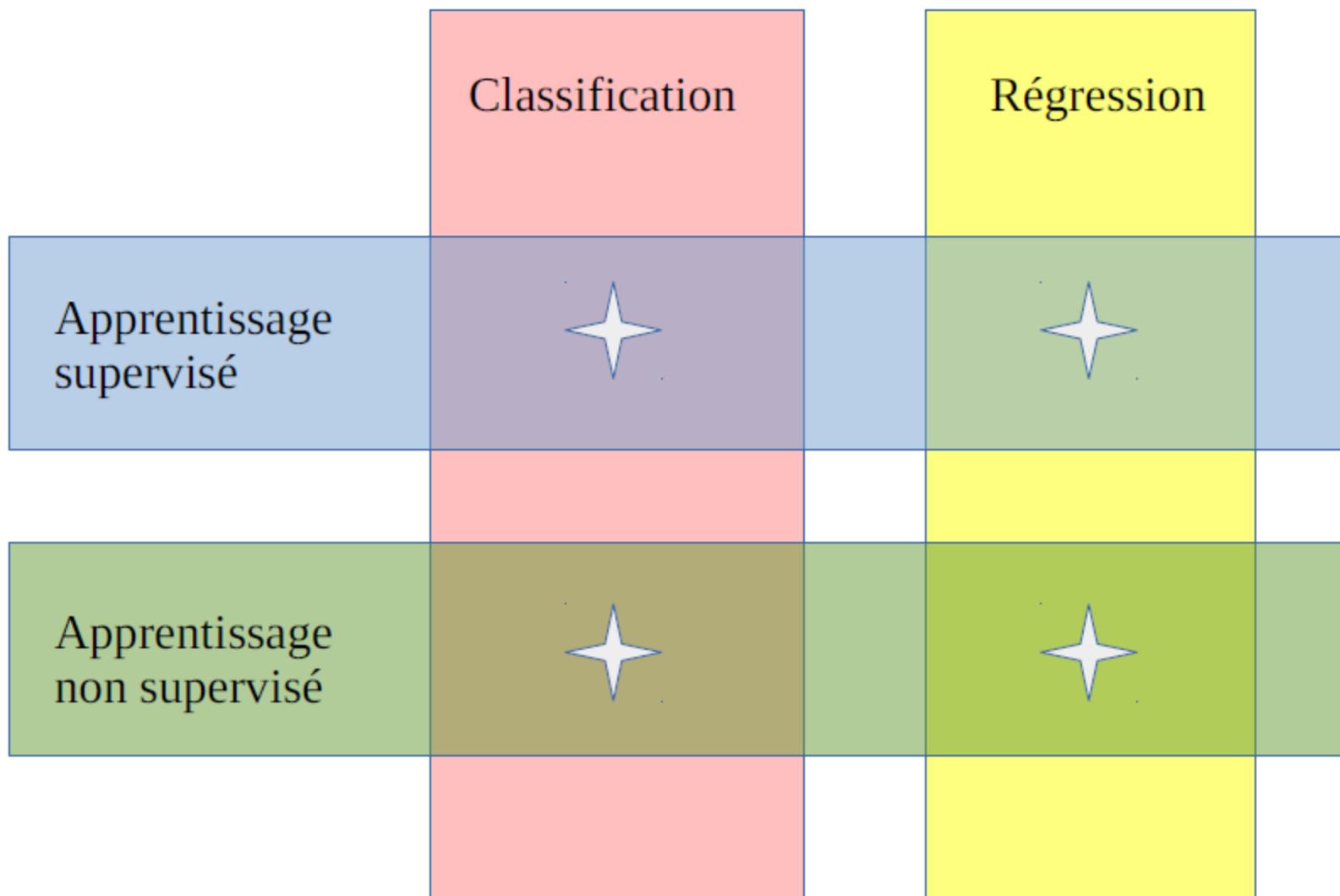




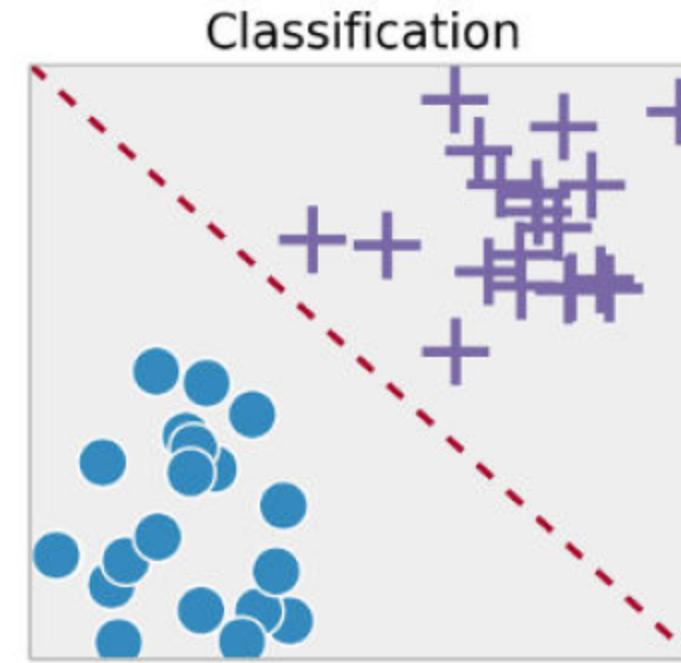
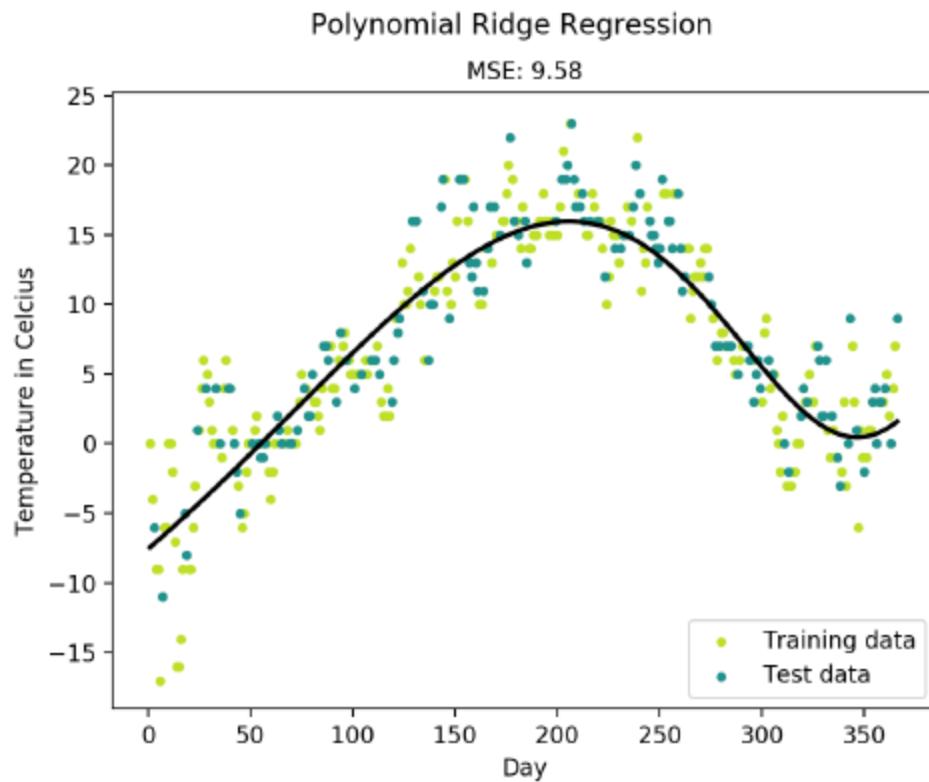
Introduction : le Machine Learning

Présentation partagée sous la licence Apache 2.0

Grandes catégories d'algorithmes de machine learning



Classification / Régression



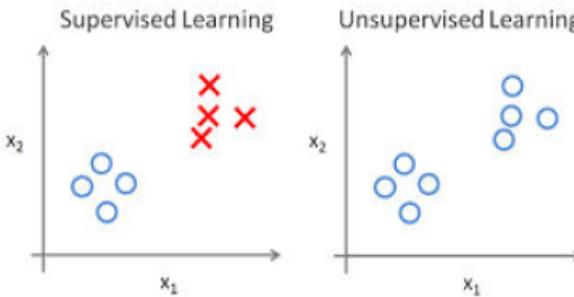
Régression

Prédire une variable quantitative

Classification

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

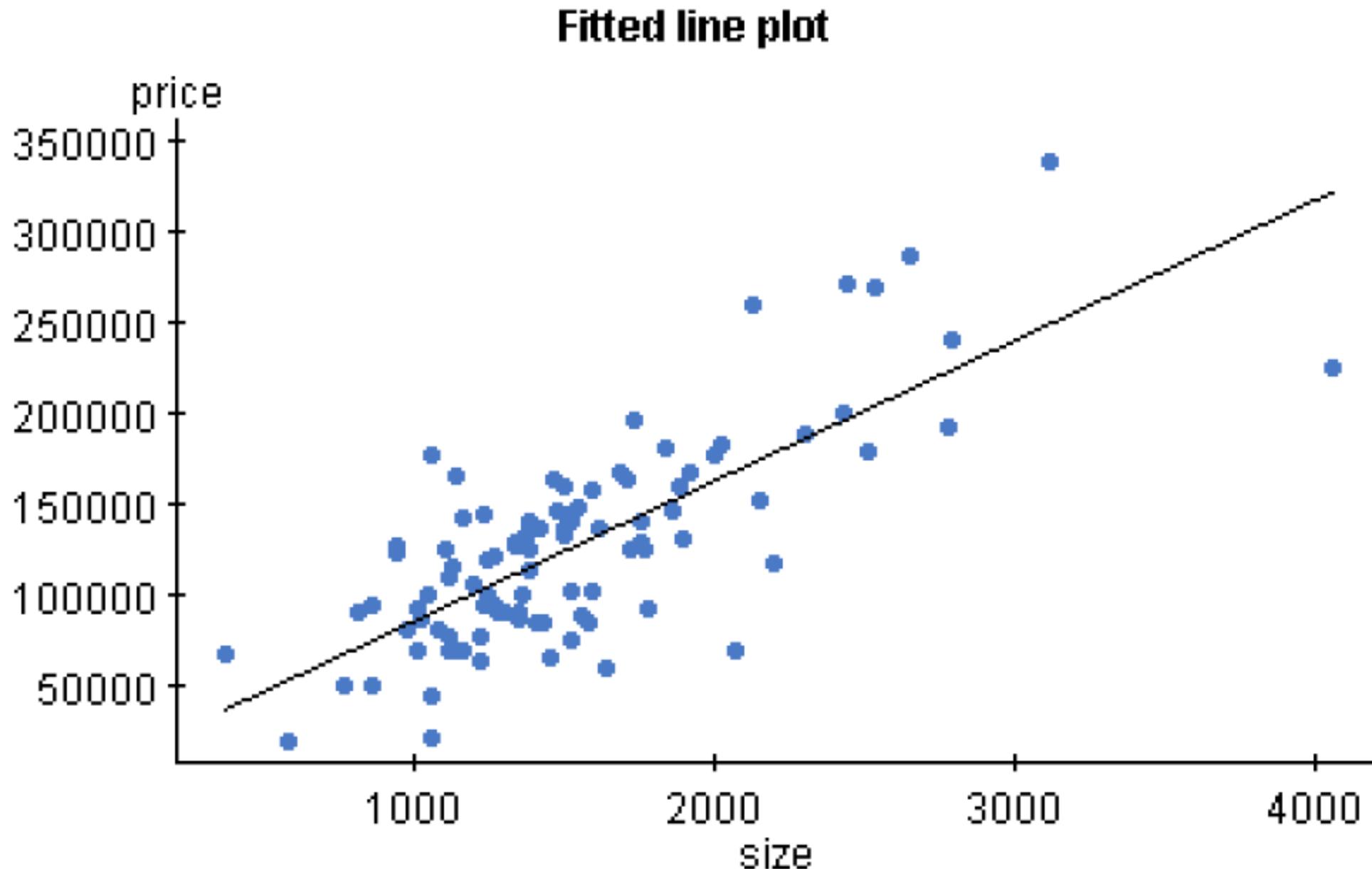
Une première méthode de Machine Learning : la régression linéaire

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

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.

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 : $h(x) = w_0 \cdot x + w_1$

■ Une fonction de coût possible :

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

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

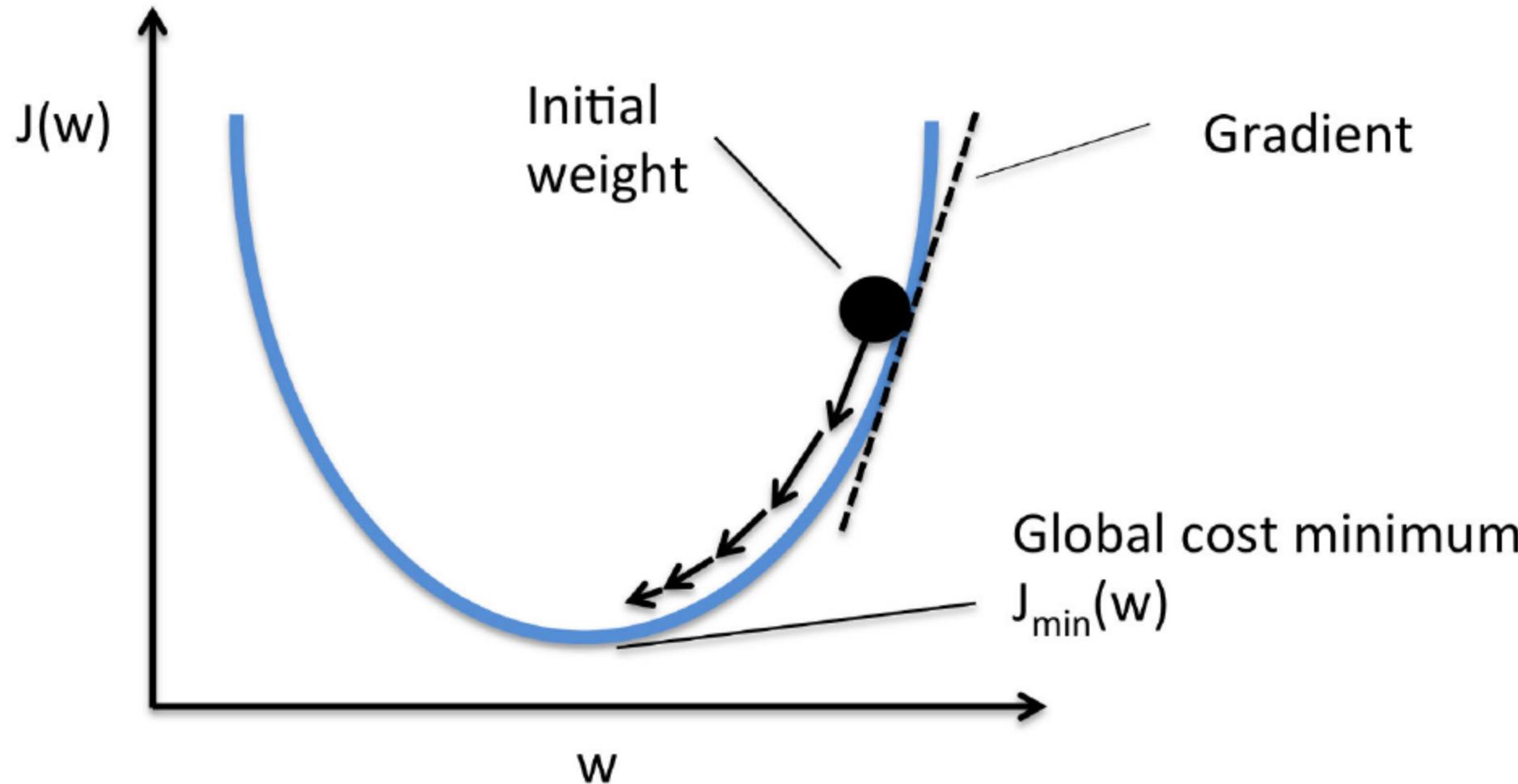
■ C'est l'écart quadratique moyen entre les prédictions et la vérité terrain

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

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

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

■ Gradient :

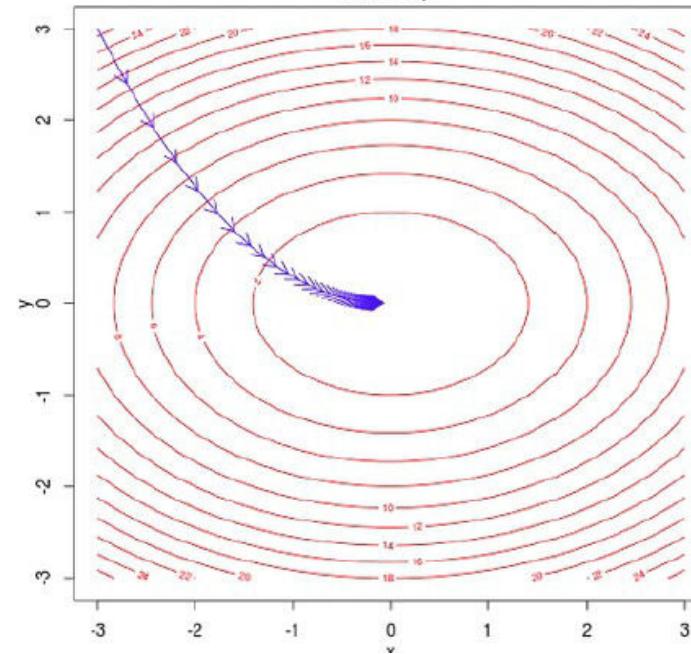
$$\frac{\partial J}{\partial w_0} = \frac{1}{m} \sum_{i=1}^m x_i \cdot (h(x_i) - y_i)$$

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

C'est parti, on dévale la pente

- Répéter autant de fois que nécessaire :

$$\begin{cases} w_0 := w_0 - \alpha \cdot \frac{\partial J}{\partial w_0} \\ w_1 := w_1 - \alpha \cdot \frac{\partial J}{\partial w_1} \end{cases}$$



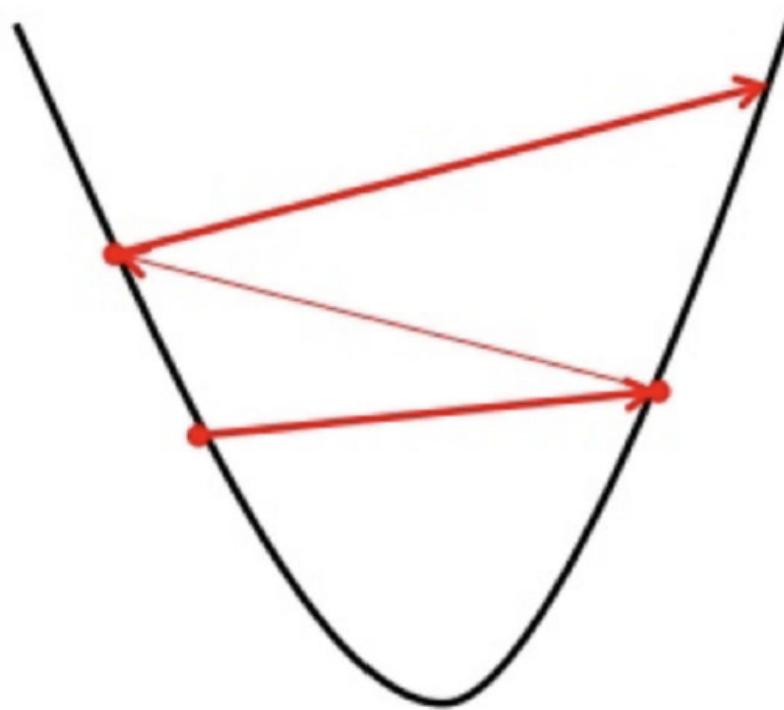
- α est le coefficient d'apprentissage (learning rate)

La convergence en images

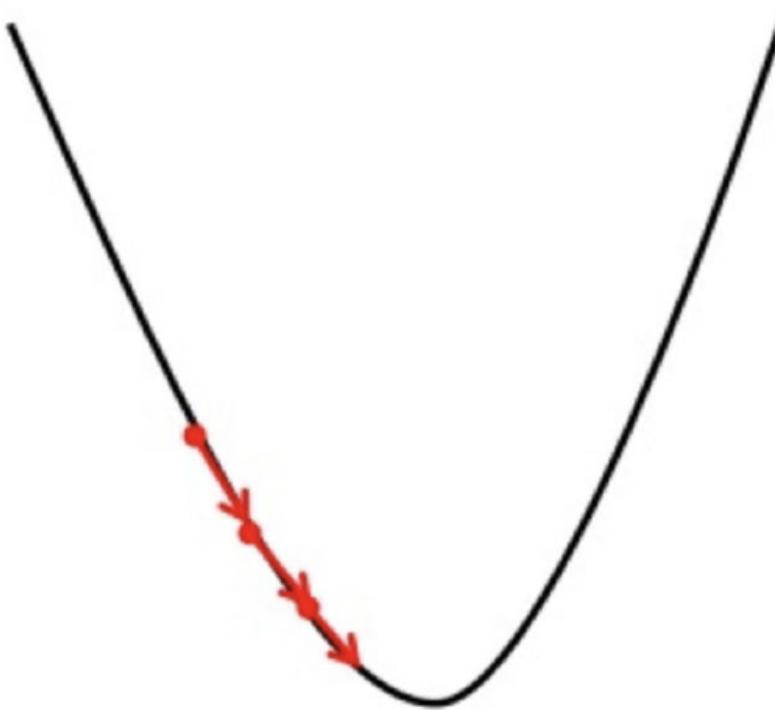
<https://www.youtube.com/watch?v=1hGsKphwC-A>

Influence du learning rate

Big learning rate



Small learning rate



Et si le jeu de données est très gros ?

- Rappel calcul des gradients pour la régression linéaire :

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

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

- 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 :

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

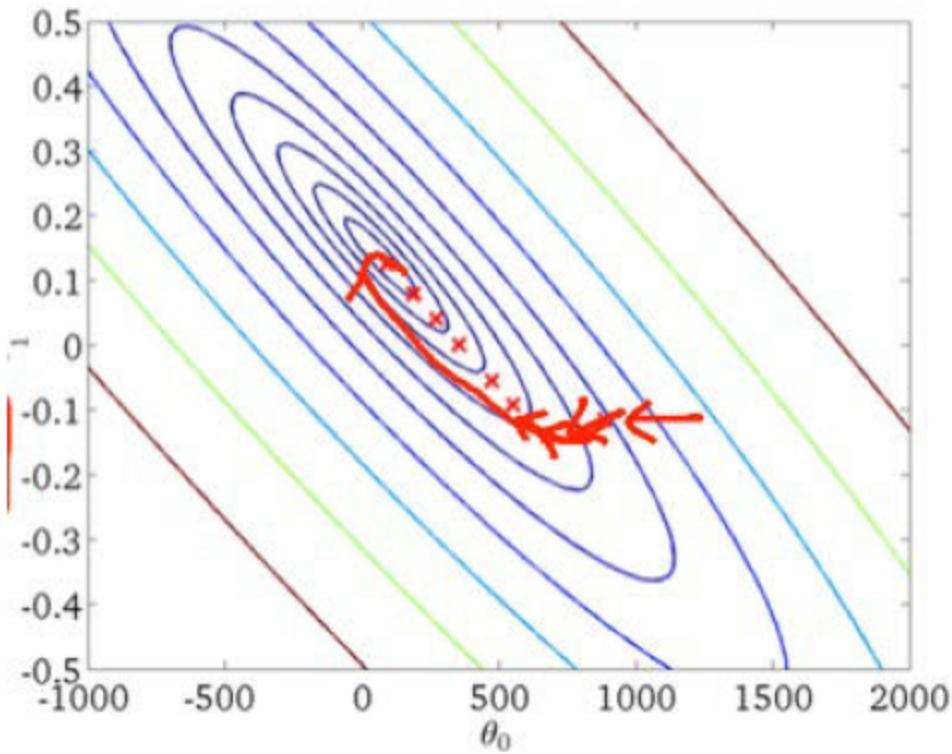
on itère sur les échantillons un par un :

pour i allant de 1 à m, répéter :

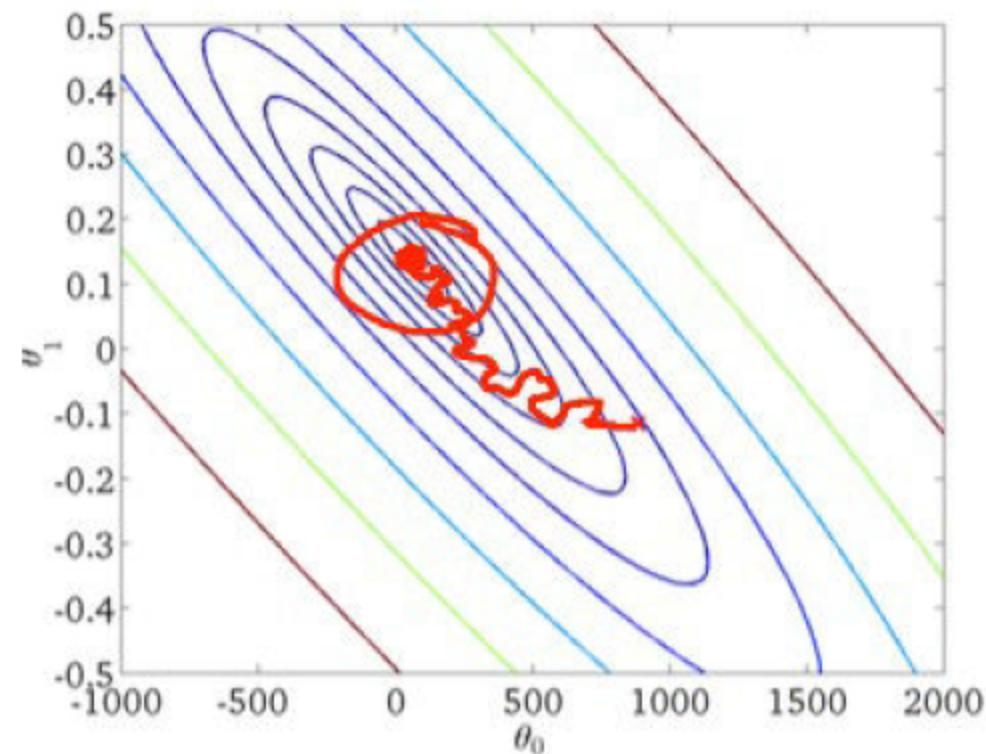
$$\left\{ \begin{array}{l} w_0 := w_0 - \alpha \cdot \frac{1}{m} x_i \cdot (h(x_i) - y_i) \\ w_1 := w_1 - \alpha \cdot \frac{1}{m} (h(x_i) - y_i) \end{array} \right\}$$

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

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'epochs 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, nombre de neurones par couche...
- Le learning rate
- La taille des mini-batches

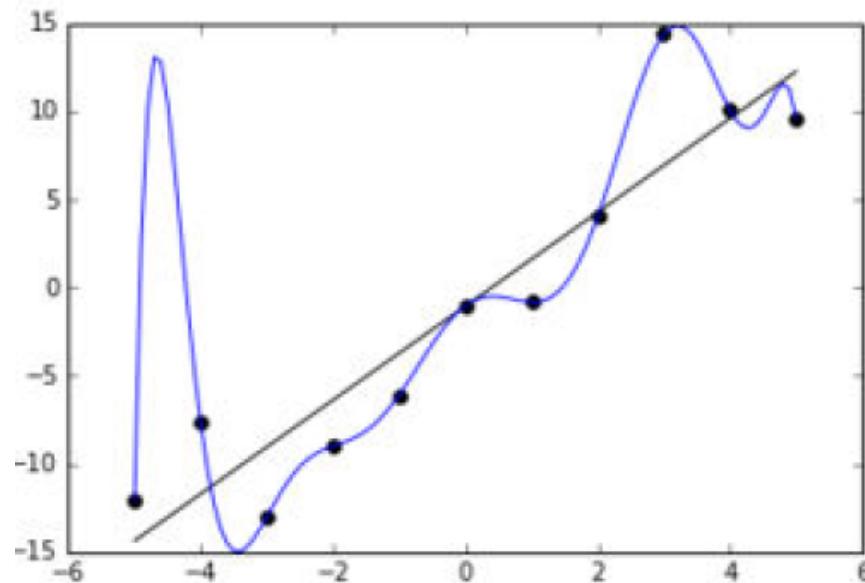
■ Comment choisir ces hyper-paramètres ?

Evaluer le modèle

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

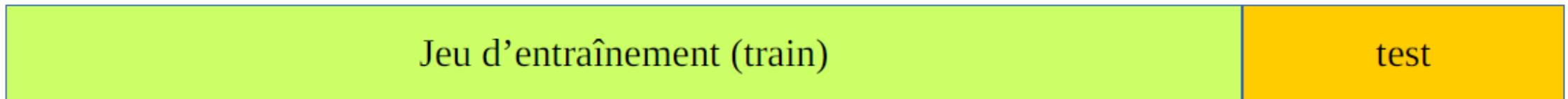
Jeu de données d'entraînement

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



Evaluer le modèle

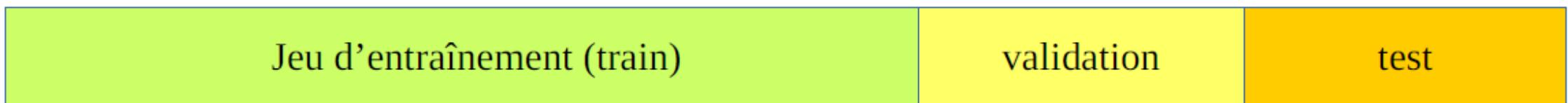
Deuxième idée : choisir les hyper-paramètres qui fonctionnent le mieux sur un jeu de test



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

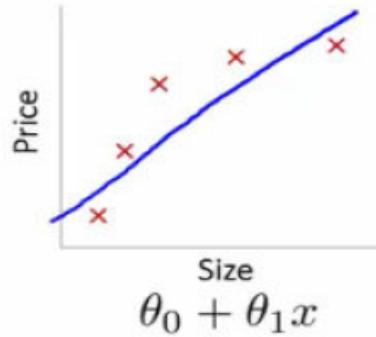
Evaluer le modèle

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.

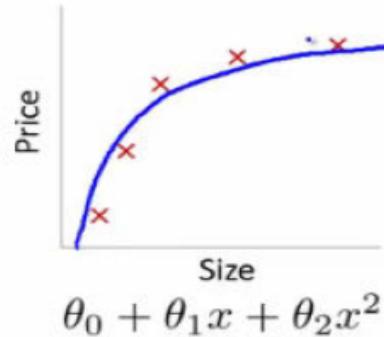


C'est mieux !

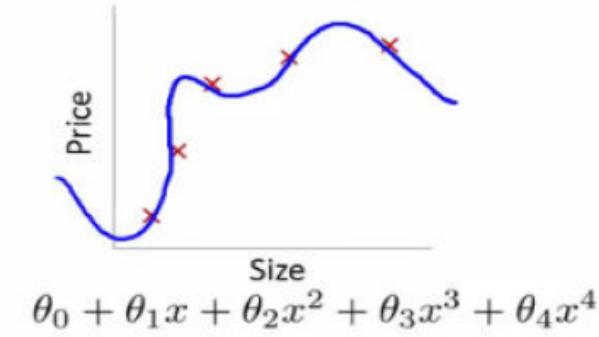
Sous-apprentissage - Sur-apprentissage



High bias
(underfit)



"Just right"



High variance
(overfit)

Sous-apprentissage

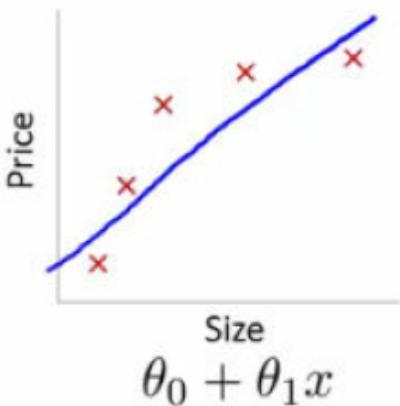
modèle trop simple pour expliquer
la variance

Bon modèle

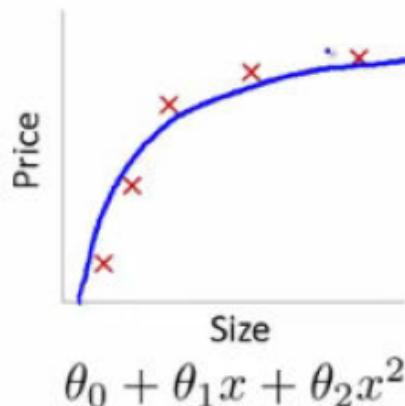
Sur-apprentissage

modèle qui colle trop au bruit du jeu
de données

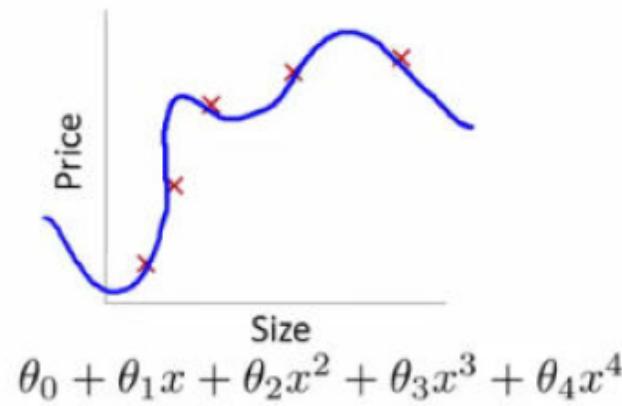
Sous-apprentissage - Sur-apprentissage



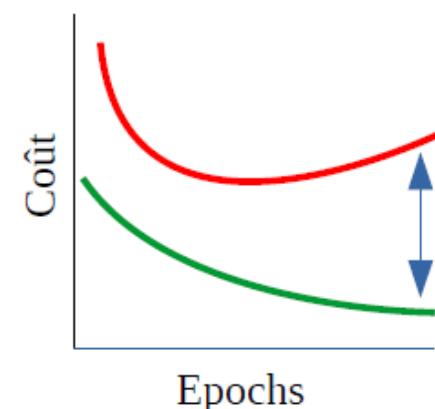
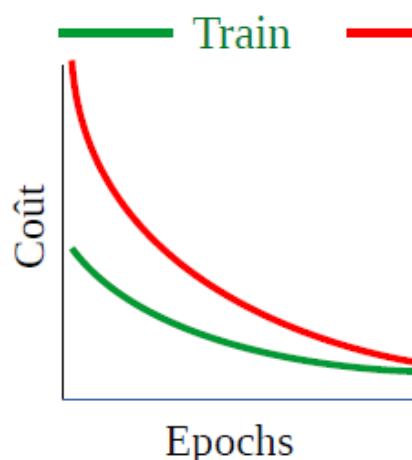
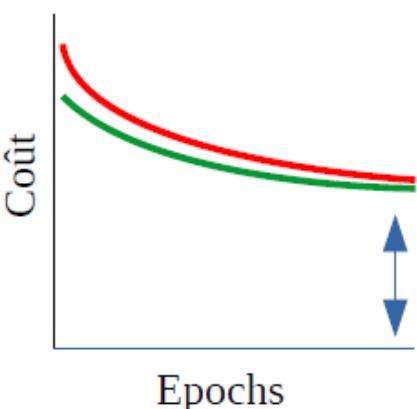
High bias
(underfit)



"Just right"



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

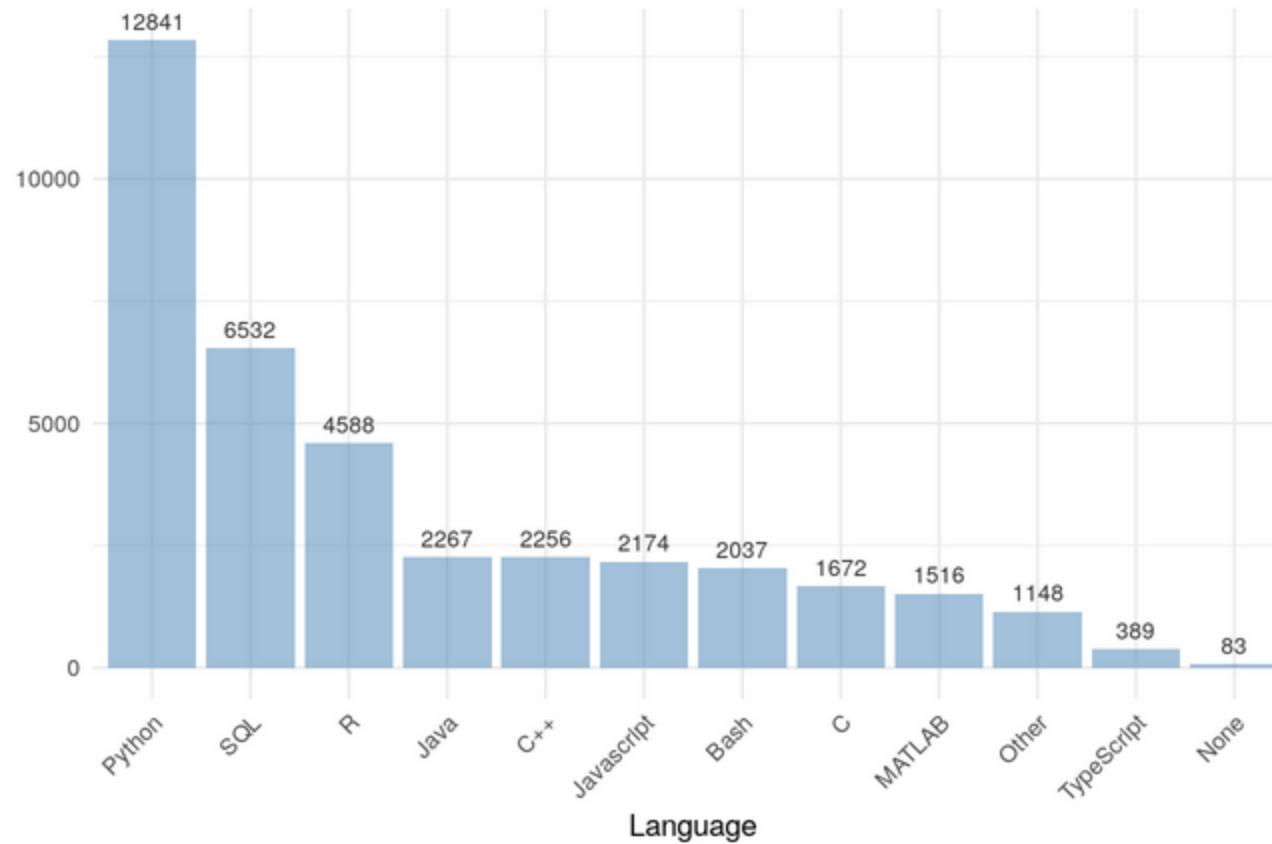
- Ajouter des données d'entraînement
 - Deux exemples de maisons ne permettent pas de créer un modèle qui généralise bien
- Simplifier le modèle ou retirer des prédicteurs
 - Eviter que le modèle parvienne à « apprendre par cœur » le jeu d'entraînement
- Entraîner le modèle moins longtemps
 - En français, l'overfitting se dit « surapprentissage ».
- Limiter la capacité d'apprentissage du modèle
 - Il existe plusieurs méthodes dont la régularisation et le dropout.
- Utiliser des ensembles
 - Comme en météo, entraîner plusieurs modèles et combiner leurs prédictions.

Des questions?



Python pour le Machine Learning

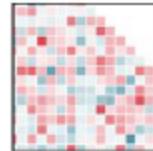
Part de marché chez les data scientists (2019)



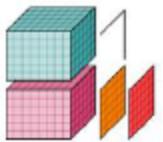
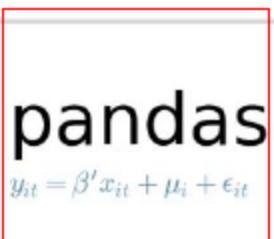
Librairies Pythons



StatsModels
Statistics in Python



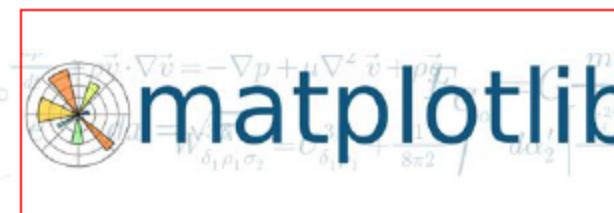
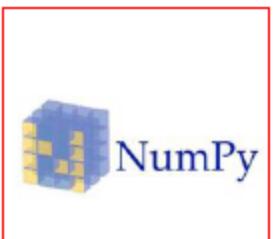
Seaborn



xarray



scikit-image
image processing in python



IP[y]:
IPython

Lire des fichiers Excel : utilisation de la Librairie Pandas

Excel = Pandas !

variable à prédire

Colonnes

E	F	G	H	I	J	K	L	M	N	O	P	Q		
1	Total Contract			From Inception to June 13, 2016			At June 13, 2016			For the Period Ended June 13, 2016				
2	Estimated Revenue	Estimated Costs	Estimated Gross Profit	Earned Contract Revenue	Contract Costs	Gross Profit	Contract Billings	Estimated Costs to Complete	Percent Complete	Under (Over) Billings	Earned Contract Revenue	Contract Costs	Gross Profit (Loss)	
3	29,831,162	22,771,956	7,059,306	12,113,470	9,246,924	2,866,546	11,987,630	13,525,092	41%	125,840	3,440,588	2,855,269	885,319	
4	4,765,875	3,915,859	850,016	4,761,592	3,912,340	849,252	4,748,777	3,519	100%	12,815	319,663	185,925	133,738	
5	3,165,949	2,635,676	530,273	3,073,180	2,558,445	514,735	3,092,332	77,231	97%	(19,152)	1,212,380	1,019,868	192,512	
6	6,845,696	5,348,200	1,497,496	5,935,890	4,637,414	1,298,476	5,727,306	710,786	87%	208,584	2,985,189	2,344,782	640,407	
7	3,202,917	2,139,767	1,063,150	3,197,769	2,136,328	1,061,441	3,199,414	3,439	100%	(1,645)	386,839	241,974	144,865	
8	3,267,627	2,402,206	865,421	3,122,086	2,295,211	826,875	3,143,402	106,995	96%	(21,316)	254,751	101,060	153,691	
9	3,513,815	2,260,925	1,252,890	2,839,759	1,827,211	1,012,548	2,573,819	433,714	81%	265,940	1,823,265	1,173,159	650,106	
10	3,913,079	3,104,573	808,506	3,591,755	2,849,640	742,115	3,503,374	254,933	92%	88,381	2,651,445	2,039,028	612,417	
11	12,187,491	13,500,000	(1,312,509)	2,193,165	3,505,674	(1,312,509)	2,476,537	9,994,326	26%	(283,372)	2,193,165	3,505,674	(1,312,509)	
12	3,274,077	2,798,357	475,720	35,779	30,580	5,199	0	2,767,777	1%	35,779	35,779	30,580	5,199	
13	3,835,139	4,296,527	(461,388)	2,578,713	3,040,101	(461,388)	2,386,461	1,256,426	71%	192,252	2,578,713	3,040,101	(461,388)	
14	13,500,000	10,227,273	3,272,727	8,553,041	6,479,577	2,073,464	8,321,142	3,747,696	63%	231,899	8,553,041	6,479,577	2,073,464	
15	3,849,262	3,137,190	712,072	274,615	223,814	50,801	1,741,936	2,913,376	7%	(1,467,321)	274,615	223,814	50,801	
16	74,614,943	64,402,779	10,212,164	46,921,464	41,803,708	5,117,756	43,715,328	22,599,071			29,854,173	27,271,295	2,582,878	
17	169,671,132	142,941,288	26,825,844	99,192,278	84,546,967	14,645,511	96,614,458	58,394,321			(651,516)	36,863,606	30,512,106	6,351,500
18														
19														

observations (X, Y)

Importer un fichier Excel avec Pandas

```
import pandas  
movies=pandas.read_excel("movies.xls")  
movies.head(4)
```

	Title	Year	Genres	Language	Country	Content Rating	Duration	Aspect Ratio	Budget	Gross Earnings	...	Facebook Likes - Actor 1	Facebook Likes - Actor 2	Facebook Likes - Actor 3	Facebook Likes - cast Total	Facebook likes - Movie	Facenumber in posters
0	Intolerance: Love's Struggle Throughout the Ages	1916	Drama History War	NaN	USA	Not Rated	123	1.33	385907.0	NaN	...	436	22	9.0	481	691	1
1	Over the Hill to the Poorhouse	1920	Crime Drama	NaN	USA	NaN	110	1.33	100000.0	3000000.0	...	2	2	0.0	4	0	1
2	The Big Parade	1925	Drama Romance War	NaN	USA	Not Rated	151	1.33	245000.0	NaN	...	81	12	6.0	108	226	0
3	Metropolis	1927	Drama Sci-Fi	German	Germany	Not Rated	145	1.33	6000000.0	26435.0	...	136	23	18.0	203	12000	1

Importer un fichier csv avec Pandas

```
import pandas  
housing=pandas.read_csv("housing.csv",sep=',')  
housing.head(4)
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY

```
Y=housing["median_income"] # la variable à prédire  
X=housing.drop("median_income",axis=1) # nos prédicteurs
```

- Préparation du jeu de données (X,Y) pour le modèle

Utilisation de Pandas pour vérifier les données

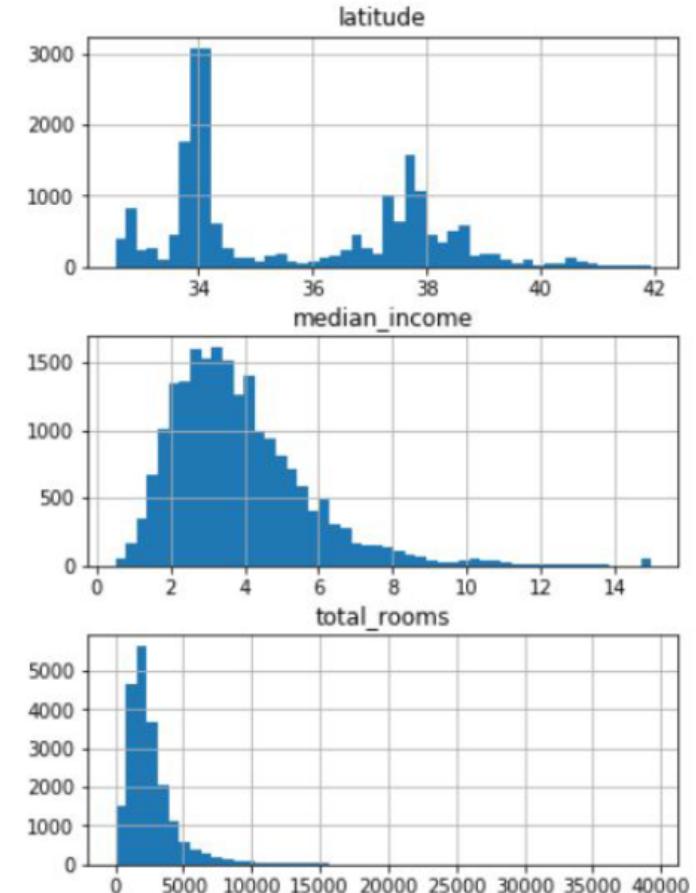
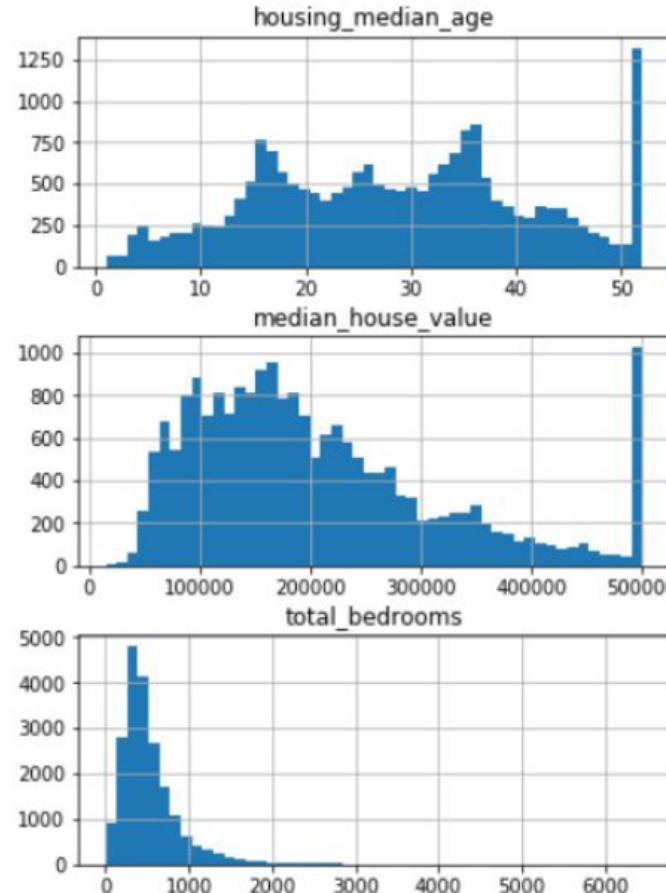
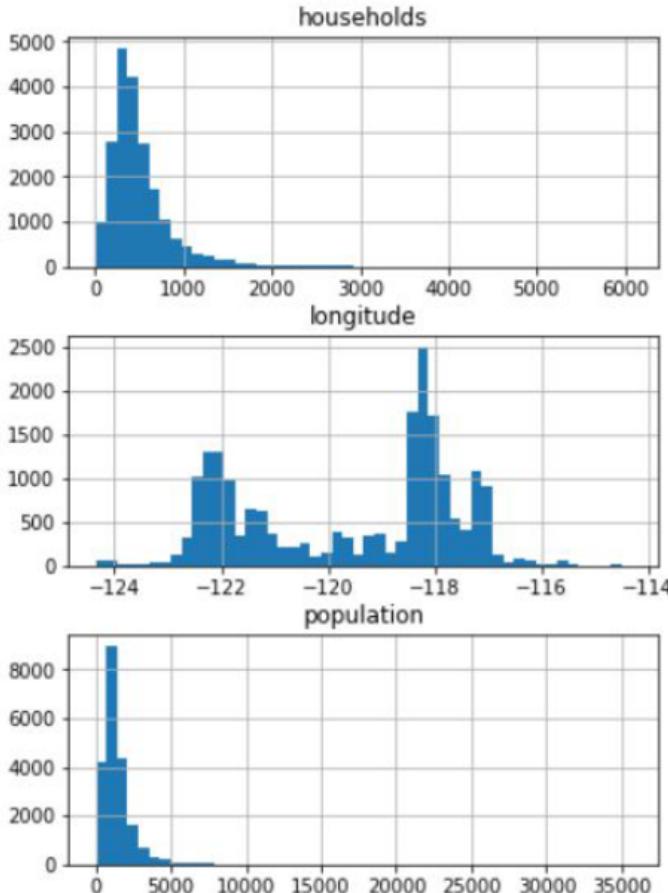
- Les données contiennent-elles des valeurs aberrantes ? (ex : une température de 6000 degrés)
- Y a-t-il des valeurs manquantes ?

Exemple de code sur le dataframe housing:

housing.describe()									
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

Utilisation de Pandas pour vérifier les données

```
housing.hist()
```



Présentation de Sklearn

Classification

Identifier à quelle catégorie appartient un objet.

Applications : détection de spam, reconnaissance d'image.

Algorithmes : SVM , voisins les plus proches , forêt aléatoire , ... Exemples

Régression

Prédire un attribut à valeur continue associé à un objet.

Applications : réponse aux médicaments, prix des actions.

Algorithmes : SVR , régression de crête , Lasso , ... Exemples

Clustering

Regroupement automatique d'objets similaires en ensembles.

Applications : segmentation des clients, résultats des expériences de regroupement

Algorithmes : k-Moyens , clustering spectral , moyenne-shift , ... Exemples

Réduction de la dimension

Réduire le nombre de variables aléatoires à considérer.

Applications : Visualisation, Efficacité accrue

Algorithmes : ACP , sélection de caractéristiques , factorisation matricielle non négative . Exemples

Sélection du modèle

Comparer, valider et choisir des paramètres et des modèles.

Objectif : Amélioration de la précision via l'optimisation des paramètres

Modules : recherche de grille , validation croisée , métriques . Exemples

Pré-traitement

Extraction de caractéristiques et normalisation.

Application : Transformation de données d'entrée telles que du texte à utiliser avec des algorithmes d'apprentissage automatique.

Modules : prétraitement , extraction de caractéristiques . Exemples

Séparer les données en jeux d'entraînement et de test

importer la fonction de séparation

```
from sklearn.model_selection import train_test_split  
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.2)
```

nouvelles variables

proportion pour jeux de test

Utilisation de Sklearn pour une régression

- Utilise la méthode de descente de gradient pour trouver les bons paramètres ● Large collection de modèles statistiques disponibles (random forest, réseaux de neurones..)

Exemple de code :

```
import sklearn.linear_model # Importer librairie
model=sklearn.linear_model.LinearRegression() # Création du modèle
model.fit(X,Y) #Entrainement du modèle
predictions=model.predict(X) # Prédiction du modèle
print(predictions)
```

Utilisation de Sklearn pour une classification

- Prédiction d'une classe et des probabilités pour chaque classe

```
import sklearn.linear_model # Importer librairie
model=sklearn.linear_model.LogisticRegression() # Création du modèle
model.fit(X,Y) #Entrainement du modèle
predictions=model.predict(X) # Prédiction du modèle
print(predictions)

['<1H OCEAN' '<1H OCEAN' 'NEAR BAY' ... 'INLAND' 'INLAND' 'INLAND']
```

```
predictions_proba=model.predict_proba(X)
print(predictions_proba[0])
```

```
[4.80454788e-01 1.37050843e-03 1.31800779e-04 2.93803587e-01
 2.24239315e-01]
```

renvoie un vecteur de probabilités pour chaque observation

Un projet Machine Learning en Python

● Les grandes étapes et les librairies associées



1. Lecture des données
Statistiques descriptives



2. Prétraitement des données



3. Modélisation



4. Évaluation



5. Présentation, mise en forme des résultats, export

