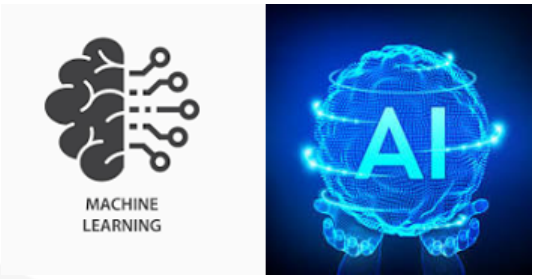
Apprentissage artificiel



Dr. BELAROUCI Sara

Université de Tlemcen

Département de génie biomédical

Email : *sara.belarouci@univ-tlemcen.dz*

1.0

Février 2024

Table des matières

I Chapitre 2 : Les différents modes d'apprentissage

1. Objectifs

À la fin de ce chapitre, l'étudiant sera capable de :

- ***Décrire*** les principes fondamentaux de l'apprentissage automatique.
- ***Distinguer*** les différentes techniques de l'apprentissage supervisé.
- ***Examiner*** les applications de l'apprentissage supervisé dans divers domaines.
- ***Évaluer*** différents modèles d'apprentissage supervisé et leur adéquation à des tâches spécifiques.

2. Introduction

En 2019, le Machine Learning était omniprésent, offrant des avancées significatives dans des domaines variés comme la médecine, notamment dans le diagnostic précoce du cancer.

Ce chapitre se focalise sur les diverses méthodes d'apprentissage automatique, en particulier la classification, et explore les techniques d'évaluation cruciales pour évaluer la performance de ces modèles.

Cf. "Qu'est-ce que la Data Science et l'Intelligence artificielle (IA) ?"

3. Les fondations du Machine Learning

3.1. Comprendre pourquoi le Machine Learning est utilisé ?

Pour comprendre ce qu'est le Machine Learning, voyons pourquoi on l'utilise. Imaginez-vous face à des défis comme construire un pont plus solide, trouver comment gagner plus d'argent, ou même comment guérir le cancer. Pour nous aider, nous avons les ordinateurs, capables de résoudre des calculs complexes rapidement. Mais voilà, les ordinateurs ne peuvent faire qu'une seule chose : résoudre des calculs qu'on leur donne [10].

Alors, deux situations se présentent[10]:

- Soit on sait comment résoudre le problème.
  Dans cette situation, il est simple : nous introduisons le calcul dans l'ordinateur, ce processus est appelé la ***programmation***, et ensuite l'ordinateur nous fournit le résultat.
  Exemple :`concevoir un pont`.
- Soit, on est bloqué, car on ne sait pas le calcul qui résout notre problème. Il est alors impossible de fournir à un ordinateur un calcul que nous ne connaissons pas, c'est un peu comme essayer de poster une lettre que nous n'avons pas écrite.
  Exemples :`Reconnaître des visages sur une photo`, `Guérir le cancer`, `Conduire une voiture...`

C'est là que le ***Machine Learning*** intervient. Il permet aux ordinateurs d'apprendre à partir de données pour résoudre des problèmes pour lesquels on n'a pas de solution toute prête.

3.2. Laisser la Machine apprendre à partir d'expériences

Dans le domaine du Machine Learning, l'idée est de permettre à l'ordinateur d'apprendre quel calcul effectuer plutôt que de lui donner explicitement les instructions. Cette approche, conceptualisée par Arthur Samuel, un mathématicien américain, se concentre sur le développement de programmes capables d'apprendre par l'expérience [10].

3.3. Mais comment apprendre ?

Pour doter un ordinateur de la capacité d'apprendre, nous utilisons des méthodes d'apprentissage largement inspirées de la façon dont nous, les êtres humains, apprenons à réaliser des tâches.

Parmi ces méthodes, on compte notamment :

- L'apprentissage supervisé (Supervised Learning)
- L'apprentissage non supervisé (Unsupervised Learning)
- L'apprentissage semi-supervisé (Semi-Supervised Learning)
- L'apprentissage par renforcement (Reinforcement Learning)
- L'apprentissage par transfert (Transfer Learning)

4. Apprentissage Supervisé

4.1. Définition d'apprentissage supervisé

Définition

L'apprentissage supervisé (Supervised learning) est peut-être le type de problèmes de machine Learning le plus facile à appréhender : son but est d'apprendre à faire des prédictions, à partir d'une liste d'exemples étiquetés, c'est-à-dire accompagnés de la valeur à prédire. Les étiquettes servent de « *professeur* » et supervisent l'apprentissage de l'algorithme [10].

Fondamental

Pour bien maîtriser l'apprentissage supervisé, il est essentiel de comprendre et de connaître les quatre notions suivantes[10]: ***Le Dataset***, ***Le Modèle et ses paramètres***, ***La Fonction Coût*** et ***L'Algorithme d'apprentissage***.

En comprenant ces notions, on peut mieux appréhender et utiliser efficacement l'apprentissage supervisé pour résoudre divers problèmes de prédiction et de classification.

4.2. Les 4 notions clés du Machine Learning

Les 4 notions clefs du Machine Learning que vous devez absolument retenir [10]:

4.2.1. Notion 1 : Apprendre à partir d'exemples (Dataset)

Dans l'apprentissage supervisé, les données sont utilisées pour entraîner et tester le modèle et aussi constituées d'une collection d'objets décrits par des caractéristiques (attributs). Ces caractéristiques fournissent des informations sur l'étiquette de chaque objet (variable de sortie ou cible).

Exemple

Ci-dessous, un Dataset qui regroupe des exemples d'appartements avec leur prix ainsi que certaines de leurs caractéristiques (features).

Cible	**Attributs**		
y	**X1**	**X2**	**X3**
Prix (Euro)	**Surface (m^2)**	**N chambres**	**Qualité**
313.000.00	124	3	1.5
2.384.000.00	339	5	2.5
351.000.00	179	3	2
420.000.00	186	3	2.25
550.000.00	180	4	2.5
490.000.00	82	2	1
335.000.00	125	2	2

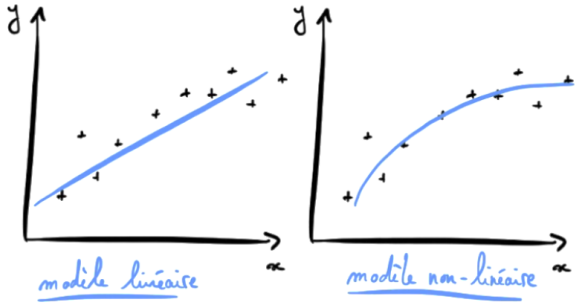
Un ensemble de données contenant des exemples d'appartements avec leurs prix et quelques caractéristiques.

4.2.2. Notion 2 : Développer un modèle à partir du Dataset

Le modèle est une représentation mathématique ou algorithmique qui apprend à partir des données pour effectuer des prédictions. Ses paramètres sont ajustés pendant l'entraînement pour minimiser l'erreur entre les prédictions du modèle et les valeurs réelles.

En Machine Learning, nous développons un modèle à partir de ce Dataset. Il peut s'agir d'un ***modèle linéaire*** ($f(x)=a.x+b$), comme illustré à gauche, ou d'un ***modèle non-linéaire*** ($f(x)=a.x^2+b.x+c$), comme illustré à droite[10].

Les paramètres du modèle, tels que a, b, c, etc., sont définis pour représenter ses caractéristiques.

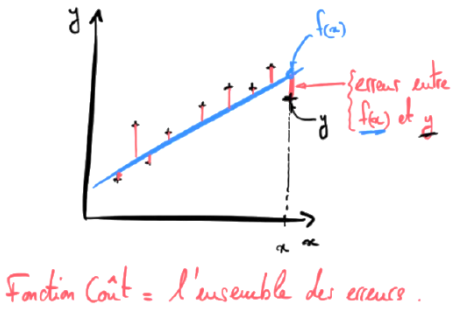


Les modèles linéaires et non-linéaires

4.2.3. Notion 3 : Les erreurs de notre modèle - la Fonction Coût

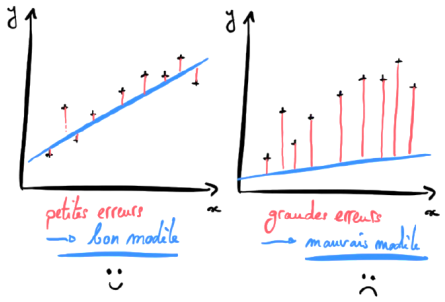
Aussi appelée fonction d'erreur ou de perte, elle mesure la différence entre les prédictions du modèle et les valeurs réelles dans le jeu de données.

Il est également important de noter qu'un modèle renvoie des erreurs par rapport à notre jeu de données. La Fonction Coût représente l'ensemble de ces erreurs, souvent calculée en prenant la moyenne quadratique des erreurs [10].



La Fonction Coût

`En résumé, un bon modèle est celui qui génère de faibles erreurs, ce qui se traduit par une petite` ***`Fonction Coût`***`.`



Les modèles performants et les modèles sous-performants

4.2.4. Notion 4 : Apprendre, c'est minimiser la Fonction Coût

En apprentissage supervisé, l'objectif principal est de trouver les paramètres du modèle qui ***réduisent au minimum la fonction de coût***. Pour cela, on fait appel à un a***lgorithme d'apprentissage***, dont l'exemple le plus courant est l'algorithme de ***descente de gradient (Gradient Descent)***, que vous étudierez dans ce chapitre[10].

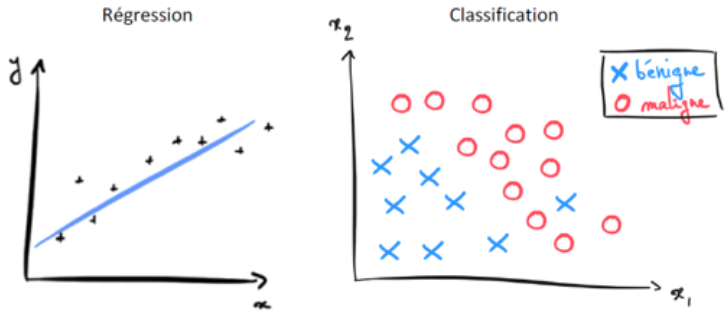
4.3. Les applications d'apprentissage supervisé

En utilisant l'apprentissage supervisé, nous pouvons développer des modèles pour résoudre deux types de problèmes principaux [10]:

- Les problèmes de ***régression***
- Les problèmes de ***classification***

🔔 *Rappel*

- Dans les problèmes de ***régression***, on cherche à prédire la valeur d'une variable continue, c'est-à-dire une variable qui peut prendre une infinité de valeurs.
  Par exemple : `Prédire le prix d'un appartement selon sa surface habitable`
- Dans un problème de ***classification***, on cherche à classer un objet dans différentes classes, c'est-à-dire que l'on cherche à prédire la valeur d'une variable discrète (qui ne prend qu'un nombre fini de valeurs).
  Par exemple : `Prédire si une tumeur est maligne ou bénigne selon la taille de la tumeur et l'âge du patient`



Régression Vs classification

4.4. La Régression linéaire

4.4.1. Récolter vos données

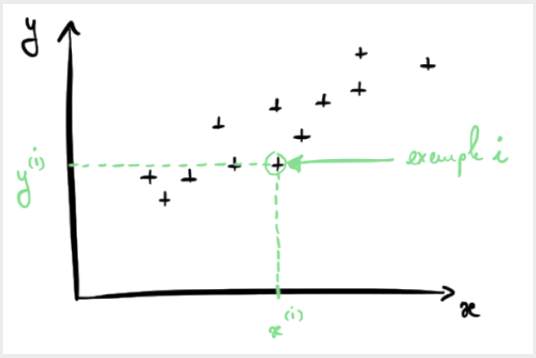
Imaginez que plusieurs agences immobilières vous aient fourni des données sur des appartements à vendre, notamment le prix de l'appartement (y) et la surface habitable (x). En Machine Learning, on dit que vous disposez de m exemples d'appartements[10].

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 :



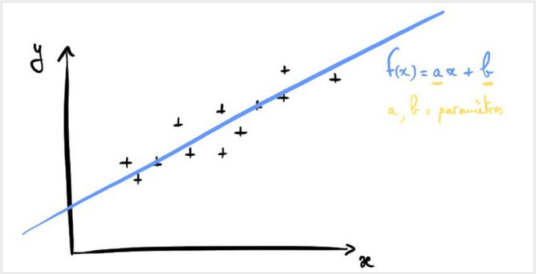
Dataset

4.4.2. Créer un modèle linéaire

A partir de ces données, on développe un ***modèle linéaire*** $f(x)=a.x+b$ 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 et , 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 :



Un modèle linéaire

4.4.3. Définir la Fonction Coût

Pour la régression linéaire, on utilise la norme euclidienne pour mesurer les erreurs entre f(x) et (y). Concrètement, voici la formule pour exprimer l'erreur i entre le prix $y^{(i)}$ et la prédiction faites en utilisant la surface $x^{(i)}$ [10]:

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

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

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

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

Note : En français, cette fonction a un nom : c'est l'erreur quadratique moyenne (Mean Squared Error)

4.4.4. Trouver les paramètres qui minimisent la Fonction Coût

La prochaine étape est l'étape la plus excitante, il s'agit de laisser la machine apprendre quels sont les paramètres qui ***minimisent*** la Fonction Coût, c'est-à-dire les paramètres qui nous donnent le meilleur modèle. Pour trouver le minimum, on utilise un algorithme d'optimisation qui s'appelle ***Gradient Descent*** (la descente de gradient) [10].

⚙ *Méthode*

L'algorithme de descente de gradient vise à trouver le minimum de la fonction coût ***J(a,b)*** à partir de coordonnées aléatoires a et b, en suivant les étapes suivantes [10]:

1. Calculez la pente de la fonction coût, autrement dit la dérivée de $J(a, b)$.
2. Déplacez-vous d'une certaine distance $\propto$ dans la direction de la pente la plus forte. Cela entraînera une modification des paramètres a et b.
3. Répétez les étapes 1 et 2 jusqu'à ce que vous atteigniez le minimum de $J(a, b)$.

💡 *Fondamental : 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.

Notre objectif final : Trouver les paramètres a et b qui minimisent $J(a, b)$.

Pour cela, nous commencerons par choisir et de manière ***aléatoire***, puis nous utiliserons la descente de gradient de manière itérative pour mettre à jour nos paramètres dans la direction de la fonction coût la ***plus faible***.

Répeter en boucle:

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

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

On appelle α la vitesse d'apprentissage (Learning rate).

- Une vitesse d'apprentissage trop faible entraîne un entraînement prolongé du modèle, tandis qu'une vitesse trop élevée peut empêcher la convergence.Il est important de trouver un juste milieu.

4.4.5. Les étapes pour programmer une Régression Linéaire

Etape 1 : Importer les librairies (Numpy, Matplotlib.pyplot, make_regression, SGDRegressor)

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.datasets import make_regression
4 from sklearn.linear_model import SGDRegressor
```

Étape 2 : Créer un Dataset

```
1 np.random.seed(0)
2 x, y = make_regression(n_samples=100, n_features=1, noise=10)
3 plt.scatter(x, y)
```

Etape 3 : Développer le modèle et l'entraîner

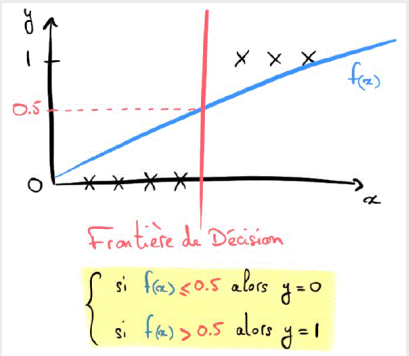
```
model = SGDRegressor(max_iter=1000, eta0=0.001)
model.fit(x,y)
print('Coeff R2 =', model.score(x, y))
plt.scatter(x, y)
plt.plot(x, model.predict(x), c='red', lw = 3)
```

4.5. Régression Logistique et Algorithmes de Classification

Les problèmes de Classification

Effectivement, dans ce scénario, où la variable cible peut prendre seulement deux valeurs (0 ou 1), nous avons affaire à deux classes distinctes. Cela correspond à ce qu'on appelle une ***classification binaire.***

Dans ces cas, pour effectuer la classification des e-mails en classe 0 ou classe 1, on introduit une ***frontière de décision*** dans le modèle. Cette frontière de décision sépare l'espace des caractéristiques en deux régions, permettant ainsi de classifier les e-mails en fonction de la classe à laquelle ils appartiennent [10].



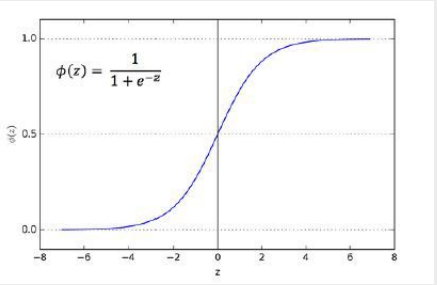
Frontière de décision

4.5.1. Le modèle de Régression logistique

Pour les problèmes de classification binaire, un modèle linéaire $F = X \cdot \theta$, comme représenté dans la figure précédente, n'est pas adapté [10].

[cf. res]

Dans le contexte des problèmes de classification binaire, une nouvelle fonction est développée, appelée fonction logistique (ou fonction sigmoïde, ou encore simplement sigma). Cette fonction présente la particularité d'assigner des valeurs toujours comprises entre 0 et 1.



Fonction sigmoïde

Pour appliquer la fonction logistique à un ensemble de données (X,y), nous effectuons un produit matriciel X.θ, ce qui donne le modèle de régression logistique :

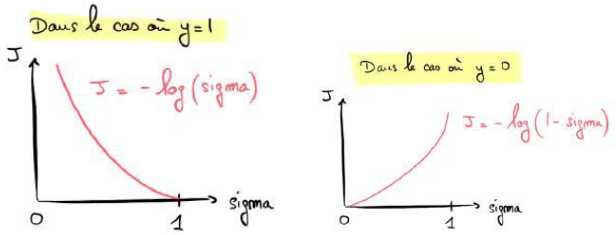
$$\sigma(X \cdot \theta) = \frac{1}{1+e^{-X \cdot \theta}}$$

a) Fonction Coût associée à la Régression Logistique

Cette fonction pour le modèle Logistique ne donnera pas de courbe convexe (dû à la non-linéarité) et l'algorithme de Gradient Descent se bloquera au premier minima rencontré, sans trouver le minimum global.

Il faut donc développer une nouvelle Fonction Coût spécialement pour la régression logistique. On utilise alors la fonction logarithme pour transformer la fonction sigma en fonction convexe en séparant les cas où y= 1 des cas où y = 0.

Fonction Coût pour la régression logistique



Fonction Coût complète

Pour écrire la Fonction Coût en une seule équation, on utilise l'astuce de séparer les cas y = 0 et y= 1 avec une annulation :

$$J(\theta)=\frac{-1}{m}\sum y\cdot\log(\sigma(X\cdot\theta))+(1-y)\cdot\log(1-\sigma(X\cdot\theta))$$

b) Gradient Descent pour la Régression Logistique

L'algorithme de Gradient Descent s'applique exactement de la même manière que pour la régression linéaire.

En plus, la dérivée de la Fonction Coût est la même aussi

Rappel

Résumé de la Régression Logistique: (Voir le document)

[cf.]

4.6. Algorithme d'apprentissage

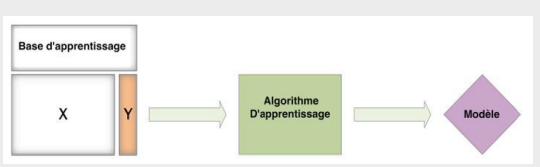
Un algorithme qui prend en entrée un ensemble de données contenant les informations nécessaires pour caractériser un problème donné et renvoie un modèle qui représente les concepts caractérisant ces données et qui doit être capable de prédire l'étiquette de nouveaux objets en fonction de leurs valeurs d'entrée.

Séparer aléatoirement les données en 3 sous-ensembles [11]:

- Les ***données d'apprentissage*** serviront à entraîner le ou les algorithmes choisis ;
- Les ***données de test*** seront utilisées pour vérifier et évaluer la performance du résultat;
- Les ***données de validation*** ne seront utilisées qu'à la toute fin du processus et ne seront, sauf nécessité, que très rarement examinées

4.6.1. Phase d' apprentissage

Dans cette phase, un algorithme d'apprentissage automatique est entraîné sur une base de données étiquetées, au cours de cette



étape, le modèle peut améliorer ses performances en se guidant avec une sous partie de la base d'apprentissage appelée base de validation [11].

Apprentissage d'un modèle a partir d'un ensemble d'apprentissage

4.6.2. Phase de validation

Lorsque la validation est impliquée, le processus d'entraînement du Machine Learning se déroule selon les étapes suivantes [11]:

1. Divisez les données d'entraînement en deux groupes: un pour l'entraînement et l'autre pour la validation. En règle générale, le rapport entre l'ensemble d'entraînement et l'ensemble de validation est de 8:2.

2. Entraîner le modèle avec l'ensemble d'entraînement.

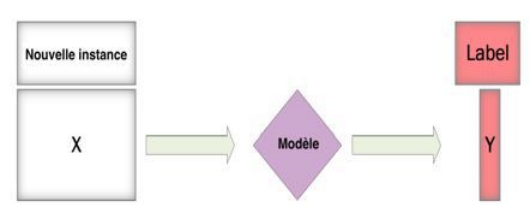
3. Évaluez les performances du modèle à l'aide du jeu de validation.

a. Si le modèle donne des performances satisfaisantes, terminez l'entraînement.

b. Si la performance ne produit pas des résultats suffisants, modifiez le modèle et répétez le processus à partir de l'étape 2.

4.6.3. Phase de test

Dans les problèmes de classification, à l'étape du test, le modèle devrait être en mesure de donner une classe à une instance de test en fonction de sa valeur d'entrée en suivant les règles apprises dans l'étape d'apprentissage [11].



Prédiction de la classe d'une nouvelle instance

4.7. Modèles d'apprentissage supervisé

Il existe de nombreuses applications qui sont le résultat d'un apprentissage automatique et qui permettent d'assister les cliniciens dans leurs procédures de diagnostic, car ils peuvent fournir un diagnostic plus précis et réduire les erreurs dues a la fatigue et aux doutes du médecin.

Ces techniques consistent a prédire la classe des nouvelles données observées en utilisant des modèles de classification comme les ***arbres de décision***, les ***réseaux de neurones***, les ***k-plus proches voisins***, etc ...

Cependant, avant d'utiliser ces applications, le médecin doit être sur de leurs fiabilités, pour cela il faudrait appliquer un certain nombre de tests et de méthodes d'évaluation des classifieurs.

[cf.]

5. Conclusion

Dans la première section, nous avons examiné les principes fondamentaux de l'apprentissage supervisé, qui représentent un prérequis crucial pour appréhender les diverses stratégies de l'apprentissage automatique. Avec

cette base établie, nous sommes prêts à plonger plus profondément dans les différentes techniques de l'apprentissage supervisé, ainsi que d'autres types d'apprentissage, dans la section suivante du chapitre