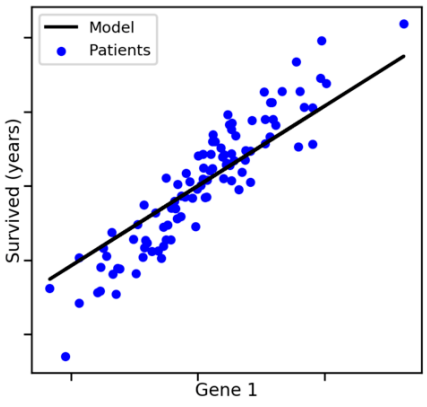
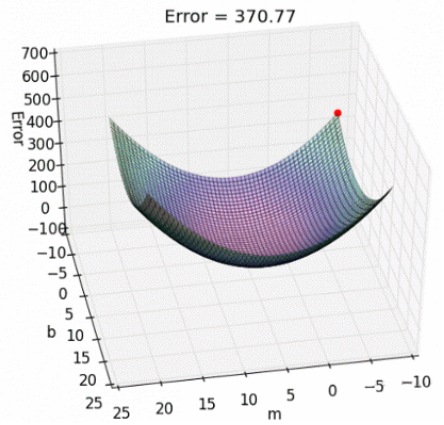
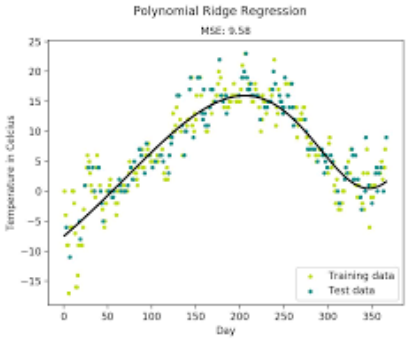
Cycle 8: Mettre en œuvre une démarche de résolution numérique

Chapitre 2 : Apprentissage SUPERVISE – les REGRESSIONS







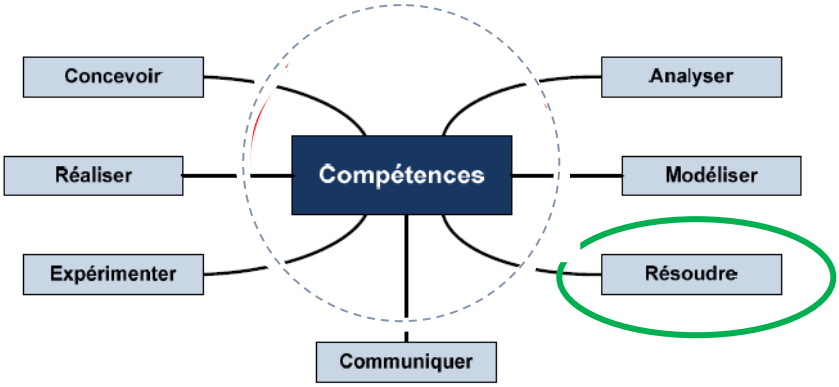
Problématique

Qu'est-ce que l'apprentissage supervisé ?
Quels sont les principes et techniques de régressions (linéaires, multiples, non linéaires) ?

Savoir

C. Résoudre:

- C3 : mettre en œuvre une démarche de résolution numérique



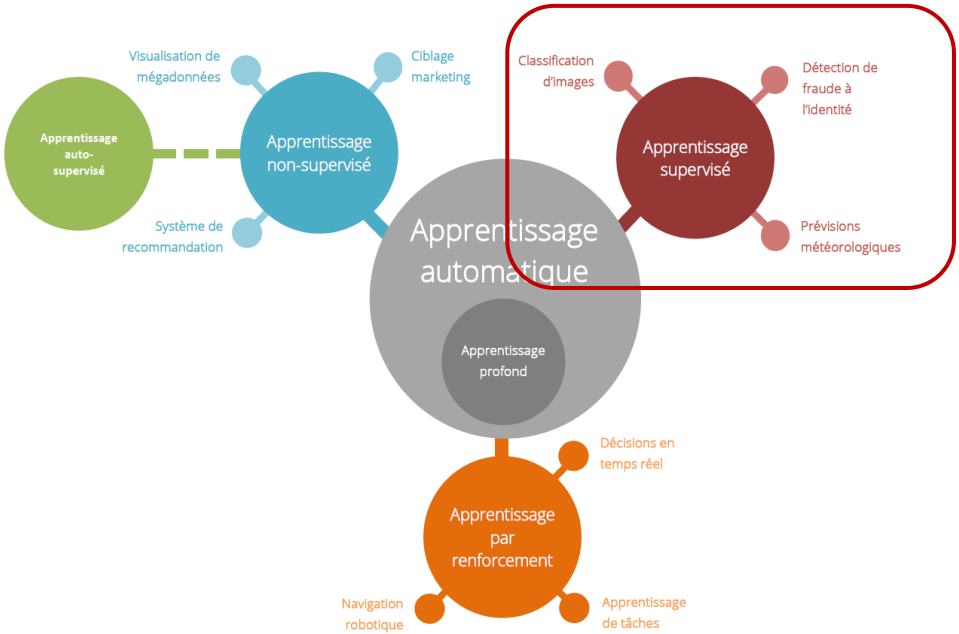
Apprentissage supervisé – les REGRESSIONS

1. Rappels sur l'apprentissage supervisé

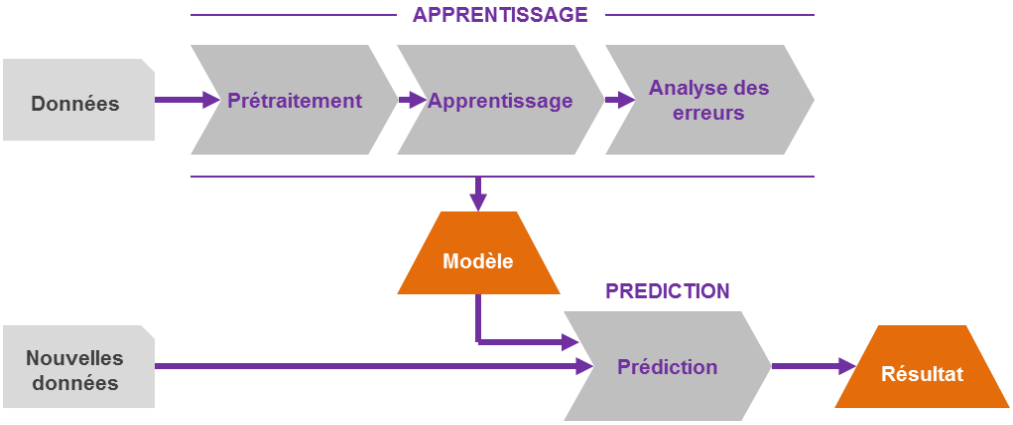
L'IA repose sur une phase très importante pour que la machine devienne intelligente, celle de l'**APPRENTISSAGE** basée sur **un très grand nombre d'exemples**. C'est ce que l'on appelle le **machine learning.**

Au sein du machine learning on distingue **3 types d'apprentissages :**

- *supervisé*
- *non supervisé*
- *par renforcement*



Après la phase d'apprentissage, **la machine affine le modèle grâce à de nouvelles entrées étiquetées** qui via des **algorithmes** que nous allons découvrir permet **d'ajuster les paramètres du modèle** permettant à la machine de s'enrichir. Une fois cette phase réalisée, le système peut alors fournir lui-même un résultat quelles que soient les données d'entrée



L'apprentissage supervisé : L'apprentissage supervisé (*supervised learning*) est une tâche d'apprentissage automatique consistant à apprendre une fonction de prédiction à partir **d'exemples annotés ou étiquetés**.

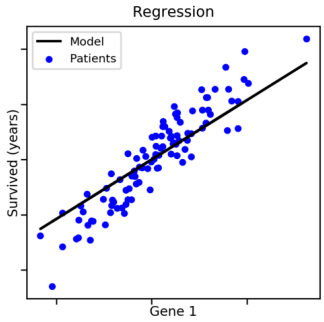
Les problèmes de **prédiction d'une variable quantitative** sont des **problèmes de régression**.

Voici une super chaine de Machine Learnia : https://www.youtube.com/channel/UCmpptkXu8iIFe6kfDK5o7VQ

Apprentissage supervisé – les REGRESSIONS

Les modèles d'apprentissage supervisé demandent beaucoup de travail préparatoire aux data scientists. Les **jeux de données en entrée doivent être étiquetés**, tandis qu'il faut **indiquer les paramètres de sortie, les résultats attendus**. Il faut également ajuster la précision pendant le processus d'apprentissage.

Ex : avec l'apprentissage supervisé, la machine peut apprendre à estimer le prix d'un appartement après qu'on lui est montré des millions de prix d'appartements fonctions de variables d'entrées données (surface, localisation…) = REGRESSION.



2. Notions fondamentales en apprentissage supervisé

Pour faire simple on a vu que l'apprentissage supervisé consistait à montrer à la machine des exemples x, y et de lui demander de **trouver l'association qui relie x à y** : **y=f(x).**

Mais quels sont les éléments indispensables pour mettre en place un apprentissage supervisé ?

Regardez cette vidéo avec de lire la suite : https://www.youtube.com/watch?v=K9z0OD22My4

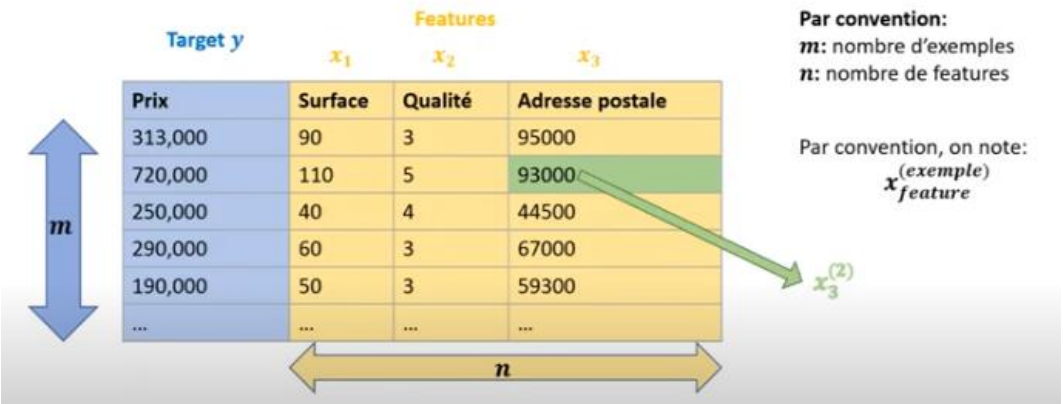
2.1. Le dataset

Les exemples que l'on montre à la machine sont mis dans un **DATASET.** C'est la 1ère notion importante.
Dans un dataset , il y a 2 types de variables :

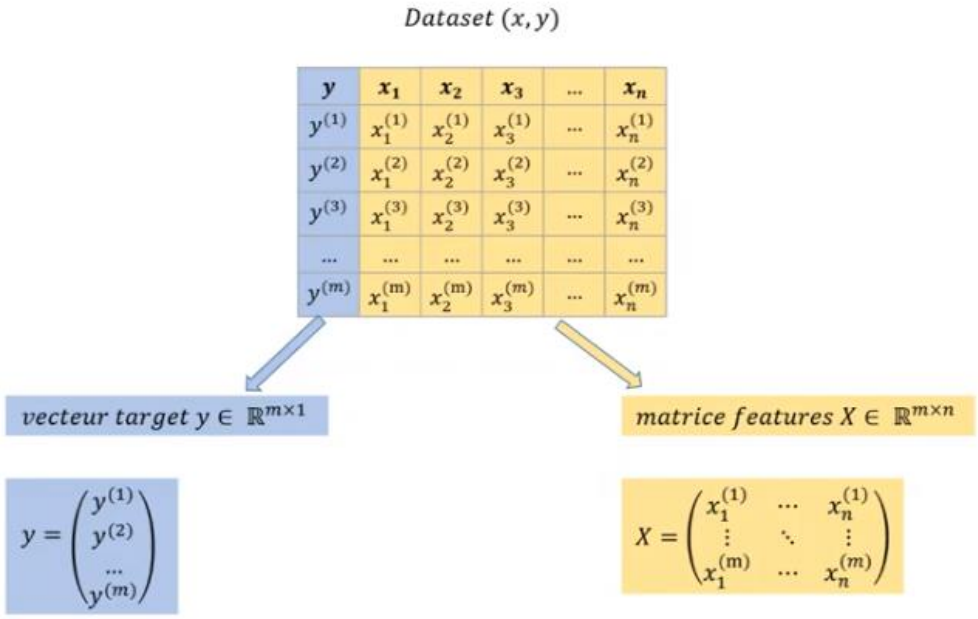
- **Target variable (y)** : c'est notre objectif, ce que l'on veut que la machine apprenne à prédire
  (ex : prix appartement, email spam ou pas…)
- **Features (x_i)** : entrées (x_i) qui viennent influencer la valeur de y
  (ex : surface, localisation, étage…)

Par convention on note **m le nombre d'exemples** que l'on a dans notre dataset *(nb de lignes)* et **n le nombre de Features** *(nb de colonnes).*

Exemple de Dataset sur des appartements

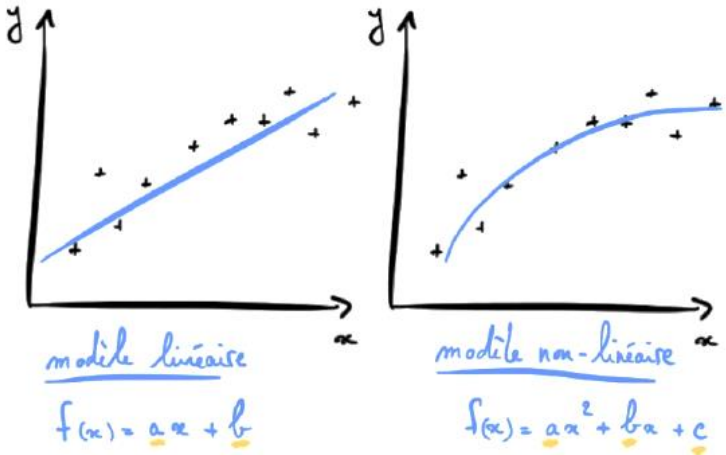


Au final, un dataset d'apprentissage supervisé ressemble à cela, avec un **vecteur « target » avec m lignes**, et une **matrice features avec m lignes et n colonnes**.



2.2. Le modèle

La 2ème notion importante en apprentissage supervisé est **le MODELE**. A partir du dataset on peut visualiser un nuage de points sur lequel on pourrait créer un modèle, par exemple linéaire y=f(x)=ax+b, qui passerait au mieux des points. Où bien un modèle polynomial non linéaire d'ordre 2 de type $y=ax^2+bx+c$ ou d'ordre 3.



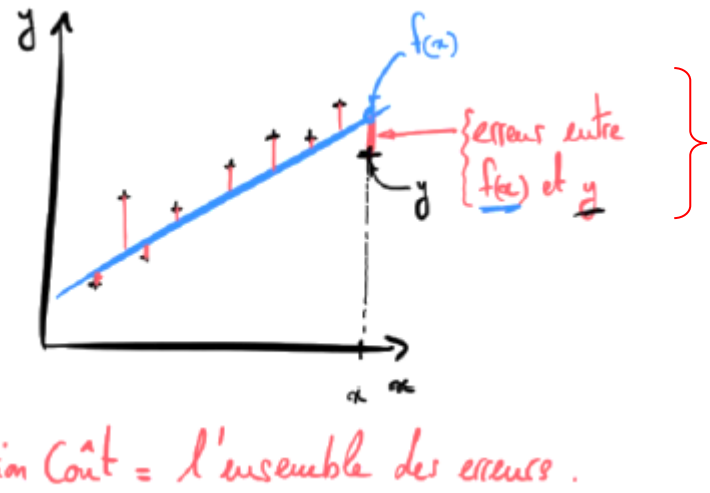
Le modèle est donc constitué de **PARAMETRES** (a, b, c …).

Attention, en apprentissage supervisé, c'est ***nous qui décidons quel modèle la machine doit utiliser*** *et c'est la machine qui doit apprendre à optimiser les paramètres pour coller au mieux au nuage de points. Au départ les* ***paramètres sont pris aléatoirement.***

2.3. La fonction coût

La 3ème notion importante en machine learning est la **FONCTION COUT**. Le modèle, lorsqu'on l'utilise par rapport au dataset, nous donne des **ERREURS** car le modèle est une généralisation de l'ensemble des valeurs. Il peut donc être loin en prédiction de certaines valeurs.

La fonction coût représente la somme de toutes les erreurs de notre modèle / dataset.

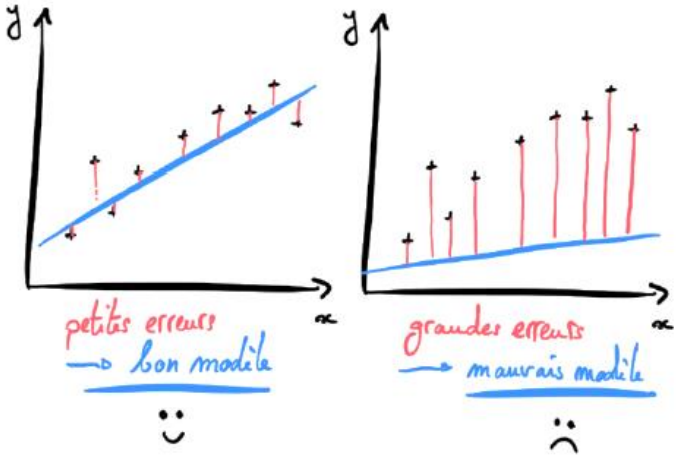


*Avoir un bon modèle c'est avoir un **modèle qui minimise les erreurs***

Fonction Coût = l'ensemble des erreurs.

2.4. L'algorithme d'apprentissage

La 4ème notion importante en apprentissage supervisé est **l'ALGORITHME d'APPRENTISSAGE**. En machine learning, on va développer une stratégie qui consiste **à trouver quels sont les paramètres a,b,c…qui minimise la fonction coût**, c'est-à-dire qui **minimise l'ensemble de nos erreurs**.



Pour cela on utilise un algorithme d'apprentissage. Il en existe beaucoup mais l'un des plus connu est celui de la **descente de gradient** que l'on va détailler plus loin dans ce cours.

*La force du Machine Learning, c'est qu'il est **très facile de développer des modèles très complexes** qui peuvent analyser des **milliers de features** (x) qu'un être humain ne serait **pas capable** de prendre en compte pour faire son calcul (et Excel non plus).*

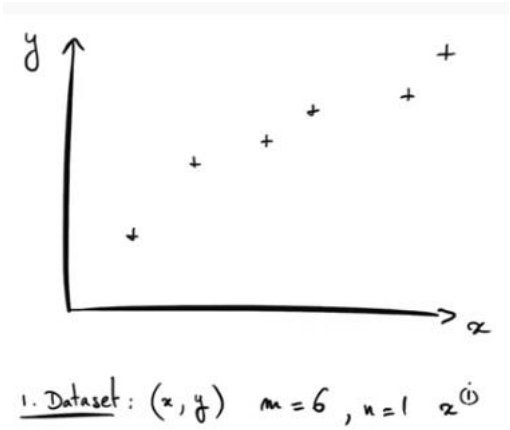
3. La régression linéaire

Si vous cherchez à prédire le cours de la bourse, le prix d'un appartement, ou bien l'évolution de la température sur Terre, alors vous cherchez en fait à résoudre un problème de **régression**.
Si vous disposez d'un Dataset (x,y) alors vous pouvez utiliser **l'apprentissage supervisé** pour développer un modèle de régression. Dans ce chapitre je vais vous montrer comment développer votre premier modèle de Machine Learning.

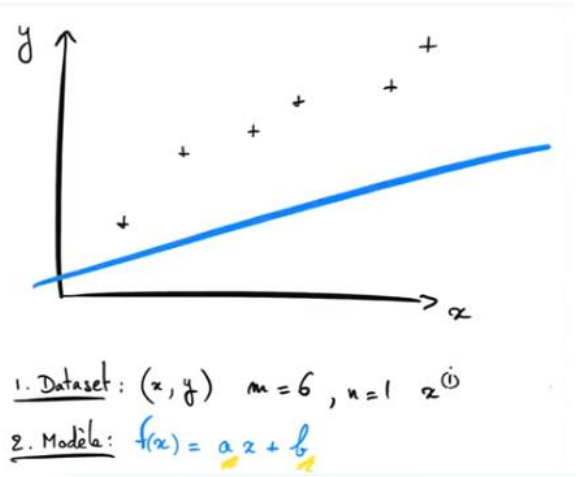
Regardez cette vidéo avec de lire la suite : https://www.youtube.com/watch?v=wg7-roETbbM

3.1. Les étapes sur un exemple

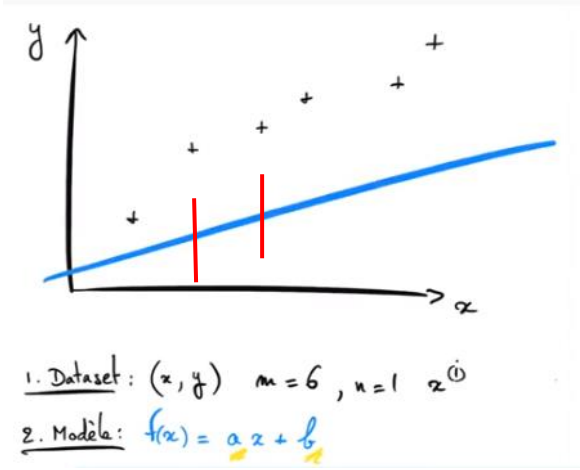
Travaillons sur un modèle avec **1 seule variable x et 6 exemples.**



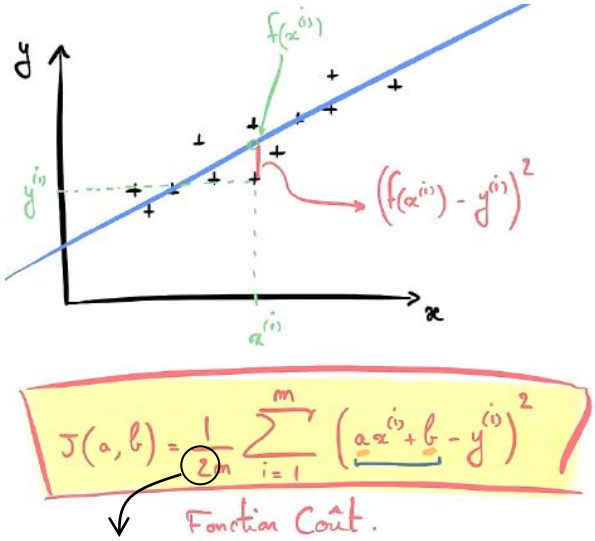
1 *On a un dataset de 6 valeurs*



2 *La machine ne connait pas les paramètres a (pente) et b (ordonnée origine), on lui donne via un random*



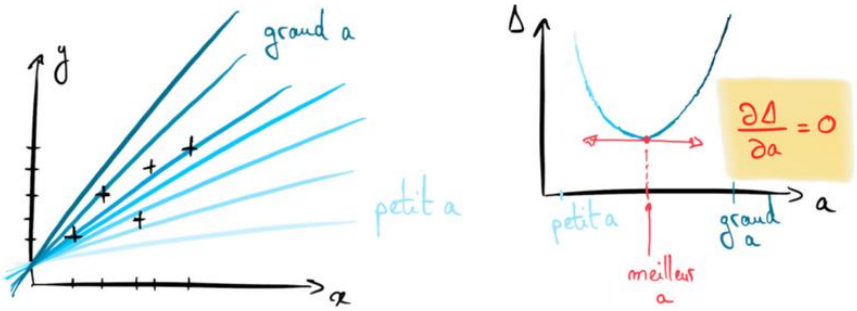
3 *La machine va calculer la fonction coût et optimiser les paramètres pour minimiser les erreurs*



On met un 2 pour simplifier calcul matriciel par la suite (change rien car on va minimiser fonction cout)

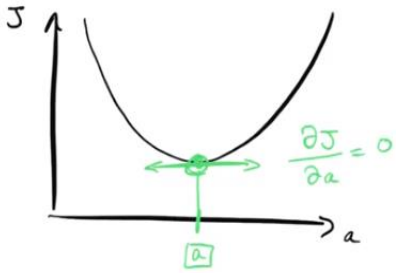
*Pour la régression linéaire, on utilise la **norme euclidienne (erreur quadratique)** pour mesurer les **erreurs***

L'idée maintenant est de trouver un **algorithme de minimisation de cette fonction coût**, c'est-à-dire de trouver les paramètres qui optimisent le modèle, pour nous déjà le paramètre a.



Pour arriver à optimiser les paramètres, plusieurs méthodes existent :

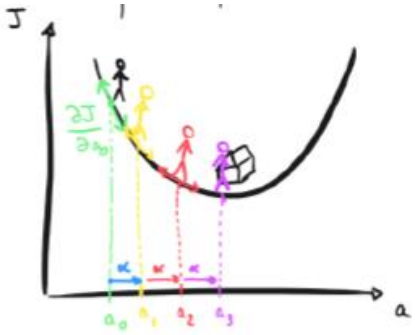
- Les **moindres carrés** (cf métrologie) …*point a pour lequel la tangente en i est horizontale…le plus optimum…*



Pour info, la méthode des moindres carrés (dite des équations normales) implique de faire des inversions de matrice et sur des énormes datasets…c'est juste impossible…

- **L'algorithme de descente de gradient** (plus efficace sur les datasets énormes)

Cet algorithme vous permet de trouver le **minimum** de la Fonction Coût J(a,b) en **partant de coordonnées a et b aléatoires**.



3.2. L'algorithme de descente de gradient

Cet algorithme permet de trouver le minimum de n'importe quelle fonction convexe qui converge en 1 seul minimum qui permettra de **trouver le meilleur modèle.**

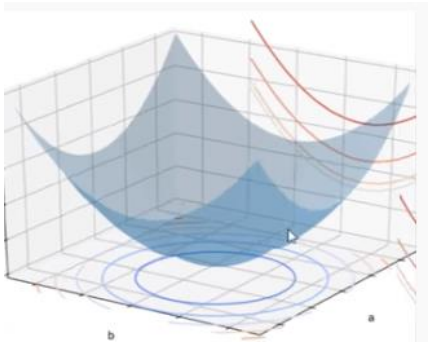
Regardez cette vidéo avec de lire la suite : https://www.youtube.com/watch?v=rcl_YRyoLIY&t=693s

Voici en résumé l'algorithme de descente de gradient :

1. Calculer la **pente** de la fonction coût, c'est-à-dire la **dérivée** de J(a,b) en un point pris au départ au hasard
2. **Evoluer** d'une certaine distance α dans la direction de la pente la plus forte. Cela a pour résultat de modifier les paramètres a et b
3. Recommencer les étapes 1 et 2 jusqu'à atteindre le minimum de J(a,b)

α est appelé **le learning rate**

Apprentissage supervisé – les REGRESSIONS



Attention, il faut faire cette descente de gradient pour a et b afin de les optimiser tous les 2 et arriver à une visualisation 3D suivante de notre surface J(a,b):

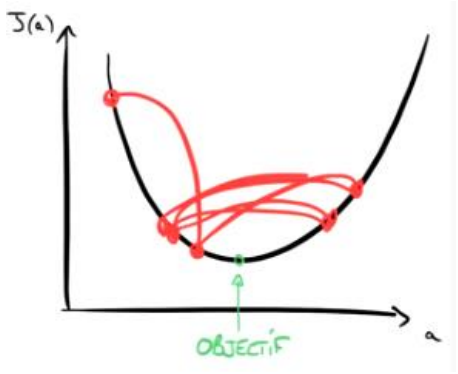
Formule d'implémentation de l'algorithme de descente de gradient :

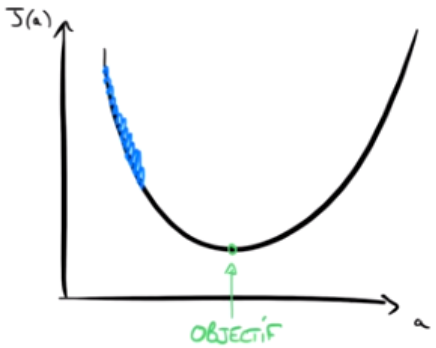
$$a_{i+1} = a_i - \alpha \frac{\partial J(a_i)}{\partial a}$$

idem pour b

Importance du learning rate α :

Si on prend une valeur de α trop grande pour apprendre très vite ou au contraire un α trop petit pour être très précis, on va jamais converger…





Pour trouver la bonne valeur de α il faut procéder par itération et tatonnement (cf programme python)

Le calcul du gradient :

Il faut calculer les **dérivées partielles** par rapport à a et b de notre fonction coût.

Notre fonction coût est :

$$J(a,b) = \frac{1}{2m} \sum (ax + b - y)^2$$

La **dérivée partielle par rapport à a** est donc :
Voilà pourquoi on a mis un 2 dans moyenne !!

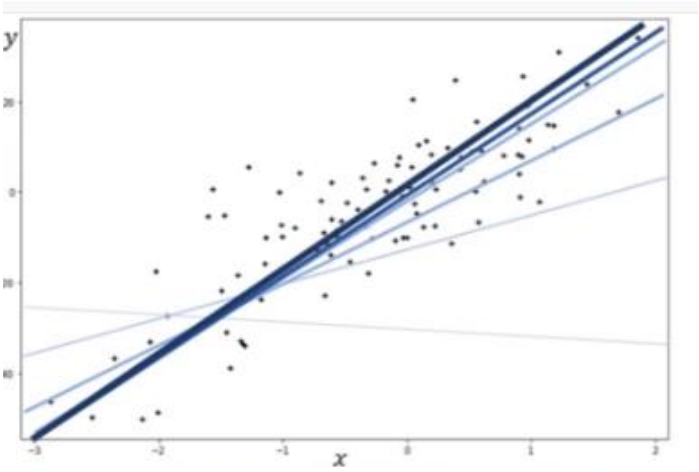
$$\frac{\partial J}{\partial a} = \frac{1}{m} \sum x(ax + b - y)$$

La dérivée partielle par rapport à b est donc :

$$\frac{\partial J}{\partial b} = \frac{1}{m} \sum (ax + b - y)$$

Apprentissage supervisé – les REGRESSIONS

SYNTHESE sur l'algorithme de la descente de gradient :



Gradient Descent

Répéter jusqu'à minimisation de la Fonction Coût:

$$a := a - \alpha \frac{1}{m} \sum_{i=1}^{m} x^{(i)} \left(a x^{(i)} + b - y^{(i)}\right)$$

$$b := b - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^{m} \left(a x^{(i)} + b - y^{(i)}\right)}_{Gradient}$$

Régression Linéaire: Résumé des équations

- **Dataset :** (x, y) avec m exemples
- **Modèle :** $f(x) = ax + b$
- **Fonction Coût :** $J(a,b) = \frac{1}{2m} \sum_{i=1}^{m} \left(a x^{(i)} + b - y^{(i)}\right)^2$
- **Gradients :**

$$\frac{\partial J(a,b)}{\partial a} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} \left(a x^{(i)} + b - y^{(i)}\right)$$

$$\frac{\partial J(a,b)}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} \left(a x^{(i)} + b - y^{(i)}\right)$$

- **Algorithme de Gradient Descent :**

$$\begin{cases} a := a - \alpha \dfrac{\partial J(a,b)}{\partial a} \\ b := b - \alpha \dfrac{\partial J(a,b)}{\partial b} \end{cases}$$

On va passer par des **calculs matriciels dans Numpy Python**

(cf programmes)

Regardez cette vidéo pour comprendre le passage par les matrices :
https://www.youtube.com/watch?v=8Y3r7F47Xfo