Régression Linéaire

Régression Linéaire

- La régression linéaire est un algorithme d'apprentissage automatique supervisé dans lequel la sortie prédite est continue et présente une pente constante.

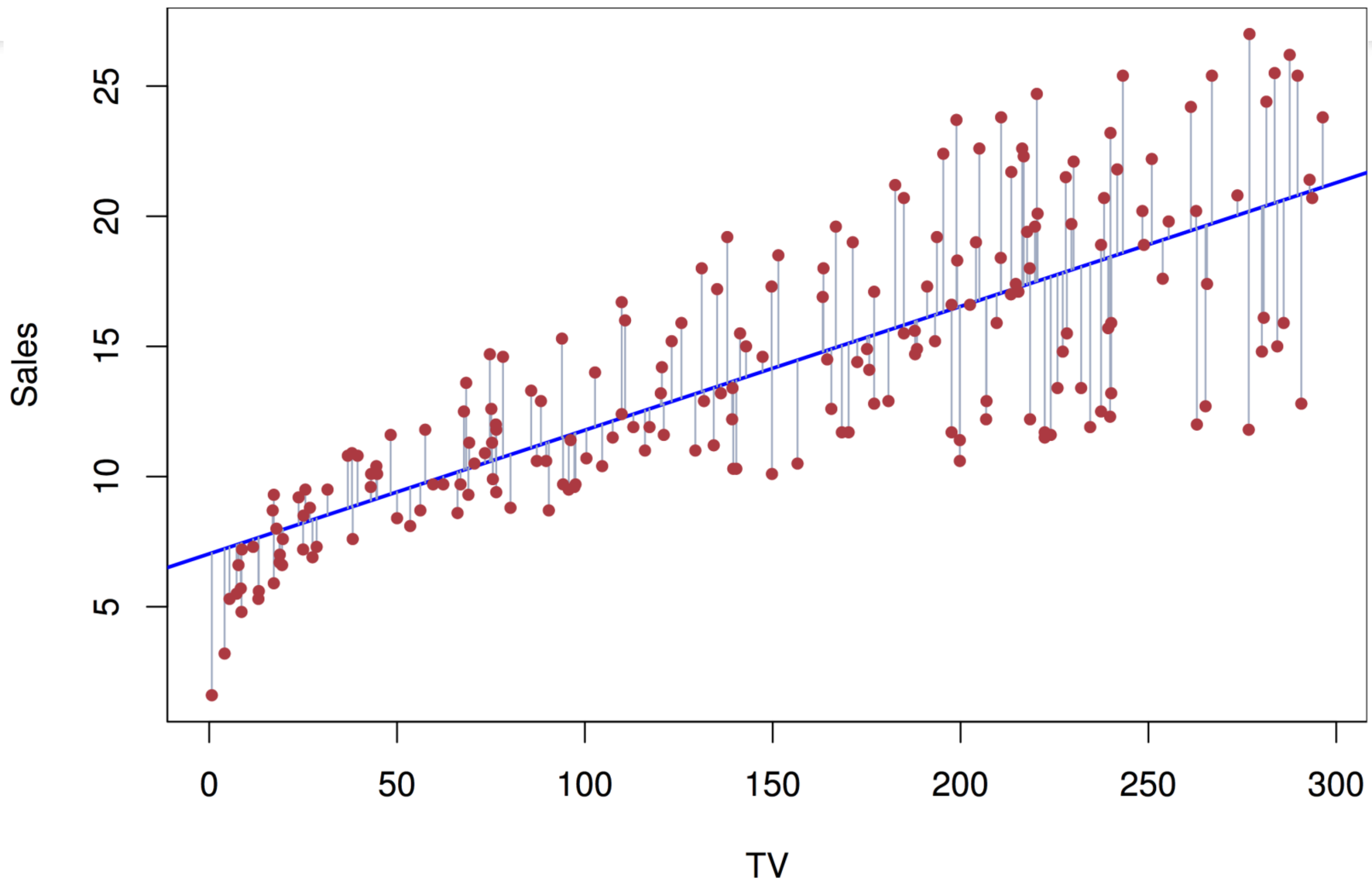
- Elle tente de trouver la meilleure relation linéaire possible entre les caractéristiques d'entrée et la variable cible (y).

- Algorithme d'apprentissage automatique **supervisé**

- Ajuste au mieux les données dont la variable cible (variable dépendante) est **une combinaison linéaire** des attributs d'entrée (variables indépendantes).

- Elle est utilisée pour prédire des valeurs continues (par exemple, ventes, prix) REGRESSION

Régression Linéaire



Régression Linéaire

Régression simple

- La régression linéaire simple utilise la forme traditionnelle d'une droite

$$\mathbf{y} = \boldsymbol{mx} + \boldsymbol{b}$$

- L'algorithme essaye d'apprendre les paramètres (poids) m et b
- Produire les prédictions les plus précises.
- X représente nos données d'entrée et Y représente notre prédiction.

Régression Linéaire multiple

Une équation linéaire multiple plus complexe pourrait ressembler à ceci, où représente les coefficients, ou poids, que notre modèle tentera d'apprendre.

$$\boldsymbol{f(x, y, z) = w1.x + w2.y + w3.z}$$

- Les variables représentent les attributs
- Les poids w_i représentent les coefficients à optimiser

Régression Linéaire simple

Entreprise	Pubs Radio (k$)	Unités vendus (k)
Amazon	37.8	22.1
Google	39.3	10.4
Facebook	45.9	18.3
Apple	41.3	18.5

Comment faire des prédictions ??

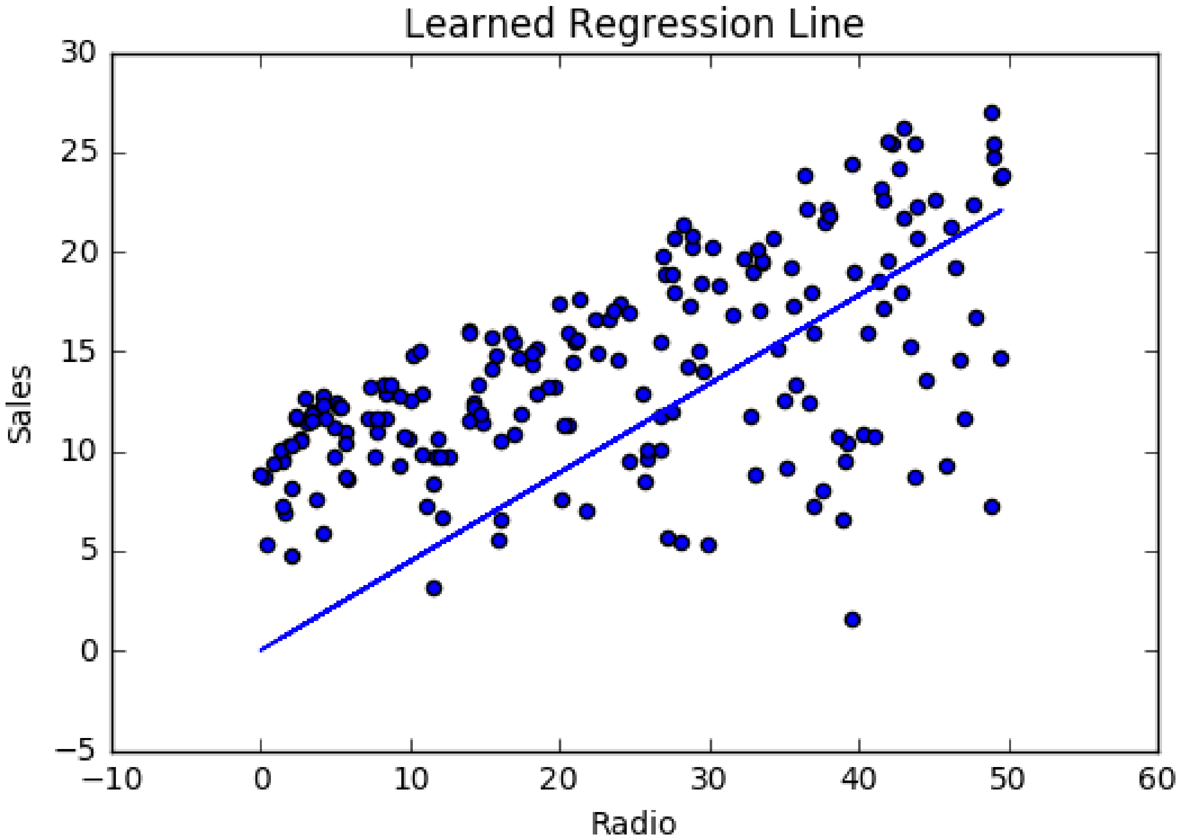
La régression linéaire va prédire le nombre d’unité vendus par une entreprise selon l’argent qu’elle a investi dans les pubs dans des Radios

$$Ventes = Poids\ .\ Radio + Biais$$

- Poids : Le coefficient (paramètres) pour la variable indépendante (pubs Radio) (appelé poids en ML)
- Radio: Représente la variable indépendante (Pubs dans Radio) (Appelée attributs «features » en ML)
- Biais: le point de où la ligne croise l'axe des y.

Régression Linéaire simple

- Le but de la régression linéaire est d'apprendre les paramètres (Poids et biais) optimales
- A la fin de l'apprentissage l'équation doit donner la droite qui approxime au mieux les données



Régression Linéaire simple

Erreur et fonction de coût

La fonction de coût est important afin de mesurer l'erreur de la fonction et optimiser les paramètres (poids, biais)

La fonction utilisée est généralement Mean Squared Error (MSE) « L2 »

$$MSE = \frac{1}{N}\sum_{i=1}^{n}\left(y_i - (m.x_i + b)\right)^2$$

- N est le nombre totale des observations
- $\frac{1}{N}\sum_{i=1}^{n}$ est la moyenne
- y_i est la sortie réelle d'une observation, tandis que $(m.x_i + b)$ est la sortie prédite

Régression Linéaire simple

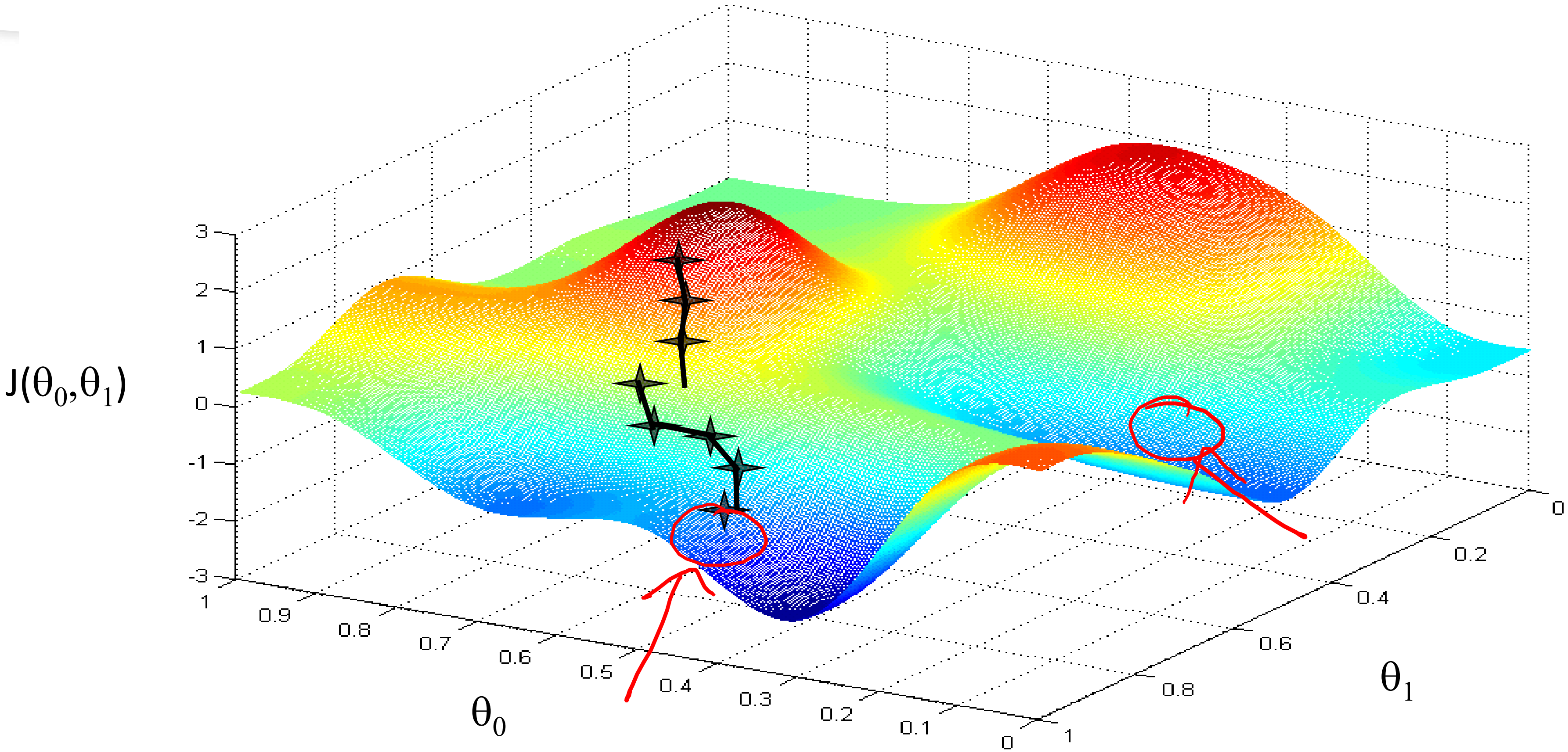
- Elle mesure la moyenne de la différence carrée entre la sortie observé et la sortie prédite

- MSE donne une valeur représentant le coût ou le score

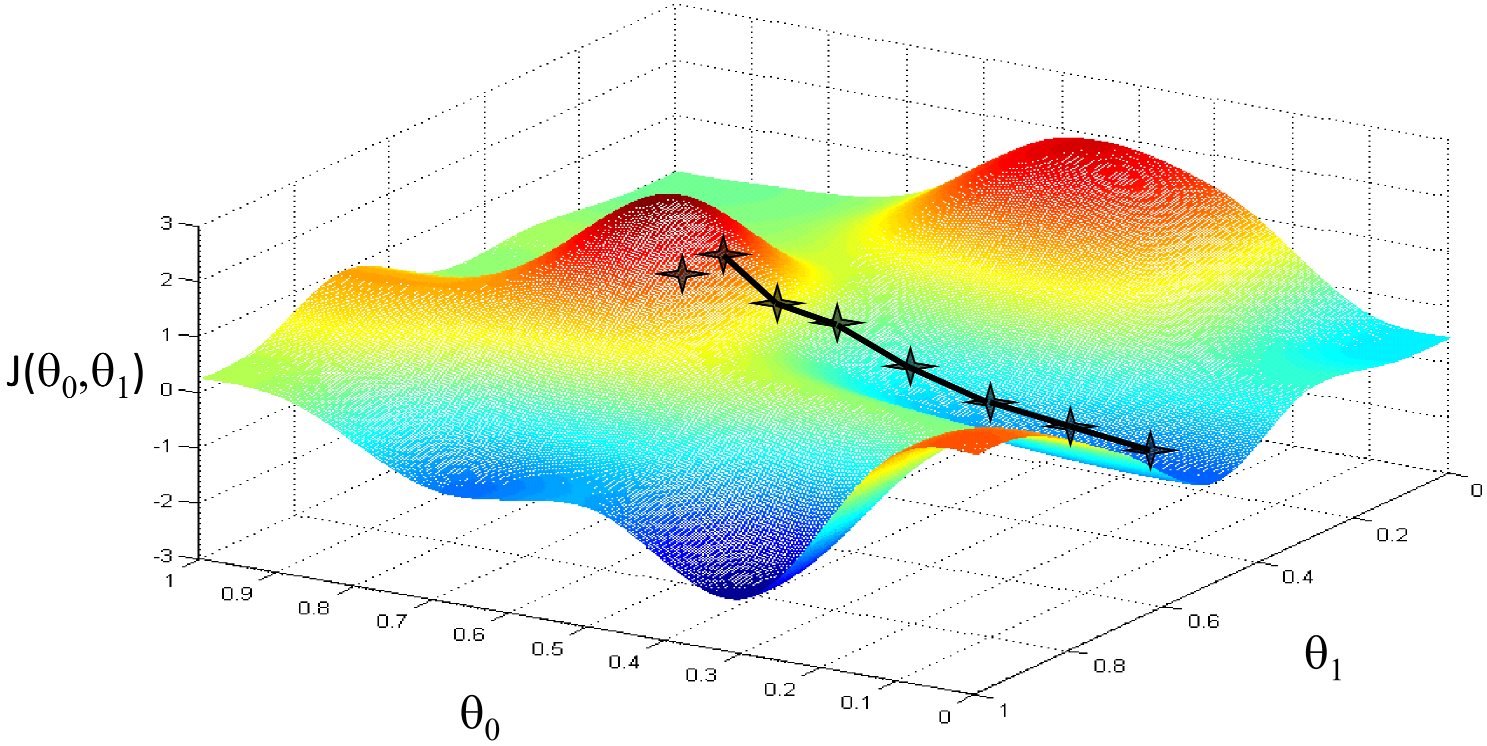
- L'apprentissage de l'algorithme consiste en la minimisation de la fonction coût afin d'optimiser les paramètres et améliorer le modèle

Comment la fonction de cout aide à optimiser les paramètres et améliorer le modèle ??

Optimisation : Descente du gradient



Optimisation : Descente du gradient



Optimisation : Descente du gradient

- On calcule les dérivé de la fonction de coût MSE par rapport aux paramètres m et b
- On obtient le gradient pour chaque paramètre
- On met à jour les paramètres
- On fait plusieurs itérations jusqu'à ce que l'erreur MSE soit minime
- On obtient le modèle qui approxime au mieux le jeu de données

Optimisation : Descente du gradient

$$J = \frac{1}{n} \sum_{i=1}^{n} (B_0 + B_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial B_0} = \frac{2}{n} \sum_{i=1}^{n} (B_0 + B_1 \cdot x_i - y_i)$$

$$\frac{\partial J}{\partial B_1} = \frac{2}{n} \sum_{i=1}^{n} (B_0 + B_1 \cdot x_i - y_i) \cdot x_i$$

$$J = \frac{1}{n} \sum_{i=1}^{n} (B_0 + B_1 \cdot x_i - y_i)^2$$

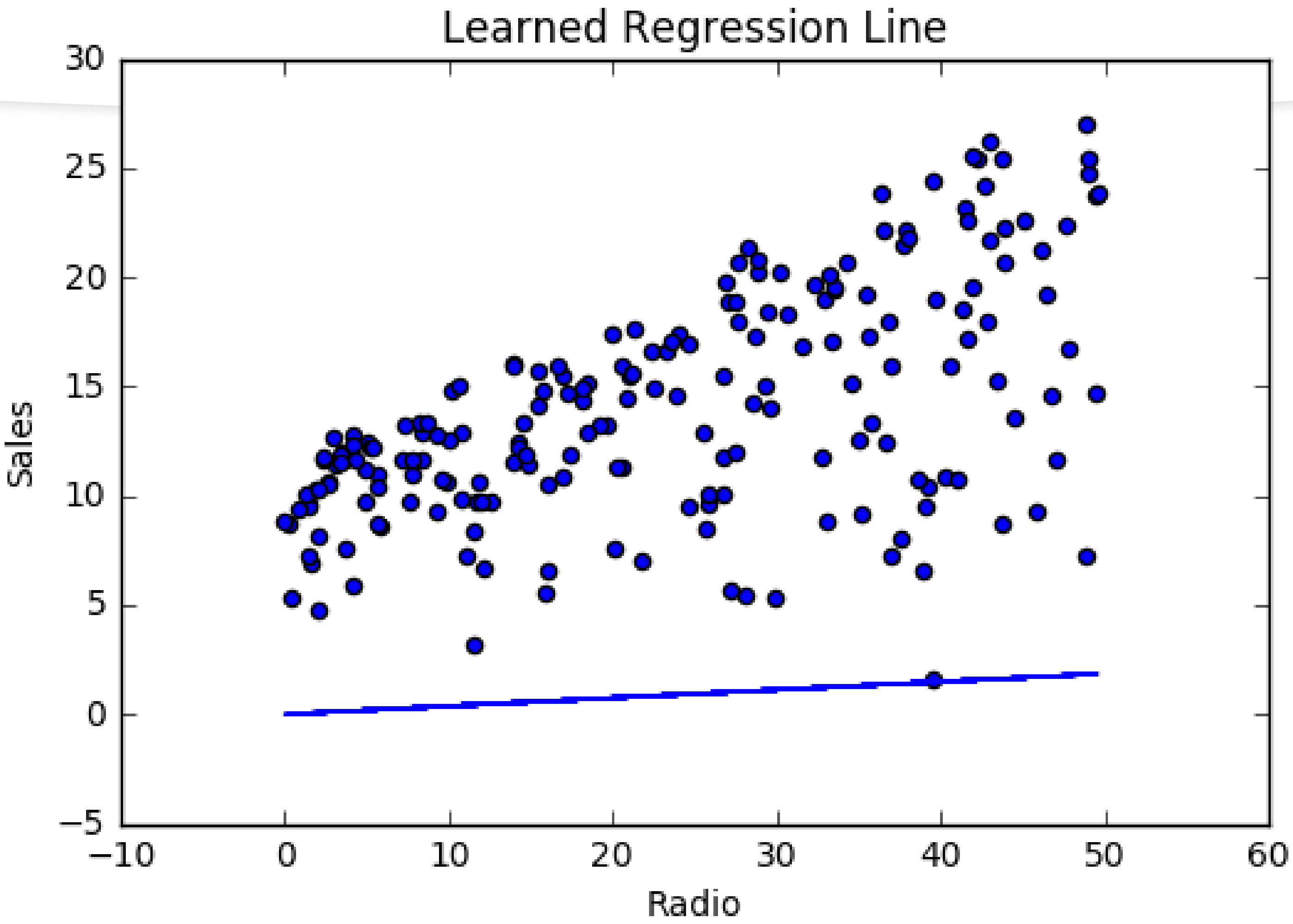
$$\frac{\partial J}{\partial B_0} = \frac{2}{n} \sum_{i=1}^{n} (B_0 + B_1 \cdot x_i - y_i)$$

$$\frac{\partial J}{\partial B_1} = \frac{2}{n} \sum_{i=1}^{n} (B_0 + B_1 \cdot x_i - y_i) \cdot x_i$$

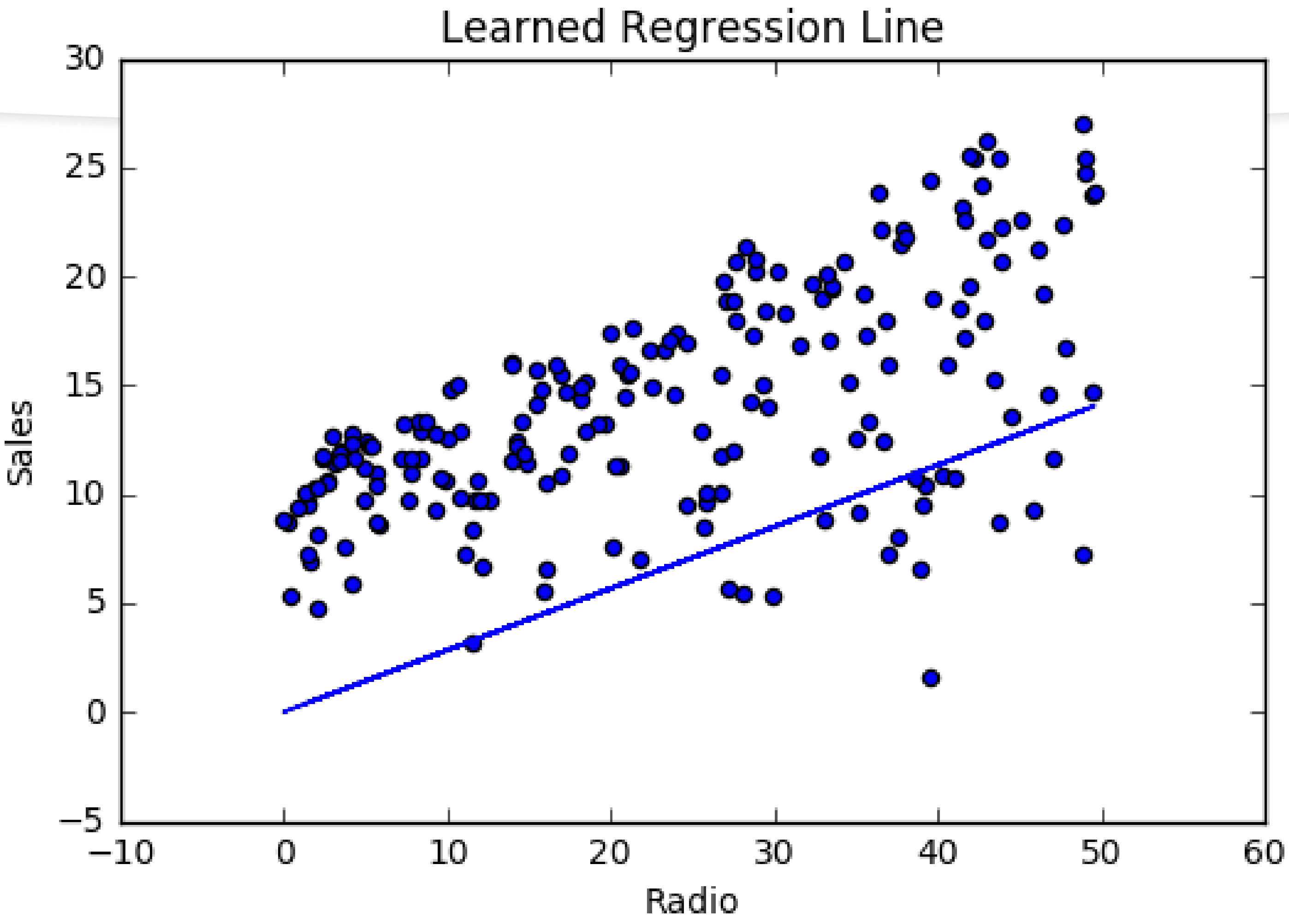
$$B_0 = B_0 - \alpha \cdot \frac{2}{n} \sum_{i=1}^{n} (pred_i - y_i)$$

$$B_1 = B_1 - \alpha \cdot \frac{2}{n} \sum_{i=1}^{n} (pred_i - y_i) \cdot x_i$$

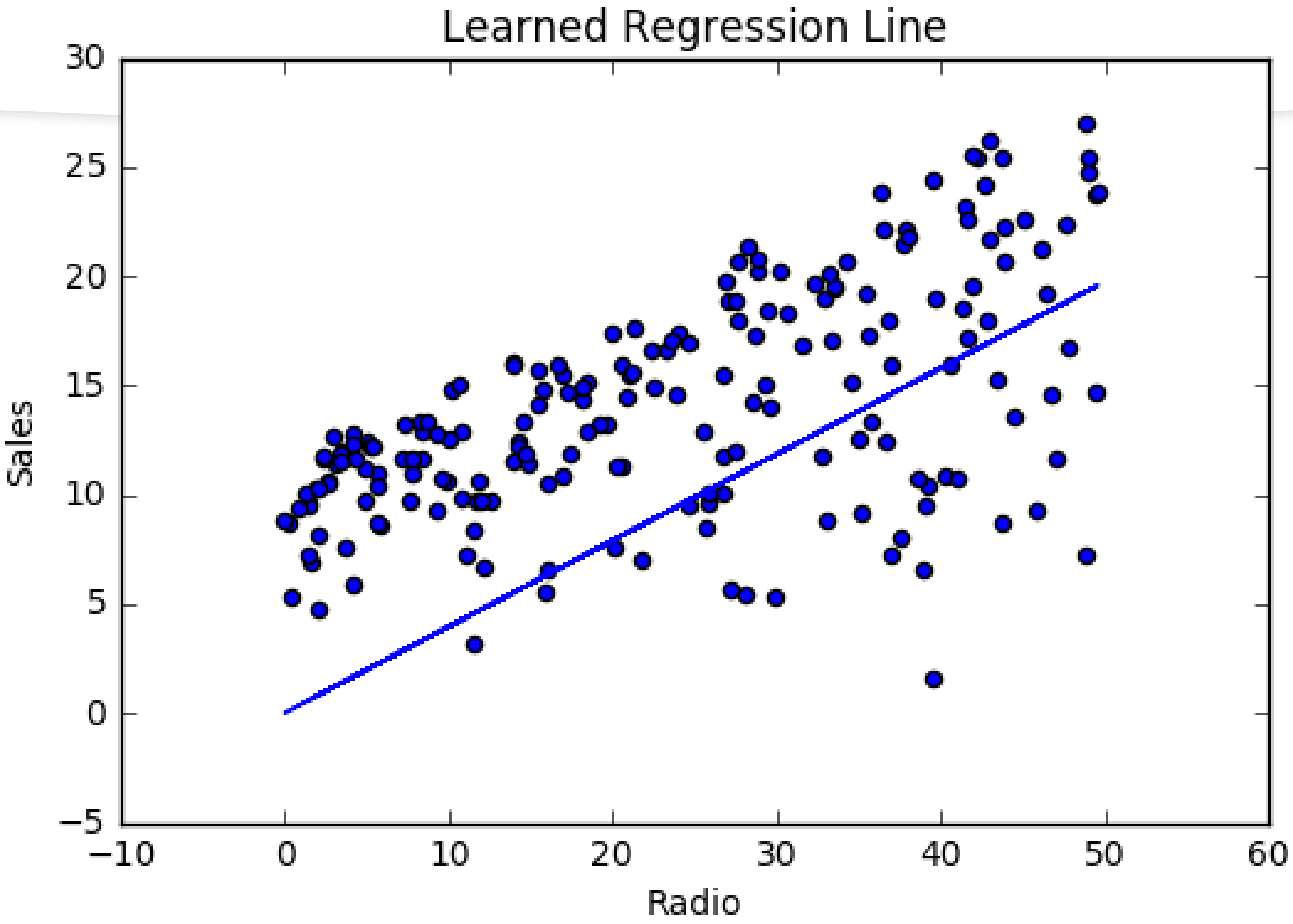
Optimisation : Descente du gradient



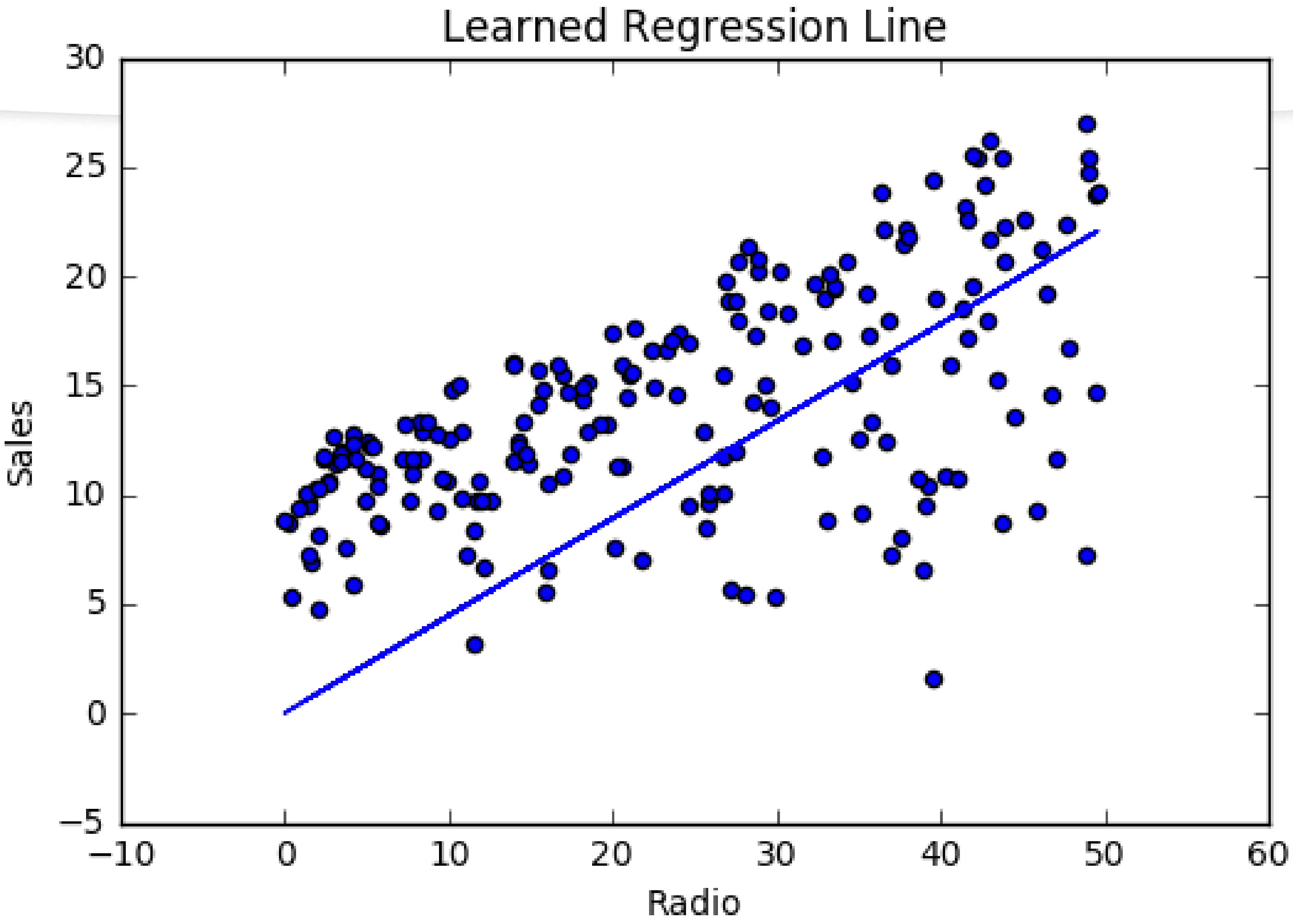
Optimisation : Descente du gradient



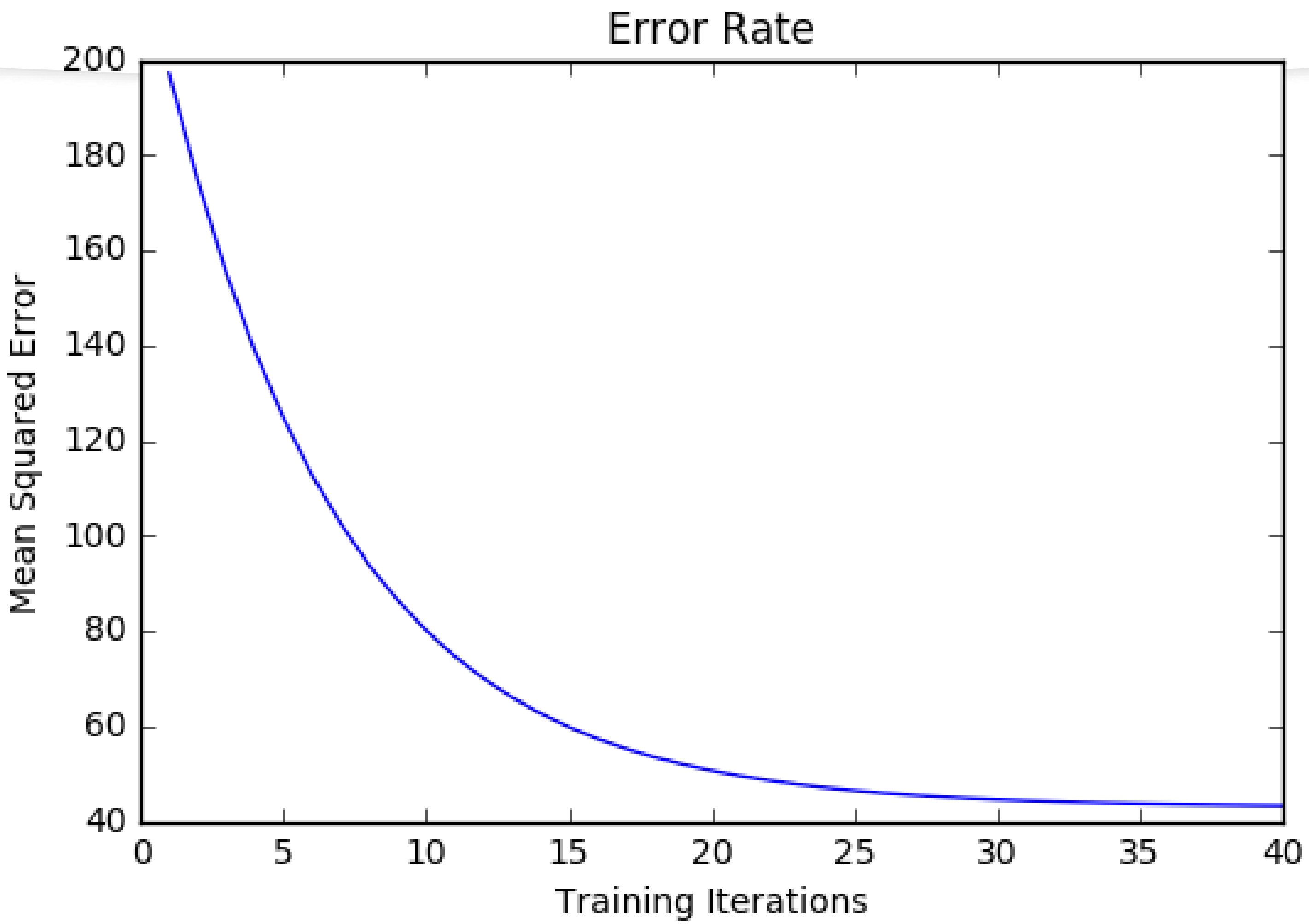
Optimisation : Descente du gradient



Optimisation : Descente du gradient



Optimisation : Descente du gradient



Prédictions

Après avoir fini l'apprentissage et évaluer le modèle

On obtient les valeurs de m et b

On peut faire des prédictions => INFERENCE

$$m = 0{,}46 \text{ et } b = 0{,}025$$

On peut maintenant donner à notre modèle l'argent investi en pubs Radio

Le modèle nous donnera une approximation du nombre d'unités vendus

Pubs Radio = 36.456 $ ➔ $X = 36.456$ ➔ $Y = mX + b = 0{,}46\,(36.456) + 0{,}025$

$$Y \approx 17.1\, milles\ unités\ vendus$$

Régression Linéaire Multiple

Company	TV	Radio	News	Units
Amazon	230.1	37.8	69.1	22.1
Google	44.5	39.3	23.1	10.4
Facebook	17.2	45.9	34.7	18.3
Apple	151.5	41.3	13.2	18.5

$$Ventes = w1.TV + w2.Radio + W3.News + Biais$$

Régression Linéaire Multiple

Sur-apprentissage
Lorsque de plus en plus de variables sont ajoutées à un modèle, celui-ci peut devenir beaucoup trop complexe et finit généralement par mémoriser tous les points de données de l'ensemble d'apprentissage.

Ce phénomène est connu sous le nom de sur-apprentissage d'un modèle. Il en résulte généralement une grande précision d'apprentissage et une très faible précision de test.

Multicollinéarité: Il s'agit du phénomène par lequel un modèle comportant plusieurs variables indépendantes peut avoir certaines variables liées entre elles. (corrélation)

Sélection des caractéristiques : En présence d'un plus grand nombre de variables, la sélection de l'ensemble optimal des attributs à partir des caractéristiques données (dont beaucoup peuvent être redondantes) devient une tâche importante pour la construction d'un modèle pertinent et de meilleure qualité.