agaetis

Data Science & Big Data

Introduction à l'apprentissage automatique: La régression linéaire

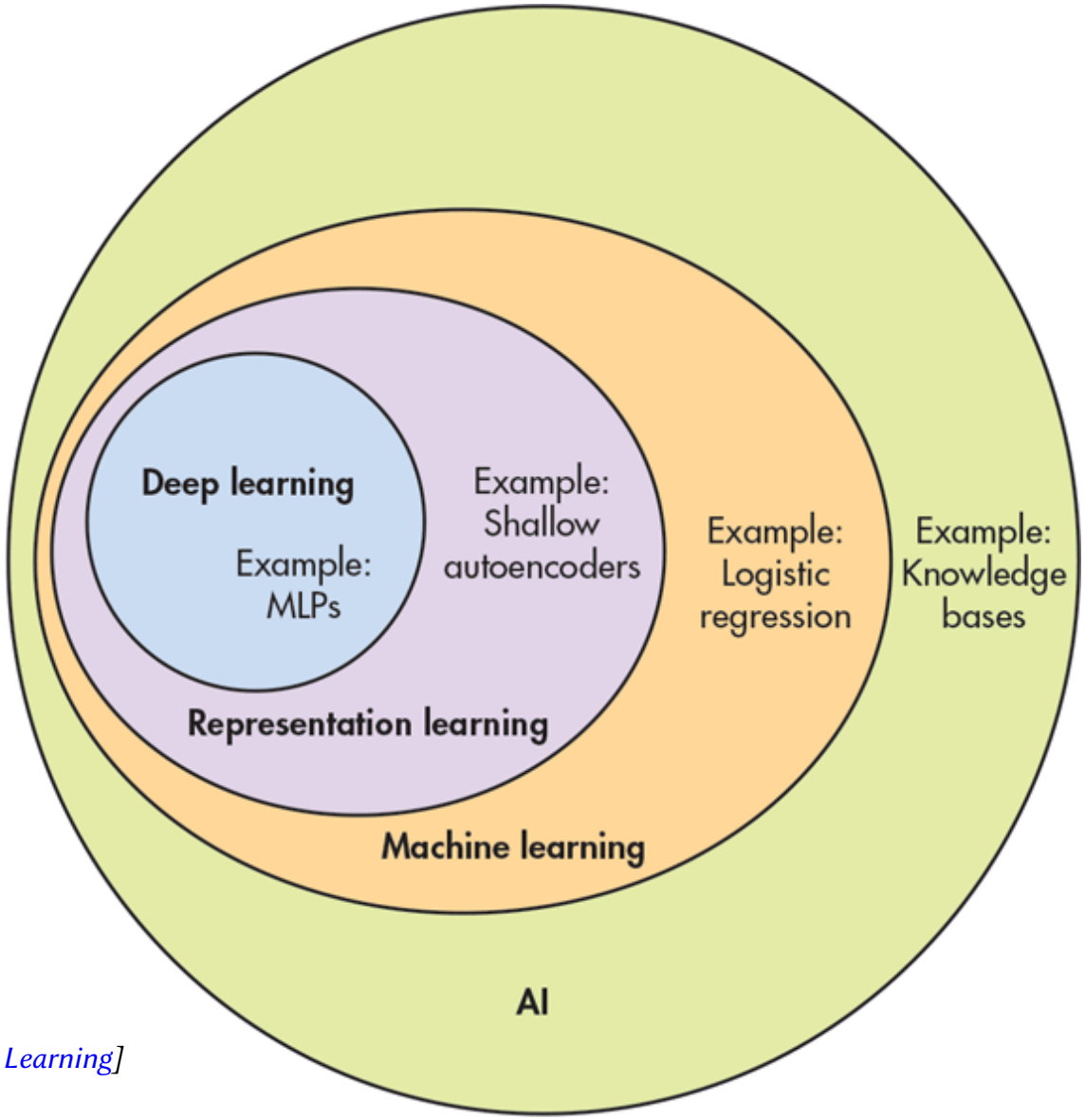
Léo Beaucourt

Pourquoi la régression linéaire?

- La régression linéaire: le "*Hello world!*" du *ML*
- Résolution d'un problème de Data science: *Prédiction d'un prix*
- En pratique: *Python*, *Jupyter*. Packages *numpy* *pandas* et *matplotlib*.
- Pas de (trop) de math ...

Allez, on démarre en douceur ...

Psycho killer, qu'est ce que c'est?



[From MIT Press book Deep Learning]

Le *Machine Learning*, qu'est ce que c'est?

- **Arthur Samuel:**
  - *The field of study that gives computers the ability to learn without being explicitly programmed.*
- **Tom Mitchell:**
  - *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

- **L'idée:** Une machine apprend *seule* à réaliser une tache complexe à l'aide de processus itératifs simple.

Les principaux types d'apprentissage

Supervisé

- Utilise des données *labélisées*
- La machine apprend par l'exemple
- *Prédis* le résultat pour de nouveaux événements
- Problèmes de régression et de classification
- Regression linéaire et logistique
- Réseaux de Neurones
- Arbres de décisions

Non-supervisé

- Données non *labélisées*
- La machine apprend par elle même à indentifier une structure
- Évaluation des performances compliqué
- Problèmes de classification, réduction de dimensions
- K-means
- Analyse en Composante Principale

Par renforcement

- Un agent A, effectue une action Ac, l'environnement E lui renvoie une récompense.
- Récompenses à court et long terme
- Utilisé par Deepmind (alphaGo)

La regression linéaire

- Déterminer une relation *linéaire* entre *input(s)* (features) et *output*:

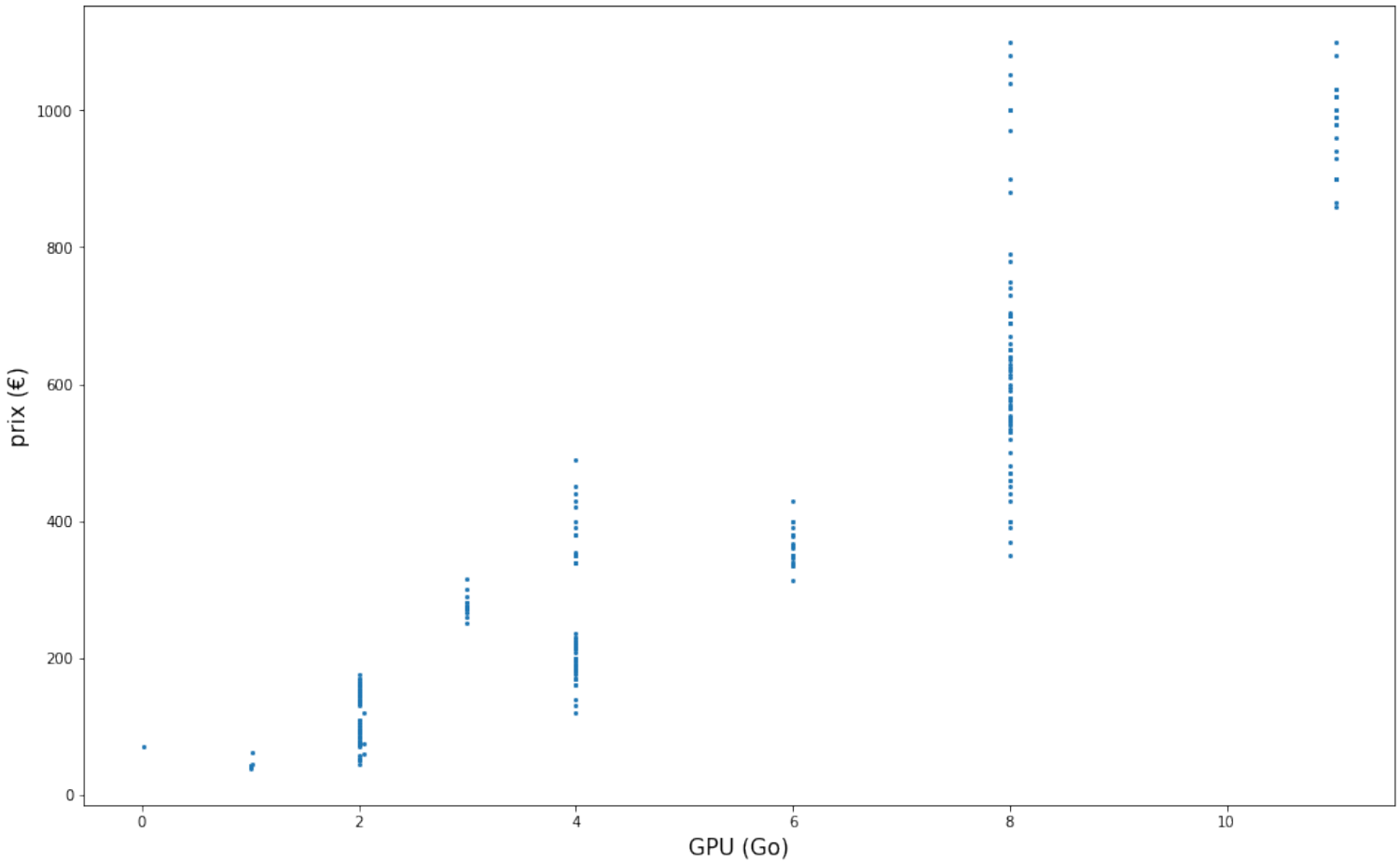
$$\Rightarrow \textbf{Apprentissage Supervisé}$$

- Prédiction d'une valeur **continue** (e.g. non discrète, non catégorielle)
- Applications:
  - Recherche de corrélations
  - En science, modélisation de phénomènes (physiques, biologiques, ...) après mesures
  - Dans le domaine médical: les études épidémiologique
  - Dans la finance/économie: prédictions des tendances, *Capital Asset Pricing Model*
  - ...

Sujet Data Science $\Rightarrow$ Premier algorithme à tester!

Un exemple: le prix d'une carte graphique

- La propriété principale d'une carte Graphique: valeur de **GPU**
- On a un jeu de données, cad une liste de carte graphiques dont on connait le couple $\{GPU; prix\}$:



Construire un modèle (regression linéaire)

- Soit: x_1 la valeur de GPU de nos m carte graphiques, et y leur prix
- On cherche à déterminer le modèle pour prédire un prix $\hat{y}$ à partir x_1:

$$\hat{y} = h_\theta(x_1)$$

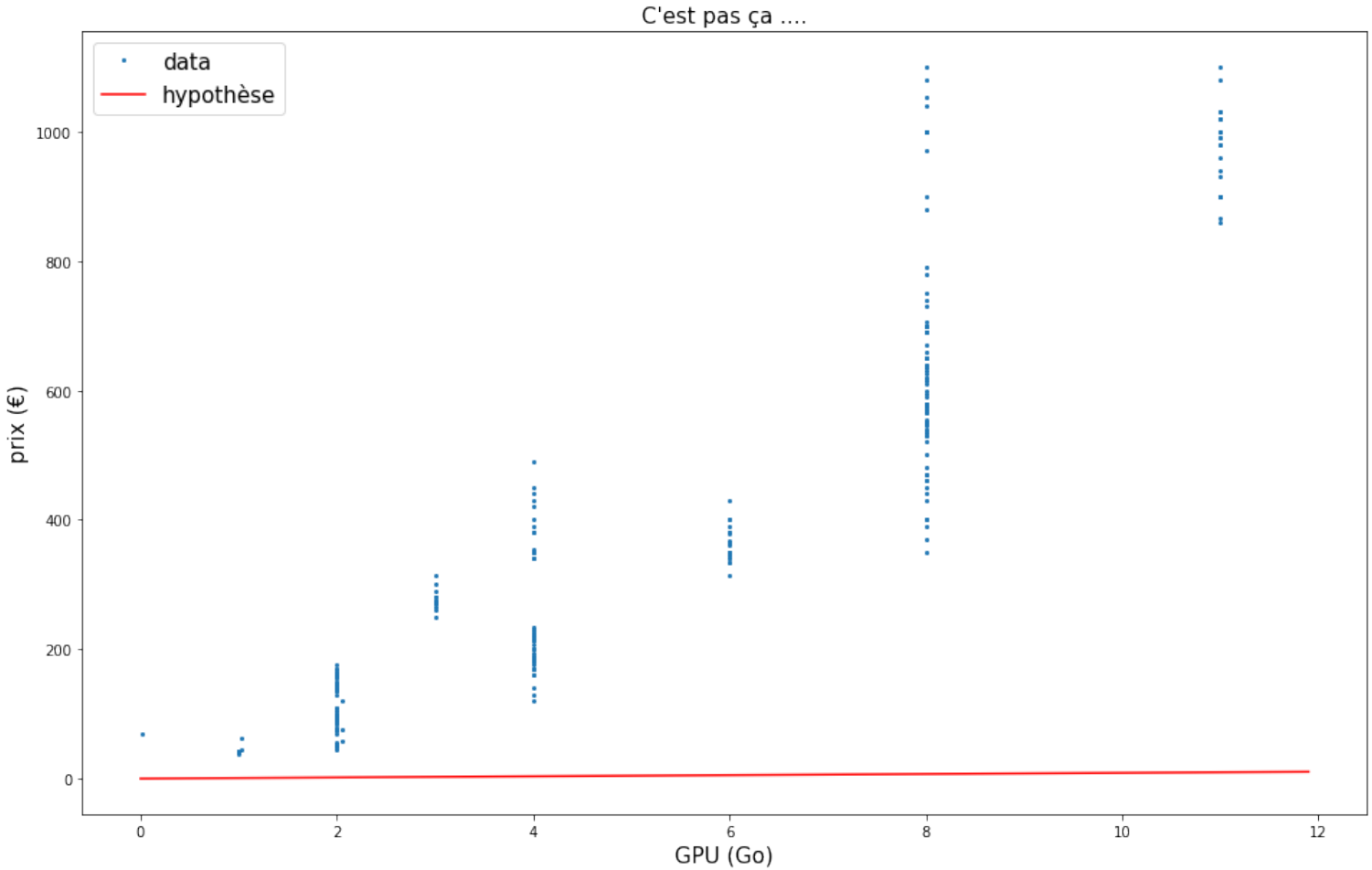
- On défini le paramètre θ_1 qui va *lier* x_1 à $\hat{y}$:

$$h_\theta(x) = \theta_1 x_1$$

- Rappel math: **fonction linéaire** $f(x) = kx$

Construire un modèle (regression linéaire)

- Initialisons aléatoirement la valeur de θ_1

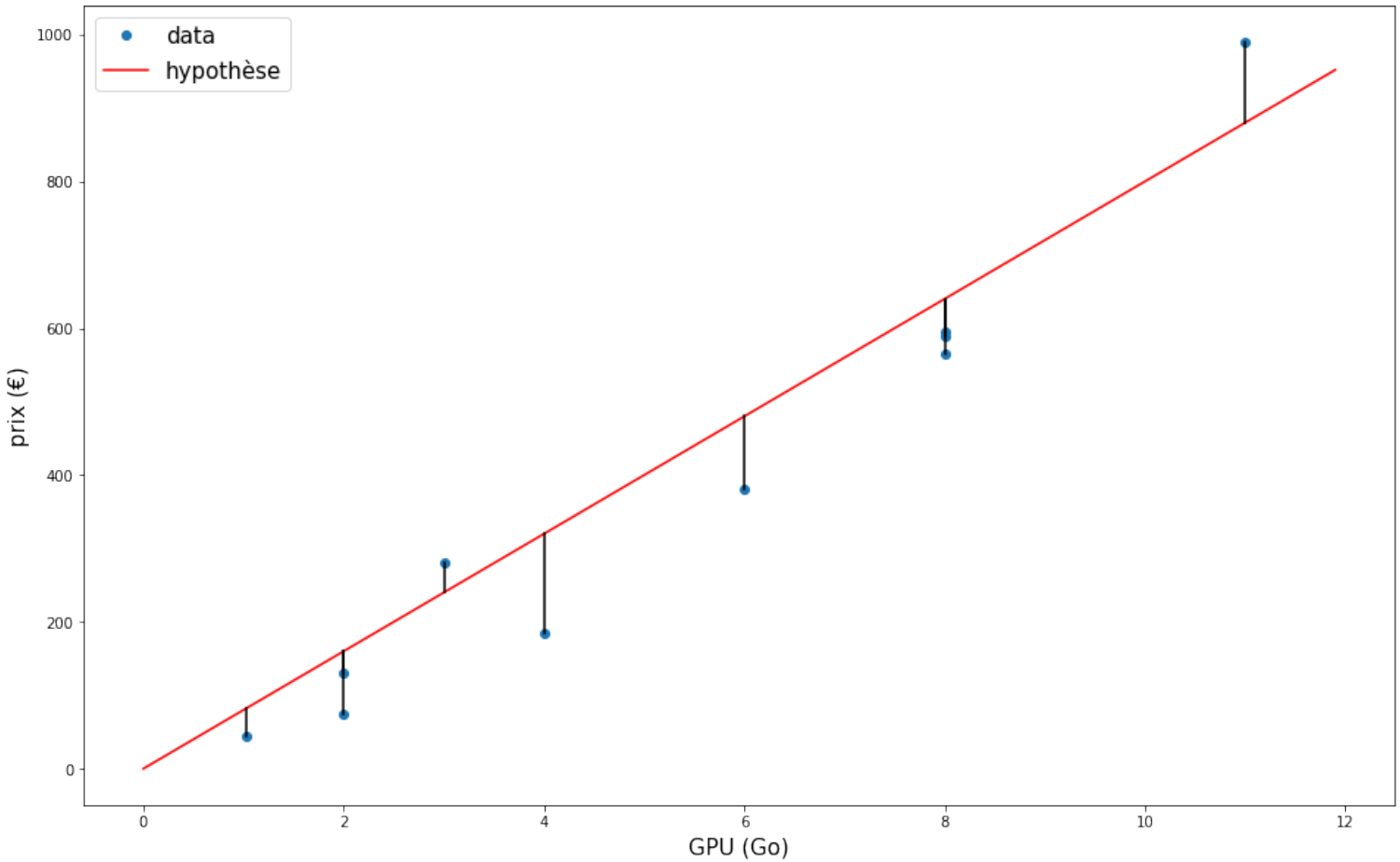


- C'est pas encore ça ...

La fonction de coût

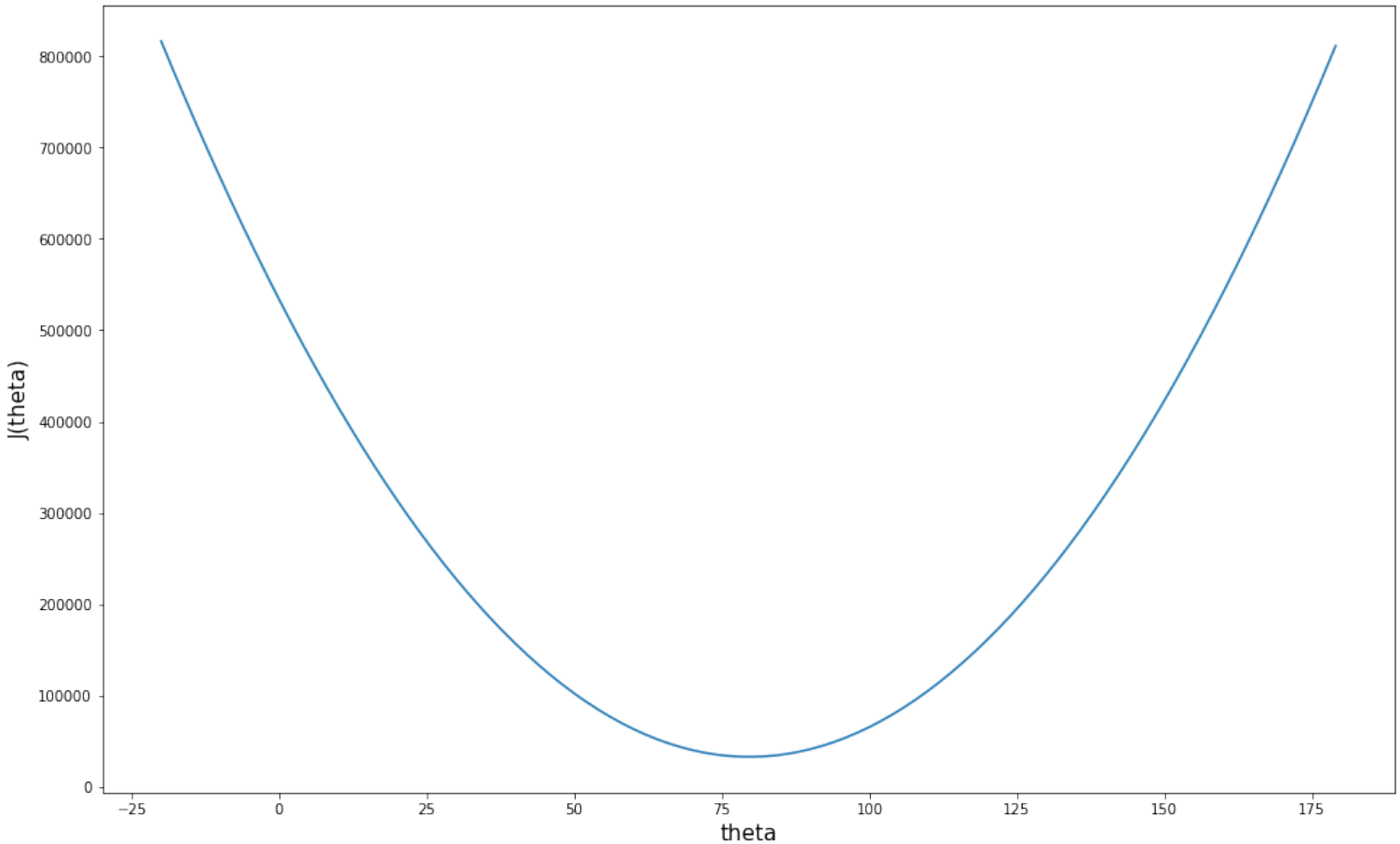
- $\mathcal{J}(\theta)$: *véracité* de notre modèle
- Une définition possible: somme quadratique des erreurs

$$\mathcal{J}(\theta) = \frac{1}{2m} \sum_{i=0}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$



La fonction de coût

- On cherche à trouver la valeur de θ_1 qui **minimise** $\mathcal{J}(\theta)$
- En Brute ...

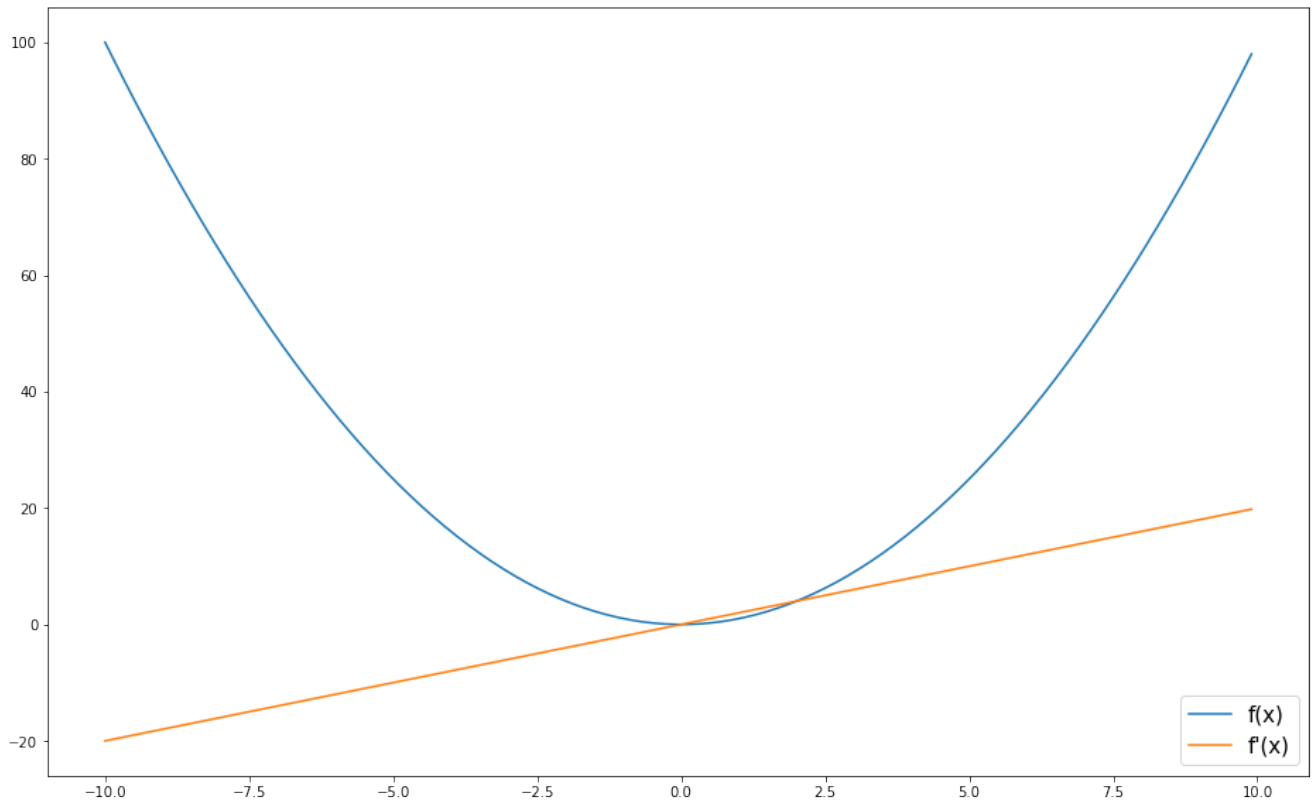


- ... essayons d'optimiser

La descente de gradient

- Algorithme pour arriver "*rapidement*" au minimum de $\mathcal{J}(\theta)$
- On va utiliser la *dérivation*: $\frac{d}{d\theta_1}\mathcal{J}(\theta)$:

Si $\mathcal{J}(\theta)$ est croissant: $\frac{d}{d\theta_1}\mathcal{J}(\theta) > 0$, Si $\mathcal{J}(\theta)$ est décroissant: $\frac{d}{d\theta_1}\mathcal{J}(\theta) < 0$



La descente de gradient

- (Encore) un peu de math, la descente de gradient s'écrit:

Descente de gradient

Répéter jusqu'à convergence: $\{$

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} \mathcal{J}(\theta)$$

$\}$

- α s'appelle le taux d'apprentissage (*learning rate*) et c'est le **seul** paramètre de l'algorithme.
- On va itérativement modifier la valeur de θ_1 en fonction de la dérivée de $\mathcal{J}(\theta)$, jusqu'à minimiser $\mathcal{J}(\theta)$ (*convergence*).

La descente de gradient

- Dérivons donc notre fonction de coût:

$$\mathcal{J}(\theta) = \frac{1}{2m}\sum_{i=0}^{m}(\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{2m}\sum_{i=0}^{m}(\theta_1 x_1^{(i)} - y^{(i)})^2$$

$$\frac{d}{d\theta_1}\mathcal{J}(\theta) = \frac{1}{m}\sum_{i=0}^{m}(\hat{y}^{(i)} - y^{(i)})x_1^{(i)}$$

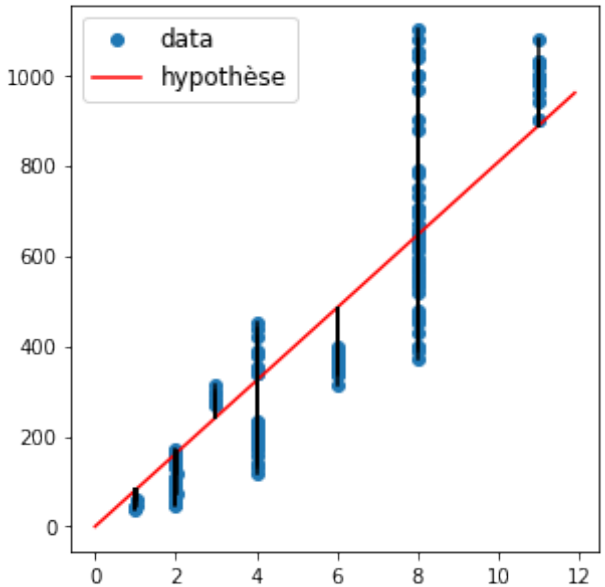
- Un peu de *hand-tunning*:
  - Le learning rate (α) est fixé à 0.03 (0.045 pour la démo)
  - Définissons une précision $\epsilon = 0.0001$ qui nous servira à arrêter la descente de gradient

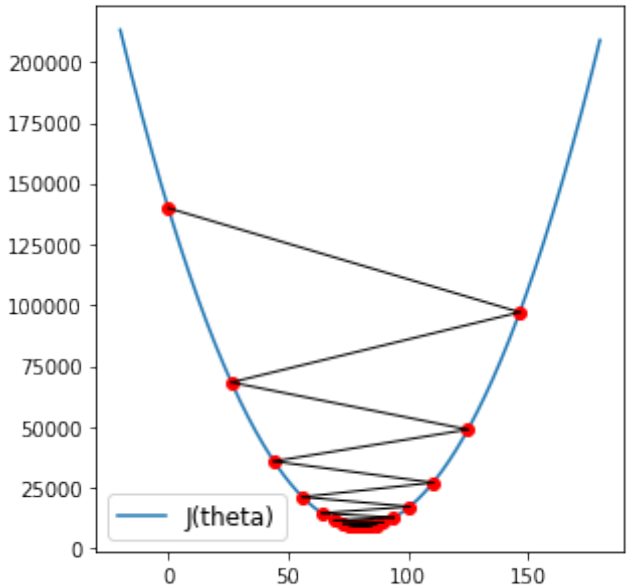
Préparation du dataset

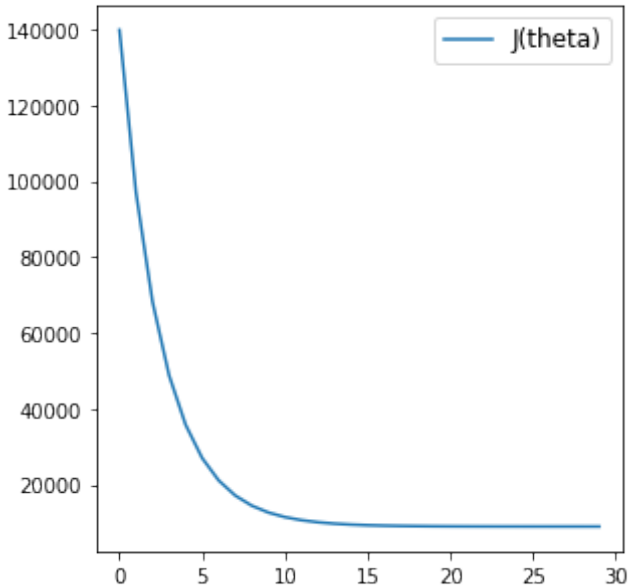
- Bonne pratique de ML, pour tout les algos!
- On sépare **aléatoirement** les données en 2 (3) échantillons:
  - Entraînement / (Validation) / Test
  - *80 / 20* (*70 / 30*) ou *60 / 20 / 20*

- *Entraînement:* utilisé pour la descente de gradient
- *Validation:* utilisé pour l'hyperparamètrage de l'algo
- *Test:* utilisé pour mesurer la performance du modèle

C'est parti !







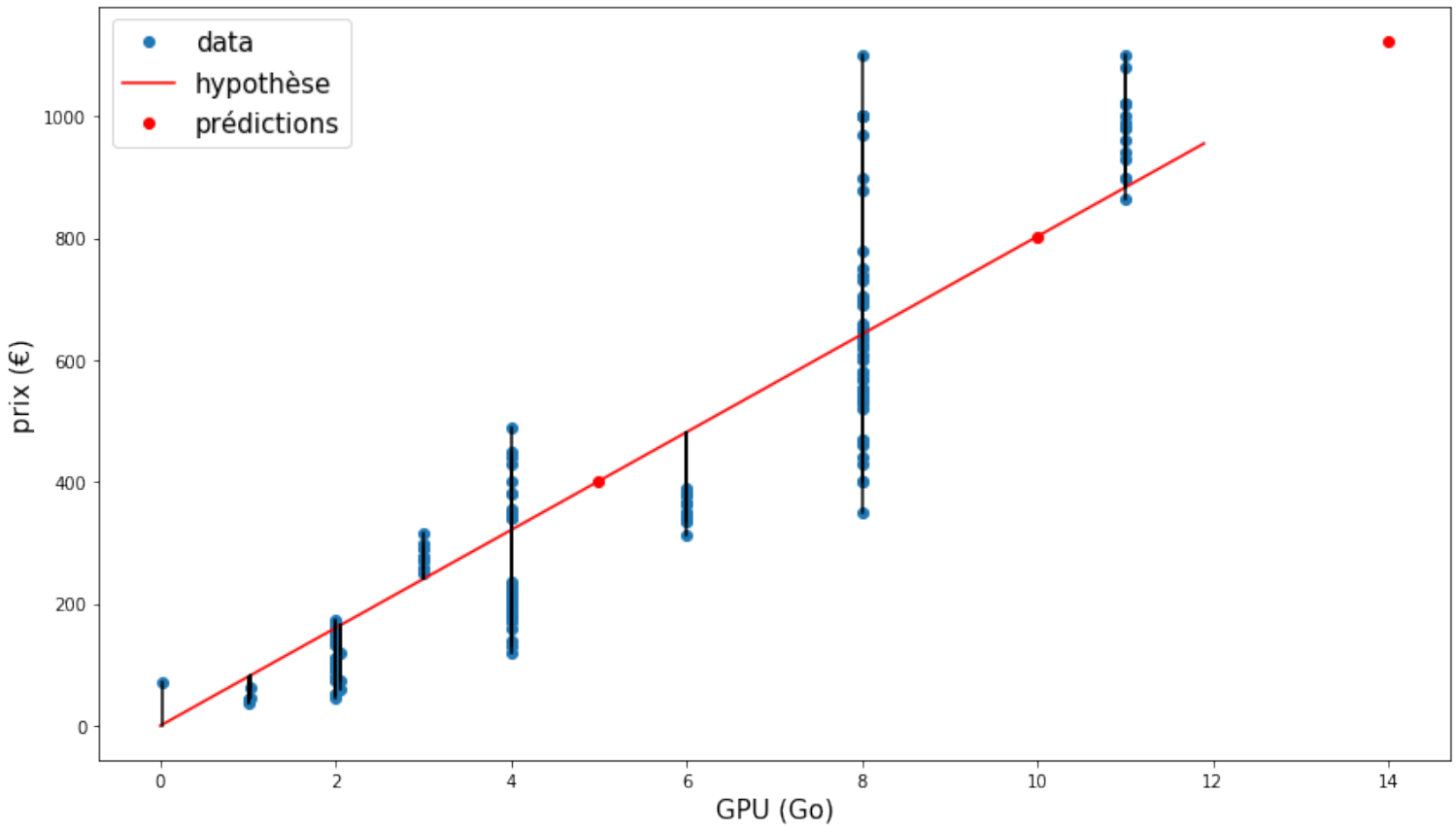
- La descente de gradient c'est achevée au bout de quelques itérations
- On peut voir que $\mathcal{J}(\theta)$ a continuellement diminué à chaque itération

$$\theta_1 \approx 80$$

$$err_{train} \approx err_{test}$$

On peut maintenant faire une prédiction

- Quel serait le prix de cartes avec 5, 10 et 14 Go de GPU?

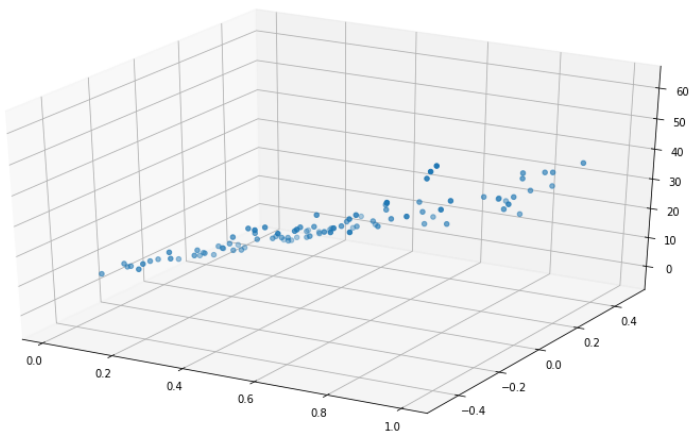


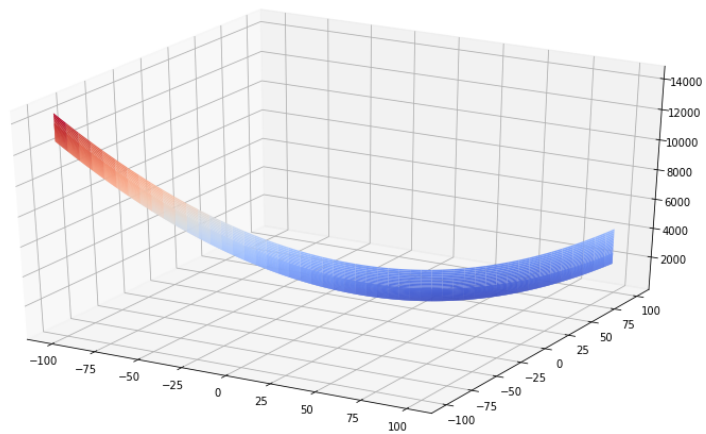
- On pourra les vendre autour de 400, 800 et 1100 euros!

La regression linéaire multivariables

- Le principe est le même, mais avec plusieurs variables x_i (donc plusieurs paramètres θ_i):

$$\hat{y} = \theta_1 x_1 + \cdots + \theta_n x_n = \sum_{i=1}^{n} \theta_i x_i$$



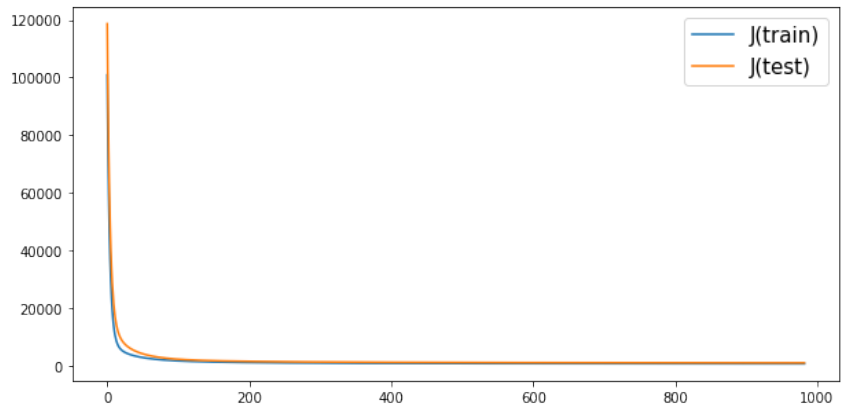


Affinons notre modèle de carte graphiques

- Plus de features: chipset, fréquence, consommation, ...
- Il va falloir explorer et nettoyer les données:
  - Gestion des données manquantes / abbérantes
  - *Features engineering*
  - Normaliser le dataset (pour accélérer la descente de gradient)

Régression linéaire multivariables: Résultats

- On utilise la même valeur de $\epsilon = 0.0001$ et $\alpha = 0.03$
- Plus long! Mais meilleur résultats:
- Modèle simple: $err \approx 100$
- Modèle multivariable: $err \approx 30$



Pour conclure sur la régression linéaire

- **Regression Linéaire:** $\hat{y}$ est une valeur *continue*
  - Valeur discrète: **Regression Logistique** (*classification*)
- Le résultat $\hat{y}$ dépend **linéairement** des variables x_i si:

$$\hat{y} = \theta_1 x_1 + \cdots + \theta_n x_n = \sum_{i=1}^{n} \theta_i x_i$$

- **Apprentissage supervisé**: y est connu pour chaque x_1 dans le jeu de données d'entrainement
- Facile à implémenter (encore plus avec Scikit-learn ...), rapide: bon point de départ sur un sujet

Merci !

Léo Beaucourt