Calcul Scientifique

Cours 10: La régression linéaire

Alexis Lechervy

Sommaire

Calibrage d'une sonde de température

Contexte

On souhaite réaliser un appareil de domotique à l'aide d'un capteur de température. Le capteur de température est une sonde qui délivre une tension en volt qui dépend directement de la température.

Problème

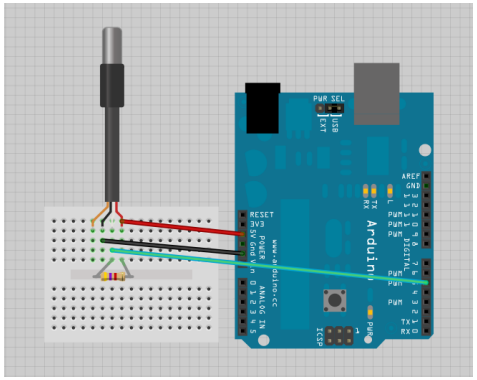
La documentation de la sonde n'est pas assez précise pour savoir quelle tension est associée à quelle température. $\Longrightarrow$ il faut calibrer la sonde de température.

Méthodologie

Méthodologie

- On va faire plusieurs mesures de température avec un thermomètre de référence.
- En parallèle, on relèvera la tension aux bornes de la sonde.
- On en déduira le lien entre les températures mesurées et les tensions au borne de la sonde.

Sonde de température

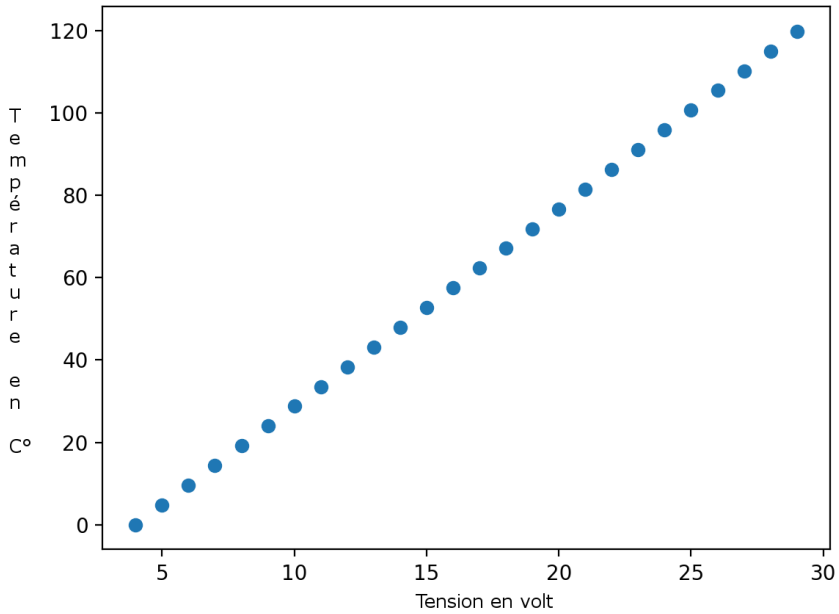


Arduino

Thermomètre de référence

Premier scénario de mesure

Nos premières mesures

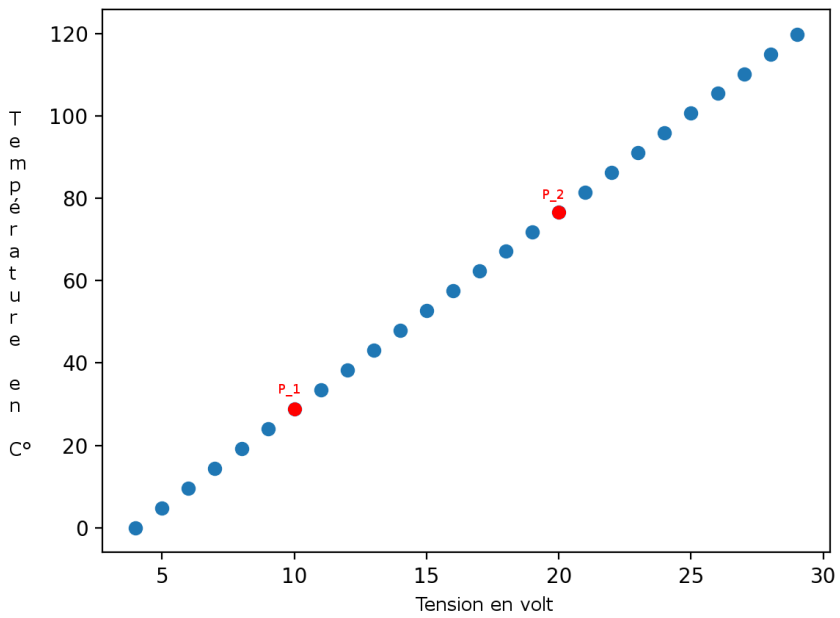


Constatation

Toutes les mesures sont parfaitement alignées. Il suffit de trouver l'équation de la droite correspondant aux mesures.

Premier scénario de mesure

Nos premières mesures

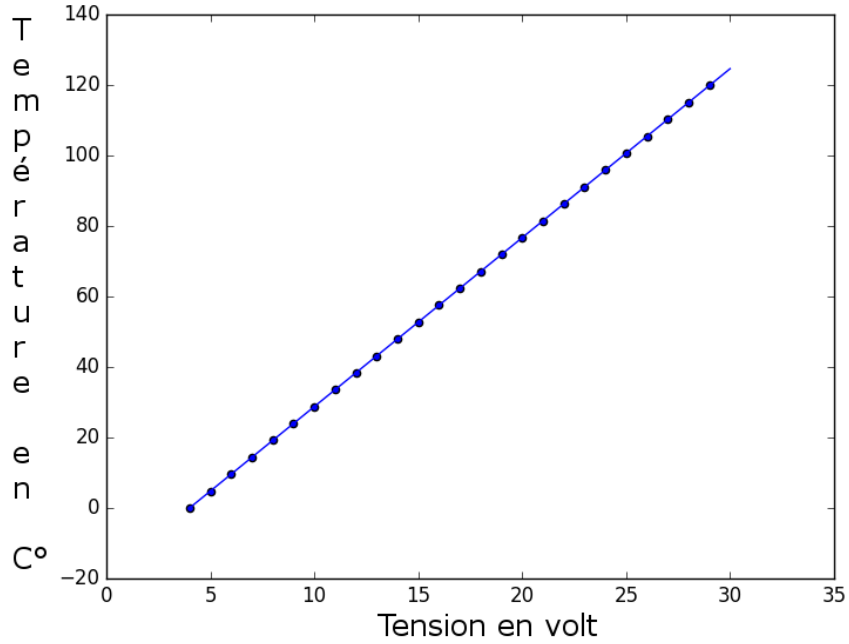


Calcul de la droite

- Pour calculer l'équation de la droite, il suffit d'avoir seulement deux points.
- Par exemple pour notre exemple P_1 (10 ; 28.76) P_2 (20 ; 76.70).
- Pour une droite de la forme $y = mx + p$, on a
$m = \frac{y_{P_1} - y_{P_2}}{x_{P_1} - x_{P_2}} = \frac{76.70 - 28.76}{20 - 10} = 4.794$ et
$b = y_{P_1} - mx_{P_1}$
$= 28.76 - 4.794x10$
$= -19.179999$

Premier scénario de mesure

Nos premières mesures

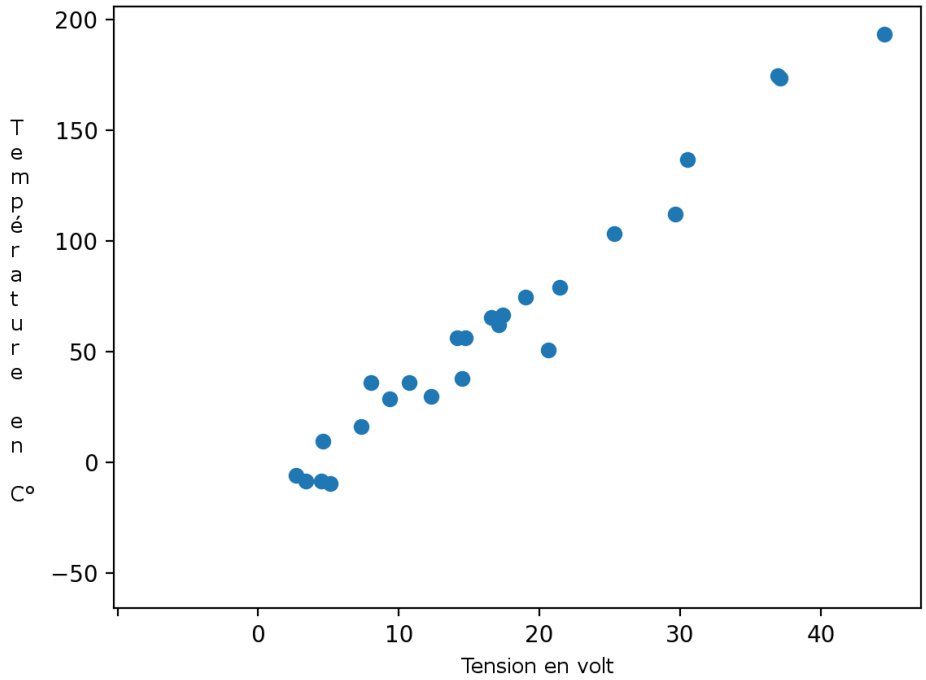


Calcul de la droite

- Pour calculer l'équation de la droite, il suffit d'avoir seulement deux points.
- Par exemple pour notre exemple P_1 (10 ; 28.76) P_2 (20 ; 76.70).
- Pour une droite de la forme $y = mx + p$, on a
  $m = \frac{y_{P_1} - y_{P_2}}{x_{P_1} - x_{P_2}} = \frac{76.70 - 28.76}{20 - 10} = 4.794$ et
  $b = y_{P_1} - mx_{P_1}$
  $= 28.76 - 4.794x10$
  $= -19.179999$

Scénario de mesure réel

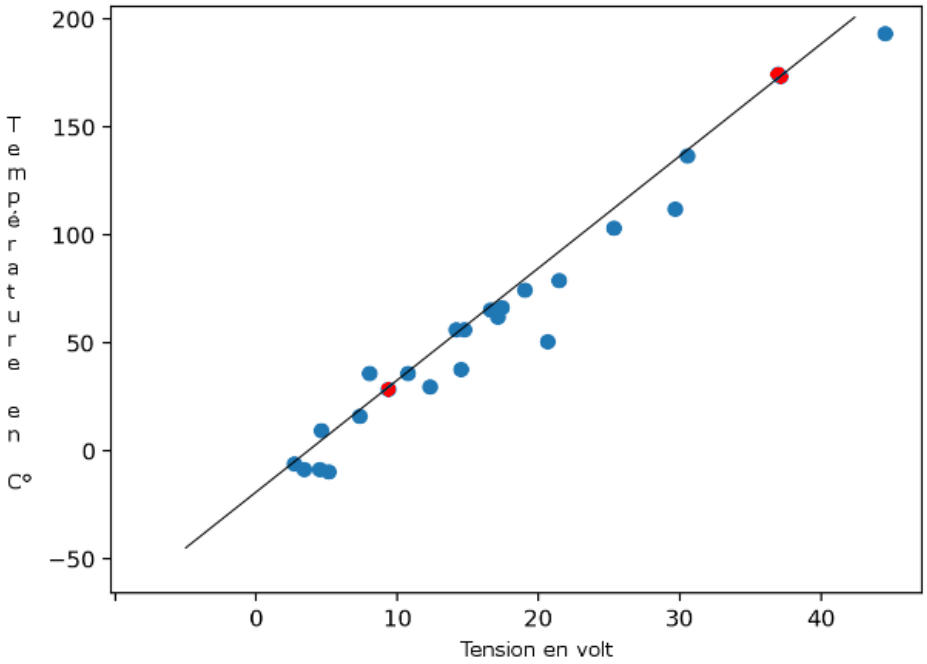
Mesures réelles



Les imprécisions de la mesure, les variabilités électroniques... font que les points réellement mesurés ne sont pas parfaitement alignés.

Scénario de mesure réel

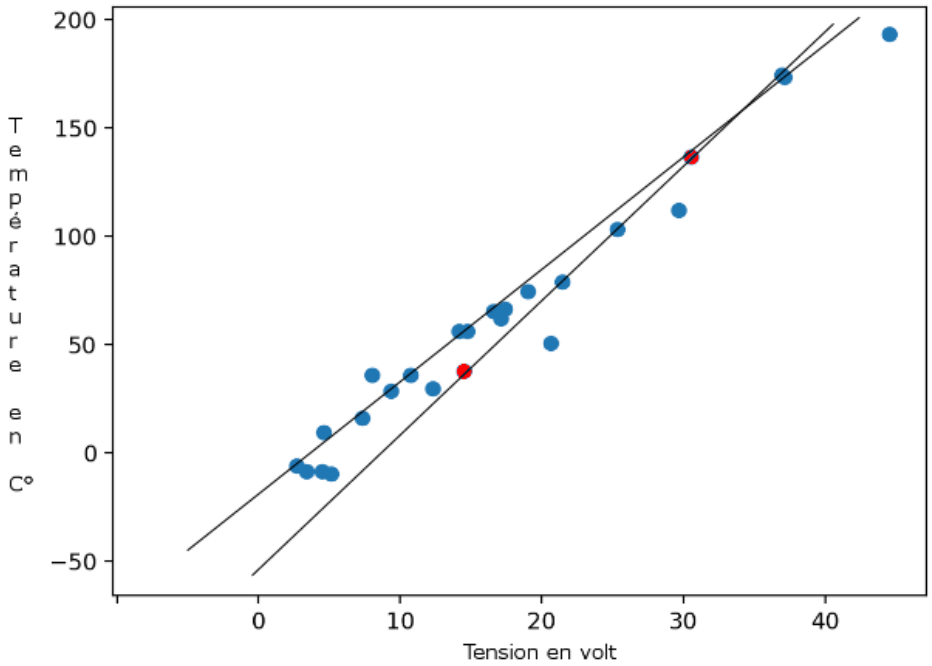
Mesures réelles



Les imprécisions de la mesure, les variabilités électroniques... font que les points réellement mesurés ne sont pas parfaitement alignés.

Scénario de mesure réel

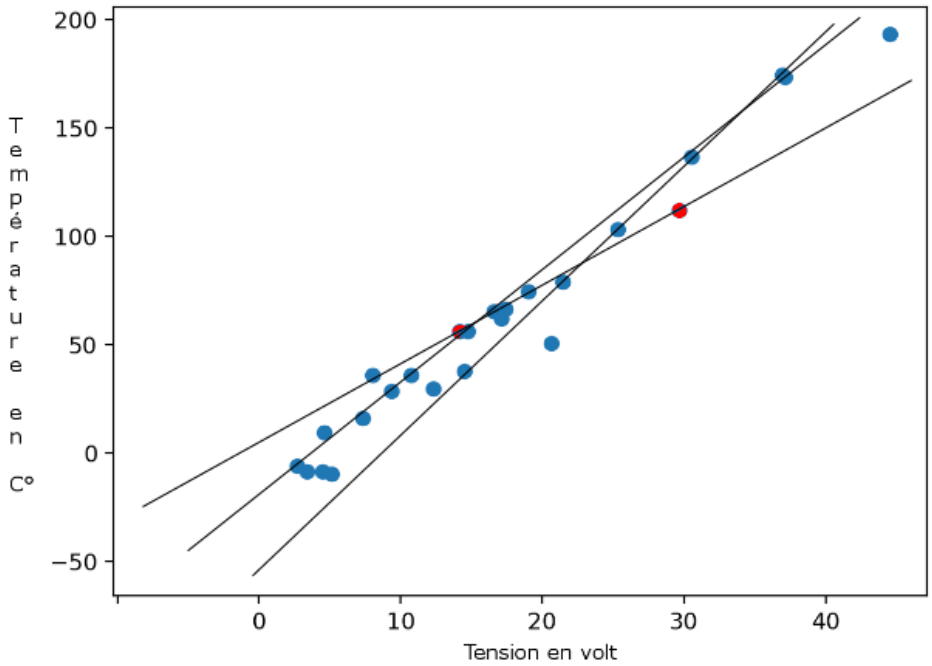
Mesures réelles



Les imprécisions de la mesure, les variabilités électroniques... font que les points réellement mesurés ne sont pas parfaitement alignés.

Scénario de mesure réel

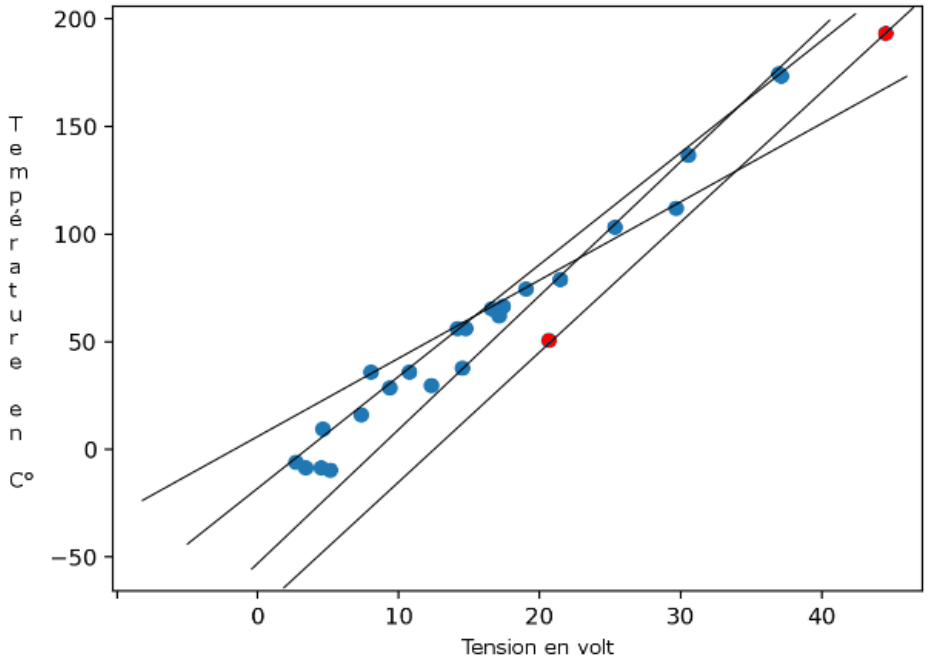
Mesures réelles



Les imprécisions de la mesure, les variabilités électroniques... font que les points réellement mesurés ne sont pas parfaitement alignés.

Scénario de mesure réel

Mesures réelles



Les imprécisions de la mesure, les variabilités électroniques... font que les points réellement mesurés ne sont pas parfaitement alignés.

Conséquences

- Plusieurs droites différentes sont possibles.

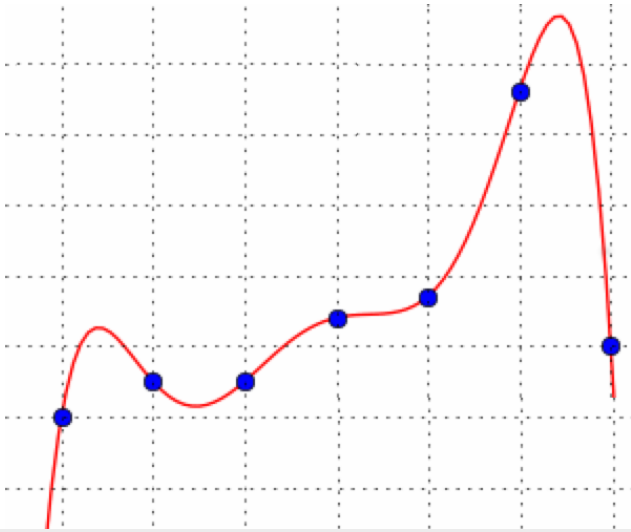
Sommaire

La régression

Principes

- La régression est un apprentissage supervisé. On connaît la "vrai valeur" associée à des exemples d'apprentissage.
- L'objectif de la régression est de pouvoir prédire une valeur réelle pour un exemple donné ne fessant pas parti des exemples d'apprentissage.
- On recherche pour cela une fonction h_θ de paramètre θ vérifiant $y = h_\theta(x)$ sur les exemples connus.

Exemple

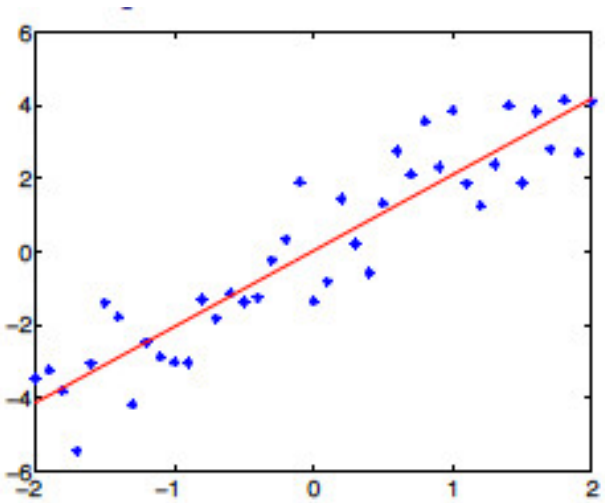


La régression linéaire

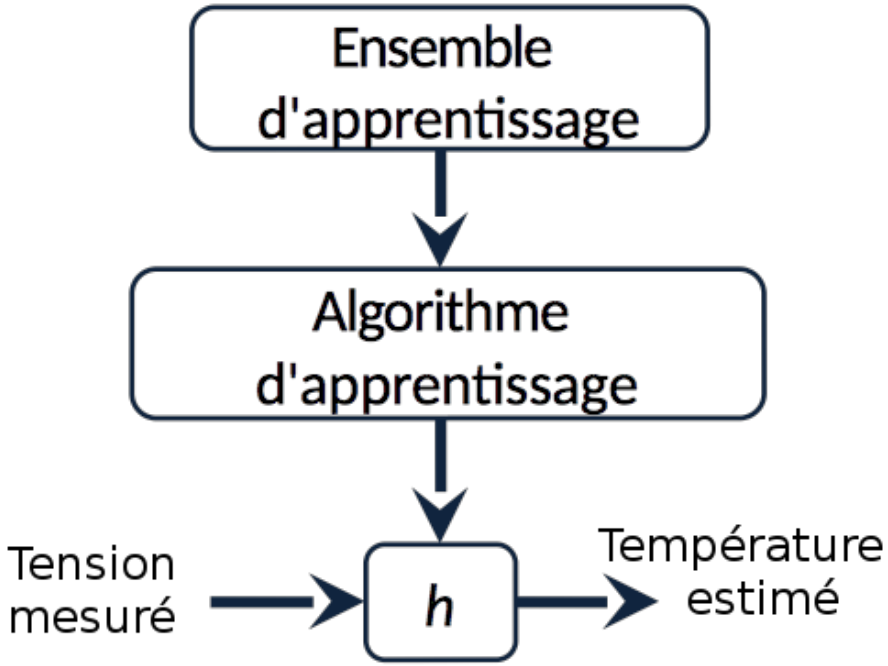
La régression linéaire

- La régression linéaire est un problème de régression dont la fonction recherchée est une droite.
- On recherche par conséquence parmi toutes les droites possibles, la droite la plus proche de nos données d'apprentissage.
- La fonction cible h_θ de paramètre $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$ est de la forme $y = \theta_1 x + \theta_0$.

Exemple



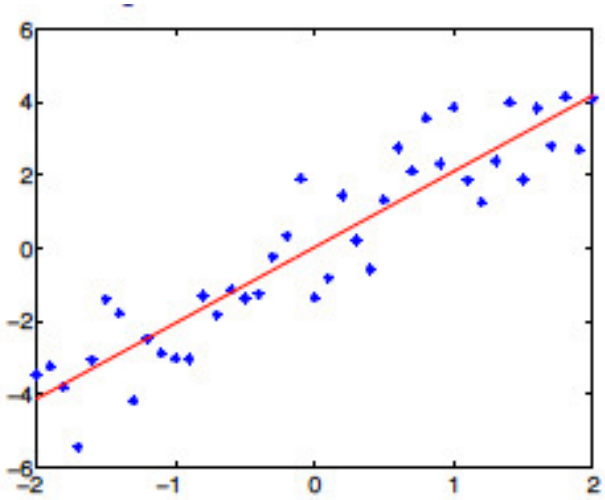
Objectif



Comment représenter l'hypothèse h ?

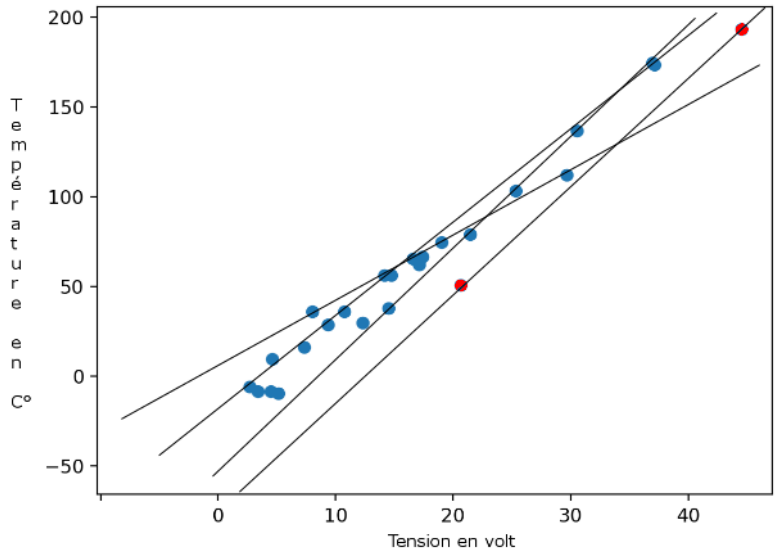
Dans le cas de la régression linéaire à une seule variable, on a :

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



Quelle droite choisir ?

Comment choisir θ définissant la droite sélectionnée ?



Idée !

Choisir θ_0, θ_1 tel que $h_\theta(\mathbf{x})$ soit le plus proche possible de $\mathbf{y}$ pour les couples connus d'apprentissage $(\mathbf{x}, \mathbf{y})$.

Erreur de prédiction

Prédictions

Pour tout les exemples d'apprentissage x_i, on peut définir une prédiction $\tilde{y}_i$ de valeur de la fonction recherchée en x_i. $\tilde{y}_i = h_\theta(x_i) = \theta_1 x_i + \theta_0$.
On peut regrouper toutes les prédictions pour chaque exemple d'apprentissage dans un vecteur $\tilde{y} = \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \tilde{y}_m \end{bmatrix}$. Avec m le nombre d'exemple d'apprentissage.

Mesure de l'erreur

On peut quantifier l'erreur commise par notre régresseur en comparant le vecteur des prédictions estimés et les vrais valeurs pour les exemples d'apprentissages :

$$\|\tilde{y} - y\|$$

Problème d'optimisation

Problème d'optimisation

On cherche donc la droite définie par θ fessant le moins d'erreur :

$$\arg\min_{\theta} \|\tilde{y} - y\|$$

Réécriture du problème

$$\arg\min_{\theta} \|\tilde{y} - y\| = \arg\min_{\theta} \|\tilde{y} - y\|^2 = \arg\min_{\theta} \frac{\|\tilde{y} - y\|^2}{m}$$

Car la norme d'un vecteur est un nombre positif et la fonction $x \longrightarrow x^2$ est croissante sur $\mathbb{R}^+$.
Cette réécriture permet de ne pas calculer la racine carré de la norme et de se ramener à une moyenne d'erreur quadratique.

Erreur de prédiction

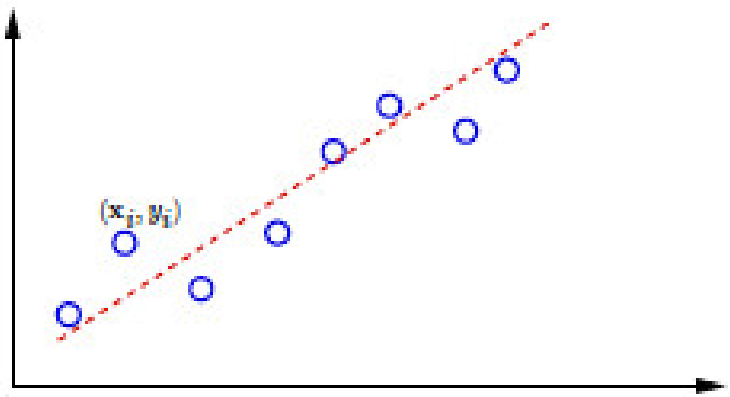
Le critère de moindre carré

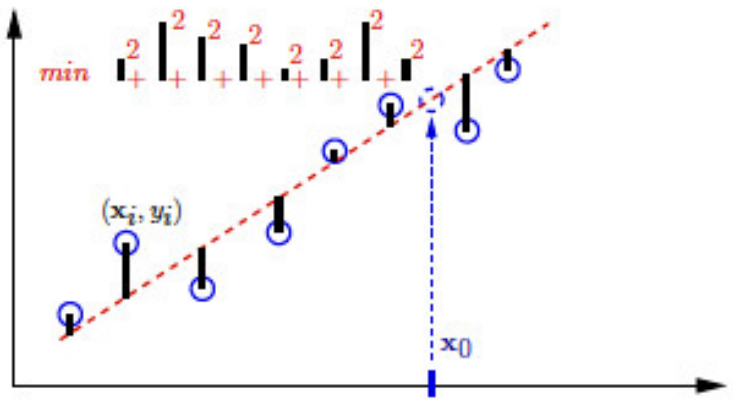
- On peut mesurer l'erreur de prédiction en termes de moyenne des distances aux carrés :

$$J(y, \tilde{y}) = \frac{\|y - \tilde{y}\|^2}{m}.$$

- On cherche donc les θ minimisant la fonction de coût suivantes :

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

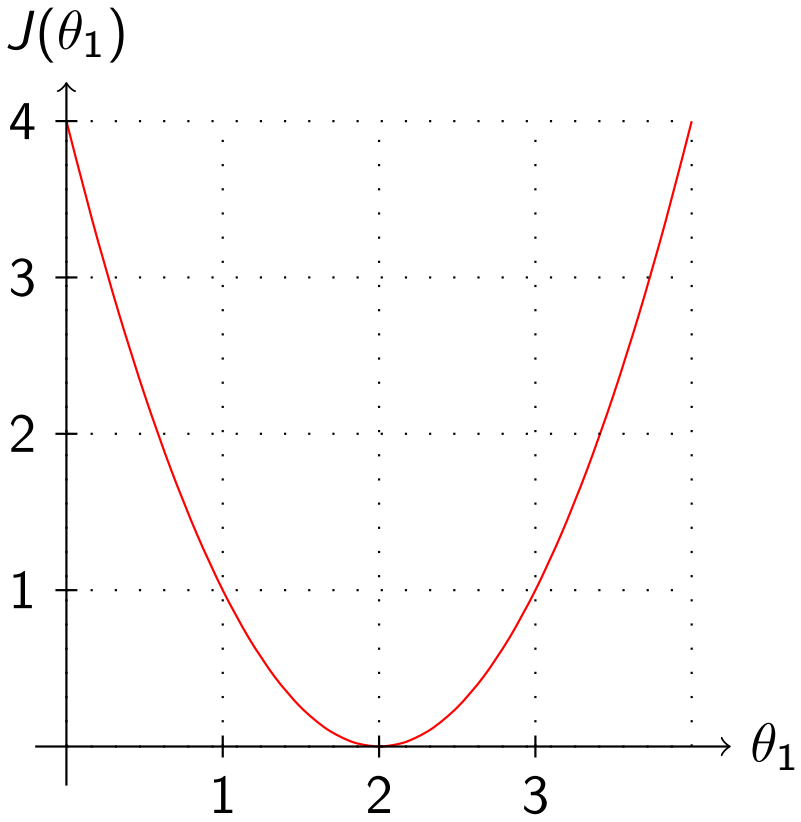




Résolution par descente de gradient

Idée

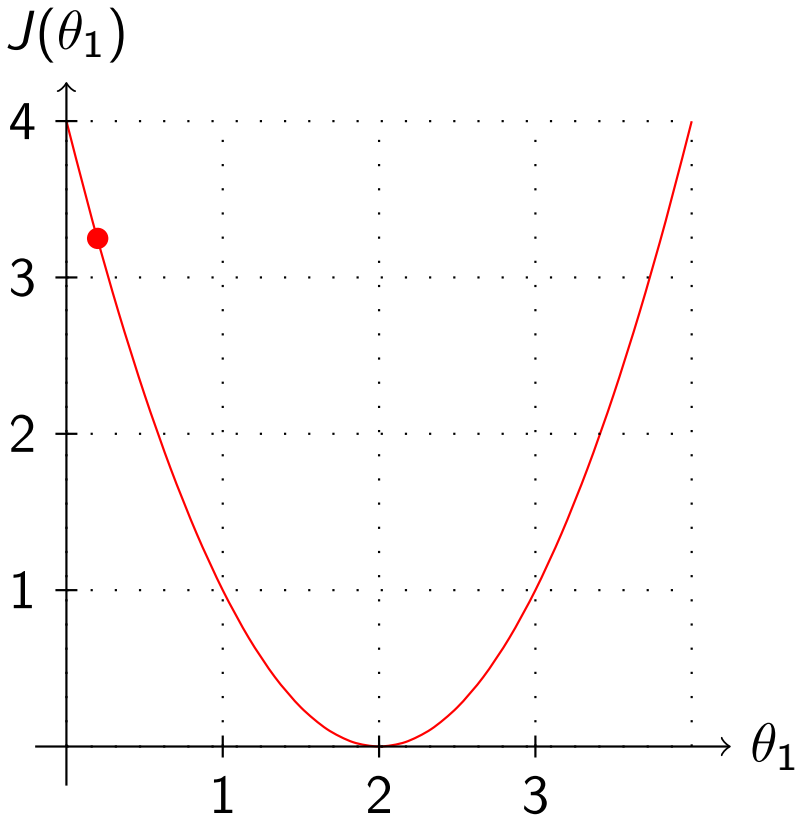
On peut atteindre le minimum en se dirigeant itérativement dans la direction de plus forte pente.



Résolution par descente de gradient

Idée

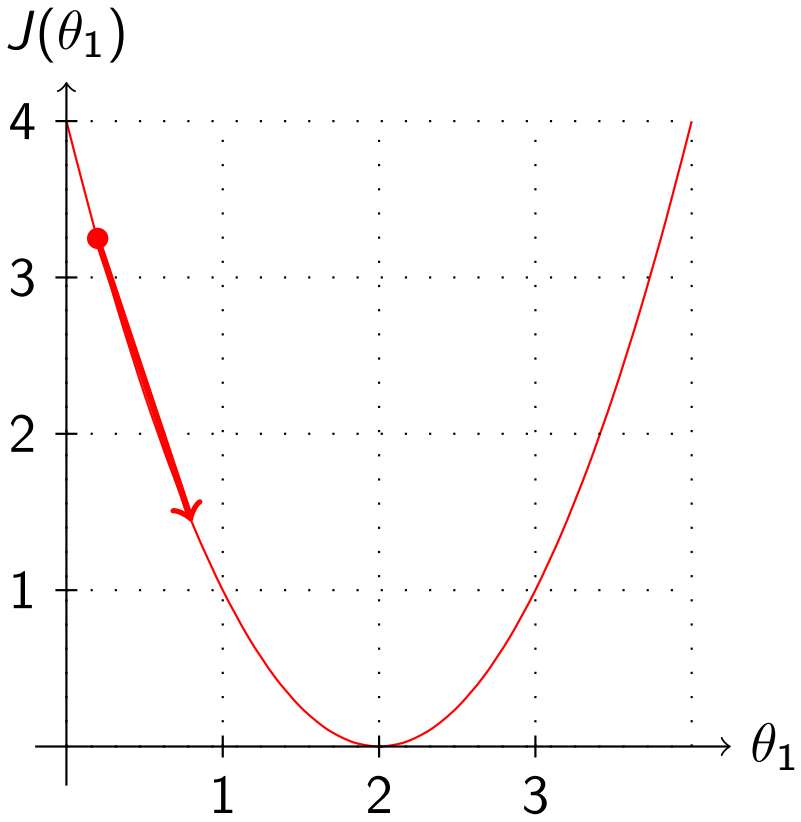
On peut atteindre le minimum en se dirigeant itérativement dans la direction de plus forte pente.



Résolution par descente de gradient

Idée

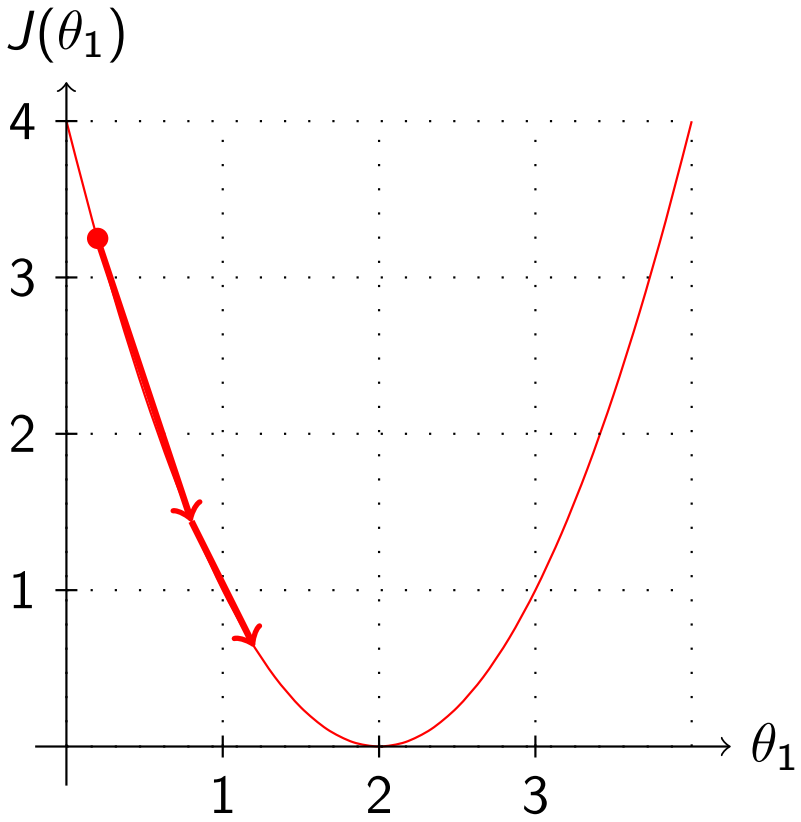
On peut atteindre le minimum en se dirigeant itérativement dans la direction de plus forte pente.



Résolution par descente de gradient

Idée

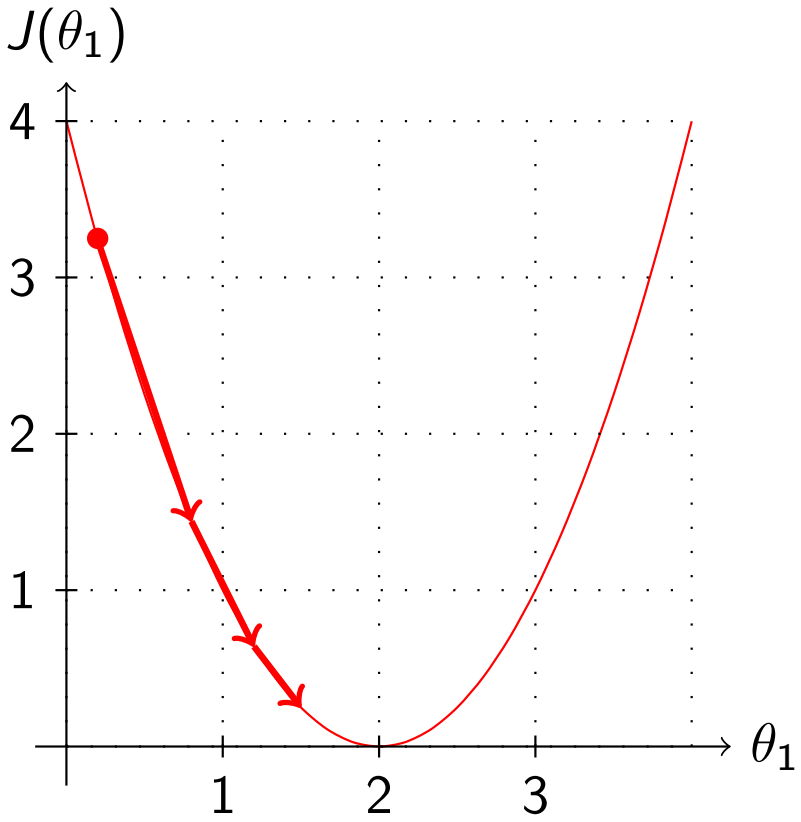
On peut atteindre le minimum en se dirigeant itérativement dans la direction de plus forte pente.



Résolution par descente de gradient

Idée

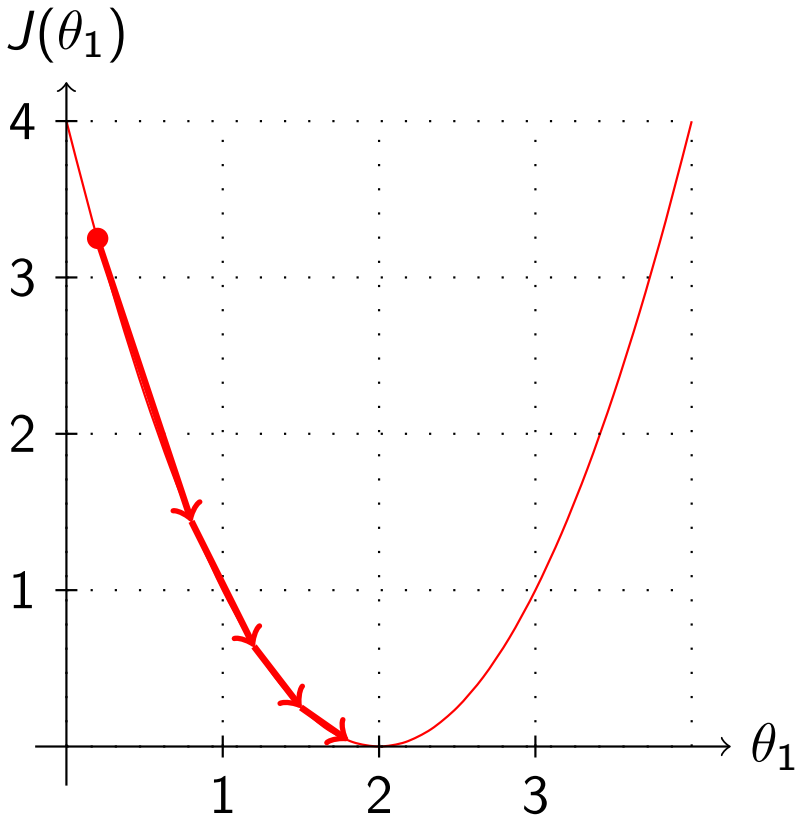
On peut atteindre le minimum en se dirigeant itérativement dans la direction de plus forte pente.



Résolution par descente de gradient

Idée

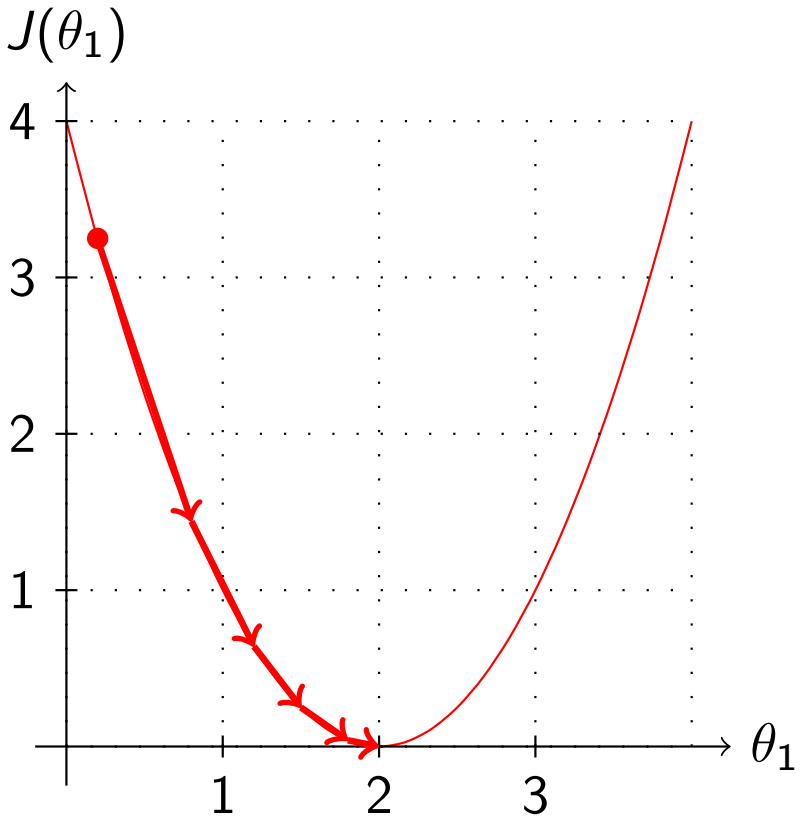
On peut atteindre le minimum en se dirigeant itérativement dans la direction de plus forte pente.



Résolution par descente de gradient

Idée

On peut atteindre le minimum en se dirigeant itérativement dans la direction de plus forte pente.



Résolution par descente de gradient

Problème d'optimisation à résoudre

$$\arg \min_{\theta_0, \theta_1} J(\theta_0, \theta_1).$$

Solution par descente de gradient

Initialiser avec θ_0, θ_1 choisies aléatoirement.
Répéter jusqu'à convergence :

$$\theta_i := \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta_i) \text{ avec } i \in \{0, 1\}.$$

α est une constante correspondant au taux d'apprentissage.
Important : θ_0 et θ_1 sont à mettre à jours simultanément.

Code scipy

Code de la fonction de coût

```
def J(theta,x,y) :
    y_pred = theta[1]*x+theta[0]
    return np.mean((y-y_pred)**2)
```

Résolution du problème d'optimisation

```
theta = scipy.optimize.minimize(lambda theta :J(theta,x,y),
        jac=False,
        x0=np.random.random(2),
        method='BFGS').x
```

Remarque : L'utilisation d'une lambda fonction permet d'éviter d'utiliser des variables globales.

Code scipy avec le gradient (pour un résultat plus précis)

Calcul du gradient de la fonction de coût

- La fonction de coût : $J(\theta) = \frac{\sum_i^m (\tilde{y}-y)^2}{m} = \frac{\sum_i^m (\theta_1 x+\theta_0-y)^2}{m}$.
- Rappel : $\frac{\partial U(\theta_i)^2}{\partial \theta_i} = 2U(\theta_i)\frac{\partial U(\theta_i)}{\partial \theta_i}$ et $\frac{\partial \sum_i^m U(\theta_i)}{\partial \theta_i} = \sum_i^m \frac{\partial U(\theta_i)}{\partial \theta_i}$
- Gradient de la fonction de coût :

$$\frac{\partial J(\theta)}{\partial \theta_0} = 2\frac{\sum_i^m (\tilde{y}-y)}{m}, \frac{\partial J(\theta)}{\partial \theta_1} = 2\frac{\sum_i^m x(\tilde{y}-y)}{m}$$

Code du gradient de la fonction de coût

```
def gradJ(theta,x,y) :
    y_pred = theta[1]*x+theta[0]
    g0 = np.mean(2 * (y_pred-y))
    g1 = np.mean(2 * x * (y_pred-y))
    return np.array([g0,g1])
```

Code scipy avec le gradient (pour un résultat plus précis)

Code de la fonction de coût

```
def J(theta,x,y) :
    y_pred = theta[1]*x+theta[0]
    return np.mean((y-y_pred)**2)
```

Code du gradient de la fonction de coût

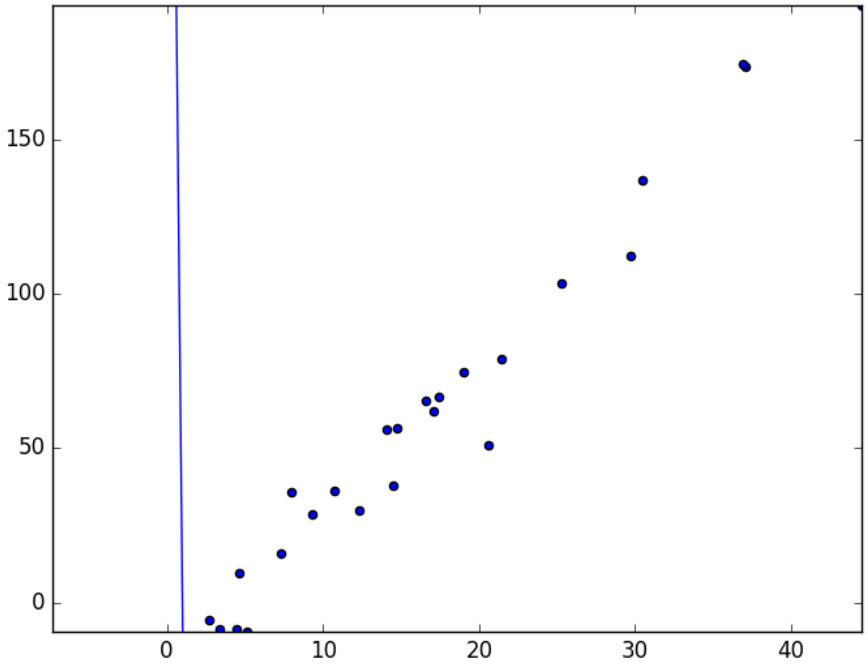
```
def gradJ(theta,x,y) :
    y_pred = theta[1]*x+theta[0]
    g0 = np.mean(2 * (y_pred-y))
    g1 = np.mean(2 * x * (y_pred-y))
    return np.array([g0,g1])
```

Résolution du problème d'optimisation

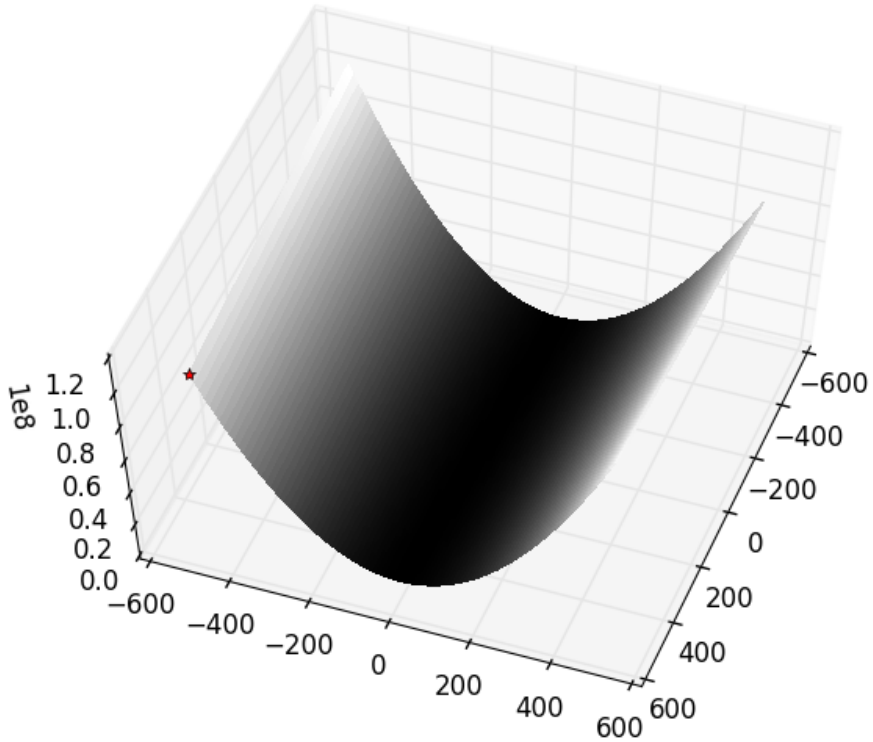
```
theta = scipy.optimize.minimize(lambda theta :J(theta,x,y),
        jac=lambda theta :gradJ(theta,x,y),
        x0=np.random.random(2),
        method='BFGS').x
```

Application de la descente de gradient à notre exemple

Évolution de la descente de gradient



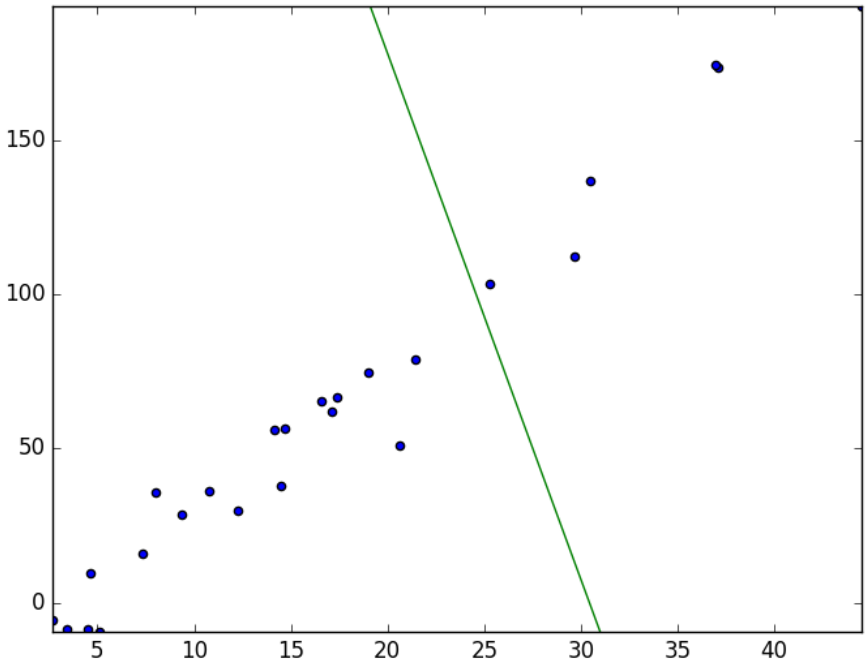
Droite θ =[-500,500]

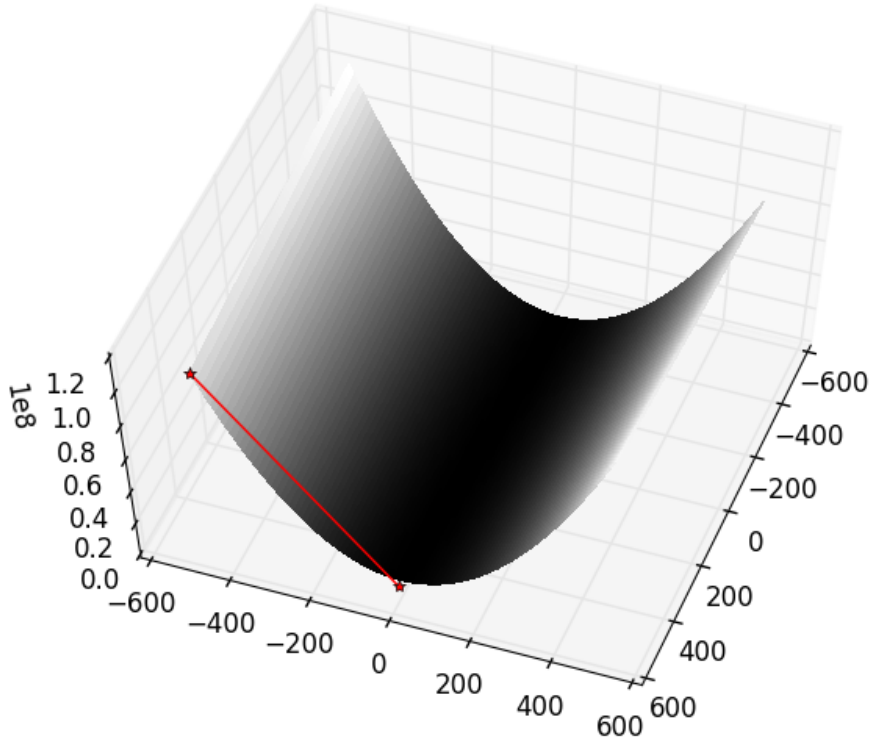


Fonction de coût :98081198.14

Application de la descente de gradient à notre exemple

Évolution de la descente de gradient

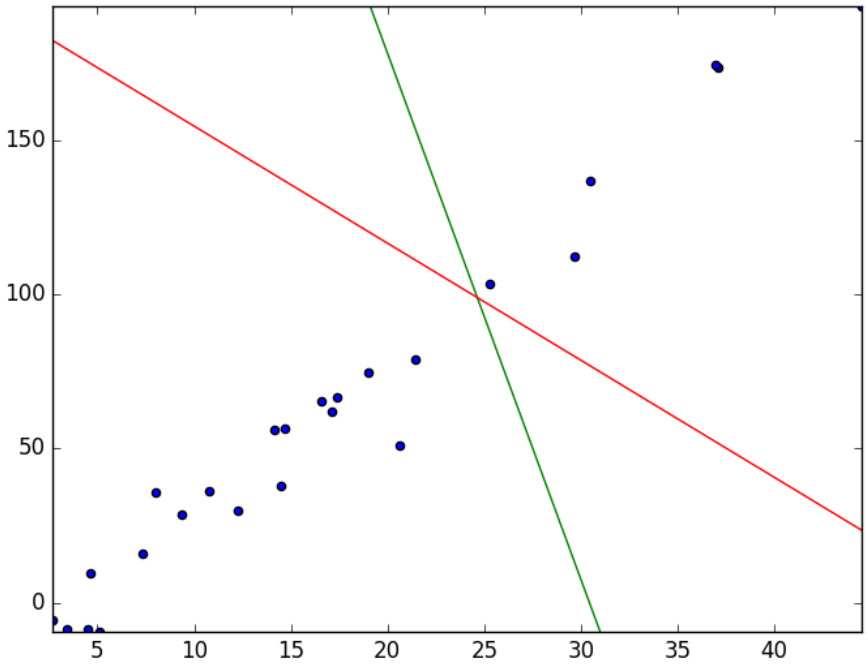


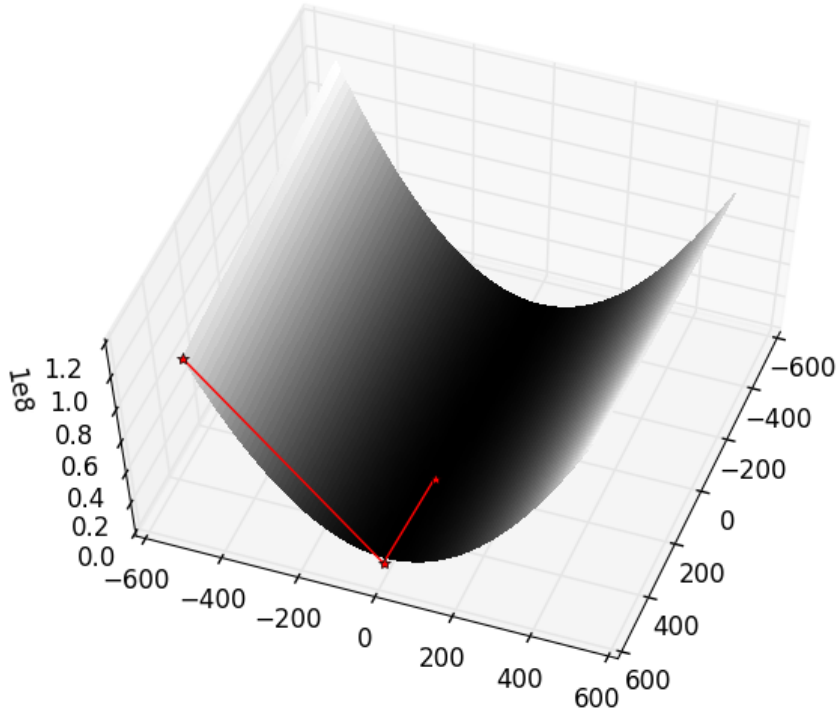


Droite θ = [519.33697608 -17.06301462] Fonction de coût : 88849.88

Application de la descente de gradient à notre exemple

Évolution de la descente de gradient

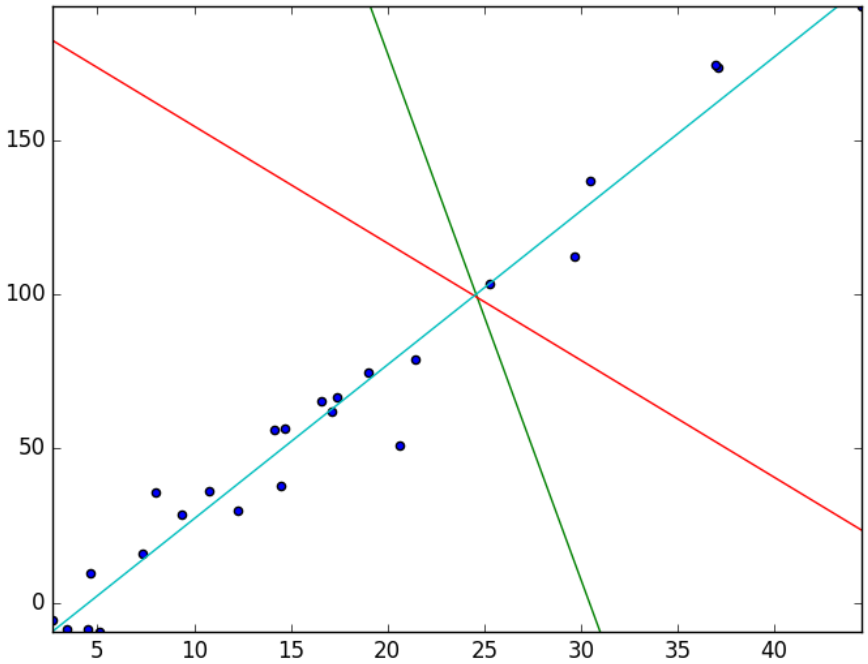


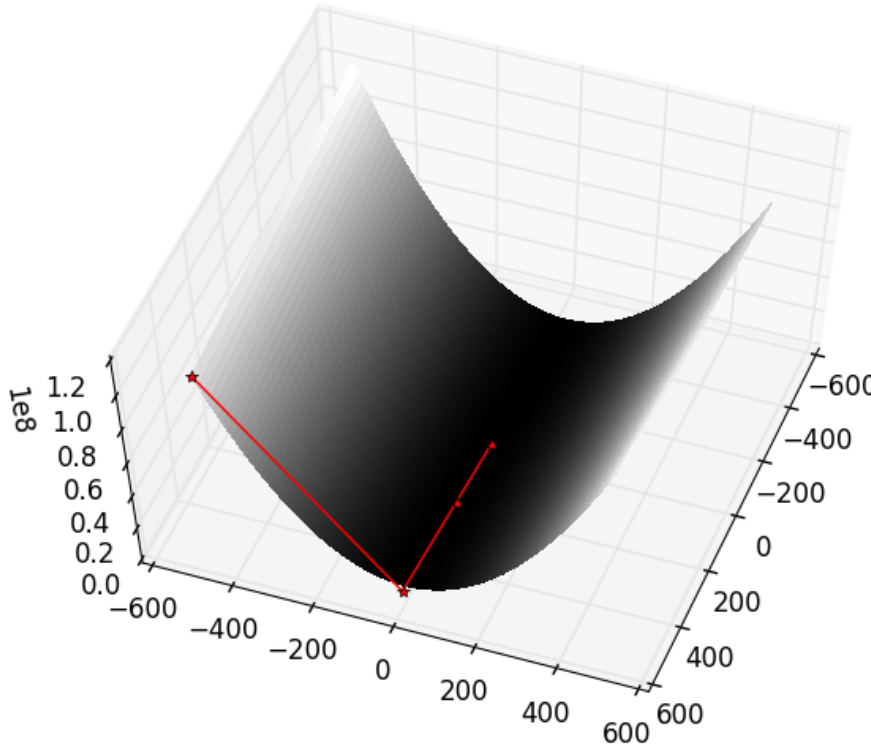


Droite $\theta =$ [192.44113047 -3.79594425] Fonction de coût : 14084.1017098

Application de la descente de gradient à notre exemple

Évolution de la descente de gradient

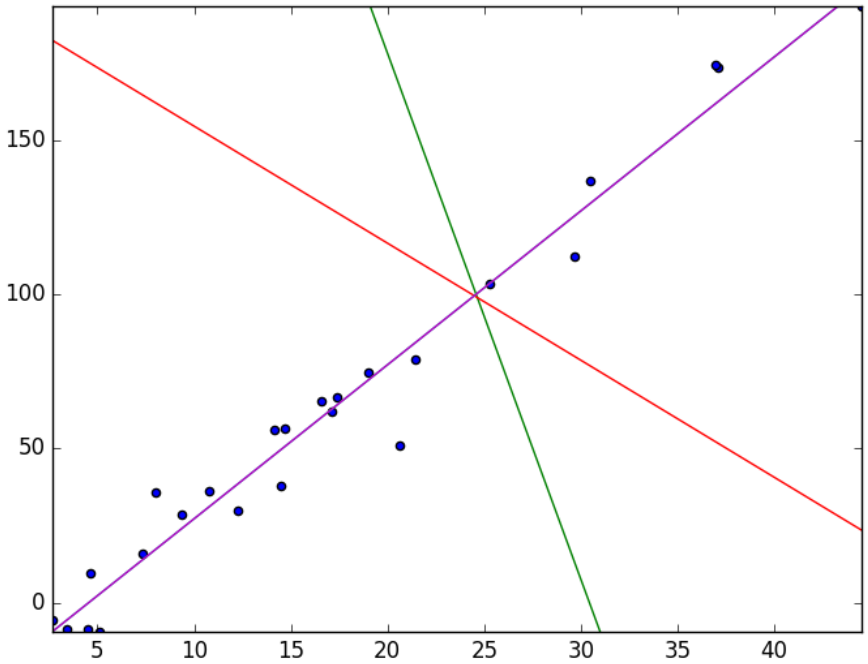


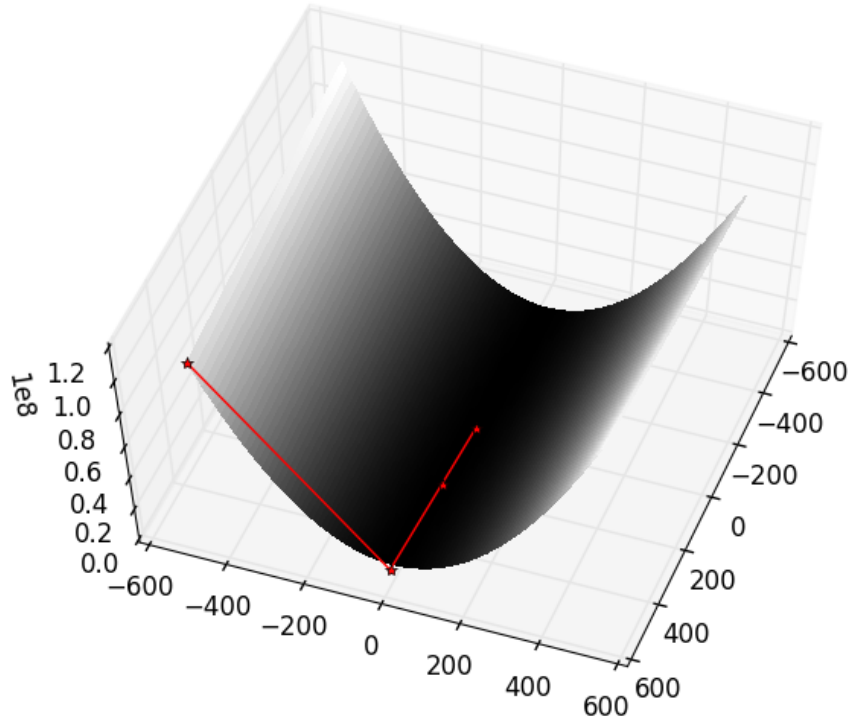


Droite $\theta =$ [-22.72835286 4.99215001] Fonction de coût : 100.90215406

Application de la descente de gradient à notre exemple

Évolution de la descente de gradient



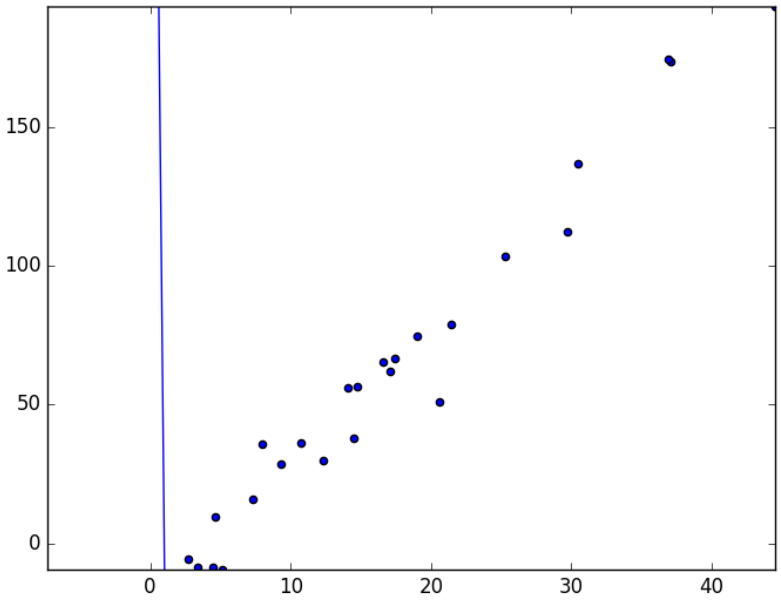


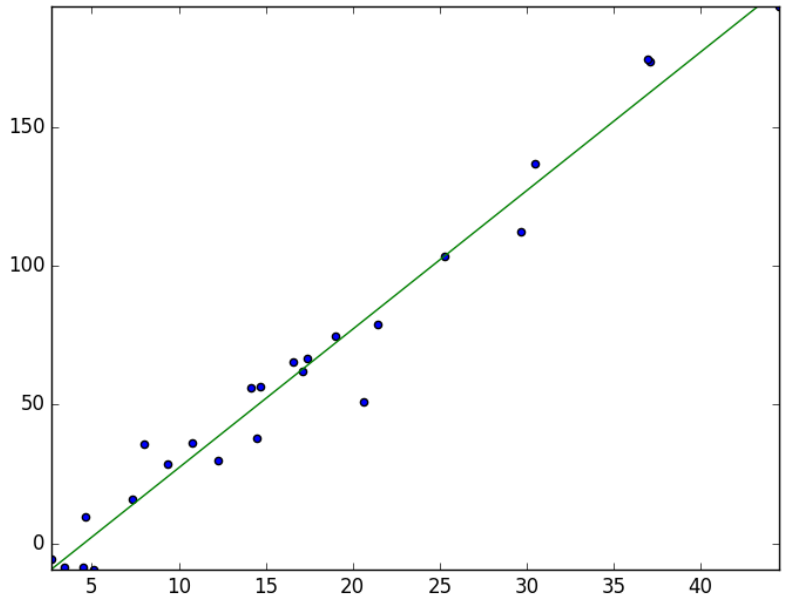
Droite $\theta =$ [-22.72625423 4.99206679] Fonction de coût : 100.90215273

Application de la descente de gradient à notre exemple

Solution finale

En partant d'une droite aléatoire, l'algorithme de descente de gradient à bien convergé vers une solution satisfaisante :





Équation finale de notre droite

$$\theta = \begin{bmatrix} -22.72625577 \\ 4.99206686 \end{bmatrix} \Longrightarrow y = 4.99206686\ x - 22.72625577$$

Utilisation de notre capteur de température

Utilisation du capteur calibré

Maintenant que nous connaissons la fonction qui relie la tension mesurée par notre dispositif à la température réelle nous pouvons effectuer des mesures avec notre capteur.

Exemple de mesure

- La tension mesurée est de 9v.
- La température est alors de $4.99206686 \times 9 - 22.72625577 = 22.20234597°C$.