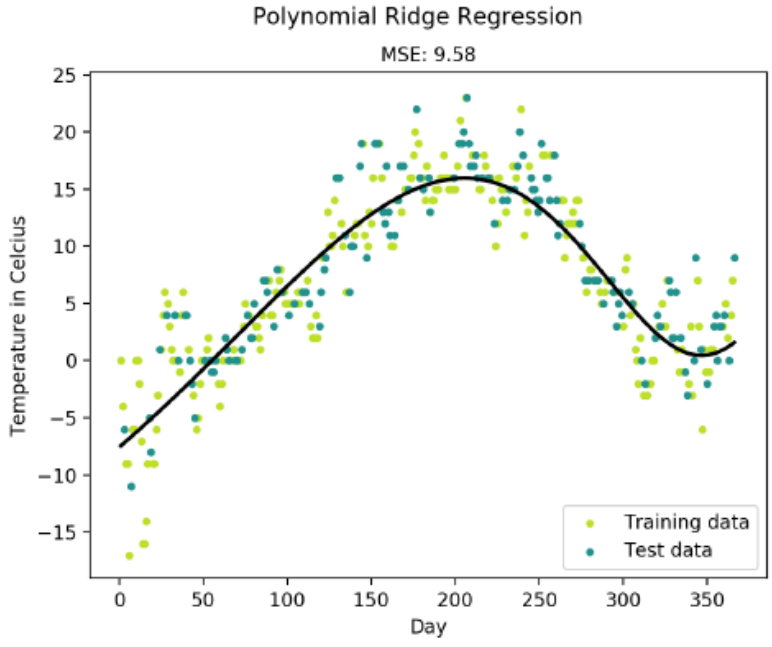
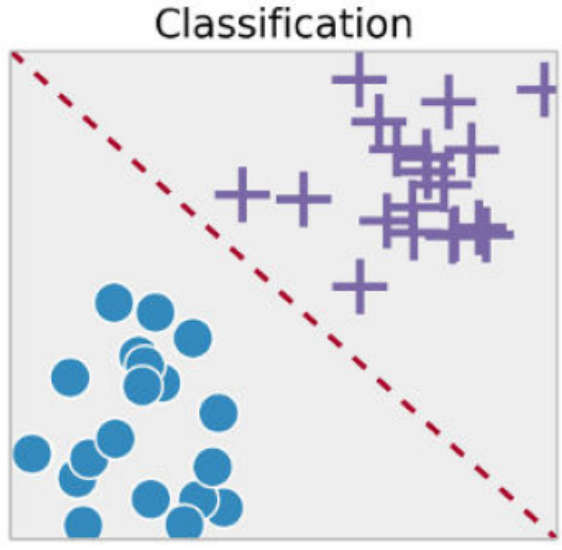
Introduction : le Machine Learning

Grandes catégories d'algorithmes de machine learning



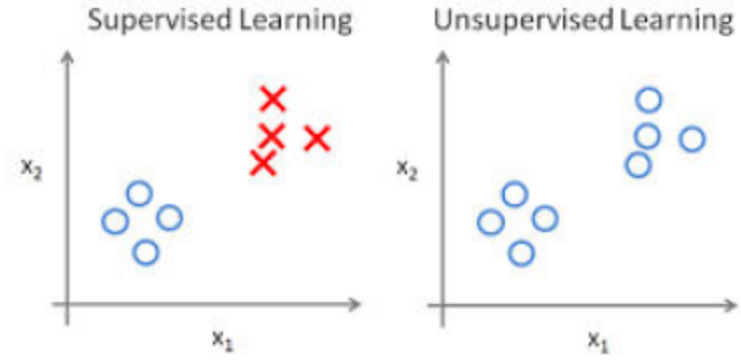
Classification / Régression





Régression	Classification
Prédire une variable quantitative	Prédire une classe (qualitative, discrète)

Apprentissage supervisé / non supervisé



■ Apprentissage supervisé :

- Nécessite un jeu d'entraînement X, y
  - X : prédicteurs
  - y : variable à prédire

■ Apprentissage non supervisé :

- Nécessite un jeu d'entraînement X
- Application principale : le clustering
- Exemple : classer des situations météo en groupes homogènes

Une première méthode de Machine Learning : la régression linéaire

La régression linéaire

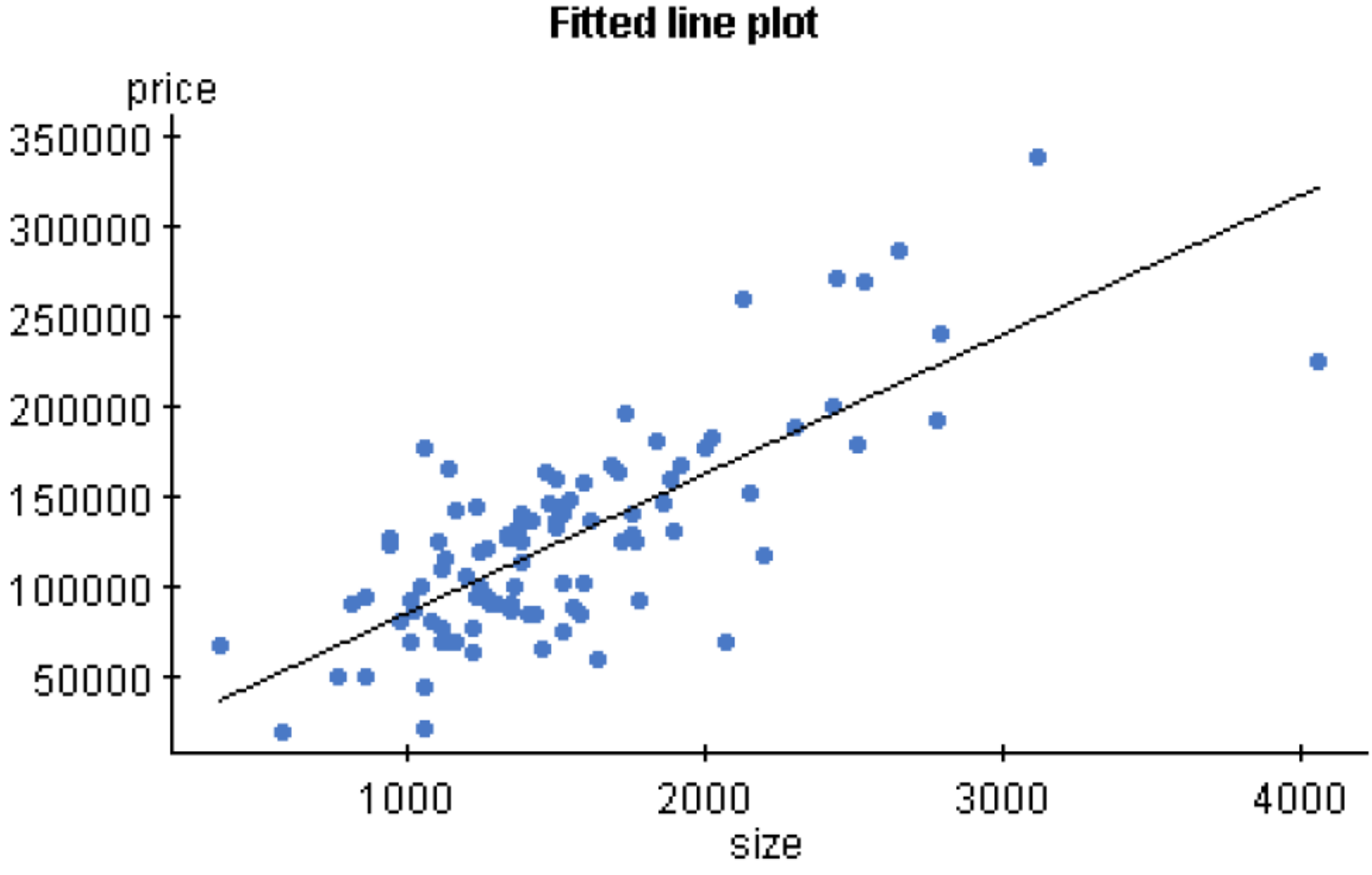
■ **Exemple : prévoir le prix de vente des maisons en fonction de leur taille**

■ **Méthode d'apprentissage supervisé :**

- Un jeu d'entraînement X, y
- X : la taille des maisons
- y : le prix

METEO FRANCE

Trouver la droite qui se rapproche le plus du nuage de points



La régression linéaire : fonction de coût

- **Comment définir la « meilleure » droite ?**

Définir une FONCTION DE COUT

- **La « meilleure » droite est celle qui minimise la fonction de coût.**

METEO FRANCE

La fonction de coût

- **Soit x,y un échantillon du jeu d'entraînement**
  - x = taille de la maison
  - y = prix de la maison
- **Soit h(x) notre prédiction : $h(x) = w_0.x + w_1$**
- **Une fonction de coût possible :**

$$J = \frac{1}{2m} \times \sum_{i=1}^{m} \left(h(x_i) - y_i\right)^2$$

m étant le nombre d'échantillons dans le jeu d'entraînement.

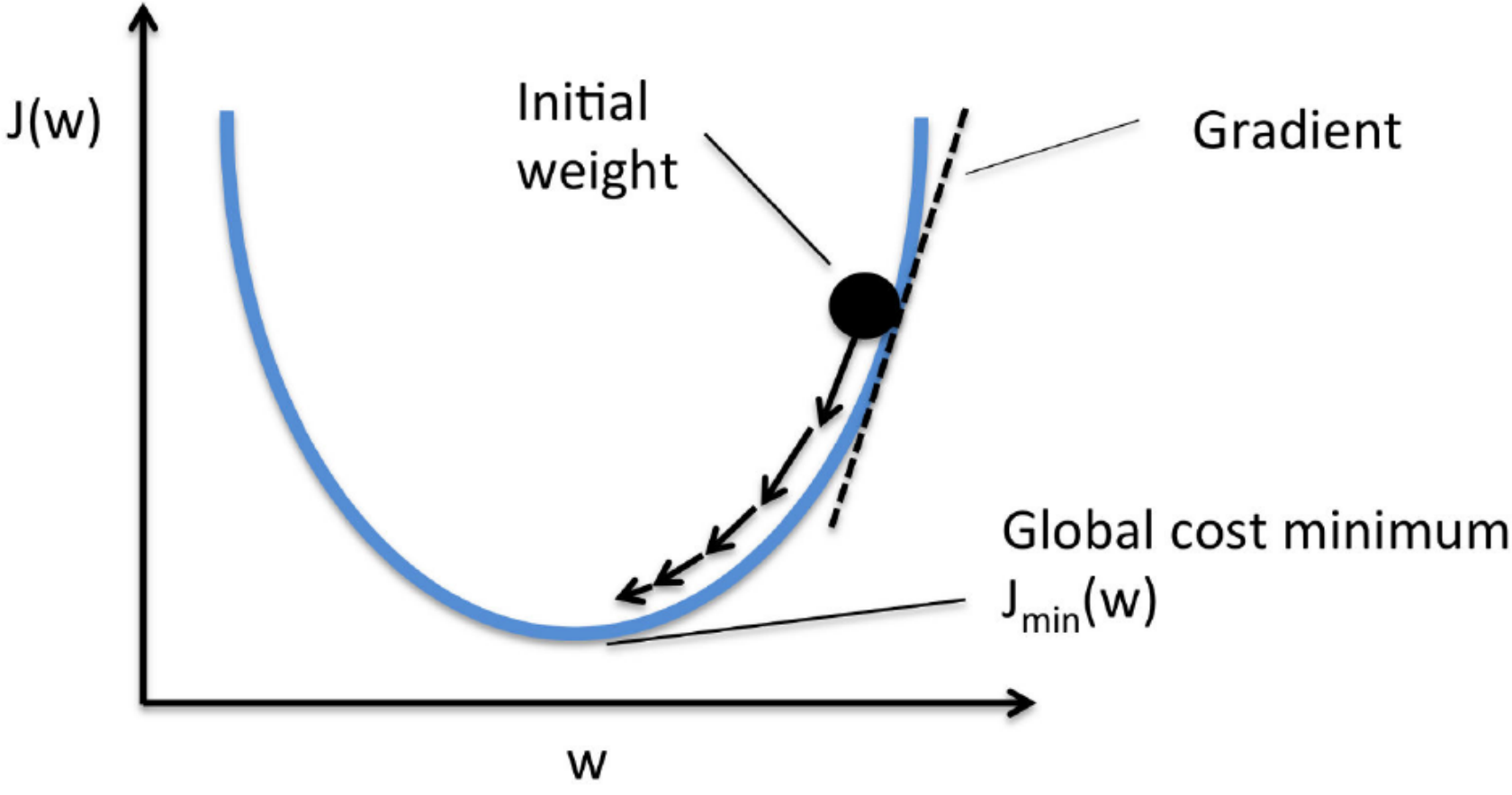
- **C'est l'écart quadratique moyen entre les prédictions et la vérité terrain**

METEO FRANCE

Comment trouver le minimum de la fonction de coût ?

Comment trouver le minimum de la fonction de coût ?

■ **La descente de gradient**



Application à la régression linéaire Calcul du gradient

■ **Application à la régression linéaire**

$$J = \frac{1}{2m} \times \sum_{i=1}^{m} \left(h(x_i) - y_i \right)^2$$

Avec $h(x) = w_0.x + w_1$

■ **Gradient :**

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} x_i . \left(h(x_i) - y_i \right)$$

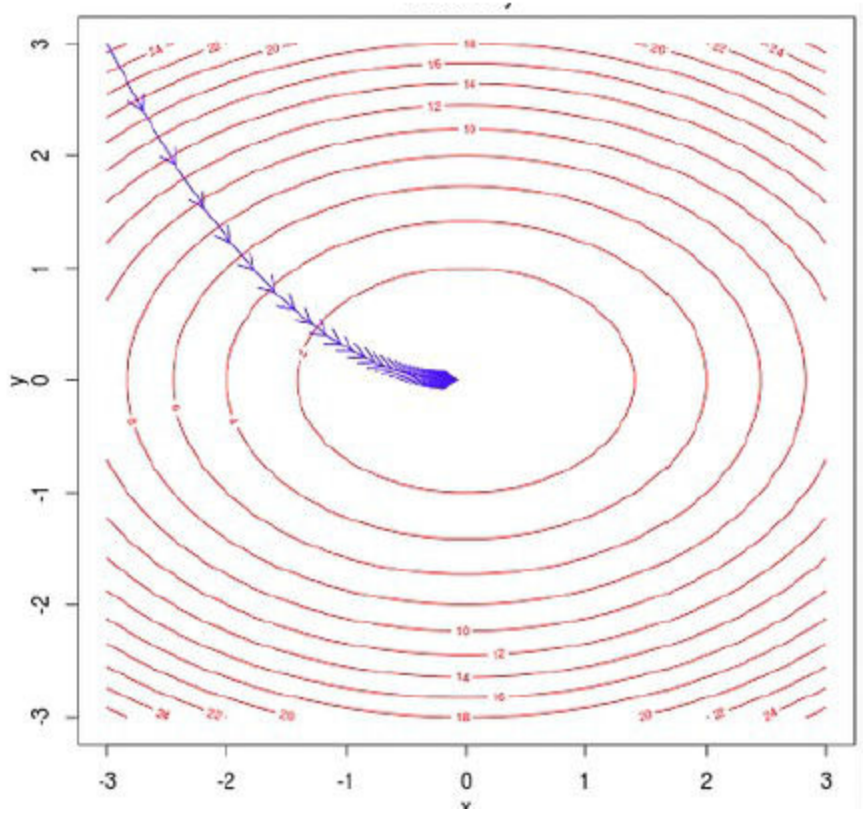
$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} \left(h(x_i) - y_i \right)$$

METEO FRANCE

C’est parti, on dévale la pente

- **Répéter autant de fois que nécessaire :**

$$\begin{cases} w_0 := w_0 - \alpha . \dfrac{\partial J}{\partial w_0} \\ w_1 := w_1 - \alpha . \dfrac{\partial J}{\partial w_1} \end{cases}$$

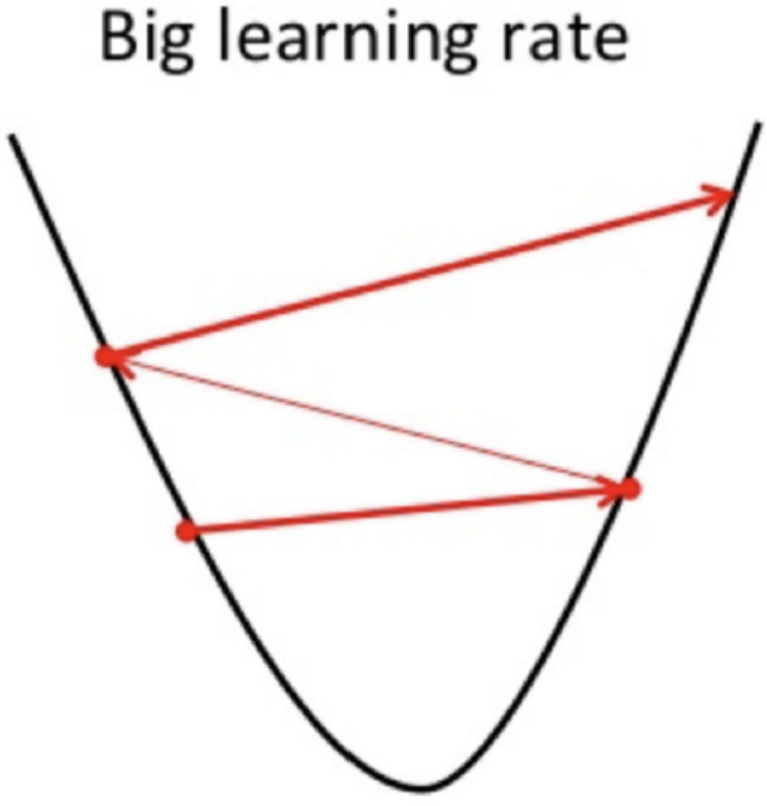


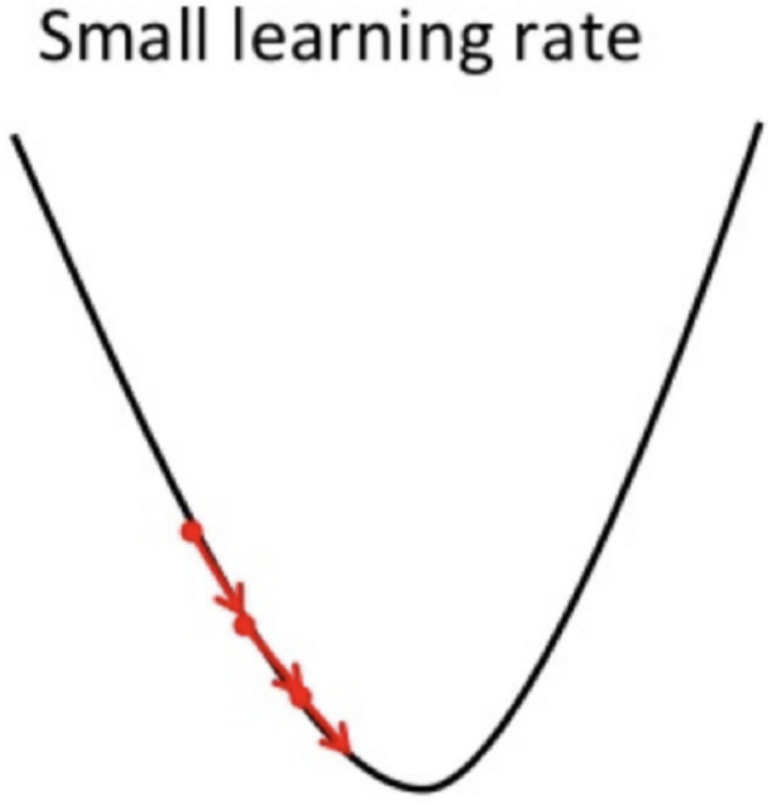
- **α est le coefficient d’apprentissage (learning rate)**

La convergence en images

https://www.youtube.com/watch?v=1hGsKphwC-A

Influence du learning rate





Et si le jeu de données est très gros ?

- **Rappel calcul des gradients pour la régression linéaire :**

$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^{m} x_i . (h(x_i) - y_i)$$

$$\frac{\partial J}{\partial w_1} = \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i)$$

- **Si m est très grand et X de dimension élevée, alors calculer la somme devient très long, voire interminable...**

- **Exemple : 1 million d'images en 1024x1024**

PROBLEME...

La descente de gradient stochastique

■ **La solution : la descente de gradient stochastique**

■ **A chaque étape de descente de gradient, au lieu de prendre l'ensemble des échantillons d'un coup comme ceci :**

$$\left\{ \begin{array}{l} w_0 := w_0 - \alpha . \frac{1}{m} \sum_{i=1}^{m} x_i . (h(x_i) - y_i) \\ w_1 := w_1 - \alpha . \frac{1}{m} \sum_{i=1}^{m} (h(x_i) - y_i) \end{array} \right\}$$

on itère sur les échantillons un par un :

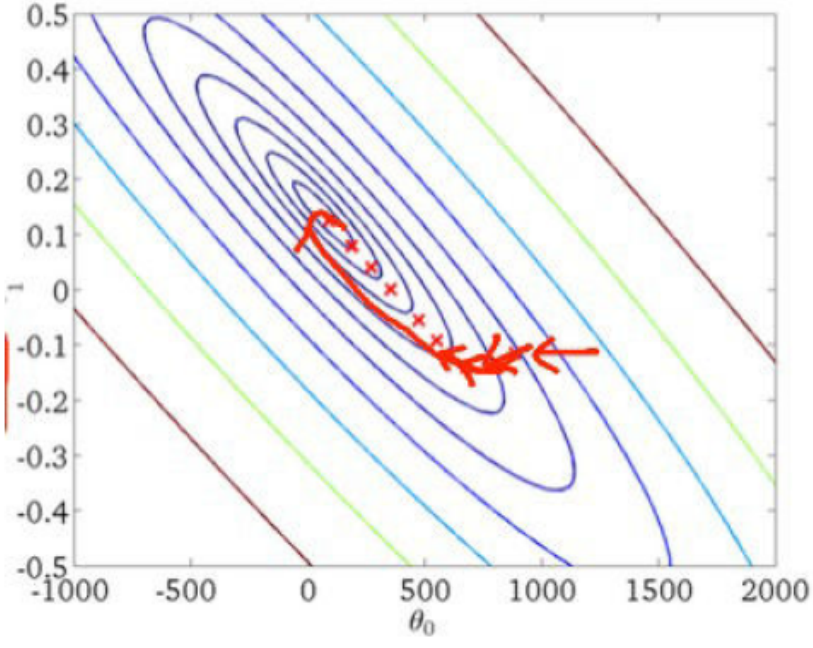
pour i allant de 1 à m, répéter :

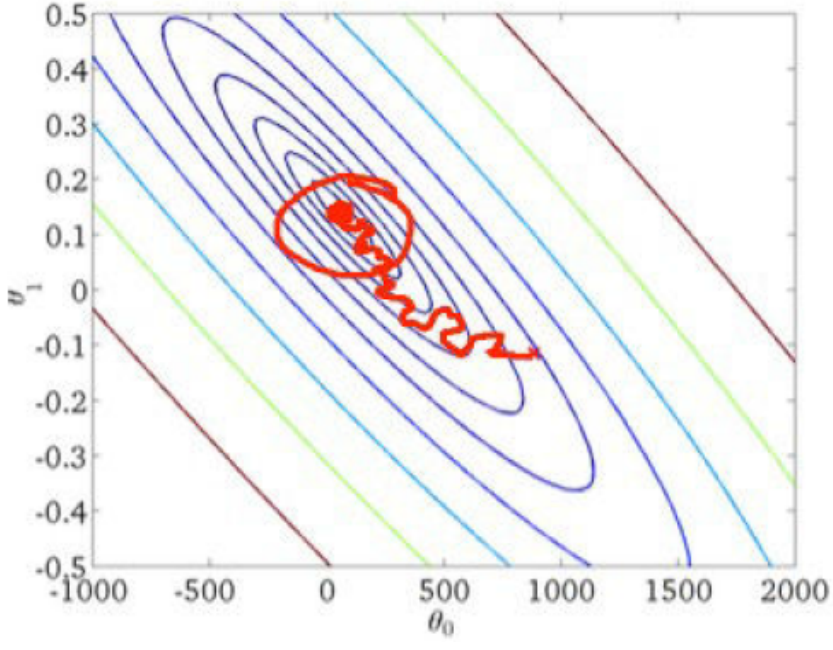
$$\left\{ \begin{array}{l} w_0 := w_0 - \alpha . \frac{1}{m} x_i . (h(x_i) - y_i) \\ w_1 := w_1 - \alpha . \frac{1}{m} (h(x_i) - y_i) \end{array} \right\}$$

METEO FRANCE

Illustration

Full batch gradient descent (FBGD)	Stochastic gradient descent (SGD)





Démo en images

https://www.youtube.com/watch?v=HvLJUsEc6dw

Mini-batch

■ **Full batch gradient descent : on calcule le gradient sur l'ensemble du jeu de données**

- Inconvénient : beaucoup trop long sur gros jeu de données

■ **SGD : on estime le gradient échantillon par échantillon**

- Inconvénient : lent et convergence plus chaotique

■ **Compromis : mini-batch gradient descent**

- On estime le gradient sur k échantillons à la fois (par exemple 32 échantillons)

C'est la méthode utilisée en pratique

METEO FRANCE

La notion d'epoch

■ **Dans la SGD, on estime le « gradient » échantillon par échantillon, ou par mini-batches de quelques échantillons**

- Une passe complète sur le jeu de données s'appelle :

UNE EPOCH

■ **Le nombre d'époch est donc le nombre de passes effectuées sur le jeu d'entraînement lors de l'apprentissage.**

Des questions sur la descente de gradient ?

Hyper-paramètres – comment « régler » un modèle de Machine Learning

■ **Que peut-on modifier dans un modèle de Machine Learning ?**

— Le type et la complexité du modèle

- La régression linéaire est un modèle simple, mais on peut le complexifier : polynôme de degré n, random forest, réseaux de neurones...

— Certains paramètres spécifiques du modèle

- Pour un réseau de neurones : nombre de couches et nombre de neurones par couche...

— Le learning rate

— La taille des mini-batches

■ **Comment choisir ces hyper-paramètres ?**

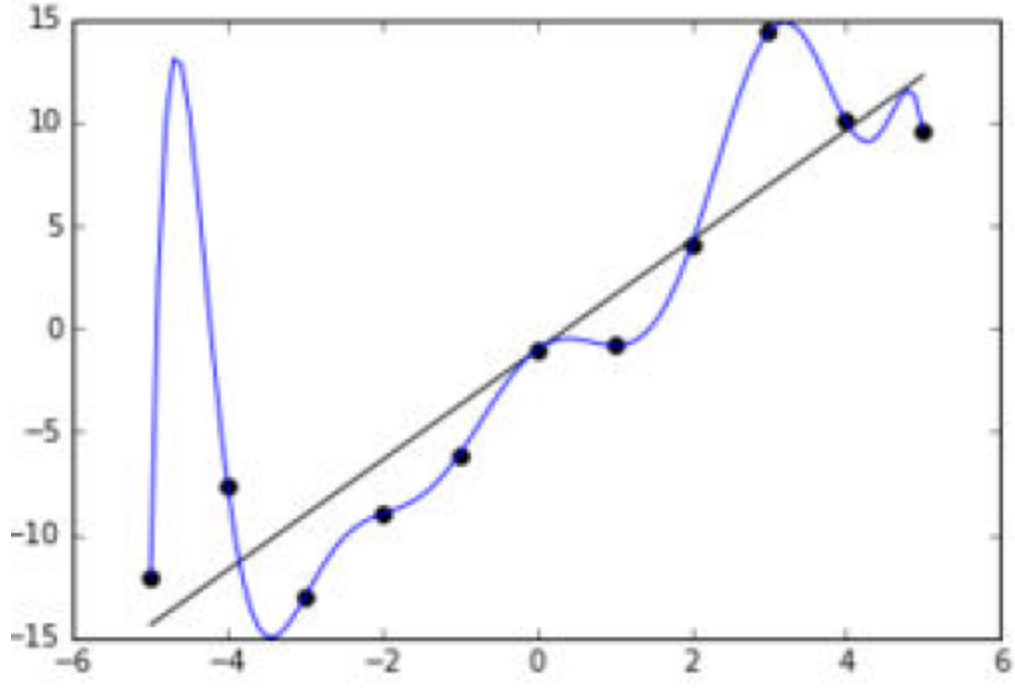
METEO FRANCE

Evaluer le modèle

Première idée : choisir les hyper-paramètres qui fonctionnent le mieux sur le jeu d'entraînement

Jeu de données d'entraînement

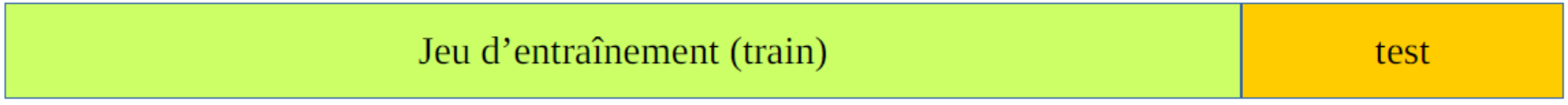
Pas bon. Le modèle risque de ne pas être capable de généraliser.



METEO FRANCE

Evaluer le modèle

Deuxième idée : choisir les hyper-paramètres qui fonctionnent le mieux sur un jeu de test



Pas bon. Aucune garantie que l'algorithme fonctionnera bien sur de nouvelles données.

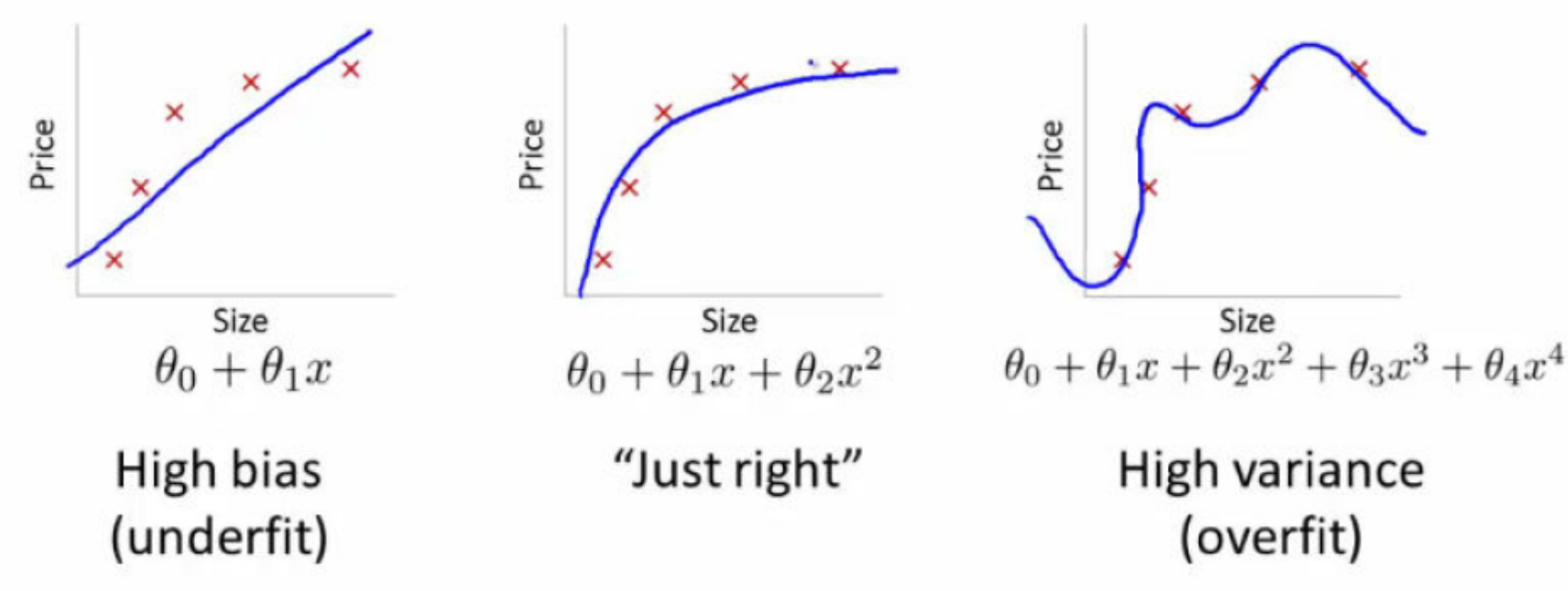
METEO FRANCE

Evaluer le modèle

Troisième idée : entraîner sur le jeu d'entraînement, choisir les hyper-paramètres qui fonctionnent le mieux sur un jeu de validation, puis une fois le modèle réglé, l'évaluer sur un jeu de test.

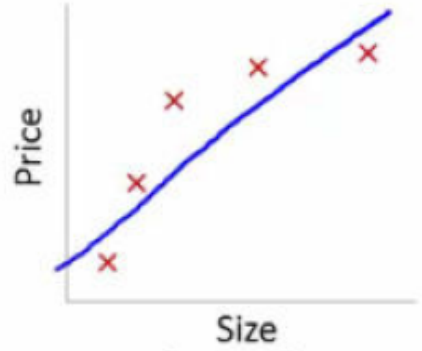
C'est mieux !

Sous-apprentissage - Sur-apprentissage



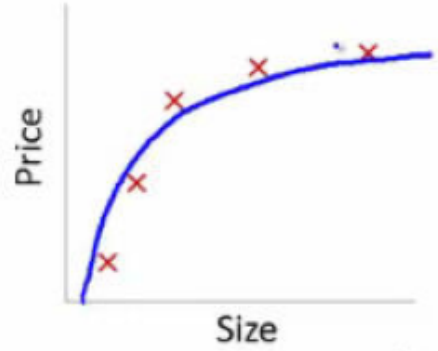
Sous-apprentissage	Bon modèle	Sur-apprentissage
modèle trop simple pour expliquer la variance		modèle qui colle trop au bruit du jeu de données

Sous-apprentissage - Sur-apprentissage



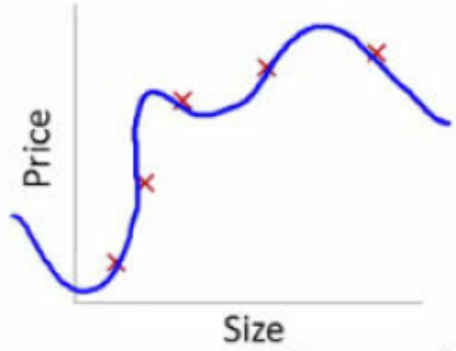
$\theta_0 + \theta_1 x$

High bias
(underfit)



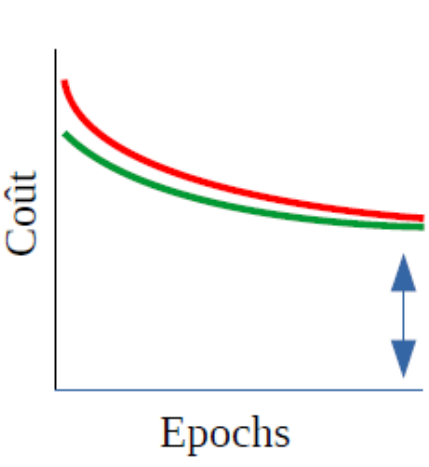
$\theta_0 + \theta_1 x + \theta_2 x^2$

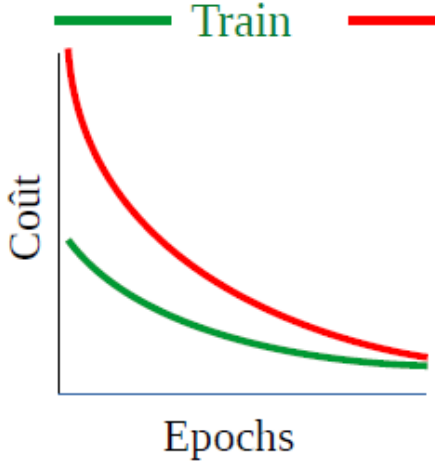
"Just right"

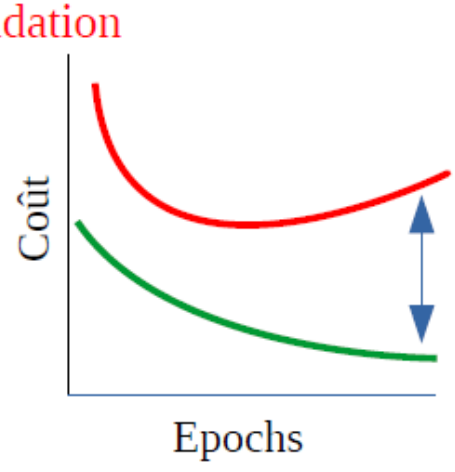


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)







Combattre l’underfitting

■ Combattre l’underfitting

— Complexifier le modèle

- Ex : modèle quadratique au lieu d’un modèle linéaire pour prédire le prix des maisons

— Ajouter des prédicteurs

- Ex : il existe d’autres paramètres que la taille qui influent sur le prix des maisons. Par exemple le nombre de chambres, la distance au centre-ville...

Combattre l'overfitting

■ Combattre l'overfitting

— Ajouter des données d'entraînement

- Ex : deux exemples de maisons ne permettent pas de créer un modèle qui généralise bien

— Simplifier le modèle ou retirer des prédicteurs

- Eviter que le modèle parvienne à « apprendre par coeur » le jeu d'entraînement

— Entraîner le modèle moins longtemps

- En français, l'overfitting se dit « surapprentissage ». Donc évitons au modèle de surapprendre.

— Limiter la capacité d'apprentissage du modèle

- Il existe plusieurs méthodes dont la régularisation et le dropout.

— Utiliser des ensembles

- C'est comme en météo. Entraîner plusieurs modèles et combiner leurs prédictions.

METEO FRANCE

Des questions?