Linear Regression

Léo

March 2025

Sommaire

1 Introduction

Les systèmes de ML peuvent être classés en fonction du taux et du type de supervision dont ils ont besoins pendant leur phase d'entraînement. Il y a 4 catégories principales : apprentissage supervisé, apprentissage non supervisé, apprentissage semi-supervisé et l'apprentissage par renforcement.

2 Apprentissage supervisé

Comme son nom l'indique, il consiste à superviser l'apprentissage de la machine, de la même manière qu'un professeur supervise l'apprentissage de ses élèves en leur montrant des exemples de questions / réponses qu'ils doivent apprendre.
La Machine étudie les exemples qu'on lui montre, et grâce à un système d'auto-évaluation et d'auto-amélioration, elle parvient à apprendre à réaliser la tâche qu'on lui demande d'effectuer.

En apprentissage supervisé, l'ensemble d'entraînement que l'on utilise pour entraîner notre algo inclut les solutions voulues. ces dernières sont les **labels**.

Une tâche typique en apprentissage supervisé est **la classification**. le filtre de spam est un exemple : il est entraîné avec une grande quantité d'exemples d'emails, et il doit apprendre à classifier les nouveaux emails qu'il aura à traiter.

Une autre tâche commune est de prédire une valeur cible numérique, comme le prix d'une voiture, à partir

d'un ensemble de features (âge, marque, etc.) appelés **prédicteurs**. C'est ce que l'on appelle une **Régression**. Pour entraîner le système, on a besoin de lui donner beaucoup d'exemples (ici, de voitures), dont leurs prédicteurs et leurs labels (ici, leurs prix).

> Un attribut est un **type de donnée** alors qu'une feature a plusieurs sens, en fonction du contexte. En général, une **feature** est un attribut + sa valeur (Ex : "âge = 21").

Algorithmes d'apprentissage supervisé les plus importants :

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVM)
- Decision Trees & Random Forests
- Neural Networks (**Attention !** : *Certaines architectures de NN peuvent être de l'ordre du non supervisé, comme les encodeurs ou les restricted Boltzmann machines*).

Cas d'utilisation :

Différencier Approche classique / déterministe et Machine Learning.

Déterministe :
- Exacte : Programmation dynamique
- Approximative : Ratio d'approximation
- Approchée : Heuristique, Métaheuristique

Problèmes efficacement résolvables par ML :
- Détermination du meilleur moment pour publier un contenu sur une page web
- Estimation du prix qu'une oeuvre pourrait atteindre lors d'une vente aux enchères.
- Développement d'un programme qui détermine si un article traite un sujet, après avoir annoté un millier d'articles en fonction de leurs sujets en utilisant la fréquence des mots clés. (Supervisé : variable à expliquer = sujet de l'article)
- Proposition de catégories pour un nouveau lien HTML en se basant sur les catégories déjà définies par les utilisateurs. (Supervisé : Classification de nouveaux liens à partir d'autres déjà étiquettés)
- Détection de fraude grâce aux données bancaires. (Apprentissage à partir des données, détection si une transaction est une fraude, métrique pour évaluer l'efficacité du modèle)

Rappel : Etapes indispensables pour résoudre un problème d'Apprentissage Supervisé :
- **Préparation des données :** importer un dataset contenant des features et une target. Noter dimension, nb de lignes et de features.
- **Définir un modèle :** développement d'un modèle avec ses paramètres
- **Fonction de coût :** développement d'une fonction de coût associée à ce modèle
- **Evaluation du modèle :** développement d'un algorithme de minimisation des erreurs

2.1 Régression Linéaire

= Modèle le plus simple.

2.1.1 Contexte

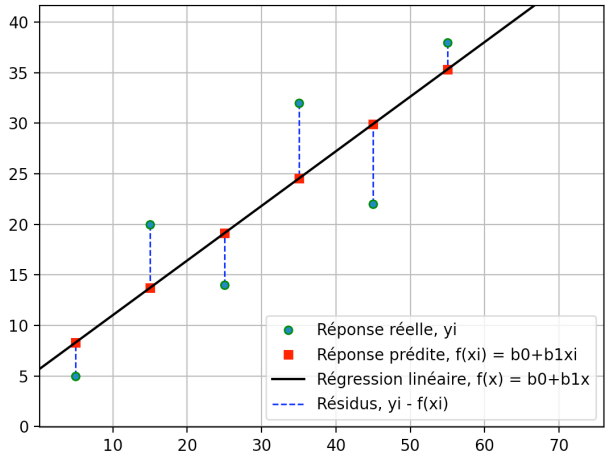
Imaginons que l'on veuille prédire le prix d'une maison en fonction de sa surface habitable. Pour ce faire, on montre des exemples de maisons à notre machine (avec leurs surfaces habitables et leurs prix).
En apprentissage supervisé, ces exemples de questions / réponses sont présentés à la machine sous forme de jeu de données (X, y), où X représente les variables d'entrées, et y la sortie attendue.

Grâce à ce jeu de données, la machine est capable d'apprendre un modèle permettant de prédire la valeur de y en fonction de X. Pour ce faire, elle effectue une auto-évaluation en recherchant le modèle qui lui offre les meilleures performances par rapport au jeu de données fourni.
Une fois ce modèle développé, il est possible de s'en servir pour faire de futures prédictions, et ainsi prédire le prix de nouvelles maisons.

2.1.2 En détails

Le modèle de Régression Linéaire peut être décrit par l'équation $y = ax + b$, avec y la valeur cible (Ex : le prix). Cette équation représente une droite.

Les paramètres a et b contrôlent l'allure de la droite : a détermine son inclinaison et b son ordonnée à l'origine.
L'objectif est de trouver les paramètres a et b qui permettent au modèle de s'ajuster au mieux au jeu de données (X, y). Et c'est la machine qui va s'en charger grâce à un algorithme d'optimisation. Ce dernier cherche à **minimiser** les erreurs entre le modèle et les points de données. La mesure de ces erreurs est la **fonction de coût**.



Cette fonction de coût mesure l'erreur entre chaque point de données et le modèle, puis fait la moyenne de toutes ces erreurs. Pour n'obtenir que des erreurs (ou résidus) positifs, on élève chaque erreur au carré. C'est ce que l'on appelle la **distance euclidienne** (D'où la somme au carré).
Enfin, on calcule la somme puis la moyenne de ces erreurs, ce qui donne la perte gobale du modèle.
En somme, on cherche le modèle qui fit le mieux notre nuage de point.
Fonction coût :

$$J = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \tag{1}$$

où, $\hat{y}$ est la valeur prédite, m le nombre de features, y_i est la i-ième valeur de la feature.

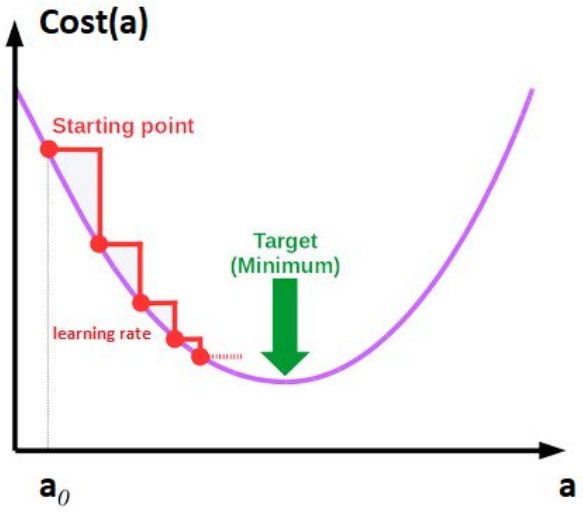
Lorsque l'on fait varier les paramètres a et b du modèle, la fonction coût prend la forme d'une parabole. On cherche alors à trouver le minimum de cette courbe.
Cependant, la machine ignore complétement où se trouve ce minimum. C'est pourquoi il est nécessaire de développer un algorithme qui lui permettra de le trouver automatiquement : la **Descente de Gradient**.

2.2 Descente de Gradient

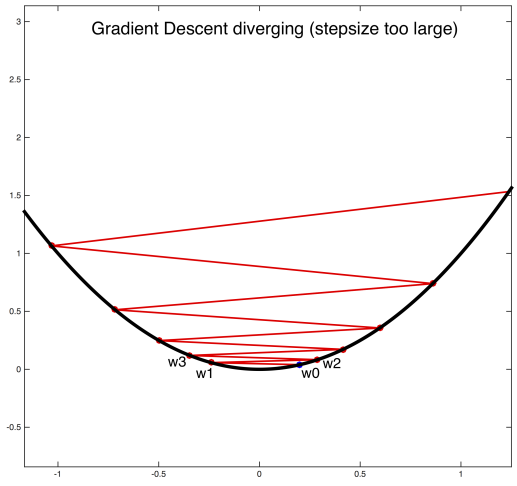
La descente de Gradient est l'un des algos les plus utilisés en ML. Il consiste à calculer le gradient de la fonction coût, i.e comment celle-ci évolue lorsque a et b varient légérement, pour ensuite faire un pas dans la

direction où la fonction coût diminue.

L'algorithme démarre avec des paramètres (a_0, b_0) choisis au hasard. On effectue ensuite un pas dans la direction de la descente, ce qui nous donne de nouvelles valeurs pour (a, b), notées (a_1, b_1). On calcule à nouveau le gradient au nouveau point afin de poursuivre la descente. On finit par converger vers le minimum de la fonction coût.



La taille des pas est un paramètre important. Elle est déterminée par l'hyperparamètrage du taux d'apprentissage α. Si ce taux est trop petit, alors l'algo va devoir passer par beaucoup d'itérations pour converger, ce qui prendra énormément de temps.
A l'inverse, si le taux d'apprentissage est trop grand, il va sauter des étapes et passer de l'autre côté de la courbe. Ca tendra à faire diverger l'algorithme, avec des valeurs de plus en plus grandes, et donc de faire échouer le processus.



(Attention ! Toutes les fonctions de coût ne ressemblent pas à des bols réguliers. Il peut y avoir des trous, des crêtes, des plateaux et toutes sortes de terrains irréguliers, rendant la converge vers un minimum difficile.)

2.3 Exercice 1

On cherche à prédire le prix de vente d'un produit en fonction des dépenses publicitaires. On suppose que la relation entre x et y suit un modèle linéaire :

$$y = ax + b \tag{2}$$

On a :

Publicité (k)	Prix (k)
1	5
2	7
3	9
4	11
5	13

2.3.1 Solution analytique

On cherche à déterminer les moindres carrées pour estimer a et b.
Dans notre exemple, on peut calculer une à une les solutions car on a un petit dataset mais très peu judicieux voire impossible pour de plus gros jeux de données.

Pour rappel, notre fonction de coût est la suivante :

$$J = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 \tag{3}$$

Appliquée à notre modèle linéaire, on a :

$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m}(ax_i + b - y_i)^2 \tag{4}$$

L'allure de cette fonction quadratique est une parabole. On cherche le minimum local de cette courbe. Pour ce faire, on utilise la Descente de Gradient dont les dérivées partielles sont :

—
$$\frac{\partial J(a,b)}{\partial a} = 0 \tag{5}$$

—
$$\frac{\partial J(a,b)}{\partial b} = 0 \tag{6}$$

(= 0 car c'est à ce point que le minimum local est atteint).

A partir de notre modèle linéaire, on obtient :

—
$$\frac{\partial J(a,b)}{\partial a} = 0 => \frac{\partial}{\partial a}\frac{1}{2m}\sum_{i=1}^{m}(ax_i + b - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}2(ax_i + b - y_i)x_i = \sum_{i=1}^{m}(ax_i + b - y_i)x_i = 0 \tag{7}$$

—
$$\frac{\partial J(a,b)}{\partial b} = 0 => \frac{\partial}{\partial b}\frac{1}{2m}\sum_{i=1}^{m}(ax_i + b - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}2(ax_i + b - y_i)(1) = \sum_{i=1}^{m}(ax_i + b - y_i) = 0 \tag{8}$$

De ces équations, on a alors :

$$a\sum_{i=1}^{m} x_i + mb - \sum_{i=1}^{m} y_i = 0 \tag{9}$$

Et on peut déterminer l'expression de b :

$$b = \frac{\sum_{i=1}^{m} y_i - a\sum_{i=1}^{m} x_i}{m} \tag{10}$$

Déterminons maintenant l'expression de a :

$a\sum_{i=1}^{m} x_i + m\sum_{i=1}^{m} x_i - \sum_{i=1}^{m} x_i y_i = 0$

$\iff a\sum_{i=1}^{m} x_i + b\sum_{i=1}^{m} x_i - \sum_{i=1}^{m} x_i y_i = 0$

$\iff a\sum_{i=1}^{m} x_i + (\frac{\sum_{i=1}^{m} y_i - a\sum_{i=1}^{m} x_i}{m})\sum_{i=1}^{m} x_i - \sum_{i=1}^{m} x_i y_i = 0$

$\iff am\sum_{i=1}^{m} x_i^2 + (\sum_{i=1}^{m} y_i - a\sum_{i=1}^{m} x_i)\sum_{i=1}^{m} x_i - m\sum_{i=1}^{m} x_i y_i = 0$

$\iff am\sum_{i=1}^{m} x_i^2 + \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i - a(\sum_{i=1}^{m} x_i)^2 \sum_{i=1}^{m} x_i - m\sum_{i=1}^{m} x_i y_i = 0$

$\iff a = \frac{m\sum_{i=1}^{m} x_i y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i}{m\sum_{i=1}^{m} x_i^2 - (\sum_{i=1}^{m} x_i)^2}$

Calculons a et b avec les valeurs numériques et informations données par le tableau de l'énoncé :

$$\begin{cases} a = \frac{5\cdot(1\times5+2\times7+...+5\times13)-((1+2+...+5)(5+7+...+13))}{5\cdot(1^2+2^2+...+5^2)-(1+2+...+5)^2} = 2 \\ \\ b = \frac{(5+...+13)-2\cdot(1+2+...+5)}{5} = 3 \end{cases} \tag{11}$$

Le modèle optimal est donc $y = 2x + 3$. Généralement, on préfera une représentation matricielle.

2.3.2 Application de la Descente de Gradient (3 itérations)

Initialement, $a = 0, b = 0$ et $\alpha = 0.01$.

Pour chaque itération :

$$\begin{cases} a := a - \alpha\frac{\partial J(a,b)}{\partial a} \\ \\ b := b - \alpha\frac{\partial J(a,b)}{\partial b} \end{cases} \tag{12}$$

3 itérations :

(1) :
$a = -0.01 \times \frac{1}{5}(-45) \times 15 = 0,31$
$b = -0.01 \times \frac{1}{5}(-45) = 0,09$

(2) :
$a = 0,31 - 0.01 \times \frac{1}{5}(0,31 \times 15 + 0,09 - 45) \times 15 = 0,58$
$b = 0,09 - 0.01 \times \frac{1}{5}(0,31 \times 15 + 0,09 - 45) = 0,17$

(3) :
$a = 0,58 - 0.01 \times \frac{1}{5}(0,58 \times 15 + 0,17 - 45) \times 15 = 0,82$
$b = 0,17 - 0.01 \times \frac{1}{5}(0,58 \times 15 + 0,17 - 45) = 0,24$

2.3.3 Observations

$a = 2$ et $b = 3$.
a et b avance. Or b avance d'un plus petit pas.

Nous ne sommes pas garantit d'atteindre ces 2 valeurs. Ceci est impacté par le α. Si ce dernier est trop petit => coûteux. Si il est trop grand => Divergence (le min n'est jamais atteint).

Note : Il existe des méthodes avec un α dynamique mais plus de calculs à effectuer car α est recalculé à chaque itération.

2.4 Exercice 2 :

Considérons un problème de régression linéaire où on cherche la fonction f qui minimise :

$$min_{f \in F} \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{13}$$

qui se ramène à chercher une constante w_0 et un vecteur de poids $w \in \mathbb{R}^p$ solution du problème

$$min_{w_0 \in \mathbb{R}, w \in \mathbb{R}^p} \sum_{i=1}^{n} (y_i - w_0 - x_i^T w)^2 \tag{14}$$

, où $x_i^T w = \sum_{j=1}^{p} x_{ij} w_j$

On suppose que l'on dispose d'une seule observation telle que $x_1 = -2$ et $y_1 = 1$.

2.4.1 Solution exacte

On remplace dans l'expression générale par les observations initiales :

$$J(\theta) = (y_1 - w_0 - x_1 w_1)^2 = (1 - w_0 + 2w_1)^2$$

On calcule maintenant les dérivées partielles de $J(\theta)$ en fonction de w_0 et w_1 :

$$\begin{aligned} \frac{\partial J(\theta)}{\partial w_1} &= 4(1 - w_0 + 2w_1) = 0 \\ &\implies (1 - w_0 + 2w_1) = 0 \end{aligned}$$

$$\begin{aligned} \frac{\partial J(\theta)}{\partial w_0} &= (-2)(1 - w_0 + 2w_1) = 0 \\ &\implies (1 - w_0 + 2w_1) = 0 \end{aligned}$$

On remarque que l'on obtient deux solutions identiques ce qui signifie que w n'est pas déterminé de manière unique. En effet, la fonction de coût est une parabole ayant une infinité de solutions satisfaisant $w_0 = 1 + 2w$. Il n'y a pas assez de solutions car il n'y a probablement pas assez de données. On peut toujours chercher le modèle adéquat mais sans une quantité de données nécessaire, on ne peut pas bien définir les contraintes et donc trouver le bon modèle pour cette situation.

2.4.2 Application de la Descente de Gradient

Initialement, $w_0 = w = 0$ et $\alpha = 0.1$.

$$(1) : \begin{cases} w_0 = w_0 - 4\alpha(1 - w_0 + 2w_1) = 0,2 \\ \\ w_1 = w_1 + 2\alpha(1 - w_0 + 2w_1) = -0,4 \end{cases}$$

$$(2) : \begin{cases} w_0 = 0,2 - 4 \times 0.1 \times (1 - 0,2 + 2(-0,4)) = 0,2 \\ \\ w_1 = -0,4 + 2 \times 0.1 \times (1 - 0,2 + 2(-0,4)) = -0,4 \end{cases}$$

$$(3) : \begin{cases} w_0 = 0,2 \\ \\ w_1 = -0,4 \end{cases}$$

La mise à jour atteint l'optimum en une seule itération ! Dès la deuxième itération, les gradients deviennent nuls, signifiant que l'on est déjà au minimum de la fonction coût.
Cela s'explique par la simplicité du problème et le peu de données (un seul point de données ici). En pratique, pour des problèmes plus complexes (données multiples, non-linéarités), la descente de gradient nécessiterait plus d'itérations et un choix plus fin du taux d'apprentissage.

Autres exemples :

(a) : $w_0 = 0, \alpha = 1.$ => Taux d'apprentissage trop grand -> Valeurs illogiques.

(b) : $w_0 = 0, \alpha = 0.001$ => Taux d'apprentissage trop faible -> Petits sauts et donc extrêmement coûteux car besoin de beacoup d'itérations.

2.4.3 Passage à la forme matricielle

On cherche à passer à la forme matricielle. Pour rappel, voici la fonction de coût du modèle :

$$f(w_0, w_1) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w_1 x_i - w_0)^2 \tag{15}$$

Tous les y seront dans un vecteur, de y_1 jusqu'à y_m (car il y a m itérations) :

$$y = \begin{bmatrix} y_1 \\ ... \\ y_m \end{bmatrix}$$

Les paramètres w_0 et w_1 seront dans un vecteur de paramètres W :

$$W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

Et X une matrice qui va contenir toutes les entrées x, de $x_1^{(1)}$ jusqu'à $x_1^{(m)}$:

$$\begin{bmatrix} 1 & x_1^{(1)} \\ 1 & ... \\ 1 & x_1^{(m)} \end{bmatrix}$$

Ainsi :

$$y_i - w_1 x_i - w_0 = y_i - \begin{bmatrix} 1 & x_i \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

Et :

$$y - XW = \begin{bmatrix} y_1 \\ ... \\ y_m \end{bmatrix} - \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & ... \\ 1 & x_1^{(m)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$= \begin{bmatrix} y_1 - (w_0 + w_1 x_1^{(1)}) \\ y_2 - (w_0 + w_1 x_1^{(2)}) \\ ... \\ y_m - (w_0 + w_1 x_1^{(m)}) \end{bmatrix}$$

Pourquoi la transposée ? :

Puisque $y - XW$ est un vecteur colonne de dimension $m \times 1$, sa transposée $(y - XW)^T$ devient un vecteur de ligne de dimension $1 \times m$. Le produit $(y - XW)^T(y - XW)$ de ces deux matrices donne :

$$(1 \times m)(m \times 1) = 1 \times 1$$

Ceci est un scalaire, et c'est exactement ce que nous voulons, car la fonction de coût doit être un nombre réel représentant la somme des erreurs au carré.

Cette transposée, une fois insérée dans notre expression est aussi bien pratique puisque dans J la première ligne est celle qui contient tous les x et la deuxième ligne est celle qui contient tous les 1. Cette façon permet d'optimiser et de calculer tous les gradients du modèle avec la même expression.

2.4.4 Calcul du Gradient de J

Nous savons, d'après la partie précédente, que :

$$J(W) = \frac{1}{2}(y - XW)^T(y - XW)$$

Et vérifier que :

$$\nabla J(W) = X^T(XW - y)$$

Pour rappel, le Gradient $\nabla J(W) = \frac{\partial J(W)}{\partial W}$

On commence par développer l'expression de $J(W)$:

$$J(W) = \frac{1}{2}[y^T y - y^T XW - W^T X^T y + W^T X^T XW]$$

Or $y^T XW$ est un scalaire et identique à $(W^T X^T y)^T y$, alors :

$$J(W) = \frac{1}{2}[y^T y - 2W^T X^T y + W^T X^T XW]$$

Calcul du Gradient $\nabla J(W)$:

Nous dérivons $J(W)$ par rapport à W.
- le terme $y^T y$ est indépendant de W, donc sa dérivée est nulle.
- Le terme $-2W^T X^T y$ est linéaire en W, donc sa dérivée est : $-2X^T y$
- Le terme $W^T X^T XW$ est de la forme $W^T AW$, donc la dérivée est : $2X^T XW$

Ainsi :

$$\nabla J(W) = \frac{1}{2}(2X^T XW - 2X^T y) = X^T XW - X^T y$$

Après factorisation :

$$\nabla J(W) = X^T(XW - y)$$