



MACHINE LEARNING

Rédigé par :

Hippolyte SODJINOU

...

Adresse mail : sodjinouhippolyte@gmail.com

Année académique 2023-2024




Table des matières

Introduction	1
1 Définition et utilité du Machine Learning	2
1.1 Qu'est-ce que le Machine Learning ?	2
1.2 Pourquoi utiliser le Machine Learning ?	2
2 Types de Machine Learning	3
2.1 Apprentissage supervisé	3
2.1.1 Apprendre à partir d'exemples (Dataset)	3
2.1.2 Développer un modèle à partir du Dataset.....	4
2.1.3 Apprendre, c'est minimiser la Fonction Coût.....	6
2.2 Apprentissage non supervisé (Unsupervised Learning).....	6
2.3 Apprentissage par renforcement	6
2.4 Apprentissage semi-supervisé	6
3 Principaux algorithmes en Machine Learning	6
3.1 Régression linéaire	6
3.1.1 Apprentissage Supervisé et problème de Régression.....	6
3.1.2 Comment pouvez-vous créer et comprendre votre premier modèle linéaire pour faire des prédictions précises?	6
3.1.2.1 Récolter vos données	7
3.1.2.2 Créer un modèle linéaire.....	7
3.1.2.3 Définir La Fonction Coût.....	8
3.1.2.4 Trouver les paramètres qui minimisent la Fonction Coût.....	9
3.1.2.4.1 Qu'est-ce que la descente de gradient ?	9
3.1.2.4.2 Comment trouver le minimum ?	10
3.1.2.5 Comment utiliser l'algorithme de Gradient Descent ?	12
3.2 Calcul des dérivées partielles de Fonction Coût $J(\mathbf{a}, \mathbf{b})$	13
3.3 Utilisation des matrices et des vecteurs	13
4 Développement du programme de Machine Learning avec Python	14
4.1 Importer les librairies	14
4.2 Créer un Dataset.....	14
4.3 Développer le modèle et l'entraîner	15
4.4 Les courbes d'apprentissage	17
4.5 Régression Polynômiale à plusieurs variables.....	18

5	Arbres de décision et forêts aléatoires	19
6	Applications du Machine Learning.....	19
6.1	En finance.....	19
6.2	En santé	19
6.3	Dans d'autres domaines	19
	Conclusion.....	20
	Bibliographie	21

Introduction

Le Machine Learning, ou apprentissage automatique, est un domaine fascinant et en constante évolution. Né de la convergence de diverses disciplines telles que les statistiques, l'optimisation, l'algorithmique et le traitement du signal, il s'est imposé comme un pilier incontournable de la société moderne. Déjà depuis plusieurs décennies, il est employé dans des applications comme la reconnaissance automatique de caractères et les filtres anti-spam. Aujourd'hui, il joue un rôle crucial dans des domaines aussi variés que la protection contre la fraude bancaire, la recommandation personnalisée de livres, films ou autres produits, la reconnaissance faciale dans les appareils photo, et la traduction automatique entre différentes langues.

À l'avenir, le Machine Learning promet d'accroître la sécurité routière, notamment grâce aux véhicules autonomes, d'améliorer la gestion des interventions d'urgence lors de catastrophes naturelles, de faciliter la découverte de nouveaux médicaments, et d'optimiser l'efficacité énergétique dans les bâtiments et les industries. Ce document a pour objectif de clarifier les concepts fondamentaux du Machine Learning et d'explorer les différentes branches qui seront abordées dans les sections suivantes.

1 Définition et utilité du Machine Learning

1.1 Qu'est-ce que le Machine Learning ?

Le Machine Learning, ou apprentissage automatique en français, est une branche de l'intelligence artificielle qui se concentre sur le développement d'algorithmes capables d'apprendre à partir des données. Contrairement aux systèmes traditionnels, où les règles sont explicitement programmées, les algorithmes de Machine Learning apprennent des modèles et des relations dans les données pour effectuer des prédictions ou des décisions.

Le concept de Machine Learning remonte aux années 1950, avec les travaux pionniers d'Alan Turing, qui introduisit l'idée d'une machine capable d'apprendre. Dans les décennies qui suivirent, des progrès ont été réalisés, notamment avec l'invention de l'algorithme de la descente de gradient et des réseaux de neurones. Cependant, ce n'est qu'au début du 21^e siècle, avec l'augmentation de la puissance de calcul et la disponibilité massive de données, que le Machine Learning a véritablement pris son essor.

Exemple :

Pour calculer le montant total dépensé par un client à partir de ses factures, un simple algorithme d'addition suffit, sans besoin de Machine Learning. En revanche, si l'on souhaite prédire quels produits un client est susceptible d'acheter le mois suivant, l'information disponible dans les factures ne suffit pas. Cependant, en disposant d'un historique d'achat de nombreux clients, un algorithme de Machine Learning peut être utilisé pour créer un modèle prédictif capable de répondre à cette question.

1.2 Pourquoi utiliser le Machine Learning ?

Le Machine Learning peut servir à résoudre des problèmes

- que l'on ne sait pas résoudre (comme dans l'exemple de la prédiction d'achats ci-dessus) ;
- que l'on sait résoudre, mais dont on ne sait formaliser en termes algorithmiques comment nous les résolvons (c'est le cas par exemple de la reconnaissance d'images ou de la compréhension du langage naturel) ;
- que l'on sait résoudre, mais avec des procédures beaucoup trop gourmandes en ressources informatiques (c'est le cas par exemple de la prédiction d'interactions entre molécules de grande taille, pour lesquelles les simulations sont très lourdes). Le machine learning est donc utilisé quand les données sont abondantes (relativement), mais les connaissances peu accessibles ou peu développées.¹

¹ Azencott, C.-A. *Introduction au Machine Learning*.

2 Types de Machine Learning

2.1 Apprentissage supervisé

L'apprentissage supervisé est la méthode la plus courante en Machine Learning. Dans ce cadre, l'algorithme est entraîné sur un ensemble de données étiquetées, c'est-à-dire des données pour lesquelles les résultats corrects sont déjà connus. L'objectif est de construire un modèle capable de faire des prédictions précises sur de nouvelles données. Des exemples d'algorithmes supervisés incluent la régression linéaire, les arbres de décision, et les machines à vecteurs de support (SVM).

Pour maîtriser l'apprentissage supervisé, il faut absolument comprendre et connaître les 4 notions suivantes :

- Le Dataset
- Le Modèle et ses paramètres
- La Fonction Coût
- L'Algorithme d'apprentissage

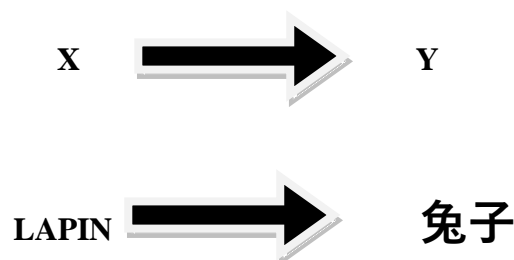
2.1.1 Apprendre à partir d'exemples (Dataset)

Imaginez que vous commenciez à apprendre le chinois.

Pour ce faire, il vous faudra soit acheter un livre de traduction chinois-français, ou bien trouver un professeur de chinois.

Le rôle du professeur ou du livre de traduction sera de superviser votre apprentissage en vous fournissant des exemples de traductions français-chinois que vous devrez mémoriser.

Comme pour apprendre la langue chinoise, on parle d'apprentissage supervisé lorsque l'on fournit à une machine beaucoup d'exemples (x, y) dans le but de lui faire apprendre la relation qui relie x à y .



En Machine Learning, ces exemples (\mathbf{x}, \mathbf{y}) sont rassemblés dans un tableau appelé Dataset :

- La variable \mathbf{y} est connue sous le nom de target (cible), représentant la valeur à prédire.
- La variable \mathbf{x} , quant à elle, est appelée feature (facteur). Un facteur influence la valeur de \mathbf{y} , et il est courant d'avoir de nombreux features $(\mathbf{x}_1, \mathbf{x}_2, \dots)$ dans notre Dataset, regroupées dans une matrice \mathbf{X} .

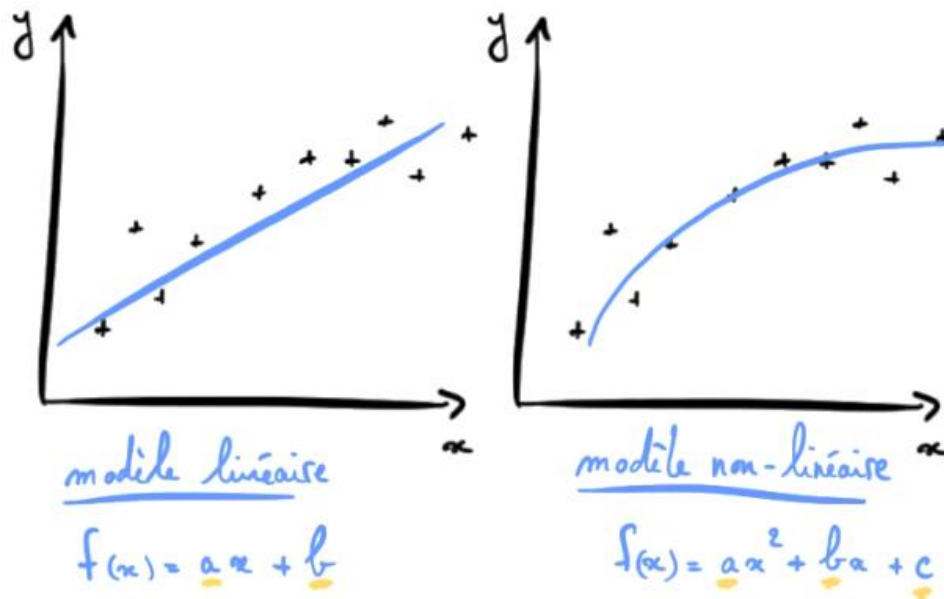
Exemple :

<i>Target</i>	<i>Features</i>		
\mathbf{y}	\mathbf{x}_1	\mathbf{x}_2	\mathbf{x}_3
<i>Prix</i>	<i>Surface en m²</i>	<i>N chambres</i>	<i>Qualité</i>
313 000 f	124	3	1,5
2 384 000 f	339	5	2,5
342 000 f	179	3	2
420 000 f	186	3	2,25
550 000 f	180	4	2,5
490 000 f	82	2	1
335 000 f	125	2	2

Dataset qui regroupe des exemples d'appartements avec leur prix \mathbf{y} ainsi que certaines de leurs caractéristiques (features).

2.1.2 Développer un modèle à partir du Dataset

En Machine Learning, un modèle est élaboré à partir de ce Dataset. Ce modèle peut être linéaire, comme illustré à gauche, ou non-linéaire, comme montré à droite.

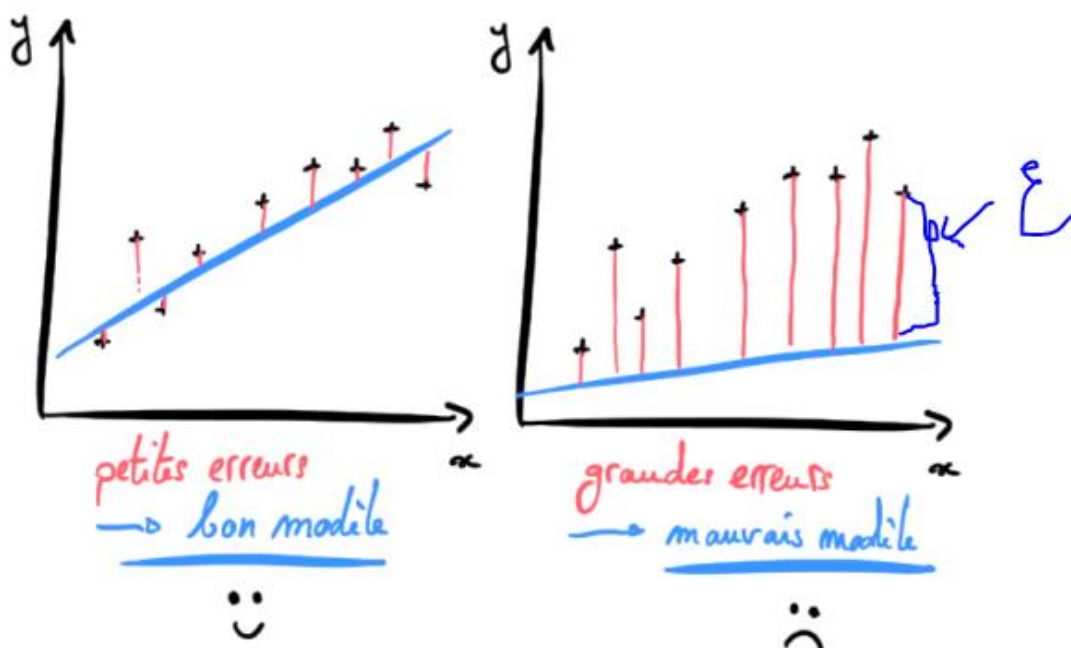


On définit a, b, c , etc. comme étant les **paramètres** d'un modèle.

2.1.3 Les erreurs de notre modèle - la Fonction Coût

Il est également important de noter qu'un modèle génère des erreurs par rapport à notre Dataset. Ces erreurs sont regroupées sous le terme de Fonction Coût, qui est généralement calculée à partir de la moyenne quadratique des erreurs.

Avoir un bon modèle, c'est avoir un modèle qui nous donne de petites erreurs, donc une petite **Fonction Coût**.



ϵ est l'erreur.

2.1.4 Apprendre, c'est minimiser la Fonction Coût

L'objectif central en apprentissage supervisé (Supervised Learning), est de trouver les paramètres du modèle qui minimisent la **Fonction Coût**. Pour cela, on utilise un algorithme d'apprentissage, l'exemple le plus courant étant l'algorithme de Gradient Descent.

2.2 Apprentissage non supervisé (Unsupervised Learning)

Dans l'apprentissage non supervisé, les données d'entraînement ne sont pas étiquetées. L'algorithme doit identifier les structures cachées dans les données. Les techniques courantes incluent le clustering, comme l'algorithme K-Means, et la réduction de dimensionnalité, comme l'analyse en composantes principales (ACP).

2.3 Apprentissage par renforcement

L'apprentissage par renforcement est une méthode où un agent apprend à prendre des décisions en interagissant avec un environnement. Chaque action entreprise par l'agent est récompensée ou pénalisée, et l'objectif est de maximiser la somme des récompenses. Cette méthode est particulièrement utilisée dans les domaines des jeux et de la robotique.

2.4 Apprentissage semi-supervisé

L'apprentissage semi-supervisé combine les approches supervisée et non supervisée. Il est utilisé lorsque l'étiquetage des données est coûteux ou difficile à obtenir. L'algorithme est entraîné sur un petit ensemble de données étiquetées et un grand ensemble de données non étiquetées, tirant parti des deux pour améliorer la performance du modèle.

3 Principaux algorithmes en Machine Learning

3.1 Régression linéaire

La régression linéaire est l'un des algorithmes de Machine Learning les plus simples et les plus utilisés. Elle cherche à modéliser la relation entre une variable indépendante et une variable dépendante en ajustant une ligne droite (ou hyperplan en cas de plusieurs variables) aux données.

3.1.1 Apprentissage Supervisé et problème de Régression

Si vous souhaitez prévoir l'évolution du cours de la bourse, estimer le prix d'un appartement, ou anticiper les variations de température sur Terre, vous êtes en réalité en train de 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.

3.1.2 Comment pouvez-vous créer et comprendre votre premier modèle linéaire pour faire des prédictions précises?

Voici les étapes à suivre pour créer votre premier modèle de Machine Learning.

Machine Learning

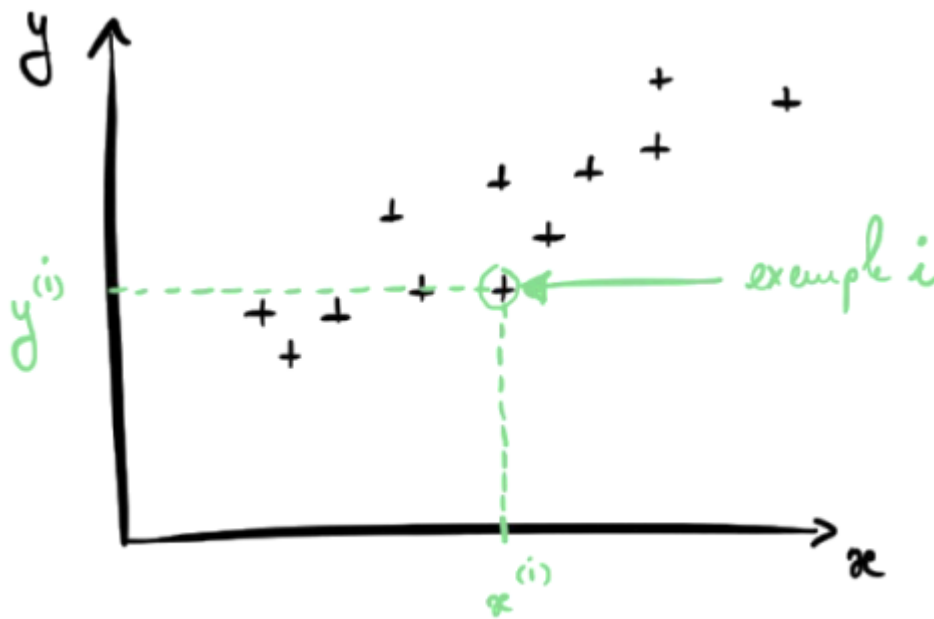
3.1.2.1 Récolter vos données

Supposez que plusieurs agences immobilières vous aient fourni des données sur des appartements à vendre, notamment le prix de l'appartement (y) et la surface habitable (x). En Machine Learning, on dit que vous disposez de m exemples d'appartements.

On désigne :

- $x^{(i)}$ la surface habitable de l'exemple i
- $y^{(i)}$ le prix de l'exemple i

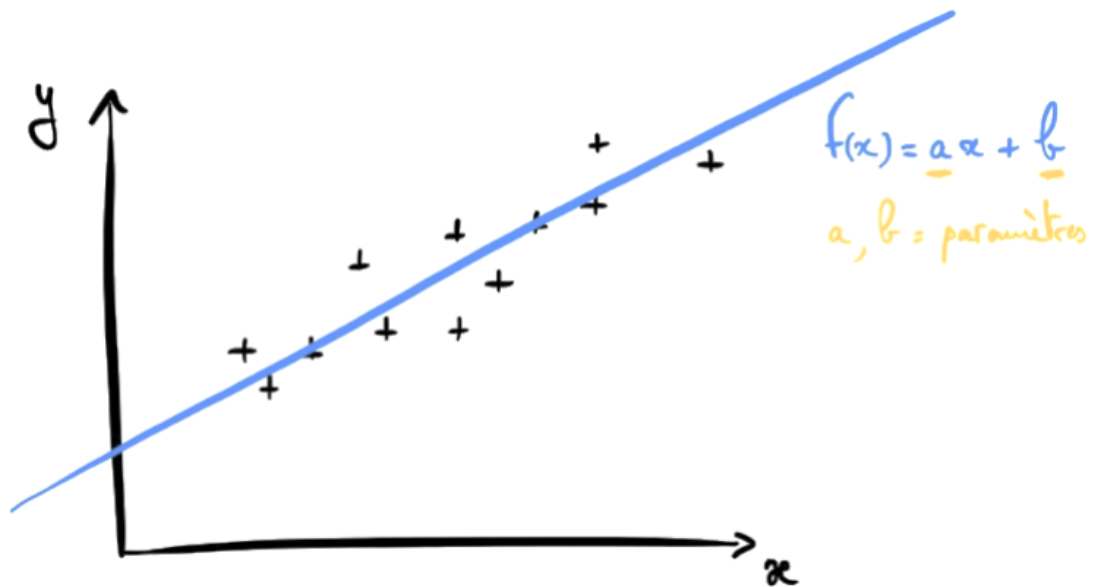
En visualisant votre Dataset, vous obtenez le nuage de points suivant :



3.1.2.2 Créer un modèle linéaire

A partir de ces données, on développe un modèle linéaire $f(x) = ax + b$ où a et b sont les paramètres du modèle. Un bon modèle donne de petites erreurs entre ses prédictions $f(x)$ et les exemples (y) du Dataset.

Nous ne connaissons pas les valeurs des paramètres a et b , ce sera le rôle de la machine de les trouver, de sorte à tracer un modèle qui s'insère bien dans notre nuage de point comme ci-dessous :



3.1.2.3 Définir La Fonction Coût

Pour la régression linéaire, on utilise la norme euclidienne pour mesurer les **erreurs** entre les valeurs prédites $f(x)$ et les vraies valeurs (y).

Concrètement, voici la formule pour exprimer l'erreur i entre le prix $y^{(i)}$ et la prédiction faites en utilisant la surface $x^{(i)}$:

$$\text{erreur}^{(i)} = (f(x^{(i)}) - y^{(i)})^2$$

Exemple :

Imaginez que le 10^{ème} exemple de votre Dataset soit un appartement de $x^{(10)} = 80 \text{ m}^2$ dont le prix s'élève à $y^{(10)} = 100,000 \text{ f}$ et que votre modèle prédise un prix de $f(x^{(10)}) 100,002 \text{ f}$.

L'erreur pour cette exemple est donc :

$$\text{erreur}^{(10)} = (f(x^{(10)}) - y^{(10)})^2$$

$$\text{erreur}^{(10)} = (100,002 - 100,000)^2$$

$$\text{erreur}^{(10)} = (2)^2$$

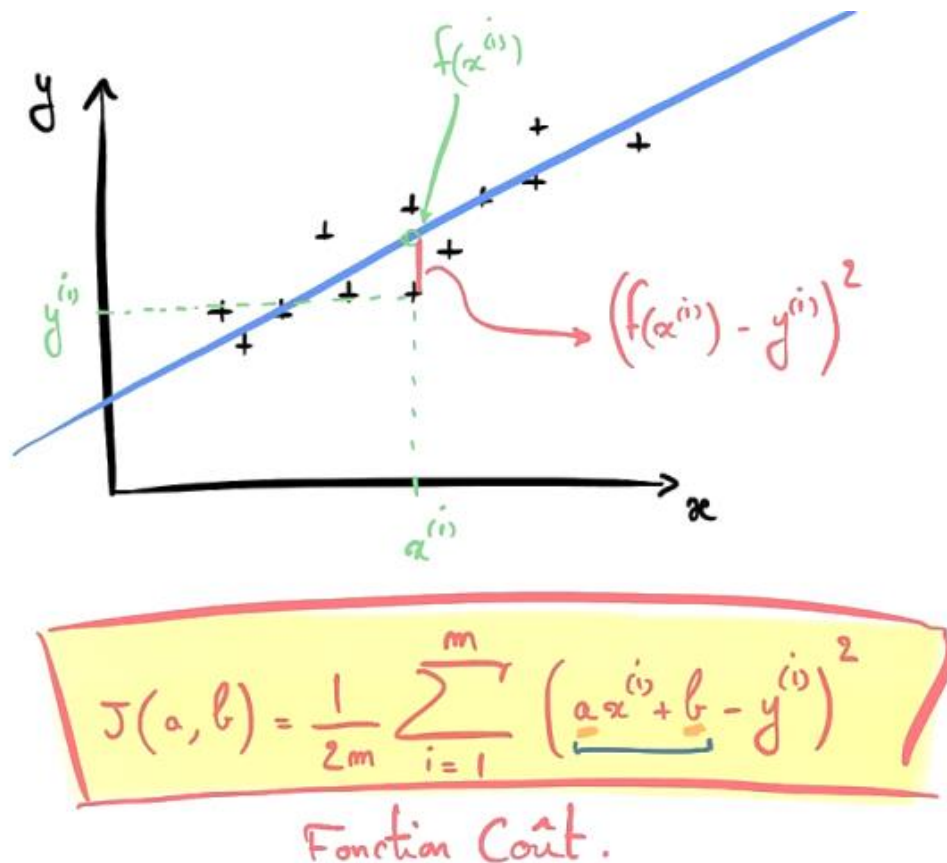
$$\text{erreur}(10) = 4$$

Chaque prédiction s'accompagne d'une erreur, on a donc **m erreurs**. On définit la **Fonction Coût $J(a, b)$** comme étant la moyenne de toutes les erreurs :

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m \text{erreur}^i$$

$$J(a, b) = \frac{1}{2m} \sum_{i=1}^m (f(x^{(i)}) - y^{(i)})^2$$

La fonction **J** est appelée l'erreur quadratique moyenne (Mean Squared Error)



3.1.2.4 Trouver les paramètres qui minimisent la Fonction Coût

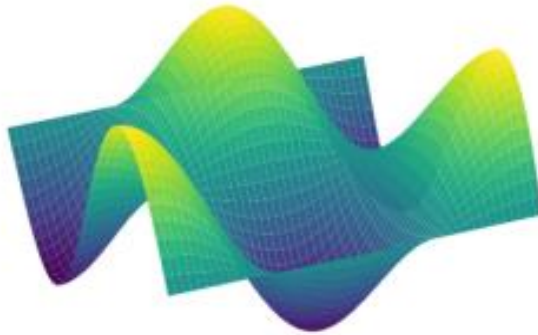
L'étape suivante, qui est la plus enthousiasmante, consiste à permettre à la machine d'apprendre les paramètres qui minimisent la fonction de coût, c'est-à-dire ceux qui nous offrent le modèle le plus performant.

Pour trouver le minimum, plusieurs méthodes sont disponibles : la méthode des moindres carrés et la méthode de la descente de gradient (Gradient Descent). La méthode des moindres carrés consiste à minimiser la somme des carrés des erreurs, mais elle impose parfois à inverser une matrice, ce qui peut être difficile à garantir. Pour contourner ces contraintes, on utilise généralement l'algorithme de descente de gradient.

3.1.2.4.1 Qu'est-ce que la descente de gradient ?

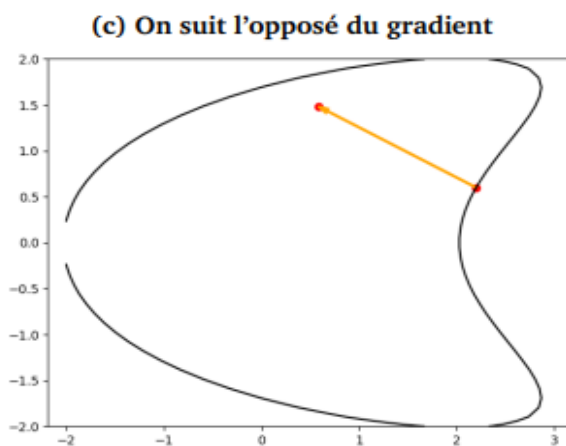
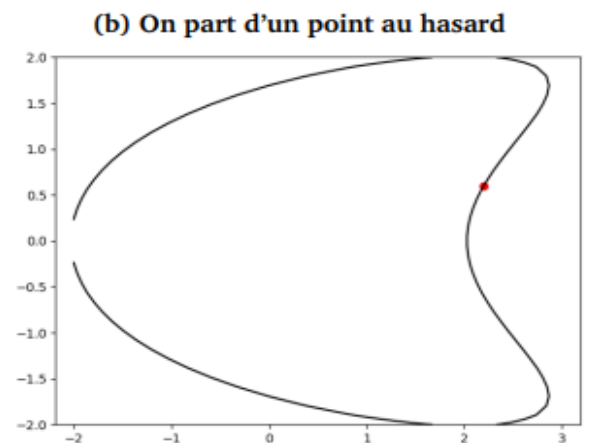
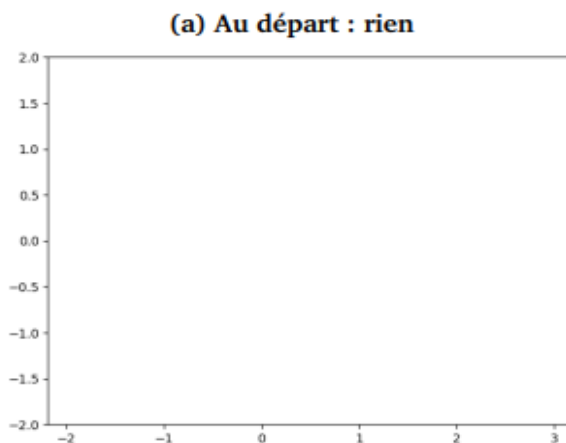
Imaginons une goutte d'eau en haut d'une colline. La goutte d'eau descend en suivant la ligne de plus grande pente et elle s'arrête lorsqu'elle atteint un point bas. C'est exactement ce que fait la descente de gradient : partant d'un point sur une surface, on cherche la pente la plus

grande en calculant le gradient et on descend d'un petit pas, on recommence à partir du nouveau point jusqu'à atteindre un minimum local.

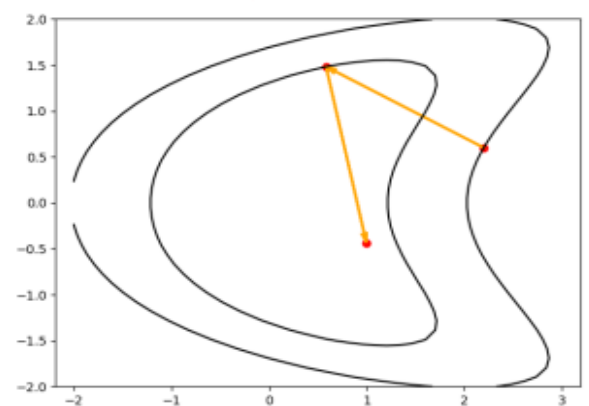


3.1.2.4.2 Comment trouver le minimum ?

On nous donne une fonction f de deux variables (a, b) et nous cherchons un point (a_{\min}, b_{\min}) en lequel f atteint un minimum. Voici la méthode expliquée par des dessins sur lesquels ont été tracées des lignes de niveau.



(d) Depuis le nouveau point, on suit l'opposé du gradient



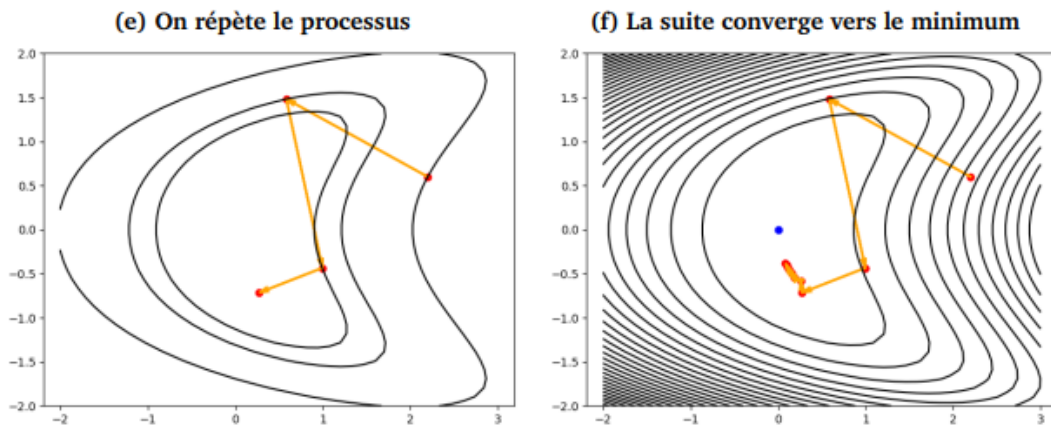
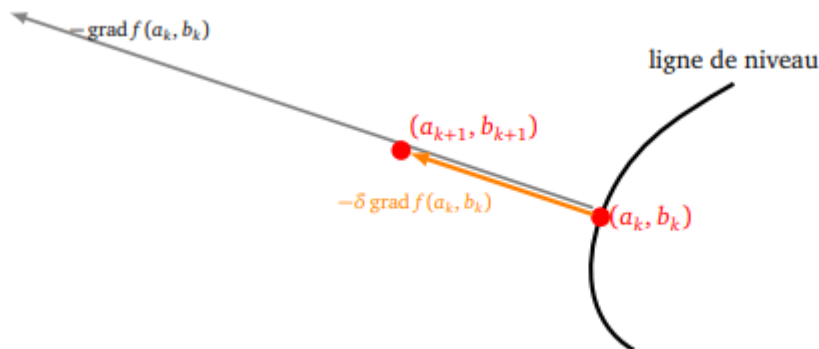


Figure (a). Au départ nous n'avons aucune information globale sur f . La seule opération que l'on s'autorise c'est calculer $\text{grad } f(a, b)$ en certains points.

Figure (b). On choisit un point (a_0, b_0) au hasard. Si on note $c_0 = f(a_0, b_0)$ la valeur de f en ce point, on sait que la ligne de niveau ($f = c_0$) passe par (a_0, b_0) .

Figure (c). On calcule en ce point le gradient de f . On trace l'opposé du gradient : $-\text{grad } f(a_0, b_0)$. On sait d'une part que la ligne de niveau est orthogonale à ce gradient et surtout que dans la direction de $-\text{grad } f(a_0, b_0)$, les valeurs de f vont diminuer.



On se dirige alors dans la direction opposée au gradient d'un facteur δ (par exemple $\delta = 0.1$). On arrive à un point noté (a_1, b_1) . Par construction, si δ est assez petit, la valeur $c_1 = f(a_1, b_1)$ est plus petite que c_0 .

Figure (d). On recommence depuis (a_1, b_1) . On calcule l'opposé du gradient en (a_1, b_1) , on se dirige dans cette nouvelle direction pour obtenir un point (a_2, b_2) où $c_2 = f(a_2, b_2) < c_1$.

Figure (e). On itère le processus pour obtenir une suite de points (a_k, b_k) pour lesquels f prend des valeurs de plus en plus petites.

Figure (f). On choisit de s'arrêter (selon une condition préalablement établie) et on obtient une valeur approchée (a_N, b_N) du point (a_{\min}, b_{\min}) en lequel f atteint son minimum. Évidemment avec la vision globale de la fonction, on se dit qu'on aurait pu choisir un point de départ plus près et que certaines directions choisies ne sont pas les meilleures. Mais souvenez-vous que l'algorithme est « aveugle », il ne calcule pas les valeurs de f en les (a_k, b_k) et n'a pas connaissance du comportement de f au voisinage de ces points.

3.1.2.5 Comment utiliser l'algorithme de Gradient Descent ?

Pour rappel, nous avons jusqu'à présent créé un **Dataset**, développé un **modèle** aux **paramètres** inconnus, et exprimé la **Fonction Coût $J(a, b)$** associée à ce modèle.

Nous allons procéder à la recherche des paramètres **a** et **b** qui minimise la **Fonction Coût $J(a, b)$** .

Pour cela, nous allons choisir **a** et **b** au hasard puis allons utiliser en boucle la descente de gradient pour mettre à jour nos paramètres dans la direction de la **Fonction Coût** la plus faible.

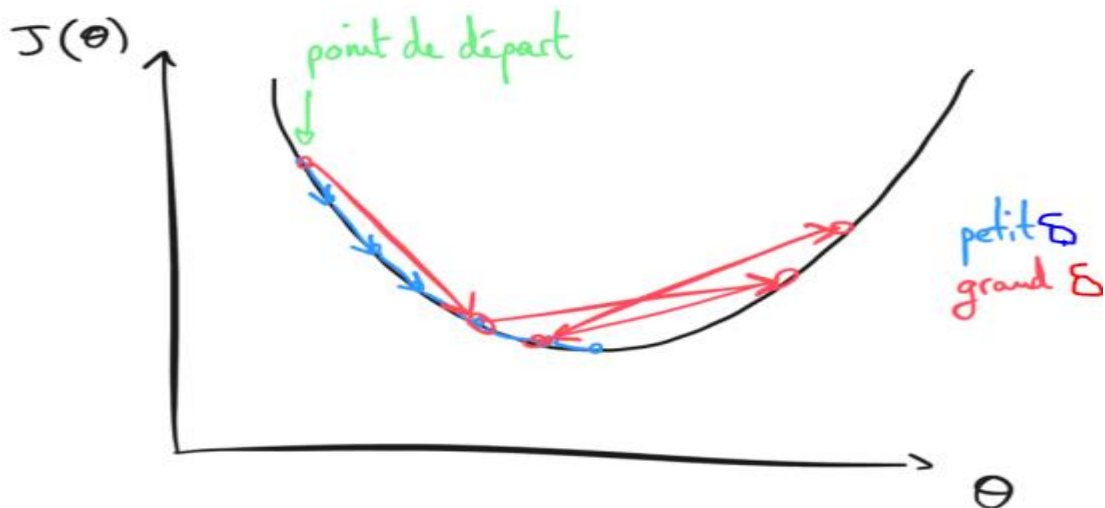
Répéter en boucle:

$$a = a - \delta \frac{\partial J(a,b)}{\partial a}$$

$$b = b - \delta \frac{\partial J(a,b)}{\partial b}$$

Commentaire : à chaque itération de cette boucle, les paramètres **a** et **b** sont mis à jour en soustrayant leur propre valeur à la valeur de la pente $\frac{\partial J(a,b)}{\partial \dots}$ multipliée par la distance à parcourir δ . On appelle δ la vitesse d'apprentissage (**Learning rate**).

Si la vitesse est trop lente, le modèle peut mettre longtemps à être entraîné, mais si la vitesse est trop grande, alors la distance parcourue est trop longue et le modèle peut ne jamais converger. Il est important de trouver un juste milieu. Le dessin ci-dessous illustre ces propos.



3.2 Calcul des dérivées partielles de Fonction Coût $J(a, b)$

Pour implémenter l'algorithme de Gradient Descent, il faut donc calculer les dérivées partielles de la **Fonction Coût**. Rappelons qu'en mathématique, la dérivée d'une fonction en un point nous donne la valeur de sa pente en ce point.

Fonction Coût :

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

Dérivée selon le paramètre a :

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

Dérivée selon le paramètre b :

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

3.3 Utilisation des matrices et des vecteurs

Dans la pratique, on exprime notre **Dataset** et nos **paramètres** sous forme matricielle, ce qui simplifie beaucoup les calculs. On crée ainsi un vecteur $\theta = \begin{pmatrix} a \\ b \end{pmatrix} \in \mathbb{R}^{n+1}$ qui contient tous les paramètres pour notre modèle, un vecteur $y \in \mathbb{R}^{m \times 1}$ et une matrice $X \in \mathbb{R}^{m \times n}$ qui inclut toutes les *features* n .

Dans la régression linéaire, $n = 1$ et on a :

- Un modèle linéaire $F(X) = X \cdot \theta$ où $\theta = \begin{pmatrix} a \\ b \end{pmatrix}$, $X, y \in \mathbb{R}^{m \times 1}$
- la Fonction Coût $J(\theta) = \frac{1}{2m} \sum (F(X) - y)^2$
- l'algorithme de **Gradient Descent**
Répéter en boucle:

$$\theta = \theta - \delta \frac{\partial J(\theta)}{\partial \theta}$$

- **Gradient:**

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T \cdot (F(X) - Y) \text{ où } X^T \text{ est la transposée de la matrice } X.$$

4 Développement du programme de Machine Learning avec Python

Prenons l'exemple de la régression linéaire.

4.1 Importer les librairies

Ouvrir un nouveau Notebook dans Jupyter. Ensuite, il faut importer les librairies et fonctions suivantes :

- **Numpy** pour manipuler notre Dataset en tant que matrice
- **Matplotlib.pyplot** pour visualiser nos données
- La fonction **make_regression** de Sklearn pour générer un nuage de point (ici on va simuler des données)
- **SGDRegressor** (qui signifie **Stochastic Gradient Descent Regressor**) et qui contient le calcul de la **Fonction Coût**, des **gradients**, de l'**algorithme de minimisation**, bref...

Code :

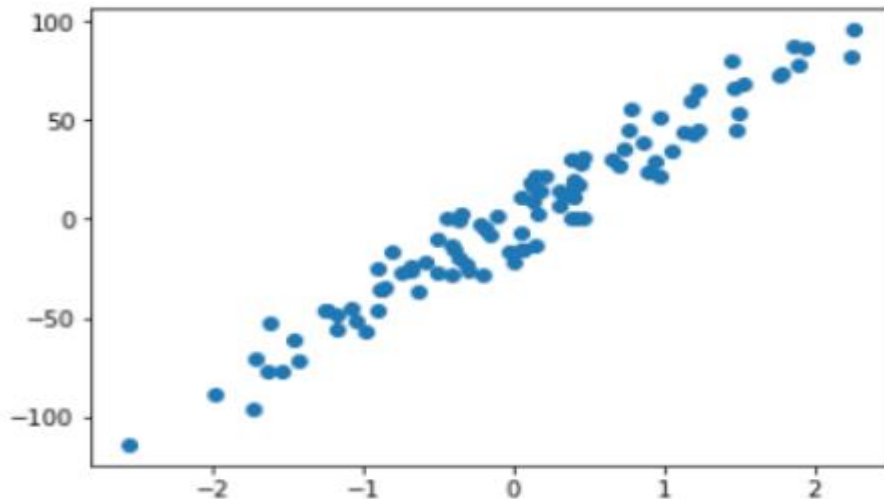
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_regression
from sklearn.linear_model import SGDRegressor
```

4.2 Créer un Dataset

Code :

```
np.random.seed(0) # pour générer les données aléatoires
x, y = make_regression(n_samples=100, n_features=1, noise=10) # générer un
nuage de points
plt.scatter(x, y) # visualiser les données
```

Après exécution



4.3 Développer le modèle et l'entraîner

Pour développer et entraîner un modèle, il a fallu beaucoup de maths dans le chapitre précédent : entre la Fonction Coût, les dérivées, l'algorithme de Gradient Descent...

Dans **Sklearn**, tout cela est déjà fait.

Code :

```
model = SGDRegressor (max_iter=100, eta0=0.0001) # créer un modèle de régression  
linéaire avec Nbrs itération=100 et Learning rate = 0,0001  
model.fit(x,y) # Descente de gradient
```

Nous pouvons évaluer la précision de notre modèle avec la fonction `score`, qui calcule le **coefficient de détermination (R^2)** entre le modèle et les valeurs **y** de notre **dataset**. Nous pouvons également utiliser la fonction **predict** pour faire de nouvelles prédictions, puis tracer les résultats avec **plt.plot**.

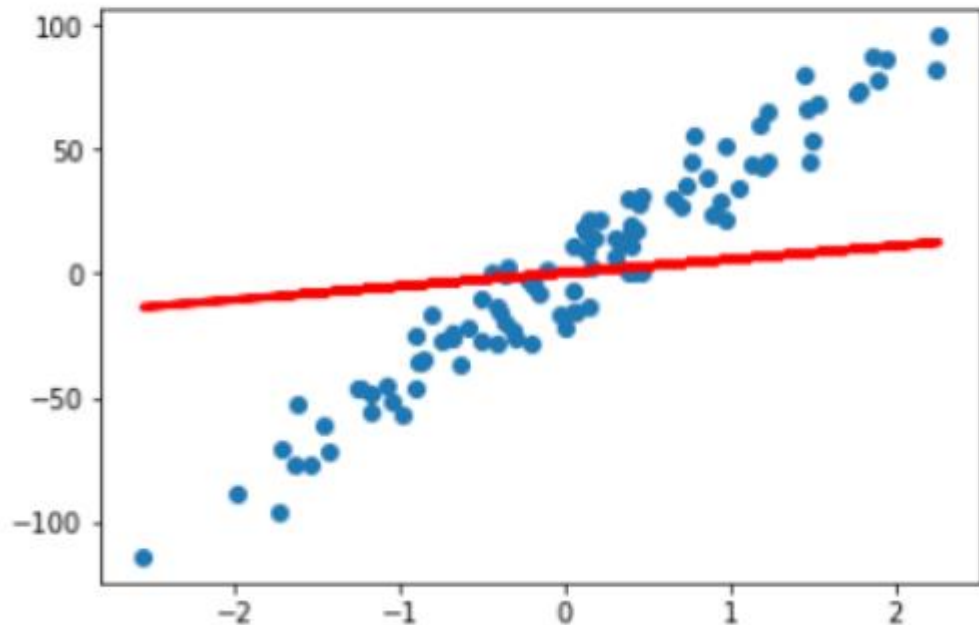
Code :

```
Print ('Coeff R2 =', model.score(x, y))  
plt.scatter (x, y)  
plt.plot (x, model.predict(x), c='red', lw = 3)
```

Après execution:

```
Coeff R2 = 0.22313211770520344
```

```
[<matplotlib.lines.Line2D at 0x1a111f552b0>]
```



Le **coefficient de détermination (R^2)** montre que le modèle est loin d'être bon. C'est parce que nous ne l'avons pas entraîné suffisamment longtemps et parce que le Learning rate était trop faible. Aucun problème, il est possible de le ré-entraîner avec de meilleurs hyper-paramètres.

En Machine Learning, les valeurs qui fonctionnent bien pour la plupart des entraînements sont :

- Nombre d'itérations = 1000
- Learning rate = 0.001

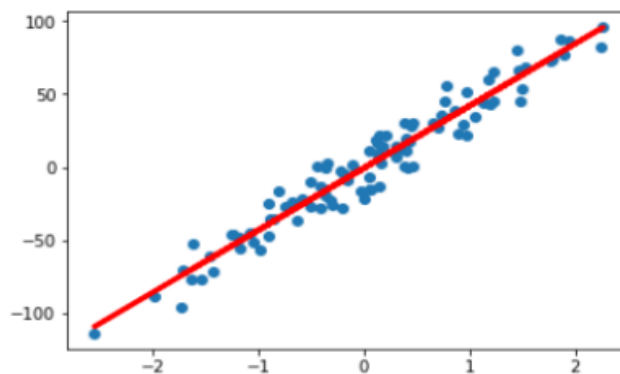
```
In [23]: model = SGDRegressor(max_iter=1000, eta0=0.001)
         model.fit(x, y)
```

```
Out[23]: SGDRegressor(alpha=0.0001, average=False, early_stopping=False, epsilon=0.1,
                       eta0=0.001, fit_intercept=True, l1_ratio=0.15,
                       learning_rate='invscaling', loss='squared_loss', max_iter=1000,
                       n_iter=None, n_iter_no_change=5, penalty='l2', power_t=0.25,
                       random_state=None, shuffle=True, tol=None, validation_fraction=0.1,
                       verbose=0, warm_start=False)
```

```
In [24]: print('Coeff R2 =', model.score(x, y))
         plt.scatter(x, y)
         plt.plot(x, model.predict(x), c='red', lw = 3)
```

```
Coeff R2 = 0.9417290436914625
```

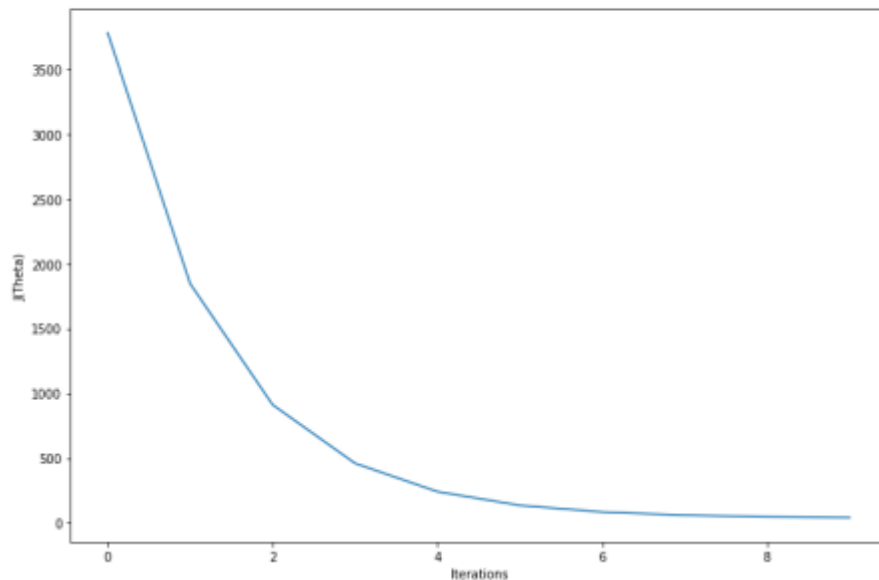
```
Out[24]: [<matplotlib.lines.Line2D at 0x1a11ffcb70>]
```



4.4 Les courbes d'apprentissage

En Machine Learning, on appelle **courbe d'apprentissage** (Learning curves) les courbes qui montrent l'évolution de la **Fonction Coût** au fil des itérations de **Gradient Descent**. Si votre modèle apprend, alors sa **Fonction Coût** doit diminuer avec le temps, comme ci-dessous :

```
|: fig, ax = plt.subplots(figsize=(12, 8))  
ax.set_ylabel('J(Theta)')  
ax.set_xlabel('Iterations')  
ax.plot(range(iterations), cost_history)
```



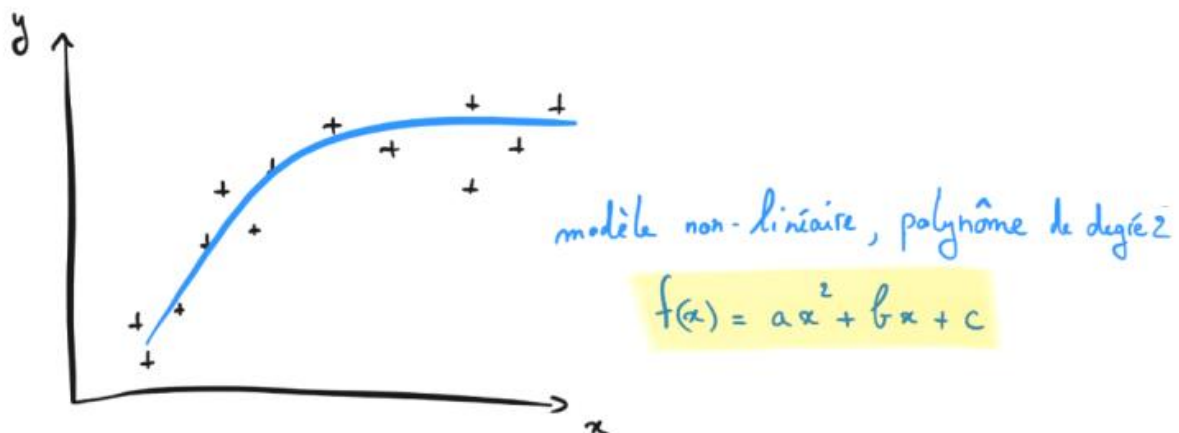
4.5 Régression Polynomiale à plusieurs variables

La régression polynomiale est une forme d'analyse de régression dans laquelle la relation entre la variable explicative et la variable expliquée est modélisée comme un polynôme.

Remarque : La régression linéaire est une régression polynomiale de degré 1.

Pour le nuage de points ci-dessous, il serait pertinent de développer un modèle polynomial de degré 2.

$$f(x) = ax^2 + bx + c$$



Machine Learning

Pour un model de régression polynomiale, le code que nous avons écrit pour la régression linéaire peut être utilisé. Il suffit de générer des variables polynômiales dans notre **Dataset** en utilisant la fonction **PolynomialFeatures** présente dans Sklearn.

Code :

```
from sklearn.preprocessing import PolynomialFeatures
```

Grâce au calcul matriciel (présent dans Numpy et Sklearn) la machine peut intégrer ces nouvelles variables polynômiales sans changer son calcul !

5 Arbres de décision et forêts aléatoires

Les arbres de décision sont des modèles qui segmentent l'espace des données en sous-espaces en fonction de critères de décision. Ils sont intuitifs et faciles à interpréter. Les forêts aléatoires sont une extension des arbres de décision qui construisent une multitude d'arbres de décision sur des sous-ensembles des données, puis agréger leurs prédictions pour obtenir une meilleure robustesse et précision.

6 Applications du Machine Learning

6.1 En finance

Le Machine Learning est utilisé en finance pour des tâches telles que le trading algorithmique, où les modèles prédisent les mouvements de marché, et le scoring de crédit, où ils évaluent la solvabilité des emprunteurs.

6.2 En santé

En santé, le Machine Learning permet de développer des modèles de diagnostic qui aident à identifier des maladies à partir d'images médicales, et d'autres modèles qui proposent des traitements personnalisés en fonction des caractéristiques spécifiques des patients.

6.3 Dans d'autres domaines

D'autres domaines d'application incluent le commerce de détail, où le Machine Learning est utilisé pour recommander des produits, et le marketing, où il permet de cibler les campagnes publicitaires de manière plus précise.

Conclusion

En conclusion, le Machine Learning représente une avancée majeure dans le traitement et l'analyse des données, permettant de développer des modèles capables d'apprendre et de s'améliorer avec l'expérience. À travers ce document, nous avons parcouru les fondements théoriques de cette discipline, en explorant ses principales techniques, telles que l'apprentissage supervisé, non supervisé et par renforcement, ainsi que leurs applications dans divers domaines. Ces approches offrent des solutions puissantes pour des problèmes complexes, bien que des défis subsistent, notamment en matière d'interprétabilité des modèles et de gestion des biais.

Malgré ces défis, le potentiel du Machine Learning pour transformer des ensembles de données en informations précieuses est indéniable. À mesure que cette technologie évolue, elle continuera d'avoir un impact profond sur divers secteurs, de la finance à la santé. Pour maximiser les bénéfices tout en minimisant les risques, il sera essentiel de favoriser une collaboration étroite entre les spécialistes des données, les experts en éthique et les décideurs. Le Machine Learning, bien qu'encore en développement, est sans aucun doute appelé à jouer un rôle central dans l'avenir de la technologie et de la société.

Bibliographie

Azencott, C.-A. *Introduction au Machine Learning*.

Deepmath - Mathématiques des réseaux de neurones. *Descente de gradient*.

Michel, H. (2020). Régression polynomiale avec python. *Le Data Scientist* .

Saint-Cirgue, G. (2019). *Apprendre-le-ML-en-une-semaine*. Royaume-Uni.