



**Dr. Christian Tsoungui Obama**

**Fiche Mémo ML #3**

# **Approfondissement : Maîtriser la Régression**

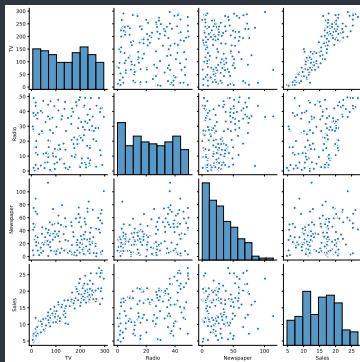
Guide fondamental pour aspirants Data Scientists

Basé sur  
"Hands-On Machine Learning with Scikit-learn, Keras & TensorFlow"  
par

Aurélien Géron

## « Tous les modèles sont faux, mais certains sont utiles » George P. Box

- Construire un modèle d'apprentissage automatique (ML) revient à « tenter » de trouver la relation « inconnue » entre la variable cible et les caractéristiques des données (features).
- Les modèles sont construits en faisant des « hypothèses » sur ce que pourrait être la « vérité » basée sur les données disponibles.
- Dans les problèmes de régression, visualiser les nuages de points entre la cible et chaque caractéristique permet de formuler des hypothèses sur le modèle réel.



## Les modèles de Régression Linéaire supposent une relation linéaire

| Ventes | TV    | Radio | Journal |
|--------|-------|-------|---------|
| 22.1   | 230.1 | 37.8  | 69.2    |
| 10.4   | 44.5  | 39.3  | 45.1    |
| ⋮      |       |       |         |
| 18.4   | 232.1 | 8.6   | 8.7     |

$\underbrace{\hspace{1.5cm}}_{\mathbf{y}} \quad \underbrace{\hspace{4.5cm}}_{\mathbf{X}}$

Modèle de régression linéaire pour la cible  $\mathbf{y}$  et les caractéristiques  $\mathbf{X} = (x_1, x_2, \dots, x_L)$  :

$$\mathbf{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_L x_L,$$

où  $\Theta = (\theta_0, \theta_1, \theta_2, \dots, \theta_L)$  sont les paramètres inconnus. Pour le jeu de données publicitaires, le modèle est :

$$\text{Ventes} = \theta_0 + \theta_1 \text{TV} + \theta_2 \text{Radio} + \theta_3 \text{Journal}.$$

## Entraîner un modèle signifie estimer les meilleures valeurs de $\theta_0, \theta_1, \dots, \theta_L$

- Le modèle réel estime Ventes = 22.1 pour TV = 230.1, Radio = 37.8, Journal = 69.2, etc.
- **Attention : il est impossible de connaître le modèle réel avec certitude.** Le meilleur modèle est celui qui prédit des valeurs de ventes les plus proches des valeurs réelles.
- Le meilleur modèle possible minimise l'erreur sur tous les échantillons, c'est-à-dire l'erreur quadratique moyenne (EQM) calculée ainsi :

$$EQM(\Theta) = \frac{1}{N} \sum_{i=1}^N \left( \hat{y}^{(i)} - y^{(i)} \right)^2,$$

où  $N$  est la taille de l'échantillon et  $\hat{y}^{(i)}$  l'estimation de la cible pour le  $i$ -ème échantillon.

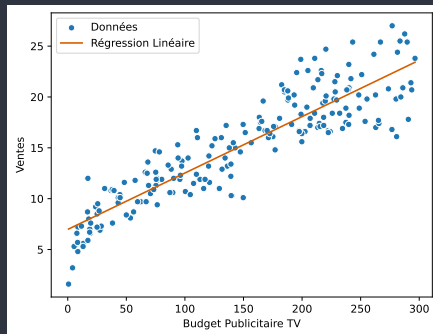
# Estimation des meilleurs paramètres $\hat{\Theta}$ dans un modèle linéaire

**Équation Normale** : Formule exacte pour minimiser la EQM

$$\hat{\Theta} = (X^T X)^{-1} X^T y,$$

où  $X^T$  est la transposée de  $X$ .

```
1 # Implementation de l'Equation Normale
2 import numpy as np
3
4 def normal_equation_fit(X, Y):
5     X_exp = np.c_[np.ones((X.shape[0], 1)), X] #
6     # Ajout d'un echantillon pour l'estimation de
7     # theta_0
8     return np.linalg.inv(X_exp.T.dot(X_exp)).dot(
9         X_exp.T.dot(Y))
10
11 # Estimation des meilleurs parametres du modele
12 # pour Sales and TV data
13 best_parameters = normal_equation_fit(adsSales[["
14     TV"]], adsSales[["Sales"]])
15
16 best_parameters
```



Visualisation : estimations des paramètres  $\hat{\theta}_0 = 6.975$ ,  $\hat{\theta}_1 = 0.055$

## Équation Normale optimisée avec Scikit-learn

- L'Équation Normale peut échouer si le nombre de caractéristiques  $L$  est grand par rapport à  $N$  ou en cas de multicolinéarité.
- Généralement lente pour les grands jeux de données.
- Scikit-learn résout cela en utilisant la **pseudo-inverse de Moore-Penrose**  $\mathbf{X}^+ = (\mathbf{U}\Sigma\mathbf{V}^T)^+$  de  $\mathbf{X}$ , soit :

$$\hat{\Theta} = \mathbf{X}^+ \mathbf{y},$$

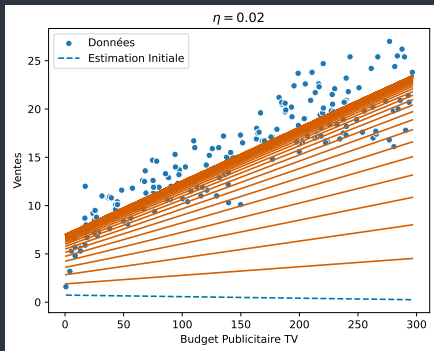
```
1 # Entraînement et valuation sur l'ensemble de formation : Regression Lineaire
2 from sklearn.linear_model import LinearRegression
3
4 lin_reg = LinearRegression()
5 lin_reg.fit(adsSales[["TV"]], adsSales[["Sales"]])
6
7 lin_reg.intercept_, lin_reg.coef_
8
```

## Approche numérique : Descente de Gradient par Batch (BGD)

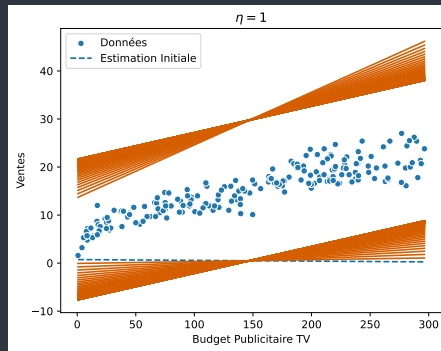
- Pour les grands jeux de données, l'Équation Normale est inefficace.
- La BGD trouve les meilleurs paramètres  $\hat{\Theta}$  du modèle de manière itérative :

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta \frac{2}{N} \mathbf{X}^T (\mathbf{X}\Theta^{(t)} - \mathbf{y}).$$

- Un **taux d'apprentissage**  $\eta$  trop petit ralentit la convergence.
- Un  $\eta$  trop grand fait diverger l'algorithme.



$\eta$  trop petit : le modèle est trouvé mais cela prend trop de temps



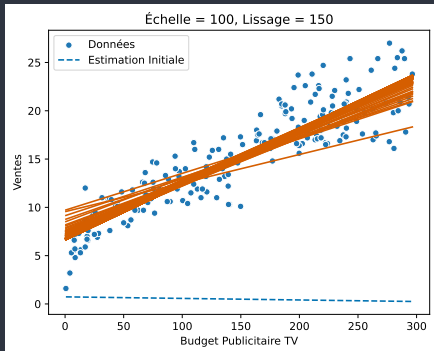
$\eta$  trop grand : le meilleur modèle risque de ne pas être trouvé

## Descente de Gradient Stochastique (SGD)

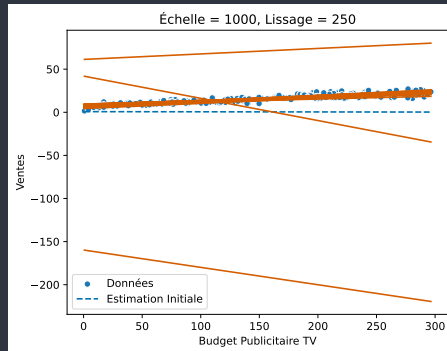
- La BGD peut rester bloquée dans un minimum local.
- La SGD peut s'en échapper en s'entraînant sur un seul échantillon aléatoire par itération :

$$\boldsymbol{\Theta}^{(t+1)} = \boldsymbol{\Theta}^{(t)} - \eta 2 \mathbf{X}_{(i)}^T (\mathbf{X}^{(i)} \boldsymbol{\Theta}^{(t)} - \mathbf{y}^{(i)}).$$

- La SGD est idéale pour les grands jeux de données et l'apprentissage en ligne.
- Elle peut osciller et ne jamais se stabiliser exactement au minimum.



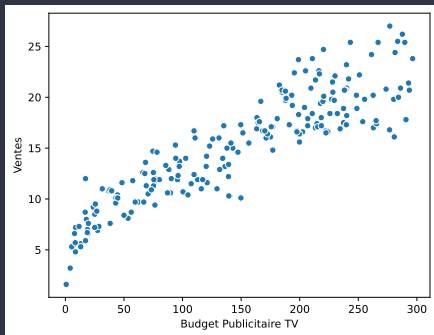
**Convergence rapide** : la SGD trouve le modèle relativement vite



**Comportement oscillatoire** : la SGD rebondit avant de se stabiliser



## Et si le nuage de points ressemble à une courbe ?



Ventes vs TV : le nuage de points ressemble à une courbe, pas une droite

### Modèle de régression polynomiale :

Pour les courbes, un polynôme de degré  $d$  pourrait être plus adapté :

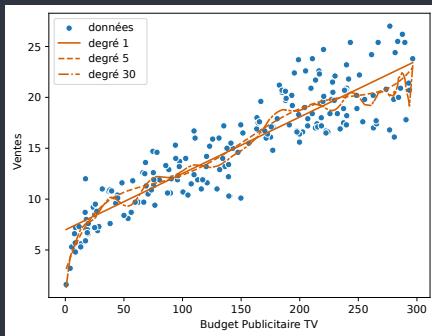
$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \dots + \theta_d x_1^d,$$

où  $\Theta$  sont les paramètres. Exemple :

$$\text{Ventes} = \theta_0 + \theta_1 \text{TV} + \theta_2 \text{TV}^2 + \dots + \theta_d \text{TV}^d.$$

## Modèles complexes : Surapprentissage vs Sous-apprentissage

- Les modèles simples (ex : linéaires) peuvent rater des motifs importants : c'est le **sous-apprentissage (underfitting)**.
- Les modèles complexes (ex : polynôme de haut degré) collent trop aux données et prédisent mal les nouvelles données : c'est le **surapprentissage (overfitting)**.
- **Compromis** : Les modèles complexes sont flexibles mais risqués ; les modèles simples sont rigides mais limités.



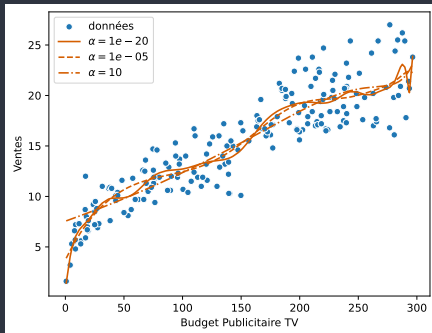
**Complexité** : La ligne est trop simple, le polynôme degré 30 trop complexe

## Régularisation : La Régression Ridge

- On peut contraindre un modèle complexe en encourageant des valeurs de  $\Theta$  petites.
- La **Régularisation** minimise la norme de  $\Theta$ .
- **Régression Ridge** : utilise la norme  $L_2$  de  $\Theta$ , minimisant :

$$EQM(\Theta) + \alpha \frac{1}{2} \sum_{k=1}^L \theta_k^2,$$

où  $\alpha$  contrôle l'intensité de la régularisation.

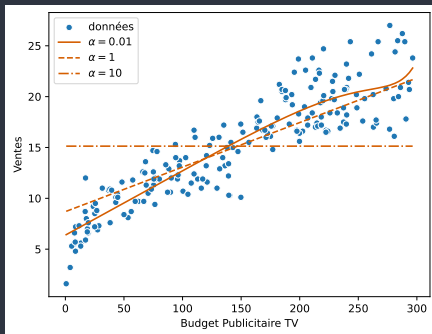


Effet de  $\alpha$  (Ridge) : le modèle se simplifie à mesure que  $\alpha$  augmente

# Régression LASSO

- Contrairement à Ridge, la **régression LASSO** peut annuler certains paramètres (les mettre à zéro).
- Cela permet une **sélection automatique de variables**.
- LASSO utilise la norme  $L_1$  de  $\Theta$  et minimise :

$$EQM(\Theta) + \alpha \sum_{k=1}^L |\theta_k|.$$



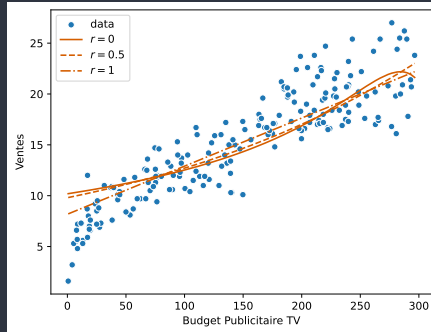
Effet de  $\alpha$  (LASSO) : le modèle finit par se réduire à une constante  $\theta_0$

## Elastic Net

- **Elastic Net** combine Ridge et LASSO avec un ratio de mélange  $r$  :

$$EQM(\Theta) + r\alpha \sum_{k=1}^L |\theta_k| + \frac{1-r}{2}\alpha \sum_{k=1}^L \theta_k^2.$$

- $r = 0$  correspond à Ridge et  $r = 1$  à LASSO.



Effet du ratio  $r$  : similarité entre les modèles Ridge et Elastic Net régularisés


## Ce qu'il faut retenir


- Évitez la régression linéaire simple (sans régularisation).
- Préférez Ridge si vous n'avez pas besoin de sélectionner des variables.
- Préférez Elastic Net à LASSO si certaines caractéristiques sont suspectées inutiles.


| Modèles                     | Hyperparamètres      | Avantages                             | Inconvénients   |
|-----------------------------|----------------------|---------------------------------------|---|
| Régression Linéaire         | Aucun                | Solution exacte, déterministe, rapide | Sensible aux valeurs aberrantes & à la multicollinéarité            |
| Descente de Gradient (BGD)  | Taux d'apprentissage | Itératif, plus rapide sur grands jeux | Peut bloquer sur un minimum local<br>Lent pour les très grands jeux |
| Gradient Stochastique (SGD) | Taux d'apprentissage | Très rapide, apprentissage en ligne   | Approximatif, nécessite un planning d'apprentissage, oscille        |




**Avez-vous trouvé cela utile ?**


 **Notebook complet :** » [Dépôt GitHub \(cliquez ici\)](#) «

 **Commentez :** Pouvez-vous expliquer pourquoi la SGD oscille ?

 **Suivez-moi :** pour la prochaine fiche de cette série.

 **Aimez ce post :** pour m'aider à toucher plus de monde.

**Restons connectés !**

 [/in/christiantsoungui](#)    [@TsounguiChris](#)