

Machine Learning & Intelligence Artificielle



 cpc
Manuel Simoes
manuel.simoes@cpc-analytics.fr

- Régression linéaire -

&

Méthode des moindres carrés

*Apprentissage Supervisé
Y quantifiable / nombre*



Prédire le prix des maisons en fonction de leur superficie

Les données brutes

PRIX DES MAISONS – PORTLAND, OREGON (USA)	
x Surface (ft ²)	y Prix (en millier de \$)
2104	460
1416	232
1534	315
852	NaN
...	...

NOTATIONS

m = nombre d'événements

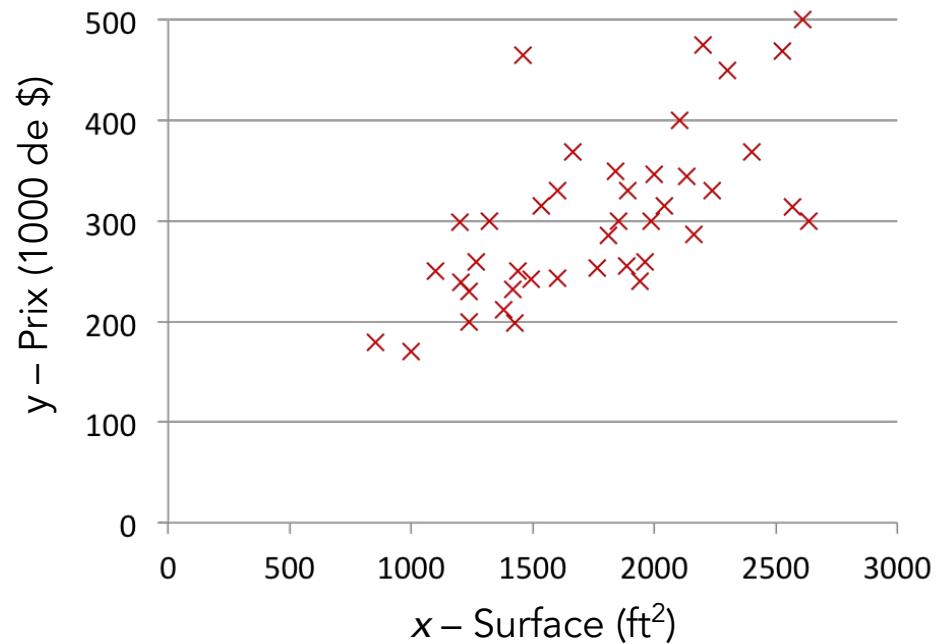
x = « input » variable / features

y = « output » variable / « target » variable

Le but pour vous est de pouvoir estimer le prix des maisons en fonction de leur surface. Pour cela vous avez un jeu de données de quelques maisons d'un quartier.



Prédire le prix des maisons en fonction de leur superficie



Visualiser les données !!!

Autant que possible il faut visualiser vos données et ne pas se contenter d'une analyse statistique.

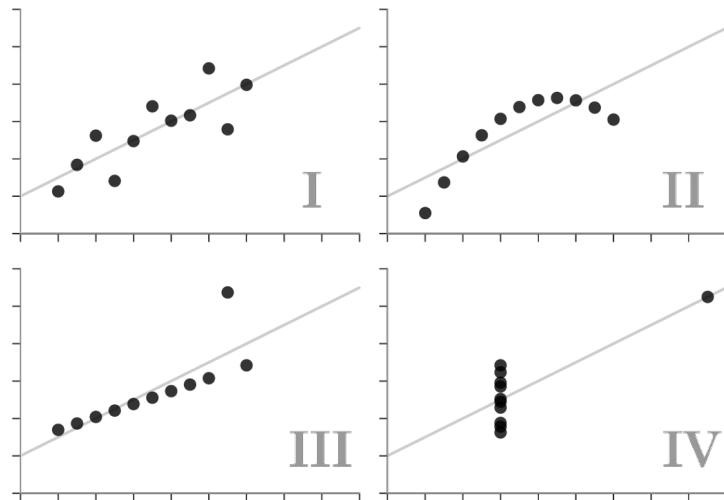


Propriétés Statistiques v.s. Visualiser vos Données

Le **quartet d'Anscombe** est constitué de quatre [ensembles de données](#) qui ont les mêmes propriétés [statistiques](#) simples mais qui sont en réalité très différents, ce qui se voit facilement lorsqu'on les représente sous forme de [graphiques](#). Ils ont été construits en 1973 par le statisticien [Francis Anscombe](#) dans le but de démontrer l'importance de tracer des graphiques avant d'analyser des données, car cela permet notamment d'estimer l'incidence des [données aberrantes](#) sur les différents [indices statistiques](#) que l'on pourrait calculer.

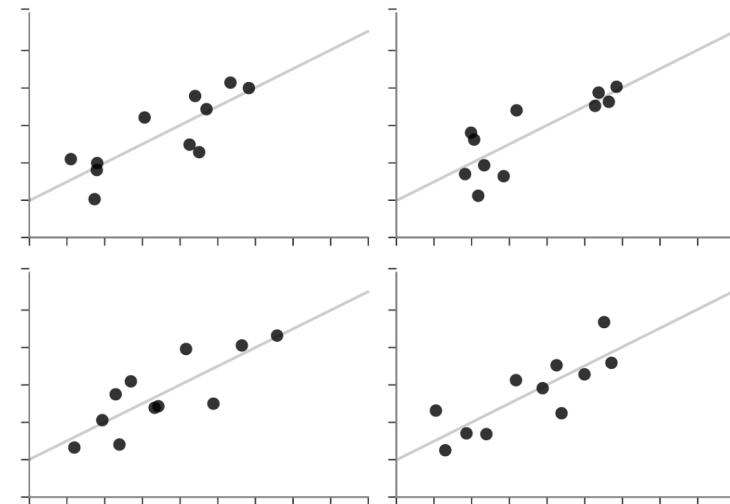
✓ Anscombe's Quartet

Each dataset has the same summary statistics (mean, standard deviation, correlation), and the datasets are *clearly different*, and *visually distinct*.



✗ Unstructured Quartet

Each dataset here also has the same summary statistics. However, they are not *clearly different* or *visually distinct*.



Une des raisons pour lequel il faut, autant que possible, visualiser les données

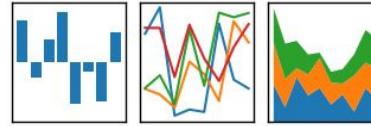


Visualiser vos Données

**Rappel des bibliothèques
que vous pouvez utiliser
pour les représentations
graphiques.**

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



<https://matplotlib.org/>

matplotlib
ou PYPLOT

<https://matplotlib.org/>



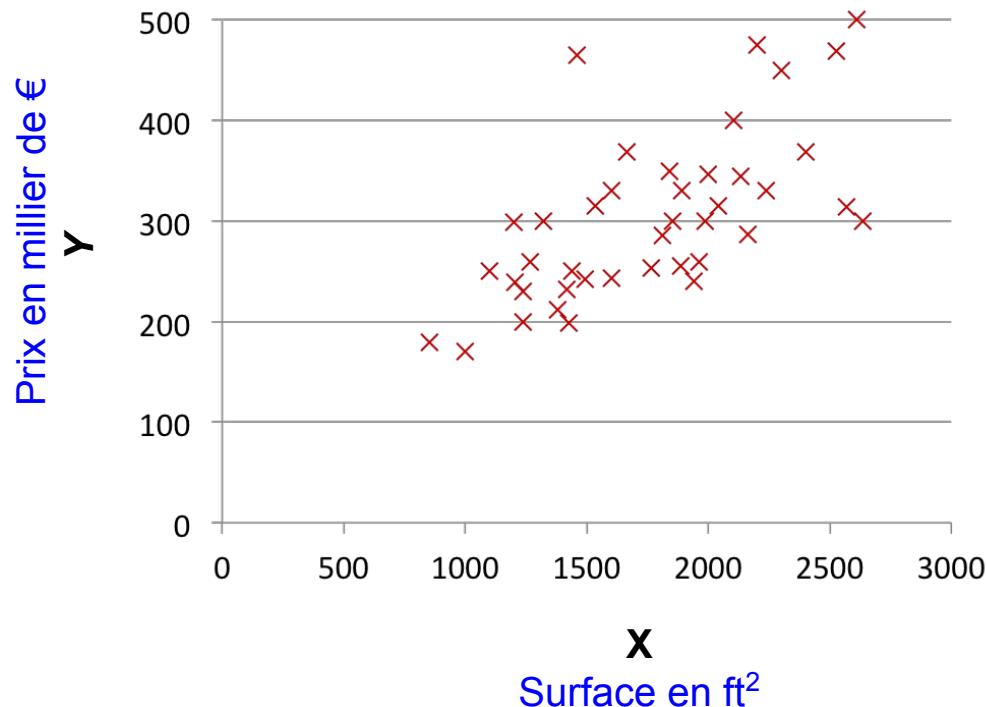
seaborn

<https://seaborn.pydata.org/>



Prédire le prix des maisons en fonction de leur superficie

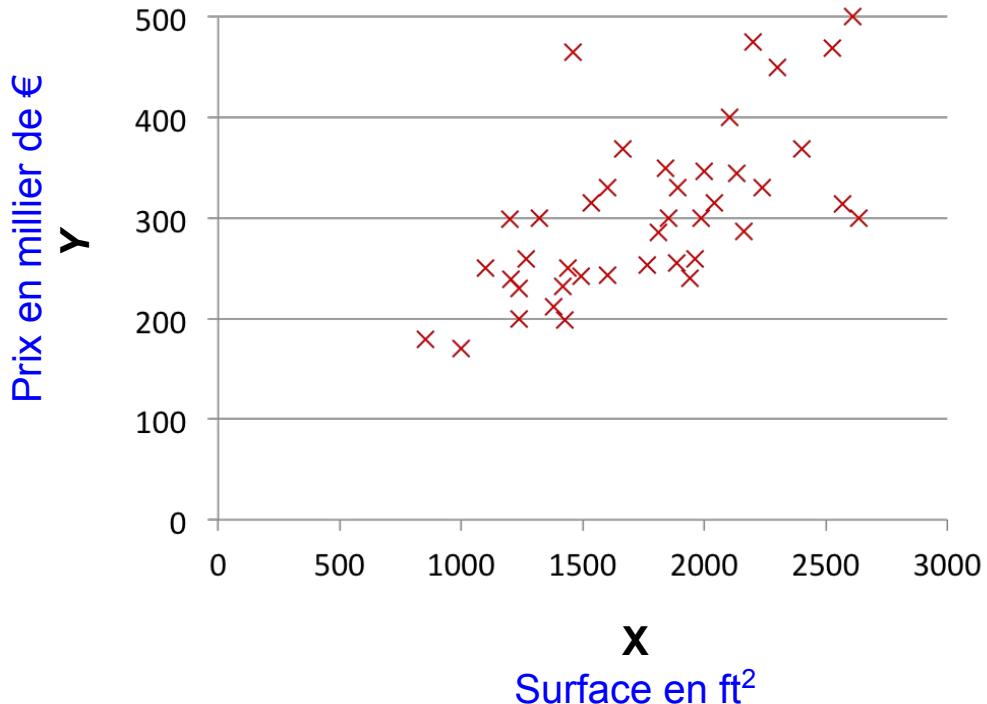
Choisir un modèle mathématique



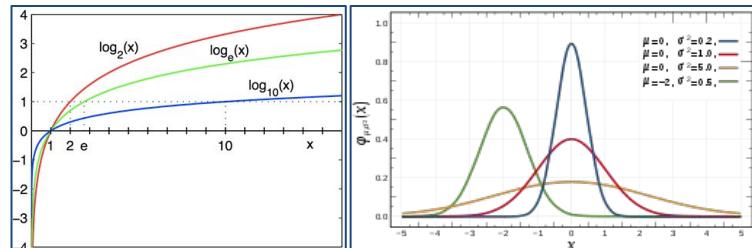
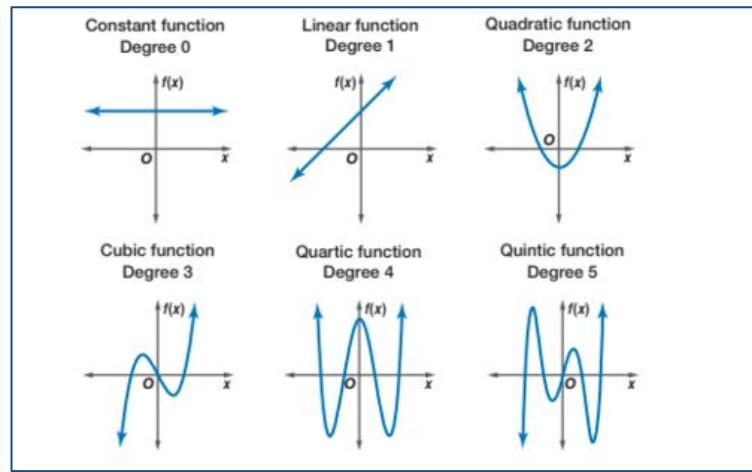
Quel type de courbe mathématique peut au mieux décrire nos données ?



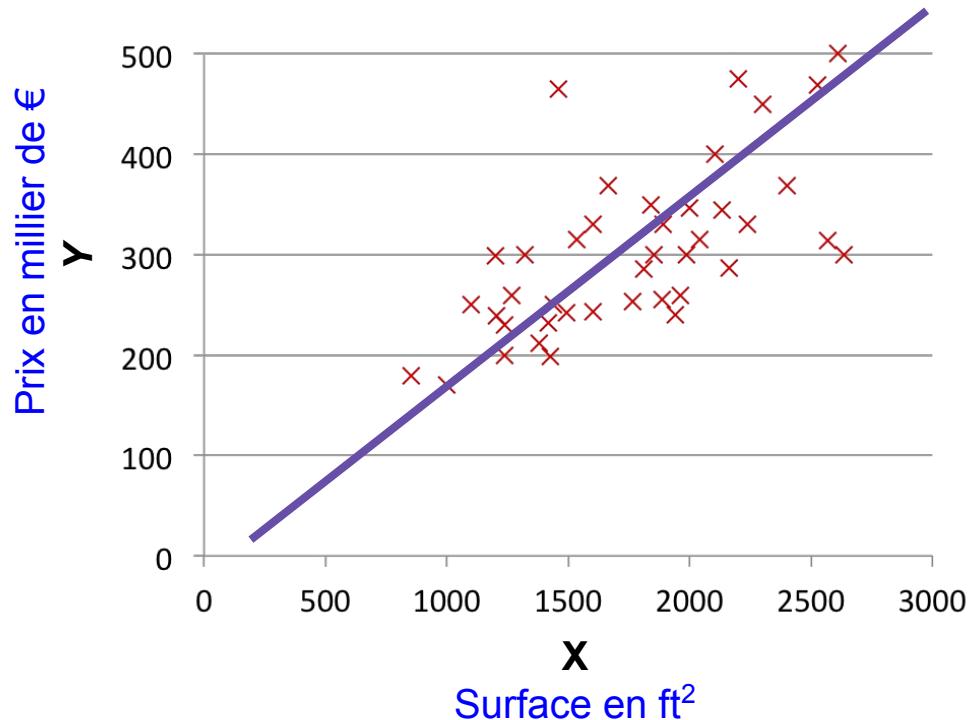
Prédire le prix des maisons en fonction de leur superficie



Choisir un modèle mathématique



Prédire le prix des maisons en fonction de leur superficie



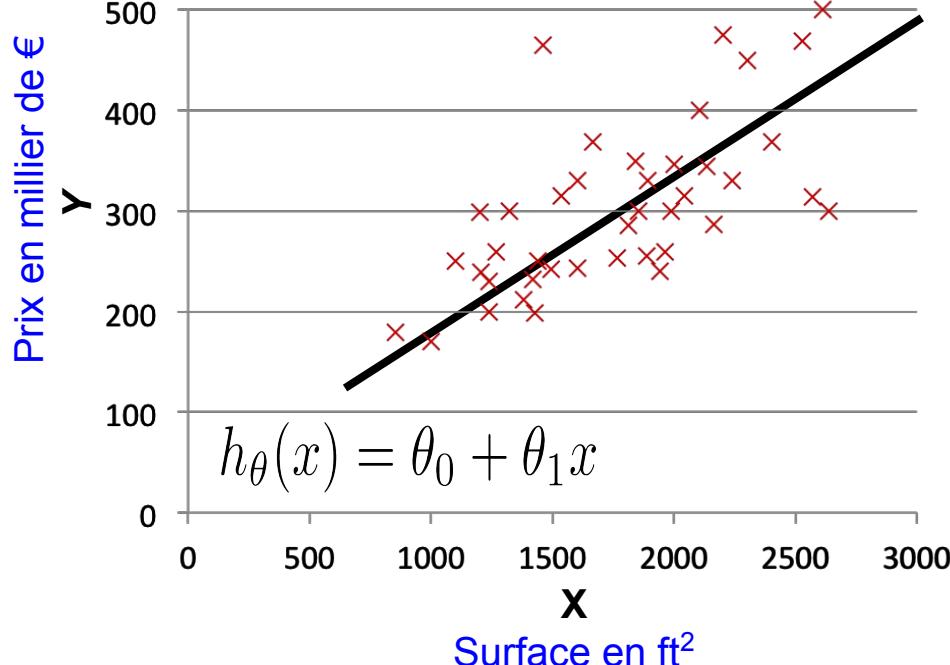
Choisir un modèle mathématique

Nous choisissons : la droite !!

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



Prédire le prix des maisons en fonction de leur superficie



Choisir un modèle mathématique

$$h_{\theta}(x) = \boxed{\theta_0} + \boxed{\theta_1} x$$

Paramètres

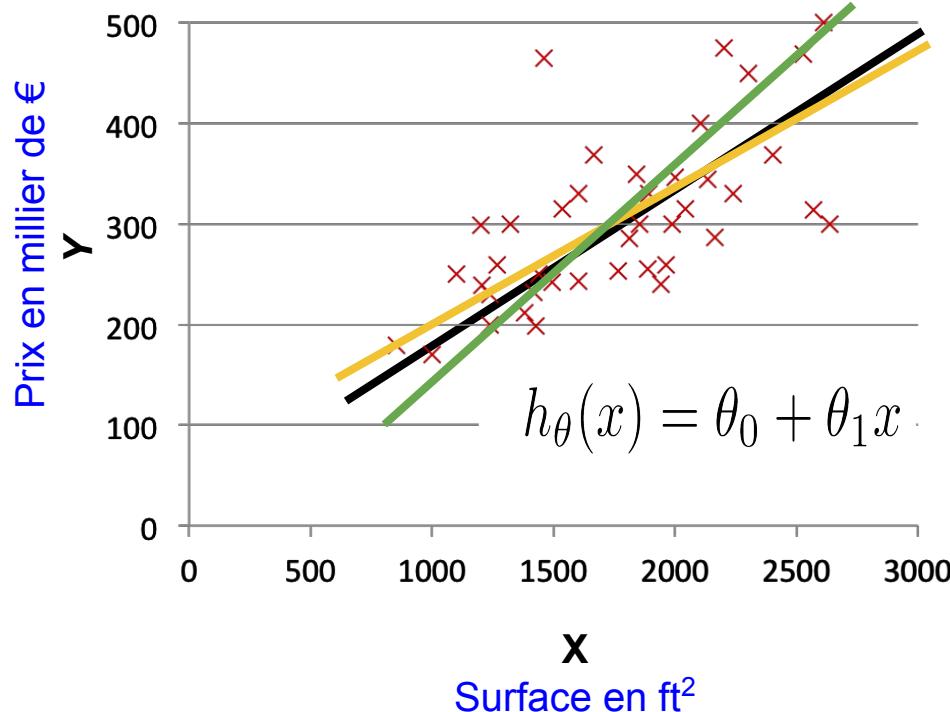
Variable / Feature

Notre modèle est paramétrique :

- une droite, 2 paramètres à déterminer durant la phase d'apprentissage.



Prédire le prix des maisons en fonction de leur superficie



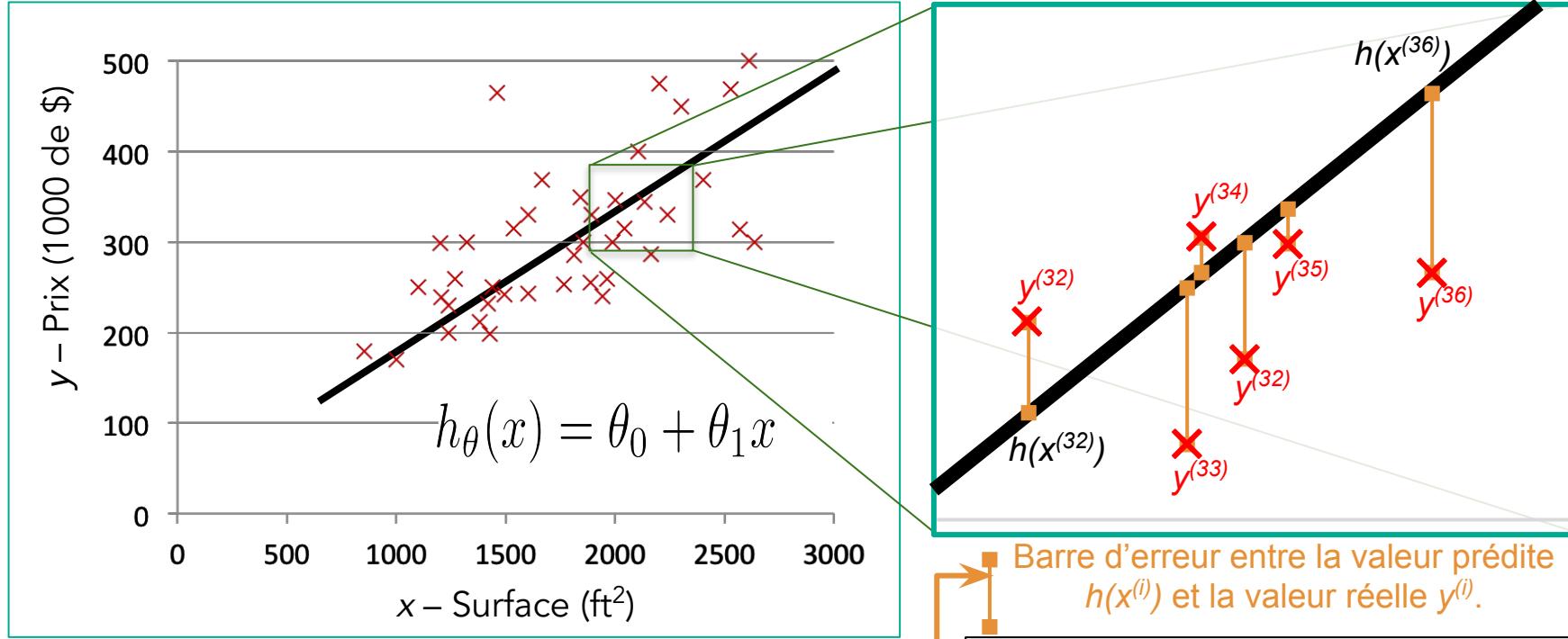
Comment ajuster les paramètres pour qu'ils définissent au mieux les données que la droite est censée représenter ?

$$h_{\theta}(x) = \boxed{\theta_0} + \boxed{\theta_1} x$$

Avoir une mesure de la qualité de notre modèle et pour chaque paramètres utilisé !



Méthode des moindres carrés : fonction coût



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

La fonction coût
Erreur quadratique moyenne entre le modèle et les données réelles.

Learning algorithm

Hypothèse du modèle mathématique :

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

Paramètres :

$$\theta_0, \theta_1$$

La fonction coût :

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

Objectif :

$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

C'est le modèle mathématique que nous avons choisi pour décrire les données Y . X et une variable est θ_1 , et θ_2 sont des paramètres.

La fonction coût décrit le comportement de l'erreur totale de notre modèle par rapport aux valeurs que l'on veut décrire. Ici θ_1 et θ_2 sont des variables et x et y sont des paramètres.

Notre objectif est de minimiser au mieux la fonction coût J pour obtenir un modèle mathématique qui décrit au mieux nos données. On va calculer la dérivé de $J(\theta_1, \theta_2) \dots$



Learning algorithm

La fonction coût à minimiser :

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

On sait :

- qu'une fonction atteint un minimum lorsque sa dérivé est nulle.
- que le signe d'une dérivée calculée en un point donne le signe de sa pente.

> 0 la fonction croît

< 0 la fonction décroît

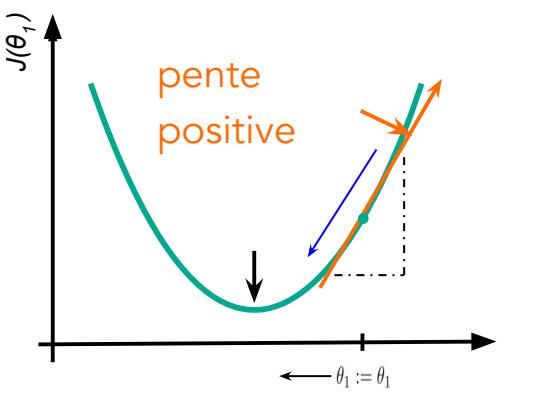


$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\ &= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h_\theta(x) - y) x_j\end{aligned}$$

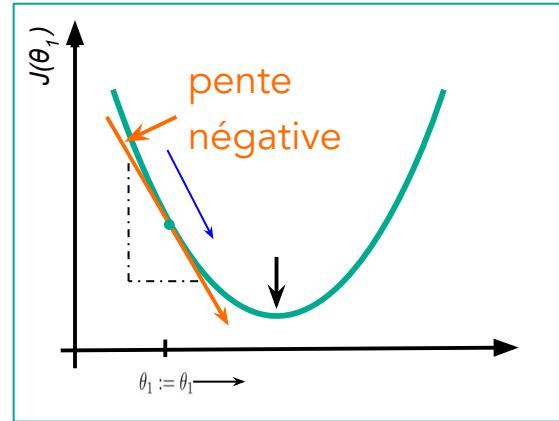


Learning algorithm

Pour comprendre, l'algorithme *Descent Gradient*, nous simplifions momentanément le problème en supprimant la constante de la droite (une fonction coût avec 1 variable).



$$-\alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$



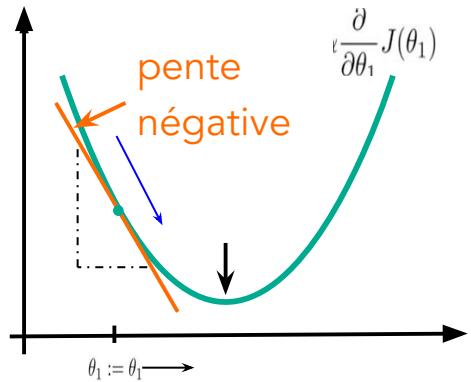
L'algorithme *Gradient Descent* va “explorer” la courbe de la fonction coût en utilisant sa dérivé

$$\frac{\partial}{\partial \theta_1} J(\theta_1)$$

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



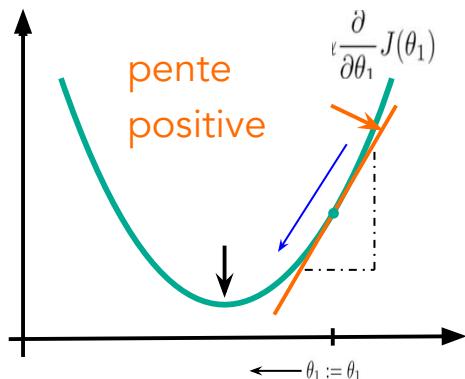
Gradient descent : Intuition



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

$$-\alpha \frac{\partial}{\partial \theta_1} J(\theta_1) > 0$$

On réduit Θ_1



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

$$-\alpha \frac{\partial}{\partial \theta_1} J(\theta_1) < 0$$

On augmente Θ_1

Quelque soit la position initiale de θ_1 , choisie, l'algorithme le déplace vers un minimum.



Gradient descent : régression

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}

GRADIENT DESCENT ALGORITHM

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

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

MODÈLE DE RÉGRESSION LINÉAIRE



$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\ &= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h_\theta(x) - y) x_j\end{aligned}$$



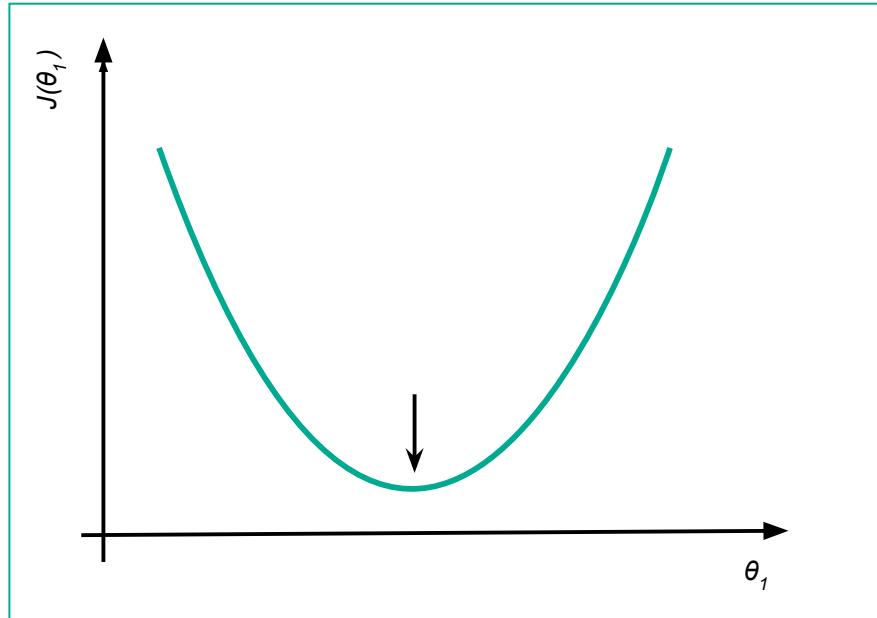
Learning algorithm

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

Pour comprendre, l'algorithme *Descent Gradient*, nous simplifions momentanément le problème en supprimant la constante de la droite (une fonction coût avec 1 variable).

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

L'objectif de l'algorithme est de trouver le minimum de cette courbe.



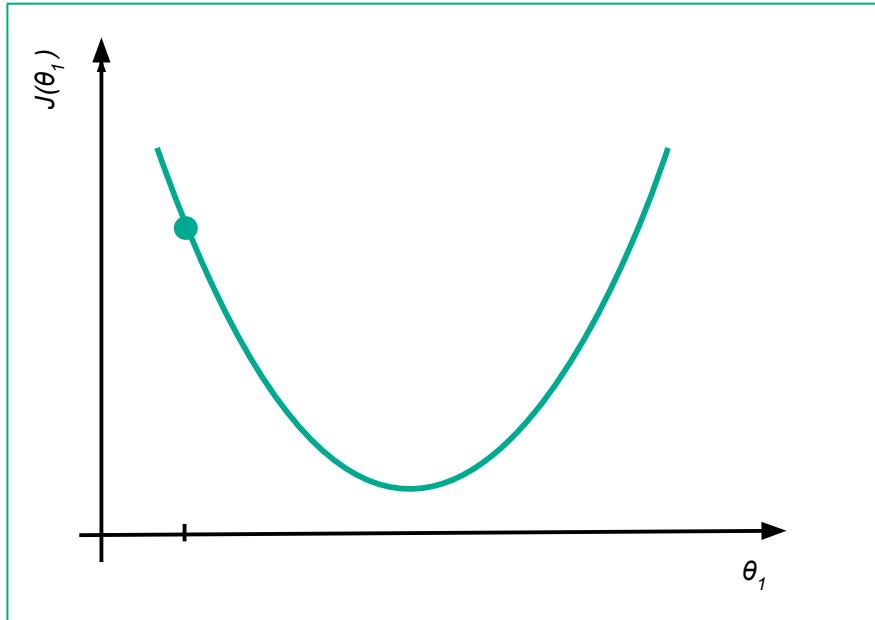
L'algorithme *Gradient Descent* va “explorer” la courbe de la dérivé de la fonction coût depuis une position initiale arbitraire.



Learning algorithm

1. On choisit une valeur de θ_1 au hasard.

L'objectif de l'algorithme est de trouver le minimum de cette courbe.



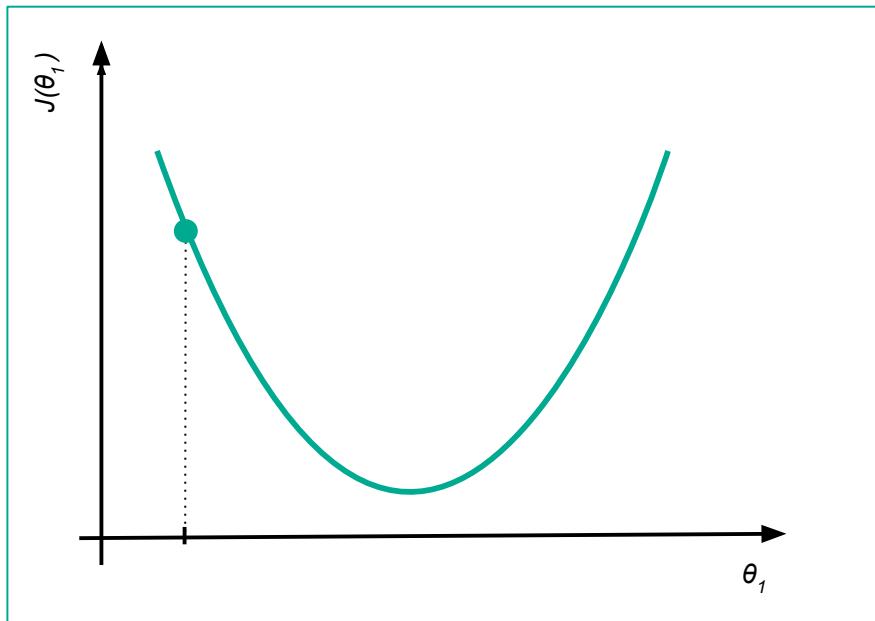
Learning algorithm

1. On choisit une valeur de θ_1 , au hasard.
2. Et on calcule la valeur de $J(\theta_1)$:

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

Nous connaissons tous les points m , $x^{(i)}$ et $y^{(i)}$, et donc $h_{\theta_1}(x^{(i)})$.

L'objectif de l'algorithme est de trouver le minimum de cette courbe.



Learning algorithm

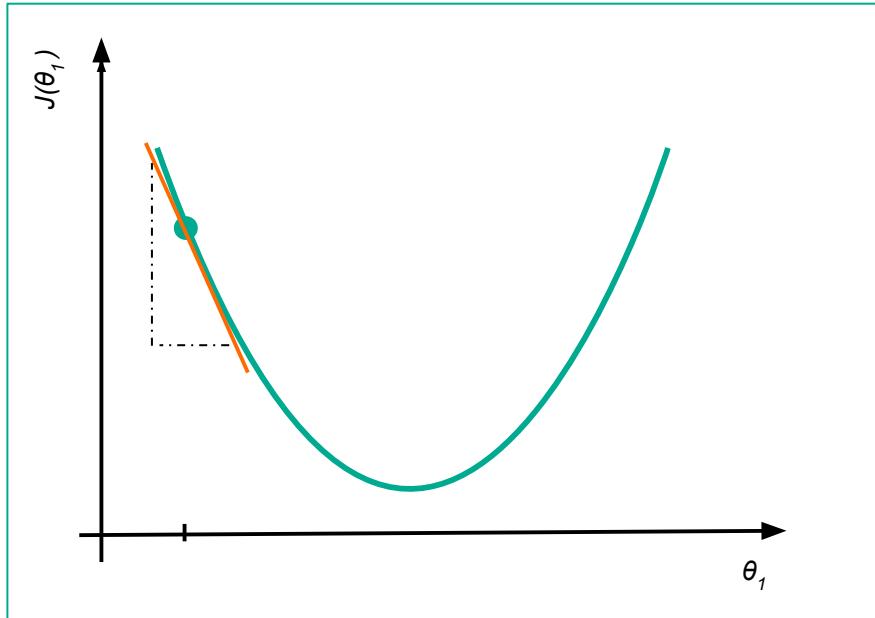
1. On choisit une valeur de θ_1 , au hasard.
2. Et on calcule la valeur de $J(\theta_1)$:

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Nous connaissons tous les points m , $x^{(i)}$ et $y^{(i)}$, et donc $h_{\theta_1}(x^{(i)})$.

3. On calcule la dérivé en θ_1 $\frac{\partial}{\partial \theta_1} J(\theta_1)$

L'objectif de l'algorithme est de trouver le minimum de cette courbe.



Learning algorithm

1. On choisit une valeur de θ_1 , au hasard.
2. Et on calcule la valeur de $J(\theta_1)$:

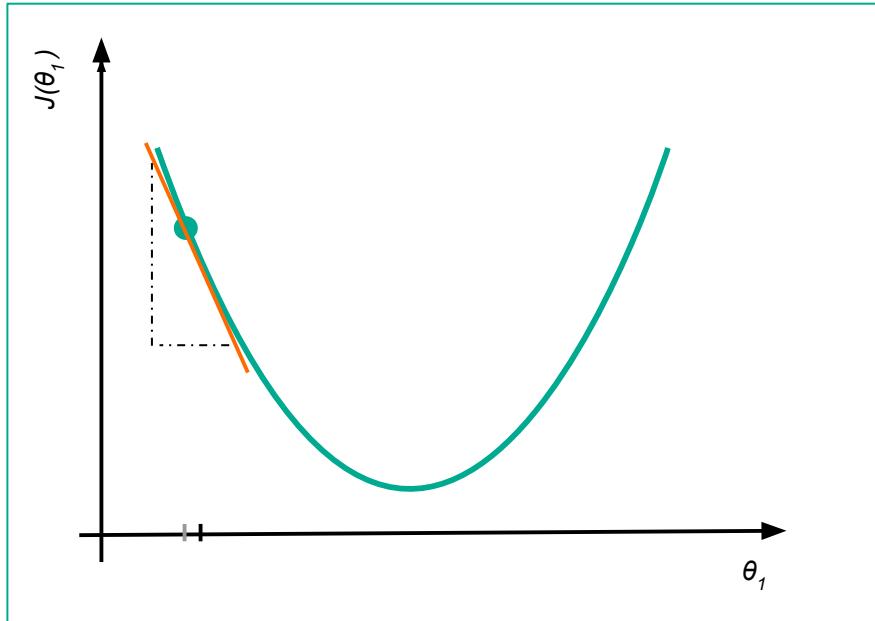
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Nous connaissons tous les points m , $x^{(i)}$ et $y^{(i)}$, et donc $h_{\theta_1}(x^{(i)})$.

3. On calcule la dérivé en θ_1 $\frac{\partial}{\partial \theta_1} J(\theta_1)$
4. On calcule la nouvelle position de θ_1

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

L'objectif de l'algorithme est de trouver le minimum de cette courbe.



Ici on introduit un hyperparamètre α : Learning Rate (taux d'apprentissage)



Learning algorithm

1. On choisit une valeur de θ_1 , au hasard.

2. Et on calcule la valeur de $J(\theta_1)$:

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

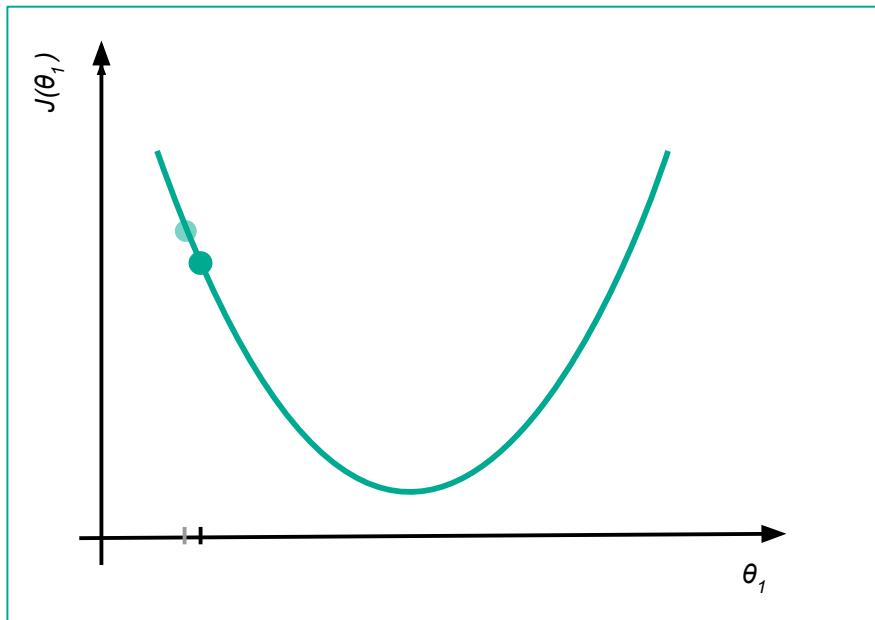
Nous connaissons tous les points m , $x^{(i)}$ et $y^{(i)}$, et donc $h_{\theta_1}(x^{(i)})$.

3. On calcule la dérivé en θ_1 $\frac{\partial}{\partial \theta_1} J(\theta_1)$

4. On calcule la nouvelle position de θ_1

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

L'objectif de l'algorithme est de trouver le minimum de cette courbe.



Learning algorithm

1. On choisit une valeur de θ_1 , au hasard.

2. Et on calcule la valeur de $J(\theta_1)$:

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Nous connaissons tous les points m , $x^{(i)}$ et $y^{(i)}$, et donc $h_{\theta_1}(x^{(i)})$.

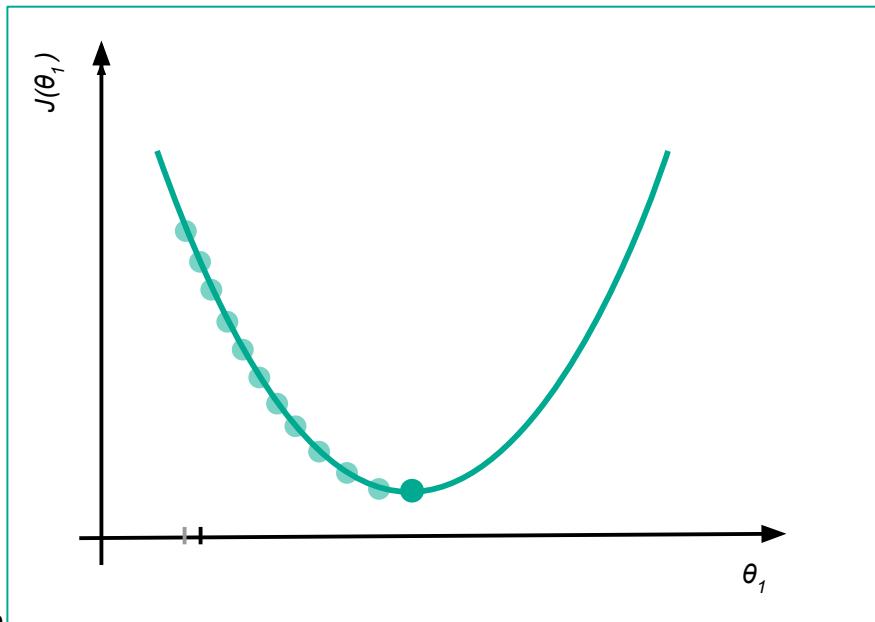
3. On calcule la dérivé en θ_1 $\frac{\partial}{\partial \theta_1} J(\theta_1)$

4. On calcule la nouvelle position de θ_1 ,

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

On recommence les itérations, jusqu'à que la différence entre la position de θ_1 et la nouvelle position θ_1 , soit inférieur à un seul de déplacement proche de zéro

L'objectif de l'algorithme est de trouver le minimum de cette courbe.



Learning algorithm

1. On choisit une valeur de θ_1 , au hasard.
2. Et on calcule la valeur de $J(\theta_1)$:

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

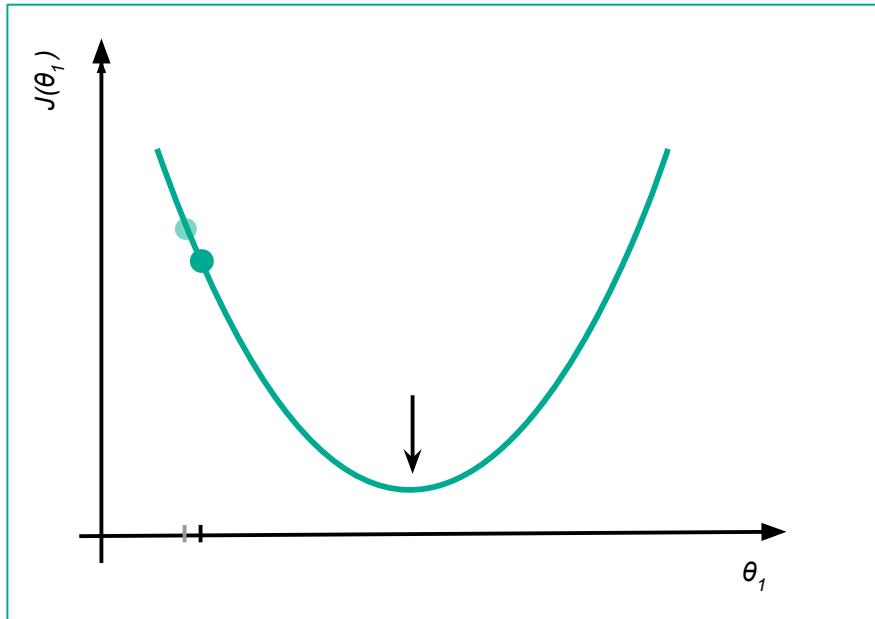
Nous connaissons tous les points m , $x^{(i)}$ et $y^{(i)}$, et donc $h_{\theta_1}(x^{(i)})$.

3. On calcule la dérivé en θ_1 $\frac{\partial}{\partial \theta_1} J(\theta_1)$
4. On calcule la nouvelle position de θ_1

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

5. Nous reprenons à l'étape 2., jusqu'à ce que la dérivé soit nulle (ou proche de nulle)

L'objectif de l'algorithme est de trouver le minimum de cette courbe.



Gradient descent : optimisation

MÉTHODE

Suivre la plus basse pente

PROCESSUS ITÉRATIF

- 1 – Choisir une position initiale $\theta^0 = (\theta_0, \theta_1, \theta_2, \dots, \theta_m)$
- 2 – Déterminer la pente à suivre $-J(\theta^t)$
- 3 – Choisir un « learning rate » α
- 4 – Mettre à jour les coordonnées $\theta^{t+1} = \theta^t - \alpha \cdot \nabla J(\theta^t)$
- 5 – Répéter depuis l'étape 2 jusqu'à un critère « stop » valide

CRITÈRES « STOP » CLASSIQUES

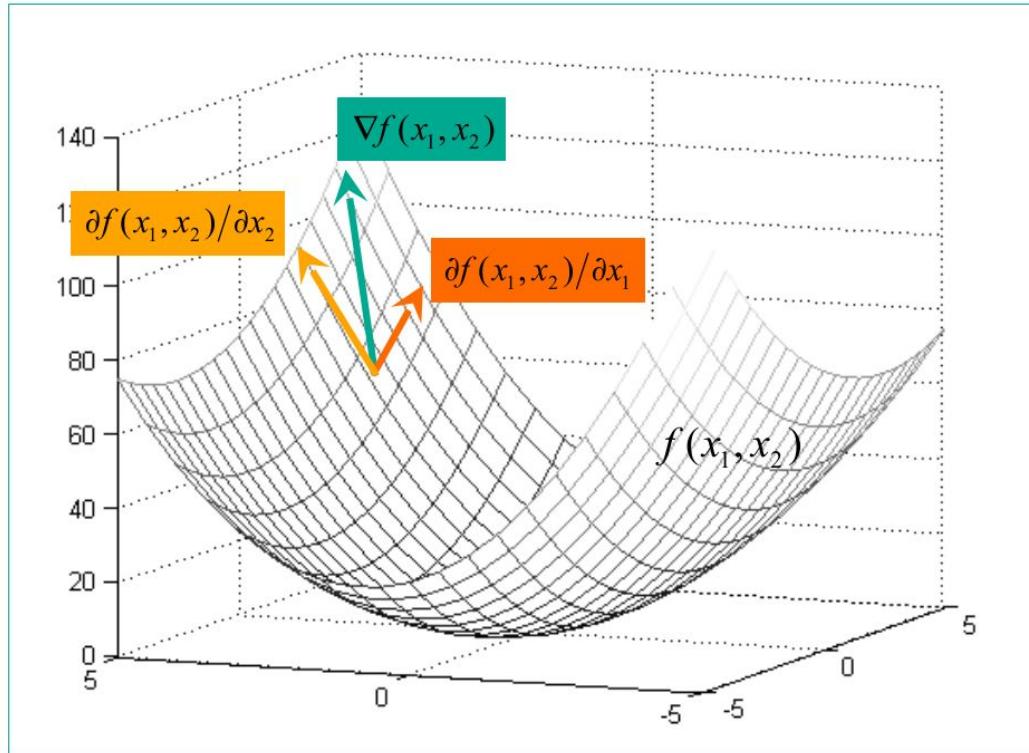
- $\nabla J(\theta^{t+1}) \sim 0$
- Nombre d'itérations maximum
- D'autres métriques ...



Rappel : vecteur gradient

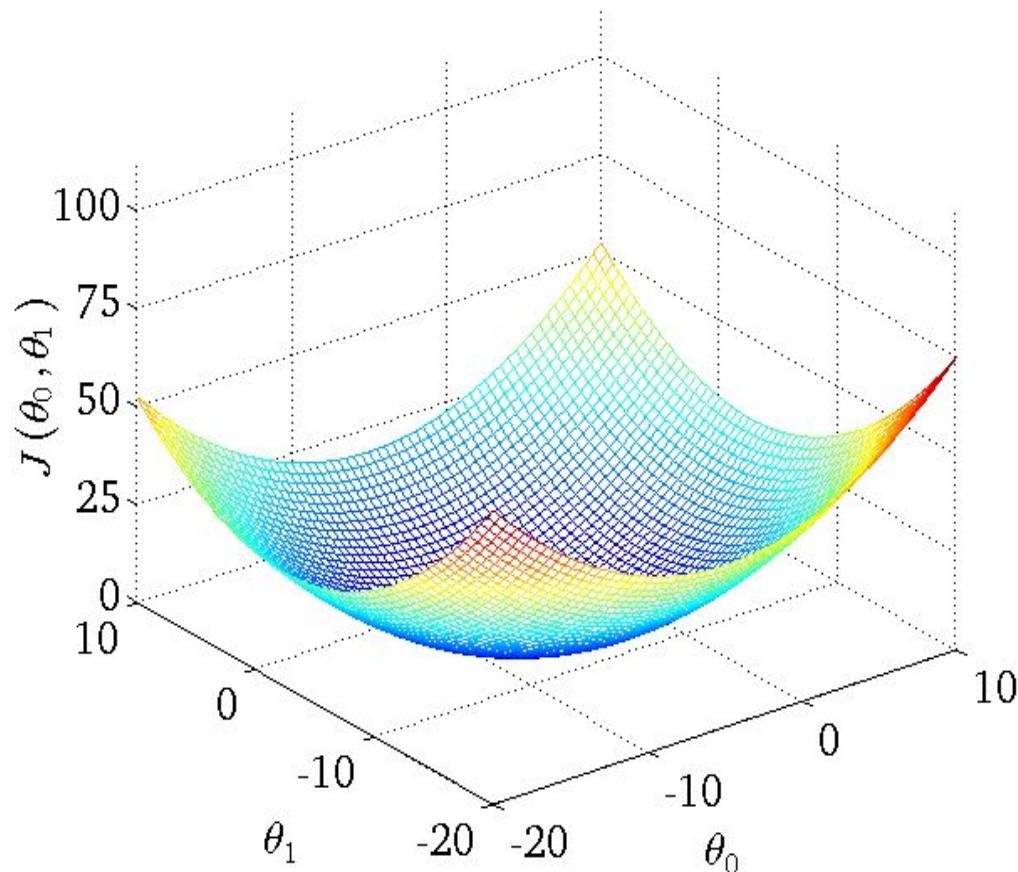
Ici, le vecteur gradient pointe vers le sens de plus forte pente (de la fonction)

Ici on reprend une fonction avec 2 variables.
On passe d'un gradient scalaire à un gradient vectoriel.



Gradient descent & régression

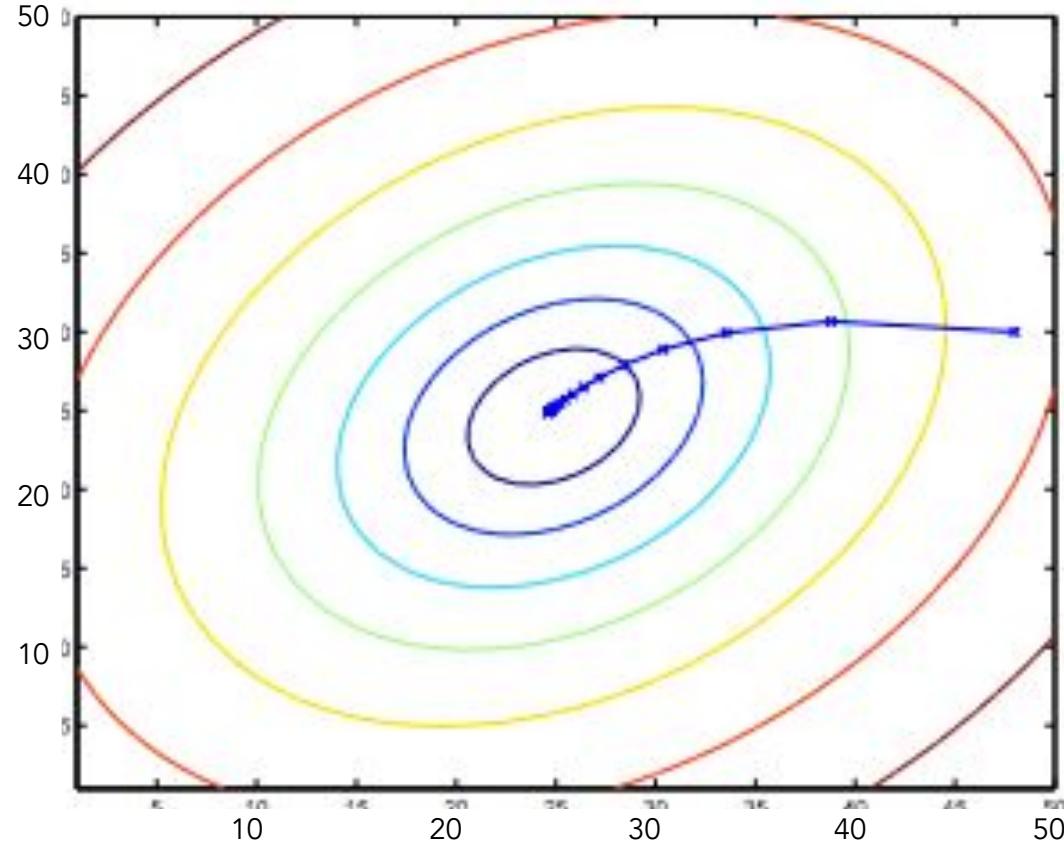
On reprend la fonction
coût avec les 2
variables θ_1 et θ_2 .



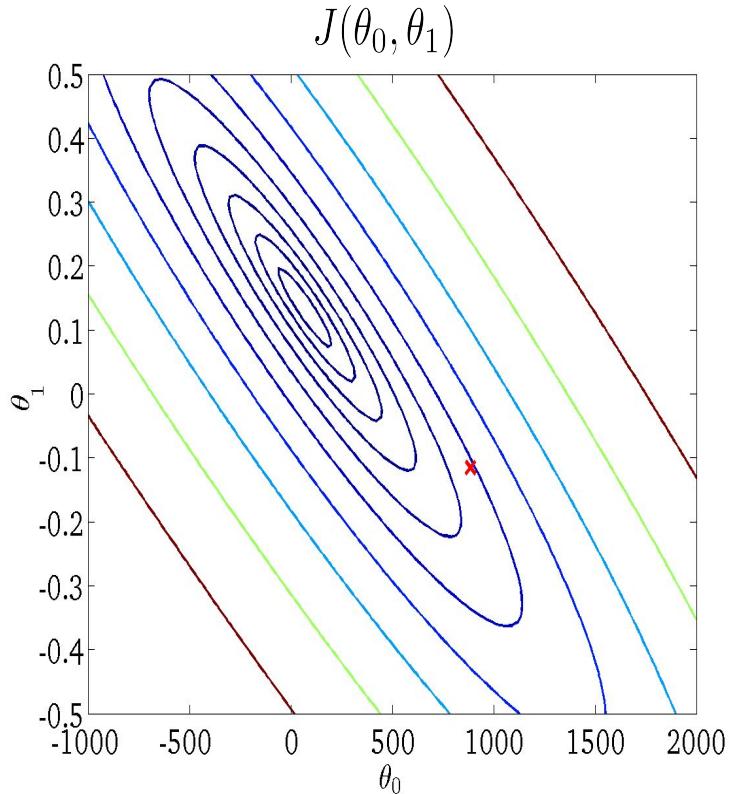
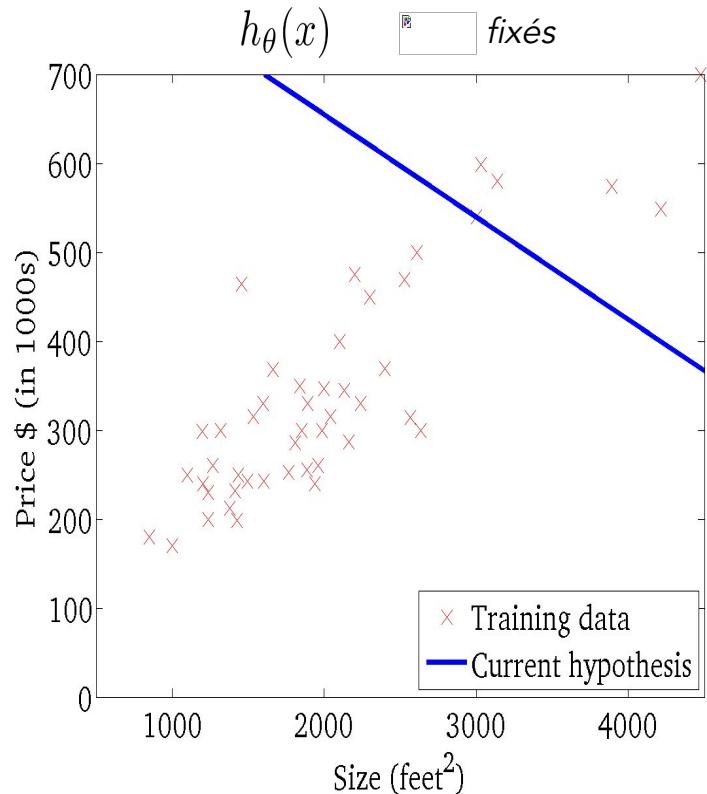
Gradient descent & régression

Projection à 2 dimensions de la fonction coût.

La couleur représente la valeur absolue de la fonction coût, elle passe du bleu au rouge (proche de 0 à fortement non nulle).



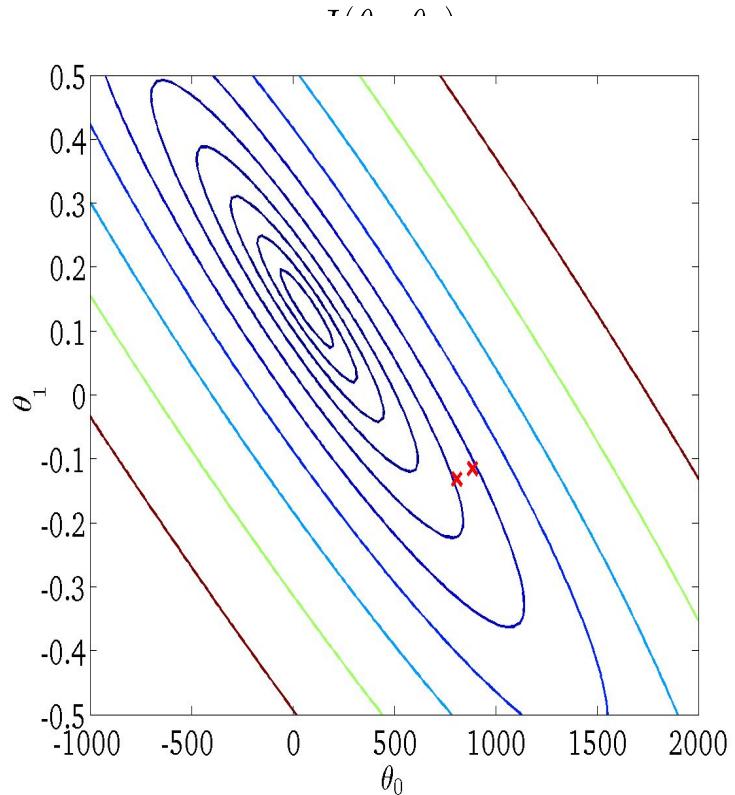
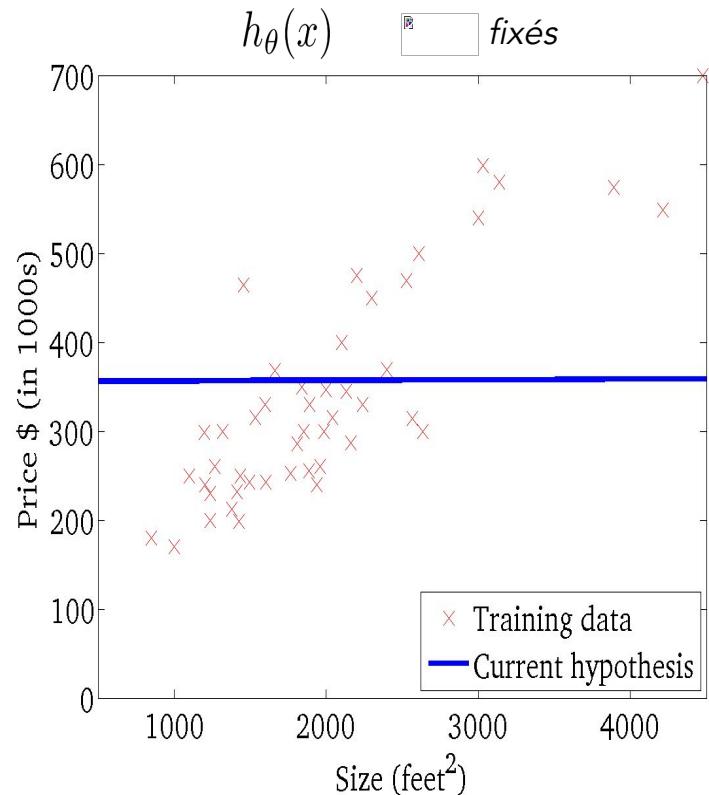
Gradient descent & régression



sont « simultanément » mis à jour



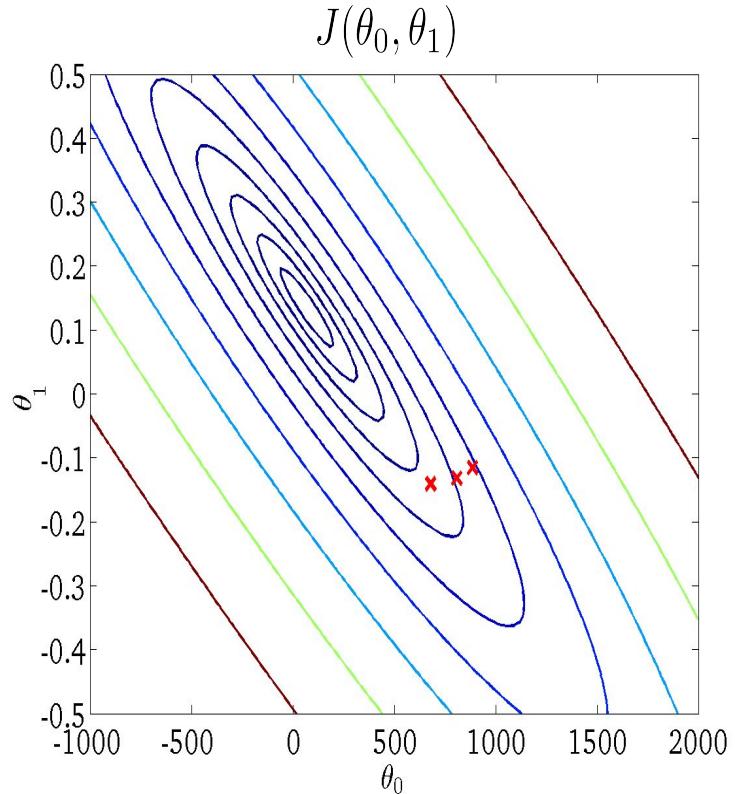
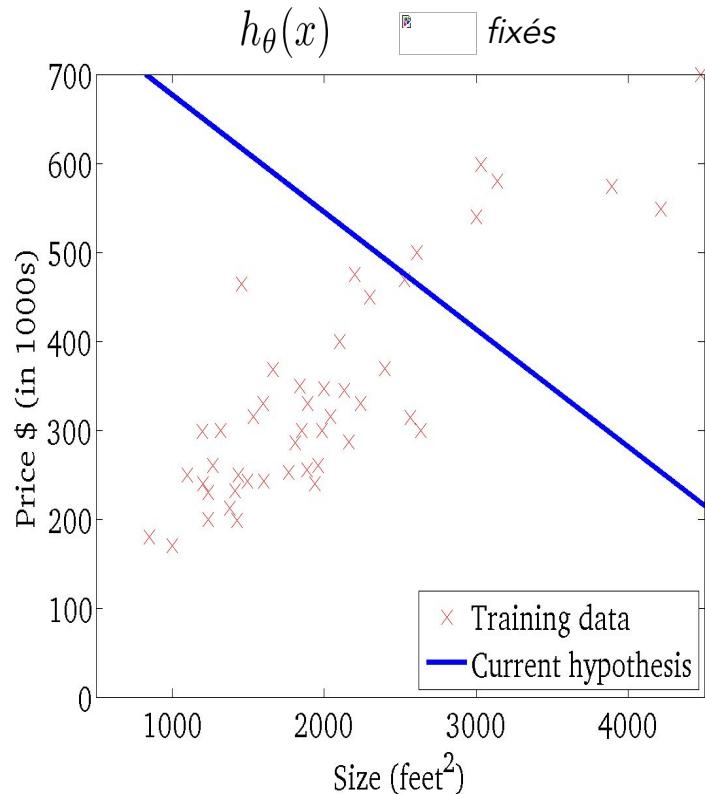
Gradient descent & régression



sont « simultanément » mis à jour



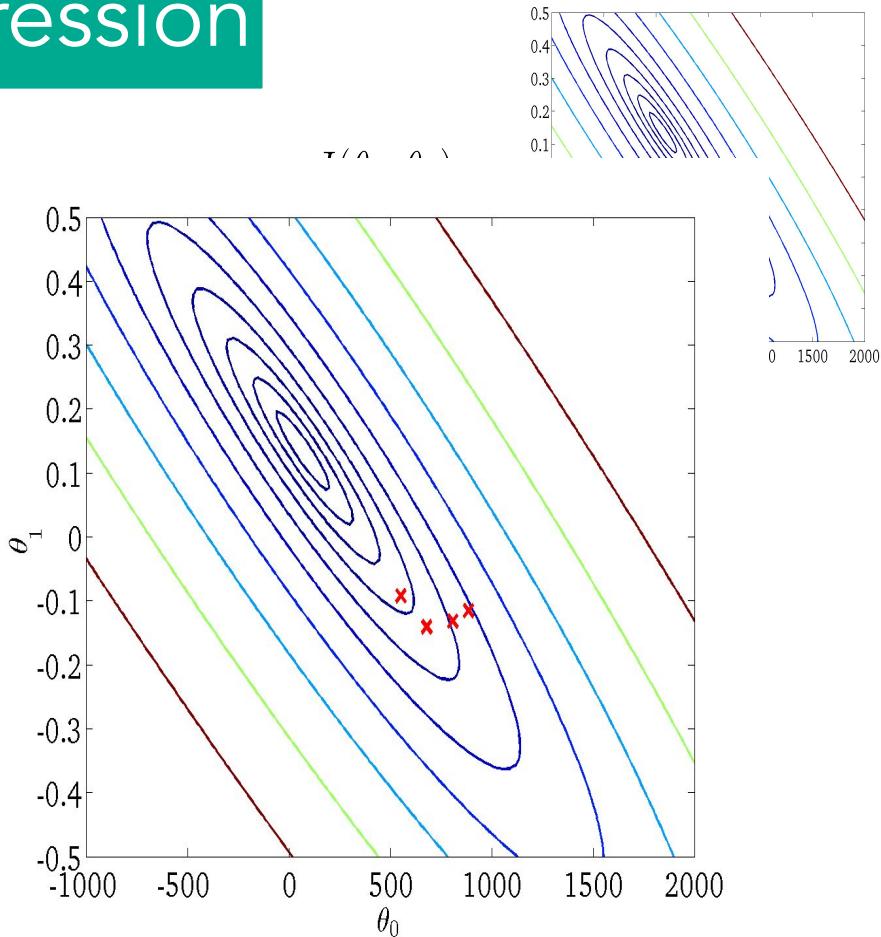
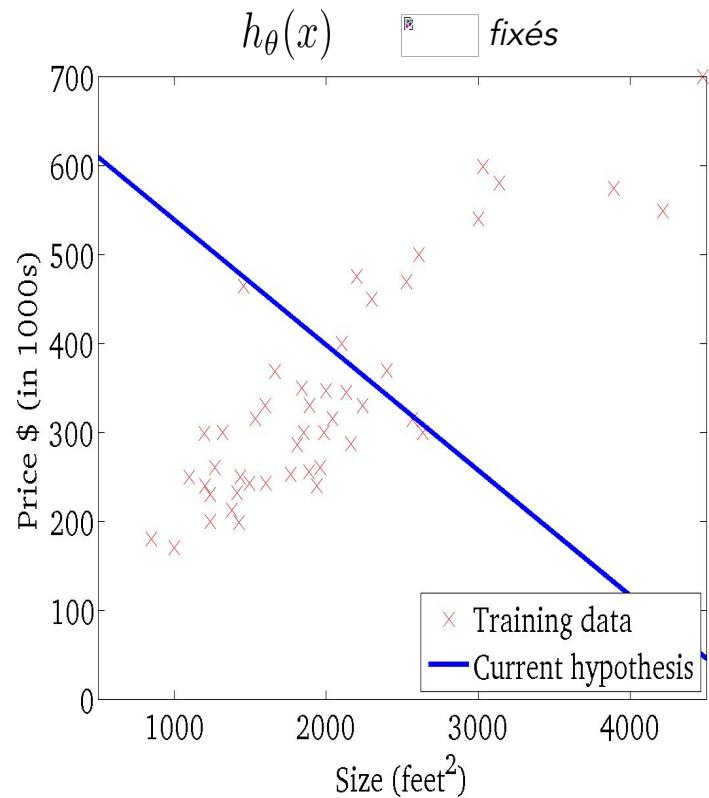
Gradient descent & régression



sont « simultanément » mis à jour



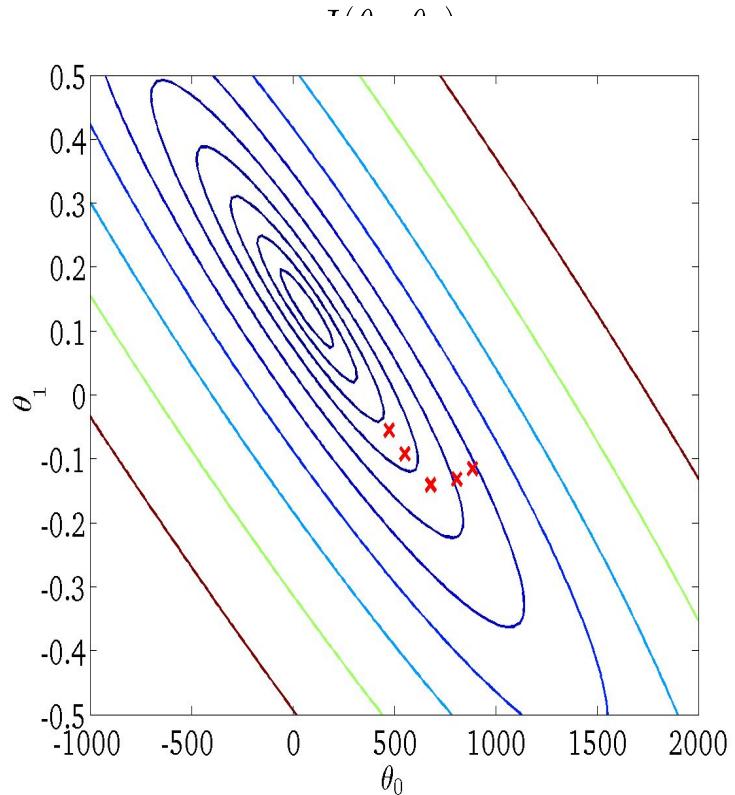
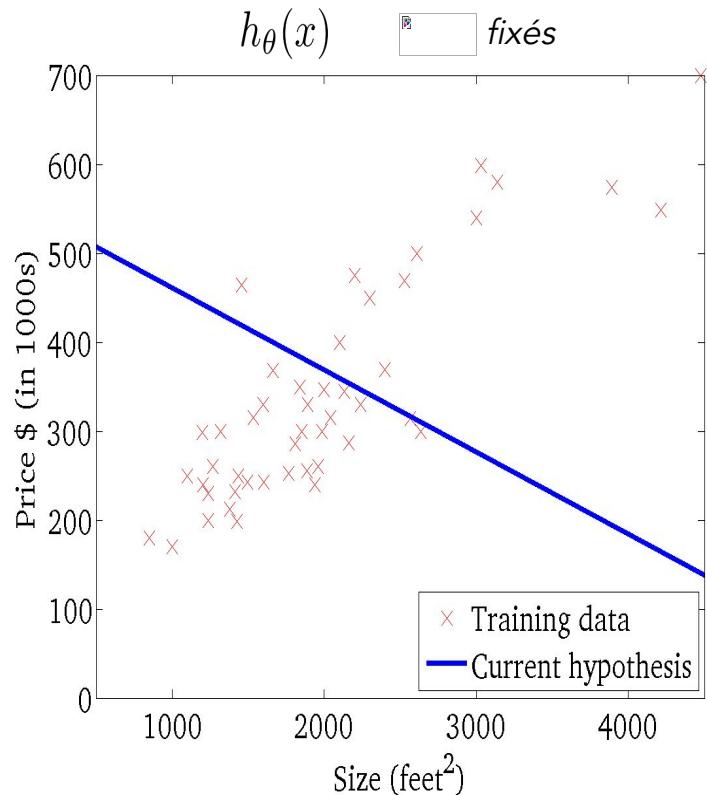
Gradient descent & régression



sont « simultanément » mis à jour



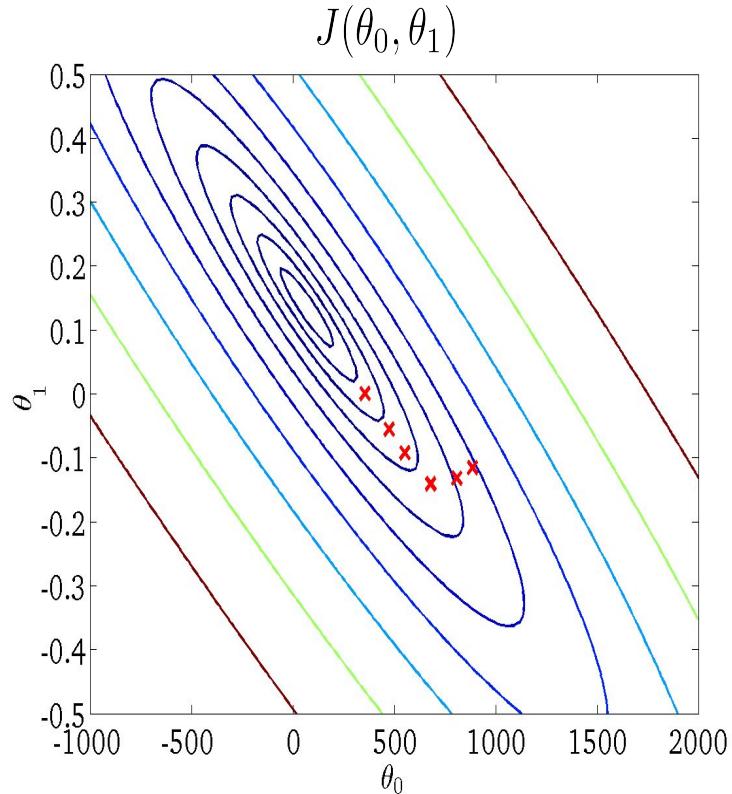
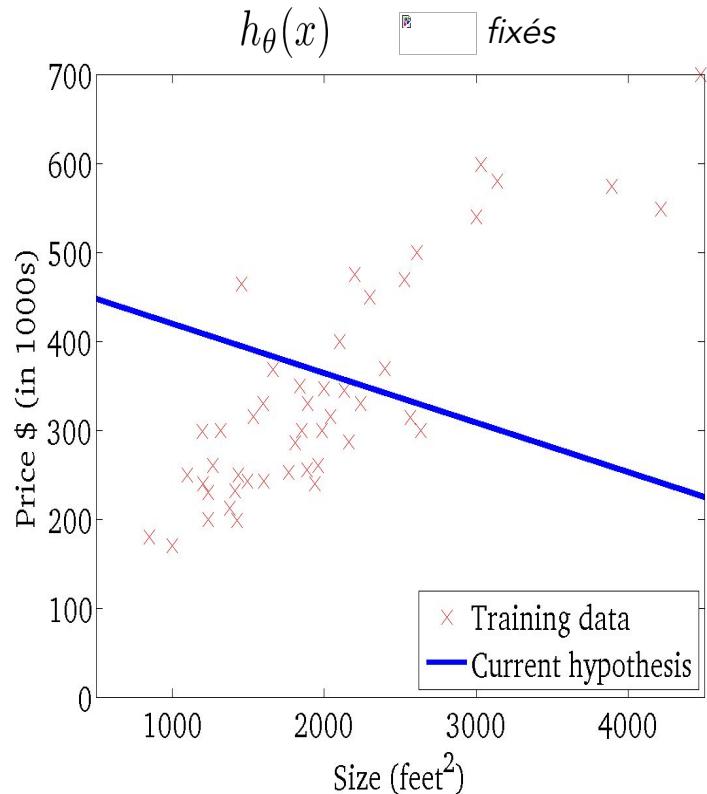
Gradient descent & régression



sont « simultanément » mis à jour



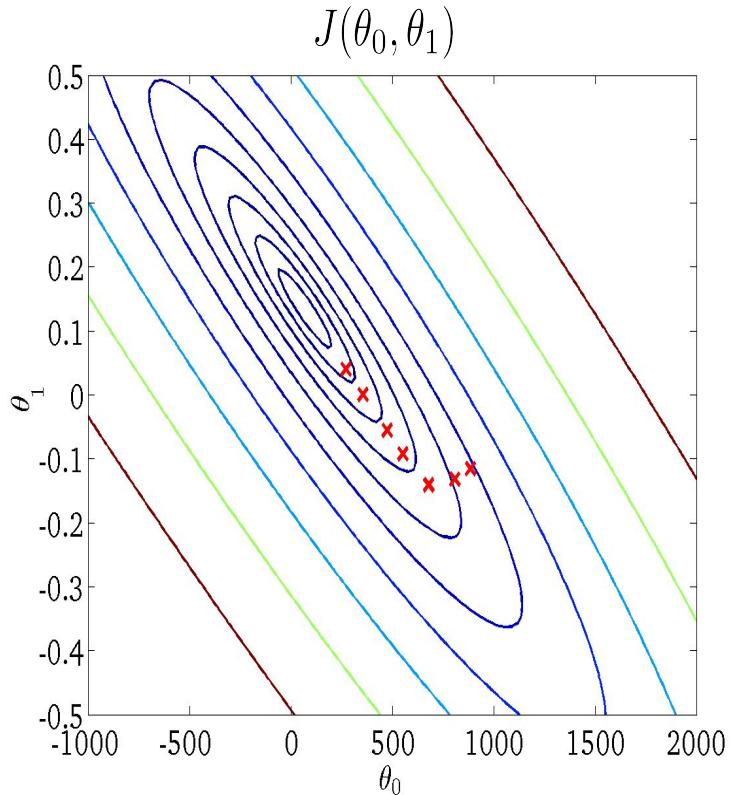
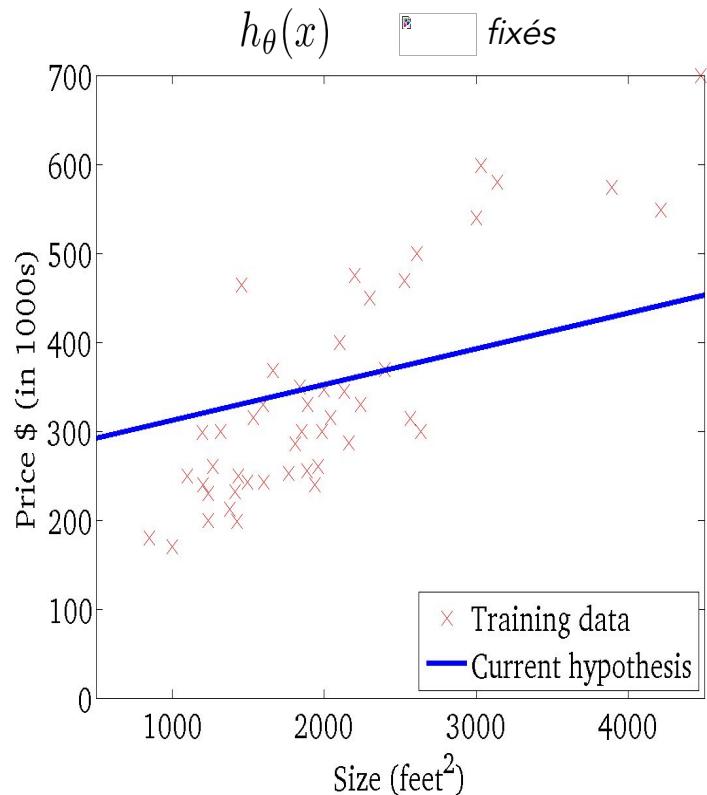
Gradient descent & régression



■ sont « simultanément » mis à jour



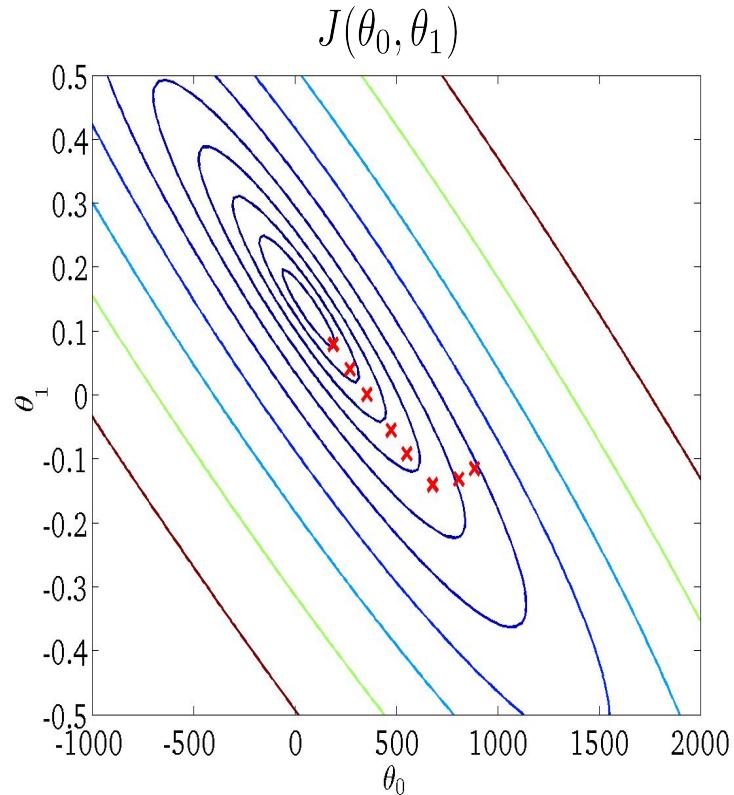
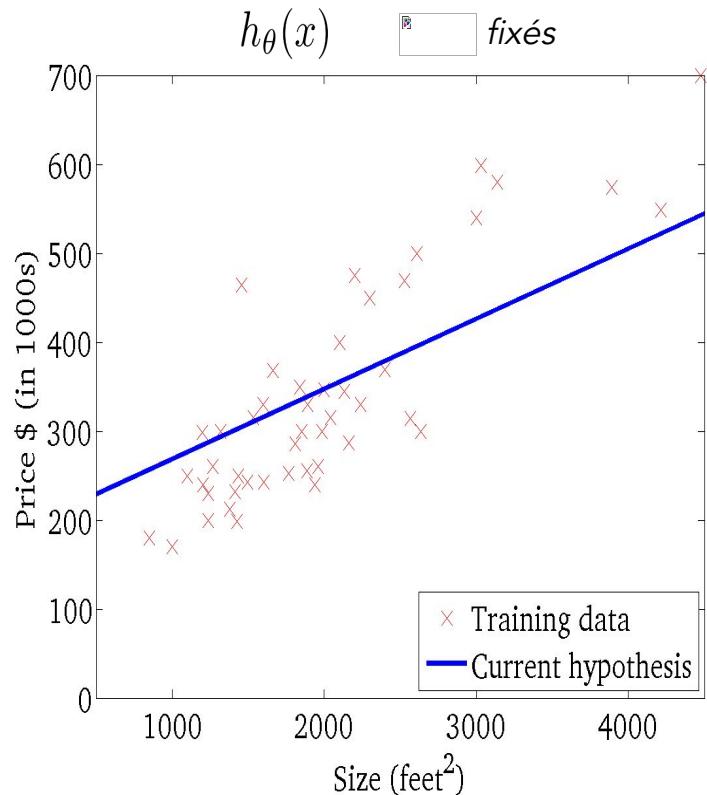
Gradient descent & régression



sont « simultanément » mis à jour



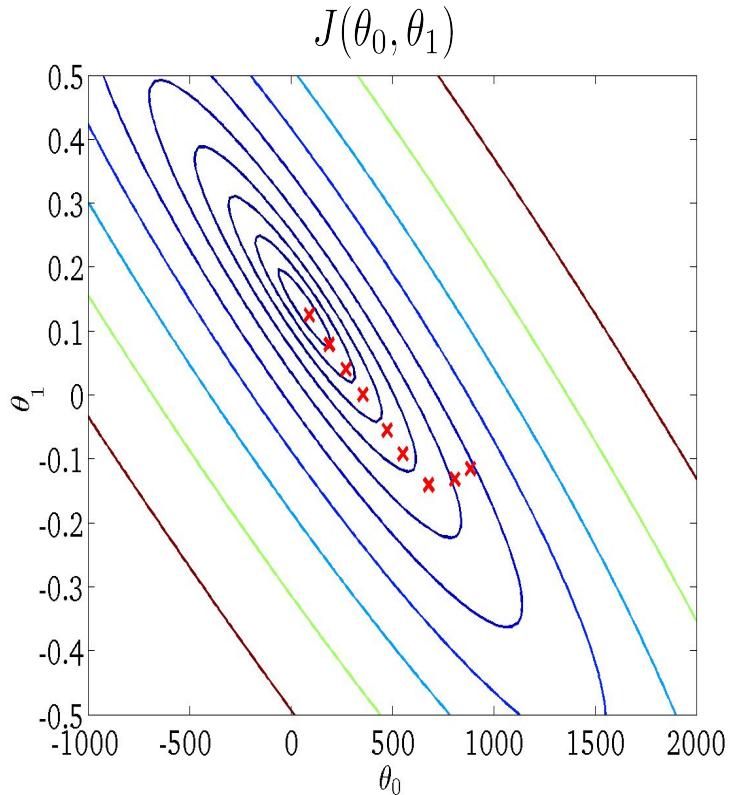
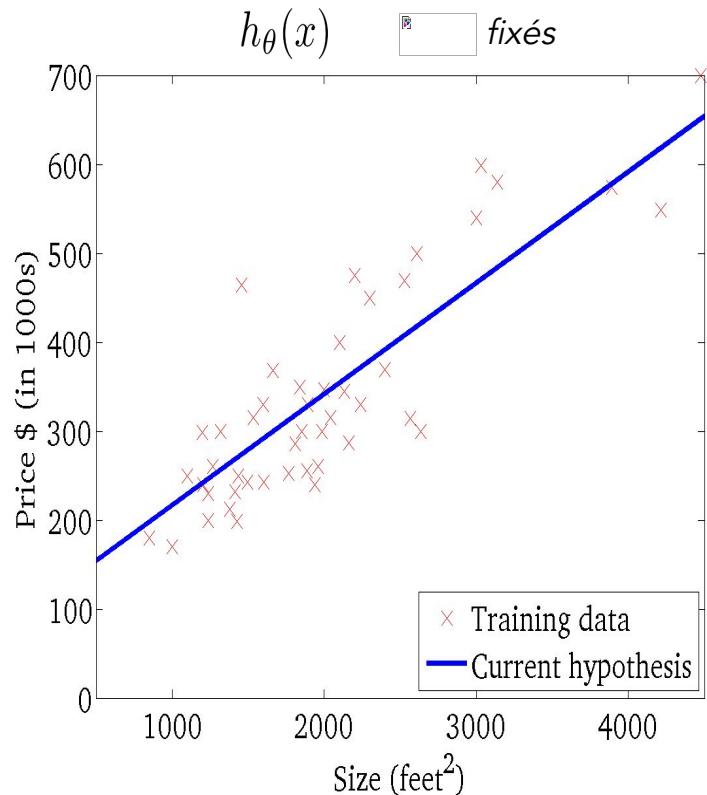
Gradient descent & régression



■ sont « simultanément » mis à jour



Gradient descent & régression



sont « simultanément » mis à jour



Gradient descent algorithm

- On définit une fonction coût $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$
- On en déduit sa dérivé $\frac{\partial}{\partial \theta_j} J(\theta) = (h_\theta(x) - y) x_j$



GRADIENT DESCENT ALGORITHM

- On débute avec des valeurs initiales de θ_0 et θ_1 (et non nul).
- On modifie les valeurs de θ_0 et θ_1 jusqu'à ce que $J(\theta_0, \theta_1)$ ait atteint un minimum

C'EST QUAND/OÙ LE MINIMUM ?

- La variation de $J(\theta_1, \theta_2)$ est inférieure à un palier (paramètre)
- Un nombre d'itérations maximum (paramètre)



Gradient descent algorithm

2 points de faiblesses de l'algorithme.

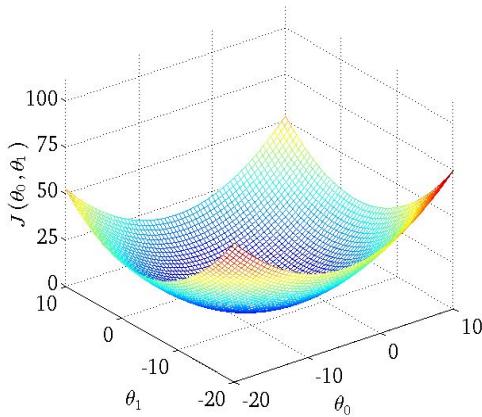
- La position initiale
- Le paramètre α , *le learning rate*.



Gradient Descent : régression

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

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



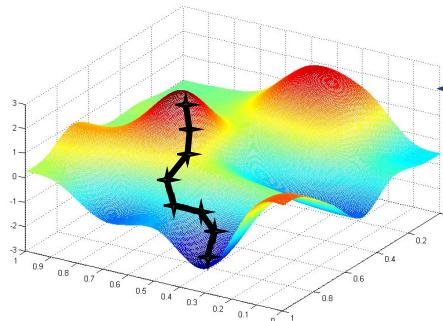
MODÈLE DE RÉGRESSION LINÉAIRE

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}



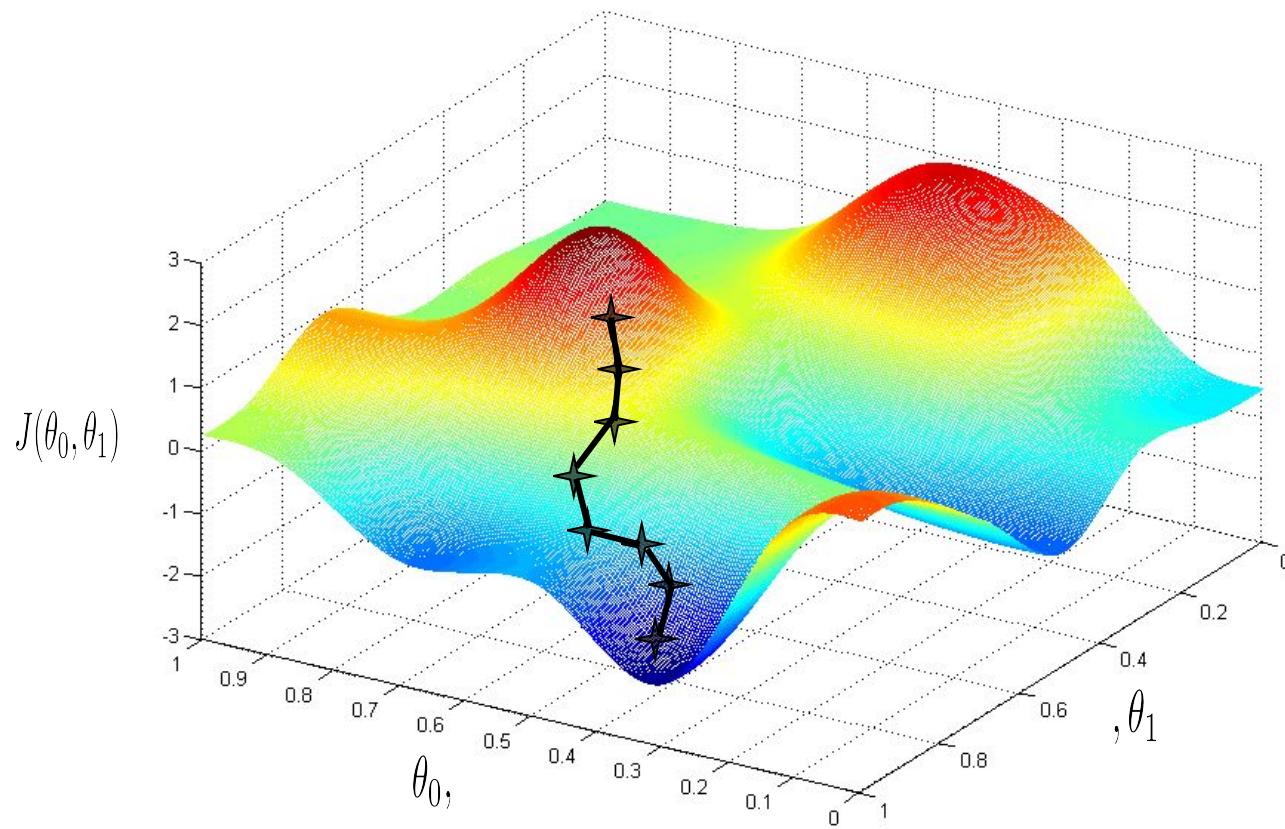
GRADIENT DESCENT ALGORITHM

Les fonctions que nous allons utiliser h peuvent être polynomiales ou multivariées. Cela va complexifier la fonction coût et sa forme.

Favoriser des formes convexes.

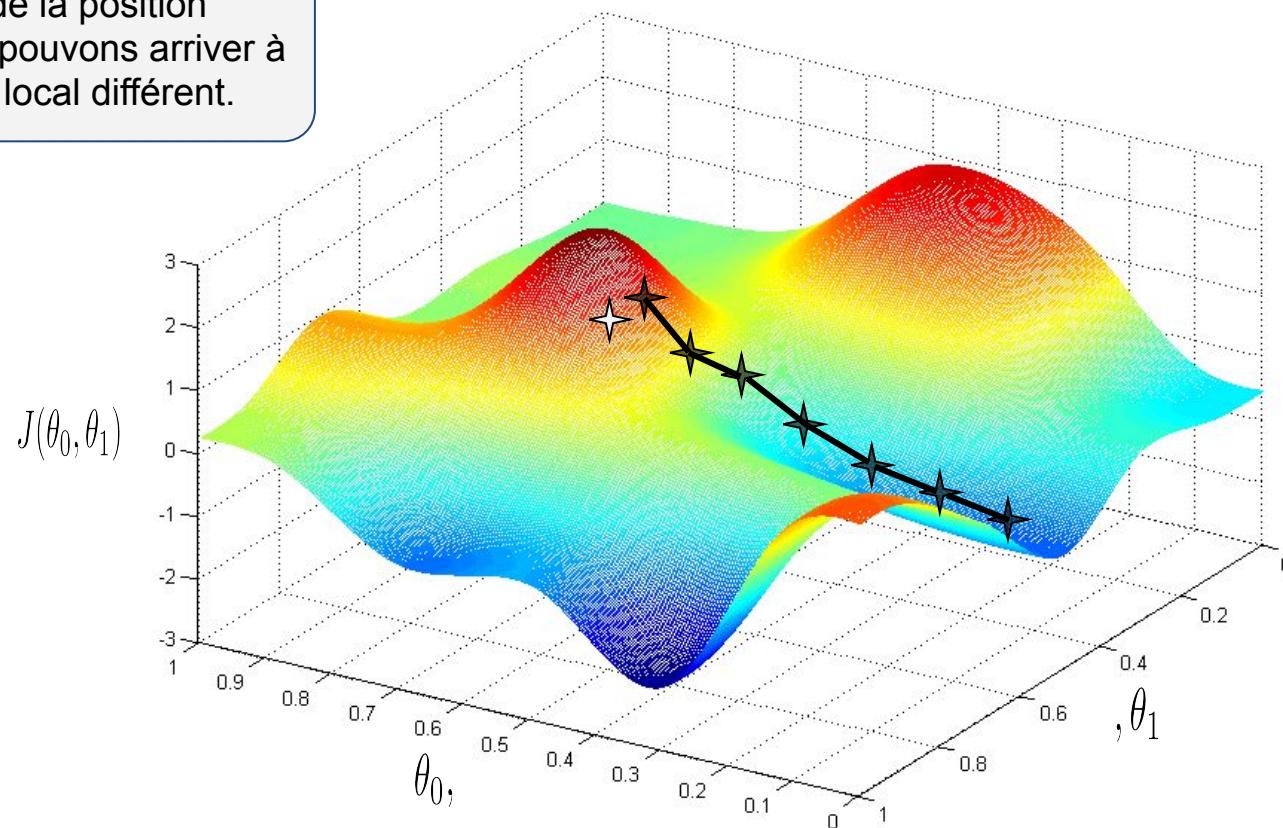


Gradient descent : valeurs initiales



Gradient descent : valeurs initiales

En fonction de la position initiale nous pouvons arriver à un minimum local différent.

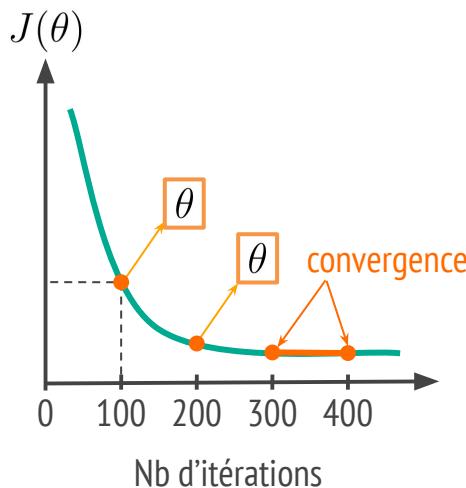


Gradient descent : vérification

S'assurer que l'algorithme du gradient fonctionne normalement

Le travail de l'algorithme du gradient est de trouver la valeur de θ qui minimise la fonction coût : $\underset{\theta}{\text{minimize}} J(\theta)$
 $J(\theta)$ doit décroître à chaque itération

Tracer la fonction coût / cost / erreur / J



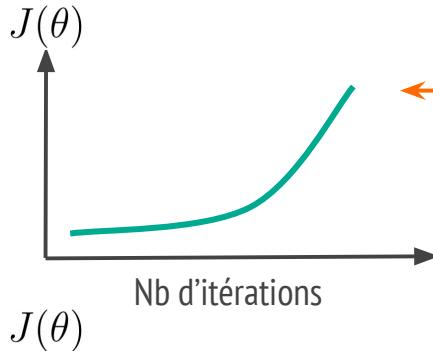
Test de convergence automatique

On déclare qu'il y a convergence si la fonction coût décroît de moins d'une certaine valeur ε , par exemple 10^{-3} , en une itération.

Problème : il est souvent difficile de déterminer ε .

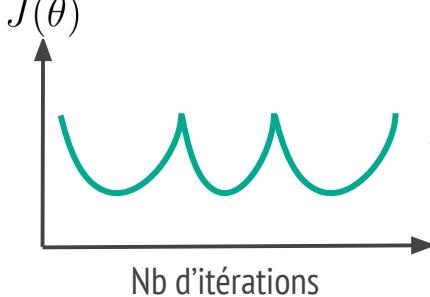
Gradient descent : vérification

Déchiffrer les courbes pour vérifier que l'algorithme fonctionne

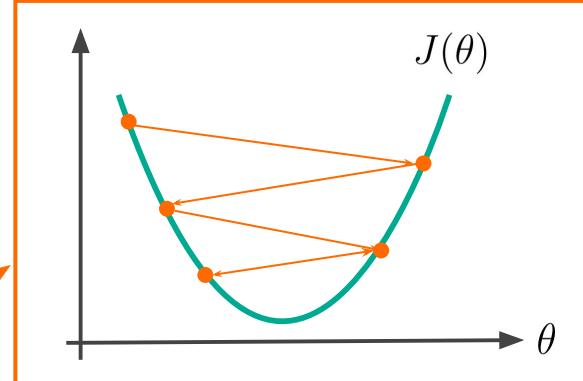


La fonction coût augmente,
l'algorithme ne fonctionne pas.
faire varier α

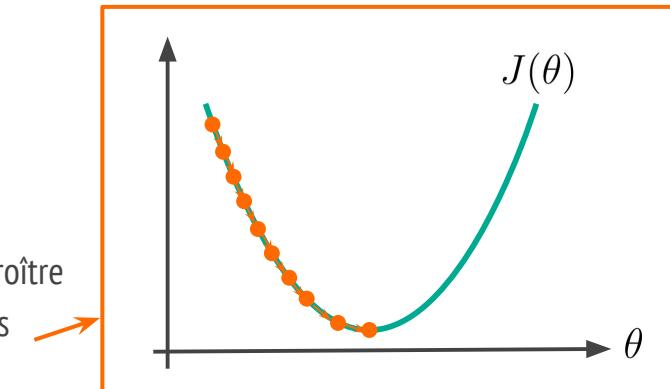
```
repeat until convergence {  
     $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$   
}
```



α est trop large



Pour des valeurs de α “suffisamment petites”, la fonction coût doit décroître à chaque itération. Mais si α est trop petit, l'algorithme convergera très (trop) lentement.



Optimization

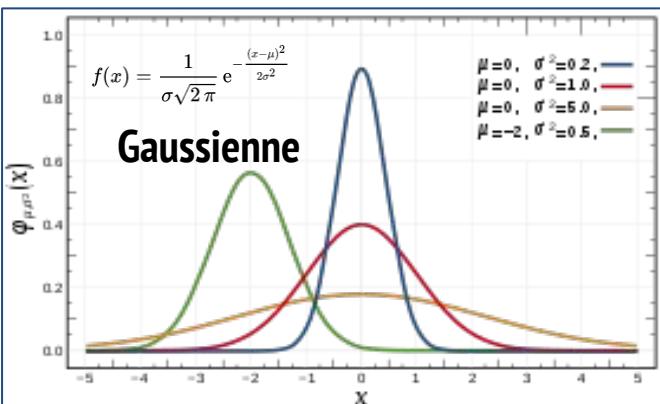
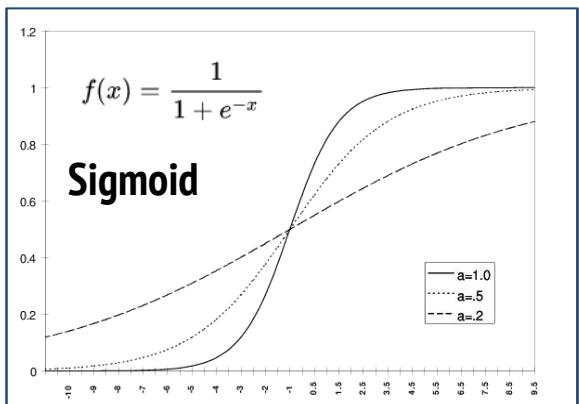
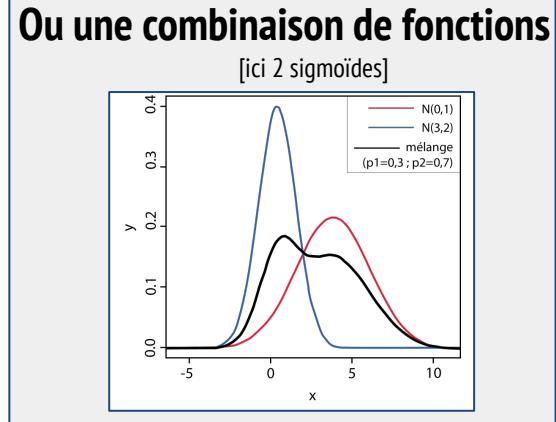
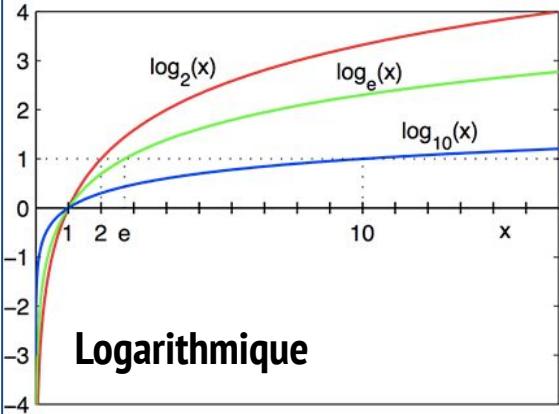
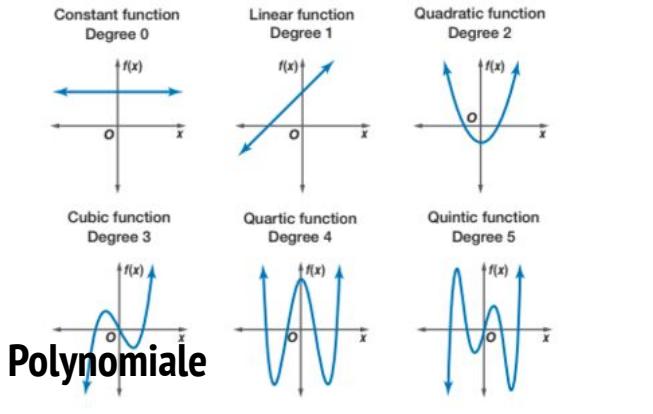
D'autres méthodes d'optimisation ont été développées

- Newton method
- Gauss-Newton
- Levenberg-Marquardt
- BFGS
- L-BFGS
- Conjugate gradient
- Etc.

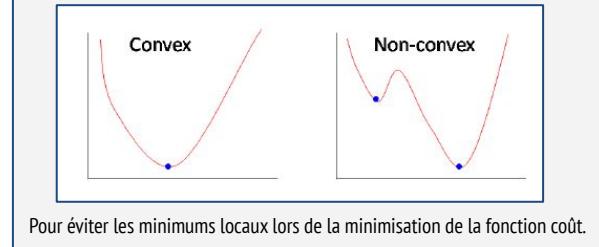
Utiliser dans d'autres champs que celui de la data science.

Fonctions non linéaires possibles

La recherche de paramètres se fait de la même manière avec des fonctions non linéaires.



Favoriser des fonctions « convexes » et connues et qui décrivent bien les données.



Pour éviter les minimums locaux lors de la minimisation de la fonction coût.



- Régression linéaire -
&
Méthode des moindres carrés II

*Apprentissage Supervisé
Y quantifiable / nombre*



Régression Multi-variable

Nombre de données

Index <i>i</i>	Surface (feet ²) x ₁	Nbr de chambre x ₂	Nbr d'étage x ₃	Age de la maison (année) x ₄	Prix (en \$1K)
0	2104	5	1	45	460
1	1416	3	2	40	232
2	1534	3	2	30	315
3	852	2	1	36	178

Index *j* Nombre de dimension des données



On travail plus communément avec des représentations des données sous forme matricielle et vectoriel (pandas, scikit-learn, numpy...)

$$X_j(i) = \begin{bmatrix} 2104 & 5 & 1 & 45 & 460 \\ 1416 & 3 & 2 & 40 & 232 \\ 1534 & 3 & 2 & 30 & 315 \\ 852 & 2 & 1 & 36 & 178 \end{bmatrix} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$X_j(i)$ = valeur de la colonne/feature *j* et de la ligne *i*.

Régression Multi-variable Linéaire

Régression linéaire univariable

$$h_{\theta}(x) = \boxed{\theta_0} + \boxed{\theta_1}x$$

Constante de la droite Pente de la droite

Régression linéaire multivariable

$$h_{\theta}(x) = \boxed{\theta_0} + \theta_1 x_1 + \boxed{\theta_2} x_2 + \cdots + \boxed{\theta_n} x_n$$

Pente de la droite de la dimension 2 Pente de la droite de la dimension n

Chaque valeur x_i correspond à une valeur du tableau (sur la même ligne i).

On fixe $x_0 = 1$



Régression Multi-variable

On a m points/lignes/événements. $i = 0, \dots, m$
On a m dimensions/colonnes/features $i = 0, \dots, n$

1. Hypothèse du modèle mathématique

On choisit le modèle mathématique (le type de courbe) qui définit au mieux les jeux de données.

Écriture matricielle

Le vecteur contient tous les paramètres des droites utilisées pour chaque dimensions/colonnes/features.

$$h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

Vecteur x (valeurs des variables)

2. Paramètres

Ici un modèle linéaire multivariable avec 1 paramètre par dimension à déterminer (+1 pour la constante).

Paramètre de la régression linéaire multivariable

On a $n+1$ paramètres.

$$\theta_0, \theta_1, \dots, \theta_n$$

3. Fonction coût

Calcul de l'erreur entre le modèle mathématique (ici la régression linéaire) et les données brutes (noté de 0 à m) pour chaque jeu de paramètres Theta (noté de 0 à n).

Moindre carré...

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

4. Gradient Descent

L'objectif est de réduire la fonction coût jusqu'à trouver les minima locaux de la fonction coût.

Répéter jusqu'à convergence

Paramètre du Gradient Descent

Learning Rate ou taux d'apprentissage un hyper-paramètre.



mise à jour des variables (j) simultanées

Dérivé partielle (j) de la fonction coût.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n)$$

$j = 0, \dots, n$

5. Résultats

Fin de l'apprentissage à la convergence du Gradient Descent. Il reste alors vérifier la qualité du résultat.



Régression linéaire

Gradient descent univariable & multi-variable

Précédemment ($n = 1$):

repeat {

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})}_{\text{gradient}} \quad \downarrow$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \quad \uparrow$$

simultaneously update : θ_0, θ_1

}

$$\frac{\partial}{\partial \theta_0} J(\theta)$$

Nouvel algorithme ($n \geq 1$):

repeat {

$$\theta_j := \theta_j - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{\text{gradient}} \quad \downarrow$$

simultaneously update θ_j for $j = 0, \dots, n$)

}

$$x_0^{(i)} = 1$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$