2020-2021

Licence économie et gestion
Parcours économie

FACULTÉ DE DROIT, D'ÉCONOMIE ET DE GESTION
UNIVERSITÉ D'ANGERS

# Introduction au Machine Learning

**GHEMMOUR Samy**

Sous la direction de M. Philippe Compaire

# REMERCIEMENTS

J'adresse mes remercîments aux personnes qui ont contribué de près ou de loin à la réalisation de ce mémoire.

Je tiens à remercier particulièrement mon encadrant, Mr Philippe Compaire, qui m'a accompagné tout au long de la réalisation de mon mémoire, et a pris le temps pour m'aider dans mon travail. Son soutien, sa clairvoyance m'ont été d'une grande aide.

Je remercie toute l'équipe pédagogique de la faculté, qui ont déployé leurs efforts pour assurer une formation aussi complète, pour aider et soutenir toute la promotion le long du cycle.

Je voudrais exprimer ma reconnaissance envers mes amis et mon frère, pour leurs idées et conseils, ils ont grandement facilité mon travail, et enfin, mes parents pour la dernière relecture.

FACULTÉ
DE DROIT, D'ÉCONOMIE
ET DE GESTION
UNIVERSITÉ D'ANGERS

# ENGAGEMENT DE NON PLAGIAT

Je soussigné Mr GHEMMOUR Samy,

Être pleinement conscient(e) que le plagiat de documents ou d'une partie d'un document publiés sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour écrire ce rapport ou mémoire.

SIGNATURE :

# Table des matières

# Introduction générale :

Nous l'utilisons des centaines de fois par jour, mais nous ne le savons pas, lorsque nous faisons des recherches sur Google, l'une des principales raisons pour lesquelles les résultats sont assez pertinents est que derrière lui se trouve un algorithme d'apprentissage automatique, qui a appris à trouver parmi des milliers de résultats possibles, ceux qui sont les plus pertinents. YouTube, Netflix ou Amazon, utilisent un algorithme d'apprentissage automatique qui a appris nos préférences et nous recommande du contenu que nous pourrions regarder ou acheter. La programmation classique est incapable d'aboutir à ce genre de résultat, mais avec la machine Learning, on donne à un algorithme la capacité d'apprendre, ce qui simplifie les choses, celle-ci est la définition même du machine Learning, **donner à une machine la capacité d'apprendre sans la programmer de façon explicite.(introduction au machine learning, Chloe-Agathe Azencott, page 1),** Ceci révolutionne le monde des transports, l'industrie aérospatiale, la santé, la finance, et de nombreux autres domaines.

Le Machine learning (apprentissage automatique) consiste à développer un modèle, en se servant d'un algorithme d'optimisation pour minimiser les erreurs, entre le modèle et nos données, Il existe différents modèles tels que les modèles linéaires, arbres de décision, support vector machine, chacun venant avec son algorithme d'optimisation, la descente de gradient pour les modèles linéaires, l'algorithme CART pour les arbres de décision, ou encore la marge maximum pour le support vector machine.

Ce travail est divisé en deux parties, la première sera d'établir ce qui relève du machine Learning, quelles sont ses différentes méthodes d'apprentissage, comment celles-ci fonctionnent, et nous verrons qu'elles diffèrent les unes aux autres de par leurs fonctionnements mais aussi leurs applications.

La deuxième partie sera consacrée à un domaine qui est dérivé du machine Learning très repondus ces dernières années, le Deep learning (apprentissage profond), sa place dans le monde de l'intelligence artificielle, nous verrons que son fonctionnement repose sur un réseau de neurones artificiels. Un derniers point sera consacré à une application du Machine Learning sur R, comment développer et évaluer un modèle de régression linéaire.

# Partie 1 : Le machine Learning.

## Introduction :

Pour comprendre le fonctionnement du machine Learning et comment les machines acquièrent des capacités d'apprentissage, demandons-nous d'abord comment nous en tant qu'être humain apprenons-nous ? Nous apprenons le plus souvent à partir d'exemples. Si nous voulons apprendre une nouvelle langue (Par exemple : l'espagnol), cette méthode fonctionne pour nous. La chose la plus raisonnable à faire est d'acheter un livre dans lequel on trouve des exemples de traduction du français vers l'espagnol, ou de demander l'aide d'un professeur, qui pourra nous encadrer en nous fournissant des exemples du français vers l'espagnol.

En machine learning, la technologie d'apprentissage la plus courante est directement inspirée de ce mode de fonctionnement. C'est ce que nous appelons l'apprentissage supervisé. Nous donnons des exemples à la machine et celle-ci les utilisent pour créer des modèles. Ces exemples sont combinés pour former un dataset. Une table de données à partir de laquelle nous pouvons effectuer un apprentissage automatique. (Voir apprendre le Machine Learning en une semaine Guillaume Saint-Cirgue, chapitre 1, Page 8-18)

**Dans l'apprentissage supervisé on compte deux problèmes :**

Les problèmes de régression, dans lesquels on cherche à prédire la valeur d'une variable continue, variable qui peut prendre une infinité de valeur, exemple : le prix d'un appartement.

Les problèmes de classification, dans lesquels on cherche à prédire la valeur d'une variable discrète, exemple : on cherche à développer un filtre anti-spam, on considérera Y=1 comme pas spam, et y=0 spam, lors de la construction de notre Dataset, on regardera la nature du courriel, soit il appartient à la classe 0, soit il appartient à la classe 1, ceci soit en tenant compte des fautes dans le courriel, ou bien le nombre de liens……,

Pour donner à un ordinateur la capacité d'apprendre, on utilise des méthodes d'apprentissage qui sont fortement inspirées de la façon dont nous, les êtres humains,

apprenons à faire des choses (voir apprendre le Machine Learning en une semaine Guillaume Saint-Cirgue, chapitre 1, Page 8-18) . Parmi ces méthodes, on compte :

•L'apprentissage supervisé (Supervised Learning)

• L'apprentissage non supervisé (Unsupervised Learning)

• L'apprentissage par renforcement (Reinforcement Learning)

## 1. L'apprentissage supervisé[1] :

L'apprentissage supervisé est la méthode la plus utilisée en machine Learning, celle-ci consiste à montrer à la machine des exemples X et Y, et lui demander de trouver l'association qui relie X a Y, tel que Y=F(x), ils permettent aussi d'expliquer les sous-jacents des prédictions.

Ces algorithmes permettent de faire des prévisions, mais surtout d'expliquer les phénomènes sous-jacents à cette prévision, ce qui intéresse les managers des entreprises, connaître le passé est indispensable pour pouvoir effectuer des prévisions, il s'agit de l'un des problèmes les plus importants du Big Data. Cette méthode est utile pour pouvoir calculer les chiffres (Exemple : Age, revenu, chiffre d'affaires…) on parle alors de régression, Calculer une catégorie (Exemple : homme/femme, sain/malade) on parle alors de classification. [i]

La complexité de cette méthode réside dans le fait que, pour pouvoir faire une prévision, il faut trouver les informations disponibles, mais supprimer celles qui ne sont pas connues au moment de la prévision, ces données sont interdites et il est impératif de les éviter, car en les gardant, on pourrait gonfler artificiellement le score de ces algorithmes de prévisions, et avoir des résultats qui sont dit biaisés, et souvent les bases de données les plus anciennes ne sont pas faites pour cela, nous pouvons citer comme exemple l'adresse d'un individus qui est souvent remplacée. [2]

**Les notions fondamentales de l'apprentissage supervisé :**

[1] Romain Rissoan, Romain Jouin. (2018). La boîte à outils de la stratégie big data. 10-35. https://www.cairn.info/la-boite-a-outils-de-la-strategie-big-data--9782100778980.htm

## 1.1. Dataset :

Les différents exemples sont compilés dans ce qui est appelé un **dataset,** celui-ci constitue la première notion fondamentale de l'apprentissage supervisé, un dataset contient deux types de variables, d'un côté la **Target variable**, ce que l'on veut que la machine apprenne à prédire (le cours d'une action, identifier si un courriel est un spam ou non…), de l'autre on a les **features**, les facteurs qui viennent influencer la valeur de Y.[3]

**Exemple de Dataset sur appartement :**

| Target $y$ | Features $x_1$ | $x_2$ | $x_3$ |
|---|---|---|---|
| Prix | Surface m2 | N chambres | Qualité |
| € 313,000.00 | 124 | 3 | 1.5 |
| € 2,384,000.00 | 339 | 5 | 2.5 |
| € 342,000.00 | 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 |

Source : apprendre le Machine Learning en 1 semaine, Guillaume Saint-Cirgues, page 13

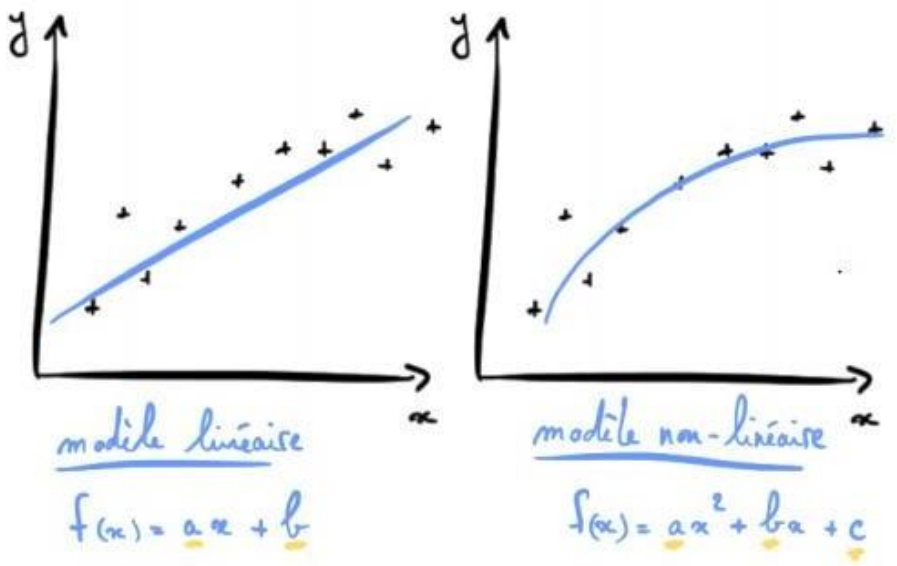
Y est une fonction de toutes ces Features, on peut dire que X représente les différentes caractéristiques qui vont servir à définir le prix de l'appartement. On note m le nombre d'exemples ou de lignes, et n le nombre de colonnes hormis la colonne Y.

Une autre convention qu'on a en machine learning, est que pour désigner une cellule dans notre Dataset, on note le numéro de l'exemple et de la features. Ce dataset sera utilisé par la suite pour créer un modèle, qui sera la deuxième notion importante du machine learning.

## 1.2. Modèle et ses paramètres[4] :

A partir du nuage de points, on peut dire qu'on peut créer un modèle, qui peut être linéaire, ou non linéaire.

[3] Voir apprendre le machine learning en une semaine Guillaume Saint-Cirgue, chapitre 1.
[4] Voir apprendre le machine learning en une semaine Guillaume Saint-Cirgue, chapitre 1.



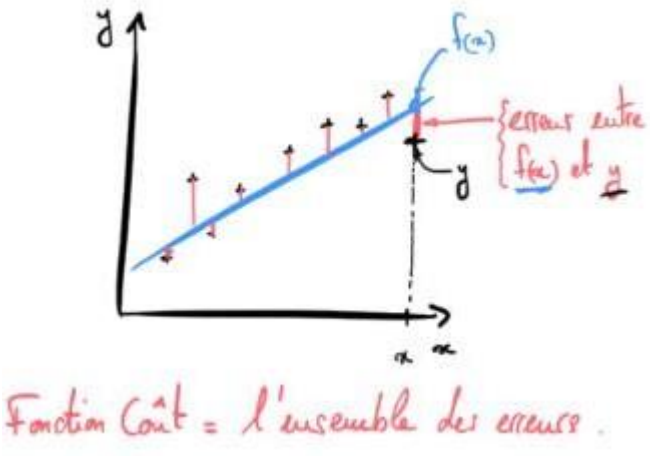

Source : Apprendre le Machine Learning en une semaine Guillaume Saint-Cirgue, p. 14

a, b et c sont les paramètres du modèle.

C'est nous qui décidons quel modèle la machine doit utiliser, celle-ci les utilisera pour en définir les paramètres.

### 1.3. La fonction cout :

Quand ce modèle peut contenir des erreurs, car c'est une généralisation d'un ensemble de points de notre dataset, ces erreurs assemblées nous donnent ce que l'on appelle la fonction cout, qui est notre troisième notion fondamentale en apprentissage supervisé.




Source : Apprendre le Machine Learning en une semaine Guillaume Saint-Cirgue, p. 14

## 1.4. L'algorithme d'apprentissage :

Un bon modèle, est un modèle qui minimise au mieux les erreurs. On développe une stratégie qui cherche à trouver quels sont les paramètres qui minimisent la fonction coût, c'est-à-dire qui minimise nos erreurs.

Une stratégie, très populaire en Machine Learning, est de considérer la Fonction Coût comme une fonction convexe, c'est-à-dire une fonction qui n'a qu'un seul minimum, et de chercher ce minimum avec un algorithme de minimisation appelé Gradient Descent. Cette stratégie apprend de façon graduelle, et assure de converger vers le minimum de la fonction Coût (si convexe).

La machine choisit des paramètres pour le modèle, puis évalue sa performance (Fonction Coût) puis cherche des paramètres qui peuvent améliorer sa performance actuelle, etc.

Le modèle d'apprentissage choisit la fonction qui réalise la plus faible erreur moyenne de prédiction sur une base d'entraînement, la fonction d'erreur quantifie le désaccord entre la prédiction de sortie donnée par la fonction qu'on souhaite apprendre pour une base d'entraînement, et sa réponse associée. Le but de cette recherche n'est pas que le modèle d'apprentissage induise une fonction donnant exactement les sorties désirées des observations, mais de trouver, la fonction qui aura de bonnes performances générales,( voir Machine learning, Massih-Reza AMINI, page 8.)

Vanpik (1999) a introduit et développé le principe de **minimisation du risque empirique,** le résultat marquant de sa théorie est une borne supérieure de l'erreur de généralisation de la fonction apprise qui s'exprime en fonction de l'erreur empirique de celle-ci sur une base d'entrainement, et la complexité de la classe de la fonction utilisée, cette complexité se traduit par la capacité de la classe de fonction à résoudre le problème utilisé. Et plus la capacité est grande, plus le risque empirique serait faible et moins on est garanti d'atteindre l'objectif principal de l'apprentissage. Voir Machine learning, Massih-Reza AMINI, page 8.

**Définition du risque empirique :**
Dans le cadre d'un problème d'apprentissage supervisé, à n observations étiquetées $\{(\vec{x}^i, y^i)\}_{i=1,\dots,n}$, le risque empirique est appelé l'estimateur :

$$R_n(h) = \frac{1}{n}\sum_{i=1}^{n} L(h(\vec{x}^i), y^i).$$

En minimisant le risque empirique, on obtient le prédicteur :

$$f = \arg\min_{h \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(h(\vec{x}^i), y^i). \quad (2.1)$$

Les solutions analytiques de l'équation 2.1 diffèrent selon le choix de F. la fonction la plus souvent utilisée est une fonction de coût convexe, ceci afin de résoudre plus facilement les problèmes d'optimisation.

Le problème dans la minimisation du risque empirique est qu'il n'admet pas qu'une seule solution unique qui dépend de la façon continue des conditions initiales, il se peut qu'un nombre infini de solutions le minimisent à zéro.

$$\forall h \in \mathcal{F}, R_n(h) \xrightarrow[n \to \infty]{} \mathcal{R}(h).$$

La question fondamentale que pose cette méthode est : est-il possible de toujours générer une fonction de prédiction qui généralise bien à partir d'un ensemble d'observations finis ? La réponse est évidemment non, l'exemple qui va suivre va nous aider à justifier notre réponse.

**Exemple :** surapprentissage.[5]

Supposons que la dimension d'entrée est d=1, prenons l'espace des observations X : l'intervalle [a, b] C R, où a et b sont des réels tel que a<b et l'espace des sorties est {-1, 1}. De plus supposons que la distribution D générant les couples d'exemples (x, y) est une distribution uniforme sur [a, b] et, pour chaque observation, la sortie désirée est -1.

Considérons maintenant un algorithme d'apprentissage minimisant le risque empirique, en choisissant une fonction dans la classe des fonctions F= {f : [a, b] --> {-1, 1}} de la façon suivante, après avoir pris connaissance d'un ensemble d'apprentissage S= {(x1, y1), …..,(xm, ym)} l'algorithme produit la fonction de prédiction fs telle que :

$$fs(x) = \begin{cases} -1 \ \ si\ x \in \{x1, \ldots\ldots, xm\} \\ +1, si\ non \end{cases}$$

Dans ce cas, le classificateur produit par l'algorithme d'apprentissage a un risque empirique égal à 0, et ceci pour n'importe quel ensemble d'apprentissage donné. Cependant, comme le classifieur fait une erreur sur tout l'ensemble infini {a, b} sauf pour les exemples d'une base d'entrainement finie, de mesure nulle, son erreur de généralisation est toujours égale à 1. Voir Machine learning, Missih-Reza Amini, p11.

[5] Voir : introduction au Machine Learning Chloé-Agathe Azencott, pages 19-21.

Cependant, une autre question en résulte, dans quel cas le principe MRE est-il susceptible de générer une règle générale d'apprentissage ?

Ceci va nous permettre de poser un deuxième principe, celui de la consistance.

**Consistance du principe MRE[6] :**

Deux conditions sont nécessaires pour cela, l'algorithme doit renvoyer une fonction dont l'erreur empirique, reflète son erreur de généralisation lorsque la taille de la base d'entraînement tend vers l'infini, l'algorithme doit trouver une fonction qui minimise l'erreur de généralisation dans la classe de fonctions considérée, si ces deux conditions sont vérifiées, on dit alors que l'algorithme est consistant.

$$(a)\ \forall \epsilon > 0, \lim_{m\to\infty} \mathbb{P}(|\hat{\mathfrak{L}}(f_S, S) - \mathfrak{L}(f_S)| > \epsilon) = 0, \text{ noté, } \hat{\mathfrak{L}}(f_S, S) \xrightarrow{\mathbb{P}} \mathfrak{L}(f_S)$$

$$(b)\ \hat{\mathfrak{L}}(f_S, S) \xrightarrow{\mathbb{P}} \inf_{g\in\mathcal{F}} \mathfrak{L}(g)$$

Ce qui implique, la convergence de l'erreur empirique de la fonction de prédiction trouvé par l'algorithme sur la base d'entrainement S, fs, vers son erreur de généralisation $\mathfrak{L}(f_S)$ et inf $_{g\in\mathcal{F}}\, \mathfrak{L}(g)$ .

Le principe MRE est consistant si est seulement si :

$\forall \epsilon > 0, \lim_{m\to\infty} P(sup_{fgF}[r(f) - \hat{r}(f, s)] > \epsilon) = 0$

Cette condition stipule que le principe de MRE est constant si et seulement si, la convergence uniforme unilatérale est assurée pour la fonction de la classe de fonctions F, pour laquelle la différence entre son erreur de généralisation et son erreur empirique est la plus grande, ce qui lui a valu le nom d'étude de pire, (voir Machine learning, Massih-Reza AMINI, étude de pire cas).

[6] Machine Learning Massih-Reza AMINI, Chapitre 1, Page 12.

## 2. L'apprentissage non supervisé :

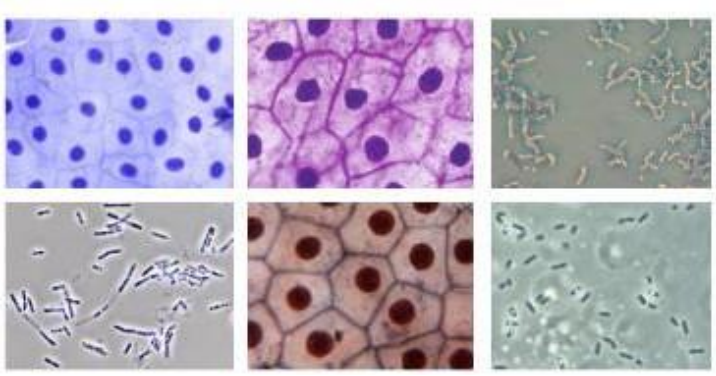
L'apprentissage supervisé consiste à enseigner à la machine des choses que nous connaissons déjà, étant donné que nous construisons à l'avance un Dataset qui contient des questions $\boldsymbol{X}$ et des réponses $\boldsymbol{y}$.

Mais que faire dans le cas où nous disposant d'un dataset sans valeur Y ? Ou si l'on veut que la machine nous aide à compléter nos enseignements ?

Dans ce cas utiliser une méthode d'apprentissage non supervisé serait plus appropriée, il s'agit de modifier les observations pour mieux comprendre, on dispose ainsi d'un Dataset ($x$) sans valeur ($y$), et la machine apprend à reconnaitre des structures dans les données ($x$) qu'on lui montre et à les regrouper uniquement selon leurs ressemblances, ceci peut servir à classer des documents, des photos, segmenter la clientèle d'une entreprise……

On appelle apprentissage non supervisé la branche du machine Learning qui s'intéresse aux problèmes pouvant être formalisés de la façon suivante : $(\vec{x}^{\,i})_{i=1,\dots,n}$ étant données n observation décrites dans un espace x, il s'agit d'apprendre une fonction sur x qui vérifie plusieurs propriétés. (Introduction au machine Learning, Chloé Agathe Azencott, p14)

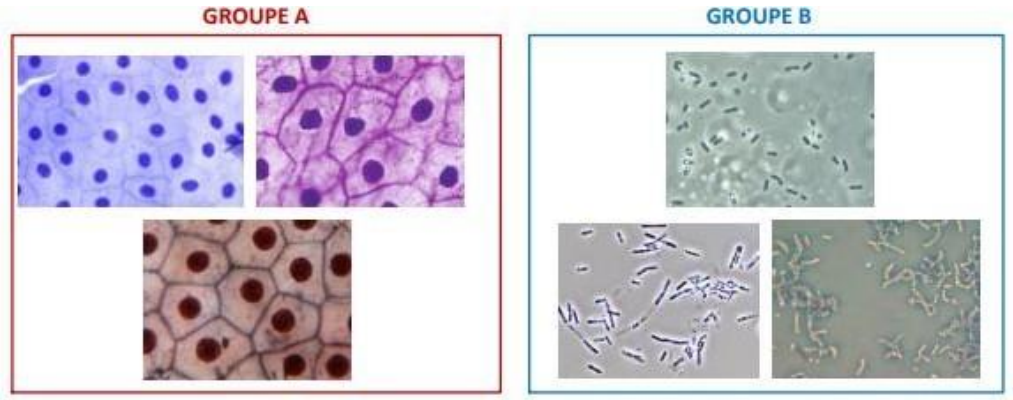
Exemple d'apprentissage non supervisé :[7]



Source : Apprendre le Machine Learning en une semaine Guillaume Saint-Cirgue, p. 72

Nul besoin de savoir s'il s'agit de cellules animales, de bactéries ou de protéines pour apprendre à classer ces images. Il suffit seulement de classer les données selon leur structure commune.

[7] Apprendre le machine learning en une semaine, Guillaume Saint-Cirgue, chapitre 4




Source : Apprendre le Machine Learning en une semaine Guillaume Saint-Cirgue, p. 72

## 2.1. Clustering :

Dans l'apprentissage non-supervisé, on dispose ainsi d'un Dataset ($x$) sans valeur ($y$), et la machine apprend à reconnaître des structures dans les données ($x$) qu'on lui montre.

On peut ainsi regrouper des données dans des clusters comme nous venons de le voir, Détecter des anomalies, la machine analyse la structure de nos données, et arrive à trouver celles dont les caractéristiques sont très éloignées de notre échantillon, ceci peut servir par exemple à détecter des fraudes banquières.[8]

Ainsi on appelle partitionnement ou clustering un problème d'apprentissage non supervisé qui peut être formalisé, comme la recherche d'une partition des n observations, cette partition doit être pertinente au vu de plusieurs critères à préciser.

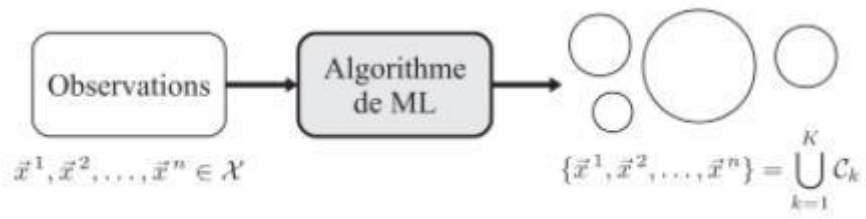



**Figure 1.3 – Partitionnement des données, ou clustering.**

Source: Introduction au machine learning, Chloé-Agathe Azencott p 8

Un des algorithmes les plus populaires est Le K-Mean Clustering (regrouper des données selon leur structure commune). Il est souvent utilisé en marketing pour cibler des groupes de clients semblables pour certaines campagnes publicitaires.

8 Apprendre le machine learning en une semaine, Guillaume Saint-Cirgue, chapitre 4

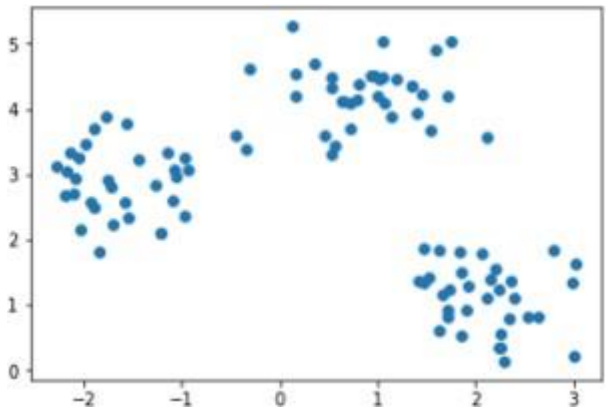
**Exemples de problèmes de clustering [9]:**

Utiliser un logiciel pour compresser une image, le problème du clustering consiste à regrouper les pixels similaires, pour ensuite améliorer leurs représentations.

Une chaine TV qui veut déterminer la répartition démographique des téléspectateurs par réseaux peut le faire en créant des clusters à partir des données disponibles sur les abonnés et de ce qu'ils regardent, une chaîne de restaurants peut quant à lui regrouper sa clientèle en fonction des menus choisis par emplacement géographique, puis modifier ses menus en conséquence.

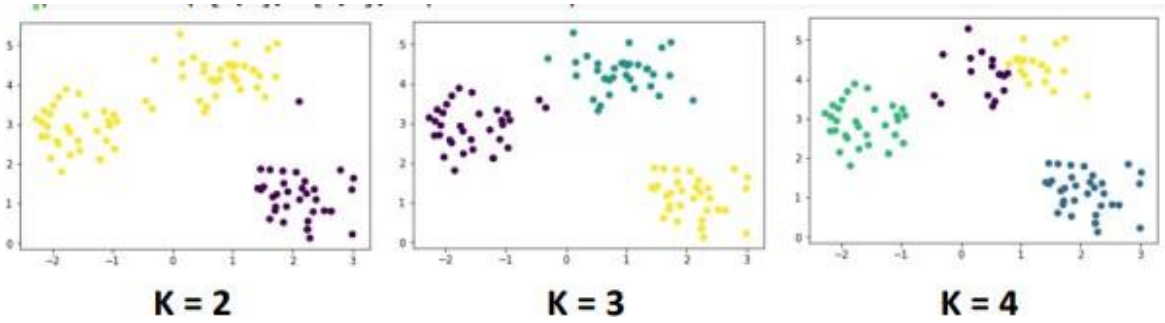
Identifier des groupes parmi les patients présentant les mêmes symptômes permet d'identifier des sous-types d'une maladie, qui pourront être traités différemment.

Voici l'exemple d'un dataset de 100 exemples, à 2 features en simulant 3 clusters.



Source : Apprendre le Machine Learning en une semaine Guillaume Saint-Cirgue, p. 75

On entraîne un modèle de K-Mean Clustering à 3 centres (K=3). Les résultats obtenus pour K = 2, K=3 et K = 4 sont les suivants :




Source : Apprendre le Machine Learning en une semaine Guillaume Saint-Cirgue, p. 72

[9] Apprendre le machine learning en une semaine, Guillaume Saint-Cirgue, chapitre 4

## 2.2. Réduction de dimension :

L'apprentissage non supervisé sert aussi à réduire la dimension de données très riches, en compilant les dimensions ensembles, en analysant la structure de nos données la machine est capable d'apprendre comment simplifier cette structure, tout en gardant les principales informations, le principe est de réduire la complexité superflue d'un dataset, en projetant ses données dans un espace de plus petite dimension, un des algorithmes qui permet de faire cela est l'analyse en composante principales.

On appelle réduction de dimension un problème d'apprentissage non supervisé pouvant être formalisé comme la recherche d'un espace Z, de dimension plus faible que l'espace dans lequel sont représentées n observations $(\vec{x}^{\,i})_{i=1,\dots,n}$ , les projections $(\vec{z}^{\,i})_{i=1,\dots,n}$ des données sur Z doivent vérifier certaines propriétés à préciser (introduction au machine learning Chloé-Agathe Azencott p 9).

Le principe est de projeter nos données sur des axes appelés Composantes principales, en cherchant à minimiser la distance entre nos points et leurs projections.

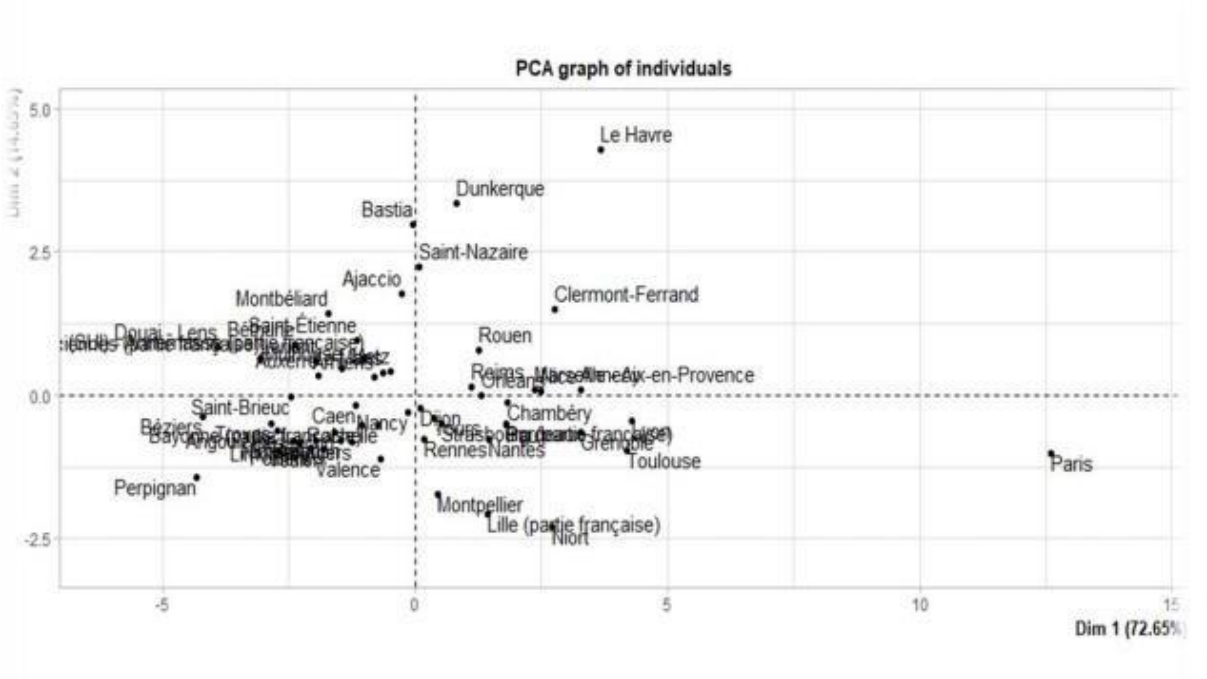
De cette manière, on réduit la dimension de notre dataset tout en préservant au maximum la variance de nos données ceci pour obtenir la projection la plus fidèle possible.

Voici les étapes à suivre pour trouver les axes de projection :

1. On calcule la matrice de la covariance des données.
2. On détermine les vecteurs propres de cette matrice : autrement dit les composantes principales.
3. On projette les données sur ces axes.

### Exemple d'une analyse par composantes principales faite sur R :

Celle-ci est réalisée sur une base de données contenant 57 individus, qui représente les départements, et 16 variables quantitatives qui représentent les salaires nets horaires moyens, par âge, sexe et catégorie socioprofessionnelle, j'ai moi-même réalisé cette ACP, sur R grâce au package FactoMineR.




## 3. L'apprentissage par renforcement[10] :

Dans ce cas, on ne donne plus d'exemples à la machine, c'est elle-même qui va générer sa propre expérience, on lui donne la liberté d'entreprendre des actions au sein de son environnement.

Considérant, par exemple, le problème d'apprendre à jouer aux échecs, un agent à apprentissage supervisé a besoin qu'on lui montre le bon mouvement pour chaque position, mais un tel signal de retour est rarement disponible, en l'absence de tel signal, l'agent ne dispose d'aucun repère pour savoir ce qu'il doit jouer, il a besoin de savoir qu'il s'est produit quelque chose de bien quand il gagne et de mauvais quand il perd, cette forme de retour s'appelle récompense, ou renforcement, elles servent à définir des politiques optimales dans les processus de décision Markov (PDM), la politique optimale est définie comme celle qui maximise la récompense totale espérée, la tache de l'apprentissage par renforcement consiste à utiliser les récompenses observées pour apprendre une politique optimale pour l'environnement. Dans plusieurs domaines, l'apprentissage par renforcement est le seul moyen d'atteindre un haut niveau de performance, l'homme a beaucoup de difficultés à évaluer de façon exacte et cohérente un grand nombre de positions, en revanche, on peut indiquer à un programme quand il a gagné ou perdu, et il peut utiliser cette information pour apprendre une fonction d'évaluation directe qui estime de façon exacte la probabilité

[10] Intelligence artificielle, Stuart Russel et Peter Norvig 3eme édition, Apprentissage par renforcement, page 879

de gagner en partant d'une position donnée. L'apprentissage par renforcement diffère fondamentalement des autres apprentissages, par l'aspect interaction, agent, environnement, l'objectif de l'agent artificiel est d'apprendre une stratégie optimale, à travers d'essais et d'erreurs pour maximiser son gain, l'agent artificiel doit Trouver un compromis entre essayer de nouvelles actions et apprendre de nouvelles choses, ou refaire les actions qui lui procurent les bonnes récompenses, ce compromis et connus sous le nom d'équilibre entre exploration et exploitation.

Considérons qu'un agent est placé dans un environnement et doit apprendre à s'y comporter avec succès, majoritairement, nous supposons un environnement entièrement observable, en revanche l'agent ne sait pas comment l'environnement fonctionne et quelle action il doit accomplir, l'agent est donc face à un processus de décision Markov inconnu, nous considérons trois conceptions d'agents :

-Un agent fondé sur l'utilité qui apprend une fonction d'utilité sur des états et l'utilise pour sélectionner des actions qui maximisent l'utilité espérée du résultat.

-Un Q-Learning qui apprend une fonction d'action valeur, qui donne l'utilité espérée de la réalisation d'une action donnée dans un état donné.

-Un agent réflexe qui apprend directement une politique qui associe directement les agents aux actions.

Un agent fondé sur l'utilité doit posséder un modèle d'environnement pour pouvoir prendre des décisions, car il doit connaître les états auxquels ces actions vont conduire, au connaître, un agent Q-learning peut comparer des choix dont il dispose sans connaitre leurs résultats, il n'a donc pas besoin d'un modèle de l'environnement, en revanche, il ne peut pas anticiper, ce qui peut limiter son aptitude à apprendre.

L'apprentissage par renforcement est l'un des domaines les plus mouvementé actuellement en intelligence artificielle, il inspire les travaux de plusieurs neurobiologistes, et psychologues dans leurs quette de compréhension du fonctionnement du cerveau et du comportement, il est applicable aussi dans l'industrie notamment dans la manipulation robotique avec dextérité dans les usines, la gestion des ressources, et les jeux vidéo.

## 4. L'application de l'apprentissage par renforcement[11] :

### 4.1.Application aux jeux :

Le programme de jeu de dames écrit par Arthur Samuel (1959,1967), fut la première grande application de l'apprentissage par renforcement, pour évaluer les postions Samuel a utilisé une fonction linéaire pondérée, allant jusqu'à 16 termes à la fois, son programme se différencie des autres méthodes de l'époque car il n'utilisait aucune récompense observée, ce qui veut dire que ce programme n'est pas destiné à converger, ou alors converger vers une stratégie perdante, pour éviter cela, Samuelson associait les poids de sa fonction à un avantage matériel positif, et cela suffisait à diriger le programme dans des zones de l'espace correspondant à un bon jeu.

En 1989, Tesauro essayait de mettre en œuvre un apprentissage par réseau de neurones de Q, à partir d'exemple de coups auxquels un expert humain attribuait des valeurs relatives, ce qui a débouché sur le NEUROGAMMON, qui présentait de bonnes performances mais ne pouvait pas rivaliser avec des experts humains. Un nouveau programme fut développé, le TD-GAMMON, l'objectif était qu'il apprenne à jouer tout seul en jouant contre lui-même, le signal de récompense était donné à chaque fin de partie, un réseau de neurones entièrement connecté représentait la fonction d'évaluation, avec environ 200 000 parties d'apprentissage et deux semaines de temps de calcul, TD-GAMMON a appris à jouer considérablement mieux que NEUROGAMMON[12].

### 4.2.Applications aux contrôles de robots :

L'un des programme les plus célèbre reste celui du **pendule inversé**, le pendule présente une position d'équilibre instable que l'on cherche à stabiliser en utilisant un chariot mobile.

Plus simplement, une application du pendule inversé est le jeu qui consiste à faire tenir un balai (la tête de celui-ci étant vers le haut) dans la paume de la main, le plus longtemps possible, les actions sont généralement discrètes, on imprime une secousse vers la gauche ou vers la droite, ce qu'on appelle le régime de **contrôle bang-bang**.

[11] Pour plus d'informations, voir [11] Intelligence artificielle Stuart Russell et Peter Norving, 3eme édition page 899.

Miche et Chambers ont publié les premiers travaux sur cette forme d'apprentissage, leurs algorithmes BOXES était capable d'équilibrer la perche pendant plus d'une heure après seulement 30 essais, contrairement aux autres travaux similaires, BOXES n'était pas une simulation, il était implanté avec un vrai chariot et une vrai perche, l'algorithme discrétisait l'espace quadridimensionnel en éléments finis, ou boites, d'où son nom, l'algorithmes exécute plusieurs essais et s'arrête au moment où la perche tombe, ou que le chariot atteigne la fin de la piste, un renforcement négatif était associé à l'action finale de la dernière boite, puis rétro propagé dans toute la séquence, voir…( page 900).

Cette étude a plusieurs intérêts[13] :

-L'homme est en fait un pendule inversé double dont les deux axes de rotation sont les chevilles et les hanches. Afin de tenir debout, les articulations travaillent sans cesse, et l'étude de ce modèle est importante pour la construction de prothèses.

La robotique utilise le principe du pendule inversé, en particulier dans de nouveaux moyens de transports à 2 roues comme le segway qui permet d'avancer en se penchant en avant.

Parmi les méthodes les plus récentes, on peut citer la CMAC( Cerrebellar Model Articulation Contrôler), qui est en substance une somme de fonctions noyau locales avec recouvrement, et les réseaux de neurones associatifs, les réseaux de neurones sont actuellement la forme la plus utilisée d'approximation de fonction, l'approximation de fonction sur l'apprentissage de base d'exemples permet au programme d'éviter d'oublier les expériences les plus anciennes, ce qui est fréquent dans les programmes à base de réseaux de neurones.

La convergence des algorithmes d'apprentissage par renforcement reste un sujet très technique, les résultats de l'apprentissage TD-GAMMON ont été améliorés pour le cas des approximateurs linéaires, Williams (1992) a proposé les méthodes de recherche de politique et a la famille d'algorithmes REINFORCE, des travaux plus récents, Marbach et Tsitsiklis (1998), Baxter et Barlett (2000)[14].

À travers ces exemples, nous avons pu commencer à cerner l'utilité de l'apprentissage par renforcement. Aujourd'hui, la complexité de la multitude de systèmes d'information qui nous environnent appelle naturellement à une automatisation des processus et de leur paramétrage adaptatif. Tandis que les méthodes d'apprentissage statistiques classiques

[13] Le Pendule inverse Et son application en robotique, BRECHET Thomas et TRIBINO Julien, 4 avril 2008. https://www.pobot.org/IMG/pdf/Pendule_inverse_en_robotique.pdf

[14] Intelligence artificielle Stuart Russell et Peter Norving. 3eme edition.

commencent à trouver un écho dans le monde du big data, le renforcement comme moyen d'inférer un comportement complexe à partir de critères simples peine encore à trouver sa voie.

### Conclusion :

Nous pouvons donner à la machine la capacité d'apprendre de différentes manières, nous ne pouvons dire quelle méthode est la plus efficace, cependant, nous pouvons affirmer dans quel contexte, une méthode s'avère plus efficace qu'une autre.

Dans le cas ou nos résultats ne seraientt pas étiquetés, l'apprentissage non supervisé sera plus approprié, il est principalement utilisé en matière de clusterisation et de réduction de dimensions, dans le cas Dataset qui contient des questions $\boldsymbol{X}$ et des réponses $\boldsymbol{y}$, en d'autres termes, le cas où nos données sont étiquetées, l'apprentissage supervisé serait plus efficace, il est généralement utilisé dans le contexte de la classification et de la régression.

Dans les cas de problème d'optimisation, un algorithme d'apprentissage par renforcement serait le plus approprié, il diffère fondamentalement des problèmes supervisés et non supervisés par l'aspect interaction, agent, environnement.

# Partie 2 : Le Deep Learning

## Introduction :

Qu'est-ce que le Deep Learning ? qu'elle est sa place dans le monde de l'intelligence artificielle ? comment fonctionne le réseau de neurones artificiels ?

Nous l'avons vue au premier chapitre, le Machine Learning est un domaine de l'intelligence artificielle, qui consiste à développer un modèle, en se servant d'un algorithme d'optimisation pour minimiser les erreurs, entre le modèle et nos données.

Il y a différents modèles tels que les modèles linéaires, arbres de décision, support vector machine, chacun venant avec son algorithme d'optimisation, la descente de gradient pour les modèles linéaires, l'algorithme CART pour les arbres de décision, ou encore la marge maximum pour le support vector machine.

Maintenant, qu'en est-il du Deep Learning, le Deep Learning est un domaine du machine Learning dans lequel, au lieu de développer un des modèles qu'on vient de citer, on développe à la place des réseaux de neurones artificiels.

Le principe reste le même, on fournit à la machine des données, et elle utilise un algorithme d'optimisation pour ajuster le modèle a ses données, mais cette fois-ci, notre modèle n'est pas une seule fonction du type $f(x = ax + b)$, mais plutôt un réseau de fonctions connecté les unes aux autres, et plus ces réseaux sont profonds, plus la machine est capable d'apprendre à réaliser des tâches complexes, comme identifier une personne sur une photo, reconnaitre un objet, conduire une voiture….

Voici donc la raison pour laquelle on parle d'apprentissage profond, c'est-à-dire de Deep Learning, quand on développe des réseaux de neurones artificiels.

## 1. Histoire du Deep Learning[15] :

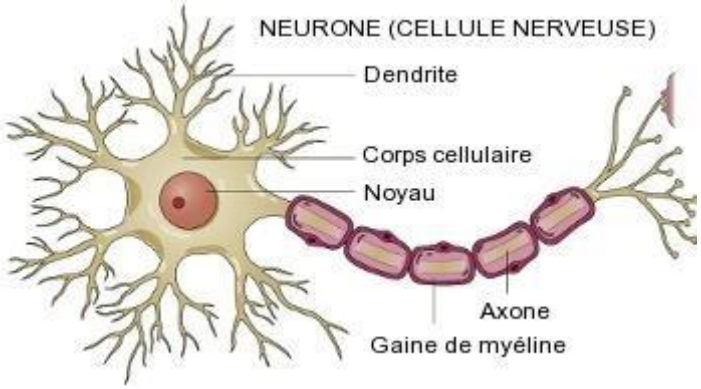
Pour bien comprendre le réseau de neurones artificiels, il faut revenir à leur histoire, et quelle fut leur évolution à travers le temps, pour arriver à la technologie que nous connaissons aujourd'hui.

### 1.1. Les premiers neurones artificiels[16] :

L es premiers réseaux de neurones ont été inventés en 1943 par deux mathématiciens et neuroscientifique du nom de **Warren Mculloch** et **Walter Pitts**, dans leur article scientifique intitulé « A logical Calculus of the ideas immanent in nervous activity », ils expliquent comment ils ont pu programmer des neurones artificiels en s'inspirant du fonctionnement des neurones biologiques. Rappelons-le, en biologie, les neurones sont des cellules excitables connectées les unes aux autres, et ayant pour rôle de transmettre des informations à notre système nerveux, chaque neurone et composé de plusieurs **dendrites,** d'une corp cellulaire et d'un **axone.**

[15] Pour plus d'information sur l'histoire du Deep Learning, voir :
Intelligene artificielle Stuart Russell et Peter Norvig 3eme édition, Chapitre 24 page 981.
Introduction au Deep Learning, Présentation et histoire du Deep Learning, J. Rynkiewicz, Université Paris 1. http://samm.univ-paris1.fr/IMG/pdf/coursdeep1-2.pdf

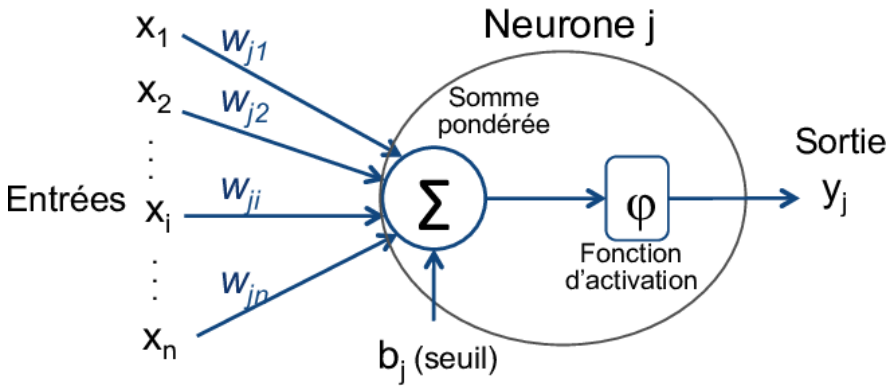
[16] Une brève histoire de l'apprentissage Gilbert Saporta, archives ouvertes.fr. https://hal-cnam.archives-ouvertes.fr/hal-02470857/document




Source : https://www.futura-sciences.com/sante/definitions/biologie-neurone-209/

Les dendrites sont en quelque sorte les portes d'entrée d'un neurone, c'est à cet endroit, au niveau de la synapse, que le neurone reçoit des signaux provenant des neurones qui le précédent, ces signaux peuvent être du type **excitateur** ou **inhibiteur**, un peu comme des signaux qui valent +1 et d'autres -1, quand la somme de ces signaux dépasse un certain seuil, le neurone s'active et produit alors un signal électrique, ce signal circule le long de l'axone en direction des terminaisons, pour être envoyé à son tour vers d'autres neurones de notre système nerveux, qui fonctionneront exactement de la même manière.

Warren Mculloch et Walter Pitts ont essayé de modéliser ce fonctionnement, en considérant qu'un neurone pouvait être représenté par une fonction de transfert, qui prend en entrée des signaux x et qui retourne une sortie Y.




Source : Approche pour la construction de modèles d'estimation réaliste de l'effort/coût de projet dans un environnement incertain : application au domaine du développement logiciel, Safae Laqrichi

À l'intérieur de cette fonction, on trouve deux grandes étapes, la première, est une étape d'agrégation, on fait la somme de toutes les entrées du neurone, en multipliant chaque entrée par un coefficient W.

Ce coefficient représente en fait, l'activité synaptique, le fait que le signal soit excitateur et vaut +1, ou bien, inhibiteur et vaut -1.

Fonction agrégation :$f = w1x1 + w2x2 + W3x3 \ldots + Wnxn.$

Une fois cette étape réaliser on passe à la phase d'activation :

Activation : $\begin{cases} y = 1 \; si \; f \geq 0 \\ y = 0 \; sinon \end{cases}$ [17]

On regarde le résultat du calcul effectuer précédemment, et si celui-ci dépasse un seuil, 0 en général, alors le neurone s'active et retourne une sortie y=1, sinon, il reste à 0.

Voilà comment Warren McClloch et Walter Pitts, ont réussi à développer les premiers neurones artificiels, plus tard renommés '' Threshold Logic Unit'', nom qui vient du fait qu'à l'origine leur modèle n'était conçus que pour traiter les entrées logiques, qui valent 0 ou 1, ils ont pu démonter qu'avec ce modèle, il était possible de reproduire certaines fonctions logiques, telles que la porte END et la porte OR.

Ils ont également démontré qu'en connectant plusieurs de ses fonctions les unes aux autres, un petit peu à la manière des neurones de notre cerveau, alors il serait possible de résoudre n'importe quel problème de logique booléenne. Cette annonce a suscité beaucoup d'engouement entour de l'intelligence artificielle, certaines personnes pensaient même qu'en quelques années, nous serions capables de développer des intelligences artificielles capables de remplacer l'homme, il n'en est rien pour le moment, car même si ce modèle pose les bases, de ce qu'est aujourd'hui le Deep Learning, il contient un certain nombres de failles, notamment le fait qu'il n'est pas d'algorithmes d'apprentissage, et qu'il faut trouver la valeur des paramètres W si on désire s'en servir.

## 1.2. Les premiers modèles à algorithmes d'apprentissage[18] :

Une quinzaine d'années plus tard, en 1957, un psychologue américain trouva comment améliorer ce modèle, en proposant le premier algorithme d'apprentissage de l'histoire du Deep Learning, il s'agit de Franck Rosenblatt, l'inventeur du perceptron.

Le modèle du perceptron ressemble de très près à celui que nous venons d'étudier, il s'agit d'un neurone artificiel qui s'active lorsque la somme pondérée des entrées dépasse un

[17] Introduction au Machine Learning, Agathe Azencott, Chapitre 7, page 94
[18] Les Echos, 1957 : le Perceptron, première machine apprenante.

certain seuil en général 0, mais avec cela, le perceptron dispose également d'un algorithme d'apprentissage, lui permettant de trouver les valeurs de ses paramètres w, afin d'obtenir les sorties y, qui nous conviennent.

Pour développer cet algorithme, Franck Ronsenblatt s'est inspiré de la théorie de Hebb, cette théorie suggère que lorsque deux neurones biologiques sont excités conjointement, alors ils renforcent leurs liens synaptiques, c'est-à-dire qu'ils renforcent les connexions qui les unissent, en neurosciences, c'est ce qu'on appelle la plasticité synaptique, et c'est ce qui permet à notre cerveau de construire sa mémoire et apprendre de nouvelles choses.

À partir de cette idée, Rosenblatt a développé un algorithme d'apprentissage, qui consiste à entrainer un neurone sur des données de référence (X,Y), pour que celui-ci renforce ses paramètres w à chaque fois qu'une entrée X est activée en même temps que la sortie Y présente dans ces données, pour cela il a imaginé la formule suivante, dans laquelle les paramètres W sont mis à jour, en calculant la différence entre la sortie de référence, et la sortie produite par le neurone, et en multipliant cette différence par la valeur de chaque entrée X, ainsi que par un pas d'apprentissage positif.

$$W = W + a(Ytrue - y)X$$[19]

Ytrue : sorite de référence.

Y : sortie produite par le neurone.

X : entrée de neurone.

a : vitesse d'apprentissage.

De cette manière, si notre neurone produit une sortie différente de celle qu'il est censé produire, par exemple Y=0, alors qu'on voudrait avoir y=1, alors notre formule nous donnera :

$$W = W + a(Ytrue - y)X$$
$$W = W + a(1 - 0)X$$
$$W = W + aX$$

Donc pour les entrées x qui valent 1, le coefficient w se verra augmenté d'un petit pas alpha, il sera renforcé, ce qui provoquera une augmentation de la fonction f=w1.x1+w2.x2+W3.x3…+Wn.xn, et qui rapprochera notre neurone de son seuil d'activation,

[19] Machine Learning, Massih-Reza Amini, chapitre 4, Pages 81-111

tant qu'on sera en dessous de ce seuil notre coefficient continue d'augmenter jusqu'au moment où Ytrue vaudra Y, ce qui nous donnera : W=W+0, nos paramètres arrêteront d'évoluer.

On pensait que grâce aux perceptrons, on pourrait créer des machines capables, de lire, écrire et marché, mais cet engouement s'effondrera en partie car le perceptron est un modèle linéaire.

On connaît alors une période, entre 1974 et 1980, période durant laquelle il n'y eut quasiment plus d'investisseurs pour financer les recherches en IA. Heureusement tout changea en 1980, lorsque Geoffrey Hinton, développa le perceptron multicouche, le premier véritable réseau de neurones artificiels.

### 1.3. Le perceptron multicouche[20] :

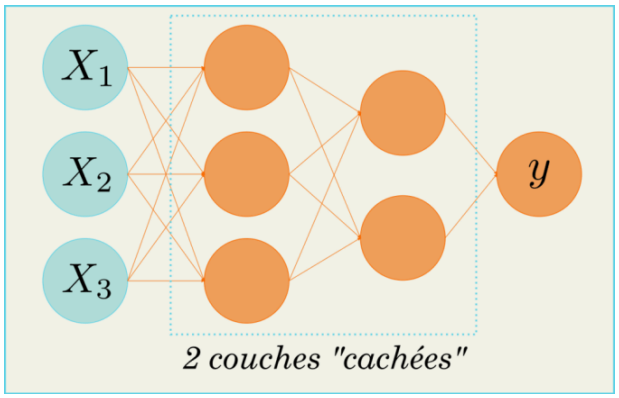
Le perceptron est un modèle linéaire dont l'inclinaison de la droite, dépendent des paramètres w, et dont la position peut être modifiée à l'aide d'un paramètre complémentaire, le biais, le seul ennui, c'est qu'une grande partie des phénomènes de notre univers ne sont pas linéaires, et dans ces conditions, le perceptron à lui seul ne suffit pas.

Comme nous l'avons vu dans le modèle de McCulloch et Pitts, en connectant plusieurs neurones, il est possible de résoudre des problèmes plus complexes qu'avec un seul, voyons ce qui se passe quand on connecte trois perceptrons ensemble :

Les deux premiers reçoivent chacun les entrées x1 et x2 et font leur calcules, ils retournent une sortie y, qu'ils envoient à leur tour vers le troisième perceptron, qui va lui aussi faire ses petits calculs pour produire une sortie finale.

Si on trace la représentation graphique finale en fonction des entrées x1 et x2, on obtient cette fois un modèle non linéaire, bien plus intéressant.

[20] Elagage d'un perceptron multicouche : utilisation de l'analyse de la variance de la sensibilité des paramètres, PHILIPPE THOMAS, ANDRE THOMAS, Centre de Recherche en Automatique de Nancy CRAN-UMR 7039), Nancy-Université, CNRS.




Source : https://www.second-glance.fr/2017/02/06/le-deep-learning-1-une-tentative-de-vulgarisation/

Trois neurones répartis en 2 couches, c'est ce qu'on appelle un **perceptron multicouche**, rajouter des couches rend le résultat plus complexe, mais plus intéressant.

Cependant, une question subsiste, comment entraîner un tel réseau de neurones, pour qu'il fasse ce qu'on lui demande de faire ? Comment trouver tous les paramètres de W et b de façon à obtenir un bon modèle ?

La solution est d'utiliser une technique qui s'appelle Back Propagation, qui consiste à déterminer comment la sortie du réseau varie, en fonction des paramètres présents dans chaque couche du modèle, pour faire cela, on calcule une chaîne de gradients, indiquant comment la sortie varie en fonction de la dernière couche, puis comment la dernière couche varie en fonction de l'avant-dernière…etc., jusqu'à arriver à la première couche de notre réseau, c'est une propagation vers l'arrière, avec ces informations, ces gradients, on peut alors mettre à jour les paramètres de chaque couche, de telle sorte a ce qu'il minimise l'erreur entre la sortie du modèle et la réponse attendus, et pour cela, on utilise une formule proche de Frank Rosennlatt, c'est la formule de la **descente de gradient.**

Pour développer des réseaux de neurones artificiels, on répète les quatre étapes suivantes :

1. L'étape de forward propagation : on fait circuler les données de la première couche jusqu'à la dernière, afin de produire une sortie Y.
2. Calculer les erreurs ente cette sortie et la sortie de référence Ytrue que l'on désire avoir, pour cela, on utilise une **fonction coût.**
3. Backward Propagation : on mesure comment cette fonction coût varie, par rapport à chaque couche de notre modèle, en partant de la dernière et en remontant jusqu'à la toute première.
4. Gradient Descent : corriger chaque paramètre du modèle grâce à l'algorithme de la descente de gradient, avant de revenir à la première étape, celle de la forward propagation, pour recommencer un cycle d'entraînement.

## 1.4. Le Deep Learning moderne :

Au fil du temps, le modèle du perceptron multicouche continua à d'évoluer, notamment avec l'apparition de nouvelles fonctions, tel que la fonction logistique, tangente hyperbolique, ou la fonction rectified linear unit, ces fonctions ont aujourd'hui remplacé la fonction **Heaviside,** qu'on a vue jusqu'à présent car elles offre de bien meilleures performances, dans les années 1990, on commençait à développer les premiers variants du perceptron multicouche, Yann LeCun inventa les premiers réseaux de neurones convolutifs, réseaux capables de traiter et reconnaître les images, en introduisant au début de ces réseaux des filtres mathématiques qu'on appelle convolution et pooling.

C'est également dans les années 90 qu'on a vus apparaître les premiers réseaux de neurones convolutifs, qui sont encore une fois, une variante du perceptron multicouche, et qui permettent de traiter efficacement des problèmes de séries temporelles, comme la lecture de textes et la reconnaissance vocale.

Les raisons pour lesquelles les technologies d'aujourd'hui ne s'utilisaient pas dans les années 90, sont que, pour bien fonctionner, un réseau de neurones doit être entraîné sur de très grosse quantités de données, or dans les années 90 on ne disposait pas d'autant de données, il a fallu attendre l'arrivée d'internet, des smartphones, et des objets connectés, pour commencer à collecter de grosses quantités de données, et d'images pouvant être exploités pour le Deep Learning.

Le Deep Learning n'a réellement pris son envol qu'en 2012, lors d'une compétition de vision par ordinateurs nommée ImaNet, où une équipe de chercheurs menée par Geoffrey Hinton, développa un réseau de neurones, capable de reconnaître n'importe quelle image avec une meilleure performance que tous les autres algorithmes de l'époque, depuis ce jour, on vante sans arrêt les mérites de cette technologie, et en allant même à la comparer au cerveau humain.

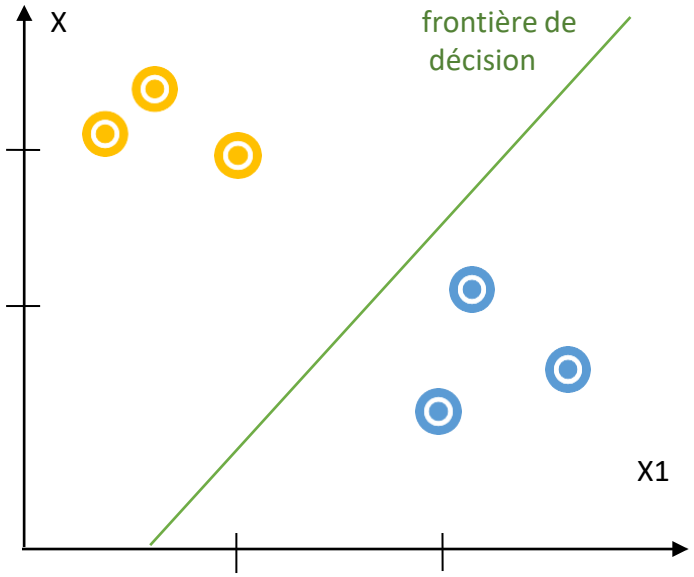
Comme le dit Yann LeCun, 'comparer un réseau de neurones à un cerveau humain revient à comparer un avion à un oiseau, car en effet, nous nous sommes peut-être inspirés de ce que l'on a vu dans la nature, mais ce n'est pas pour autant que les avions volent en battant des ailes.'

## 2. LE perceptron [21]:

Le perceptron est l'unité de base des réseaux de neurones, il s'agit d'un modèle de classification binaire, capable de séparer linéairement deux classes de points.

Pour nous aider à comprendre, prenons un exemple, imaginez qu'on ait deux types de plantes, des plantes toxiques qu'on va noter Y=1, et non-toxiques qu'on note Y=0, imaginons que l'on décide de mesurer certains attributs de ces plantes, comme la langueur et la largeur de leurs feuilles que l'on va noter respectivement x1 et x2.

| Y | X1 | X2 |
|---|---|---|
| 1 | 0.5 | 2.0 |
| 1 | 1.1 | 2.1 |
| 1 | 0.7 | 2.6 |
| 0 | 2.0 | 1.0 |
| 0 | 2.5 | 0.7 |
| 0 | 2.2 | 0.3 |

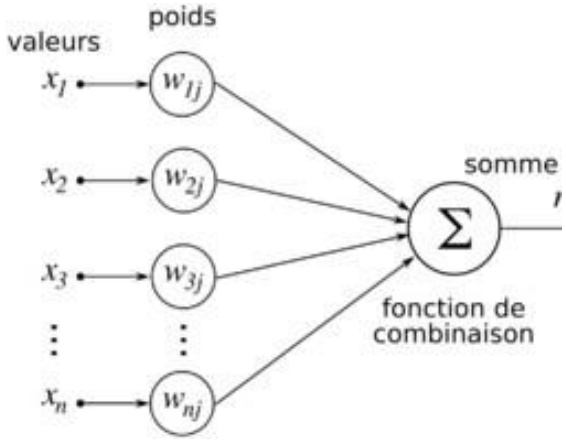



On représentant ces deux classes de plantes dans un graphique, on observe qu'elles sont linéairement séparables, on se basant sur cette droite, on peut prédire à quelle classe appartient une plante, en se basant sur cette droite, communément appelé la frontière de décision, si un point est à droite de cette droite, il sera considéré comme étant une plante non toxique, et il appartient à la classe 0, s'il se trouve à gauche, il sera considérer comme une plante toxique et il appartiendra a la classe 1, pour pouvoir donc classer ces plantes et développer ce modèle nous devons donc connaitre l'équation de cette droite.

Pour ça, on va développer un modèle linéaire en fournissant toutes nos variables x1 et x2 à un neurone, et en multipliant chaque entrée du neurone par un poids w, on fait passer

[21] LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME Claude Touzet, archives ouvertes.fr. https://hal-amu.archives-ouvertes.fr/hal-01338010/file/Les_reseaux_de_neurones_artificiels.pdf.
Apprendre le machine Learning en une semaine Guillaume Saint-Cirgue.
TP5 : Perceptron, Université Paul Sabatier. https://www.math.univ-toulouse.fr/~xgendre/ens/m2se/M2SE_TP5.pdf.

également un coefficient complémentaire qui correspondra au biais b, ce qui nous donnera la fonction suivante :




Source : Centre national de la recherche scientifique.

$$Z(x1, x2) = w1x1 + W2x2 + b^{22}$$

Sur notre graphique, à gauche de la frontière, notre fonction Z sera positive, à droite elle sera négative, notre droite correspond aux valeurs de x1 et x2 où : $Z(x1, x2) = 0$.

Pour trouver ou appartiennent nos futures plantes, il va falloir régler nos paramètres W et b, de façon à séparer du mieux possible nos deux classes.

$$\begin{cases} ypred = 0 \; si \; Z < 0 \\ ypred = 1 \; si \; Z \geq 0 \end{cases}$$

C'est de cette façon que fonctionne le premier neurone de l'histoire du Deep Learning, le perceptron, maintenant, pour améliorer ce modèle et le rendre plus performant, une bonne chose à faire, serait d'accompagner chaque prédiction d'une probabilité, de cette manière, plus un point est éloigné de notre droite de décision, plus il est probable que ce point appartienne vraiment à cette classe, plus un point est proche de cette droite, moins il a de chance d'appartenir réellement à cette classe, pour cela on peut utiliser une fonction d'activation qui nous retournera une sortie de 0 ou de 1 au fur et à mesure que l'on s'éloigne de la frontière de décision ou Z est égal à 0.

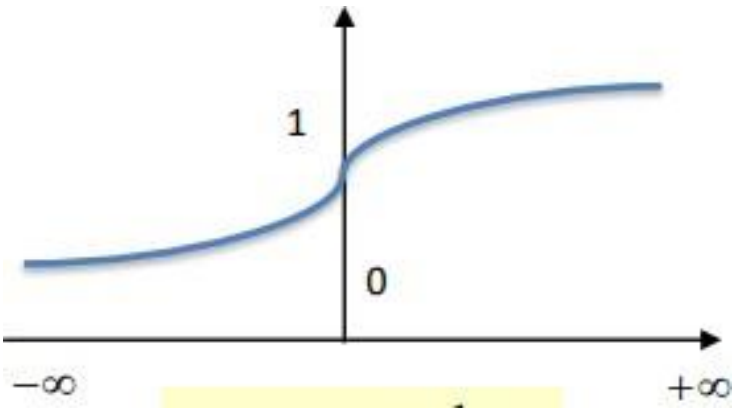
La fonction qui va nous permette de faire ça, est la **fonction sigmoïde** (logistique)

## 2.1. La fonction sigmoïde :

$$a(z) = \frac{1}{1 + e^{-z}}^{23}$$

[22] Apprendre le Machine Learning en une semaine, chapitre 5, page 61.

## Fonction sigmoïde (logistique)




Source : Ricco Rakotomalala, Deep Learning, Perceptrons simples et multicouches, univ Lyon 2 p13.

Cette fonction permet de convertir la sortie Z, en une probabilité a(z), celle qu'une plante appartienne à la classe 1, une plante dont la valeur de Z=1.4, sa probabilité a(Z) sera de :

$$a(1{,}4) = \frac{1}{1 + e^{-1{,}4}}$$

$$a(1{,}4) = 0.8$$

D'après notre modèle, cette plante a 80% de chance d'appartenir à la classe 1, ce qui est logique car Z>0.

Une plante dont la valeur de Z est de -2.1, cela nous donne une probabilité a(z)=0.1, d'après, notre modèle, cette plante a seulement 10% de chance d'appartenir à la classe 1, une probabilité bien plus faible, mais ceci est logique car celle-ci se situe à gauche de notre frontière de décision ou Z<0, on pourra dire à la place qu'elle a 90% de chances d'appartenir à la classe 1, probabilité complémentaire de celle que nous avons calculé.

Ce que l'on peut remarquer est que ces probabilités suivent une loi de Bernoulli, car pour Y=0 et Y=1 :[24]

$$P(y = 1) = a(z).$$

$$P(y = 0) = 1 - a(z).$$

[23] Ricco Rakotomalala, Deep Learning, Perceptrons simples et multicouches, univ lyon 2 p13.
[24] Voir Apprentissage supervisé, Aurélien Garvier, université de Toulouse. https://www.math.univ-toulouse.fr/~agarivie/mydocs/apprentissageSupervise.pdf

## 2.2. Loi de Bernoulli :

$$\mathrm{P}(\mathrm{Y}=\mathrm{y}) = \mathrm{a}(z)^{y} \times (1-a(z))^{1-y}$$

En décomposant les deux cas, celui où Y=1, et celui où Y=0 on obtient :

$$\mathrm{P}(\mathrm{Y}=0) = \mathrm{a}(z)^{0} \times (1-a(z))^{1-0}$$

$$\mathrm{P}(\mathrm{Y}=1) = \mathrm{a}(z)^{1} \times (1-a(z))^{1-1}$$

Ce qui nous amène aux deux expressions d'avant :

$$P(y=1) = a(z).$$

$$P(y=0) = 1-a(z).$$

**Résumé :**

Ce qui se trouve à l'intérieur d'un neurone est une fonction linéaire de la forme :

$$Z(x1,x2) = w1x1 + W2x2 + b$$

Celle-ci est suivie d'une fonction d'activation, la plus simple étant la fonction sigmoïde, qui nous retourne une probabilité qui suit une loi de Bernoulli.

## 2.3. La vraisemblance :

Notre but est de régler les paramètres W et b de façon à obtenir le meilleur modèle possible, celui qui minimise au mieux nos erreurs entre les sorties a(Z) et les vraies données y. Pour ce faire nous allons commencer par définir une fonction coût qui va permettre, de mesurer ces erreurs :

Une fonction coût est une fonction qui permet de quantifier les erreurs effectuées par un modèle. Dans notre modèle, celle-ci va permettre de mesurer les écarts entre les sorties a(Z), et les données Y dont nous disposons, pour cela, la fonction coût qui sera utilisée, est la fonction Log Loss.

L'une des façons de calculer la performance d'un modèle est de calculer sa **vraisemblance**, en statistiques celle-ci nous indique la plausibilité de notre modèle vis-à-vis des vraies données.

Nous connaissons les caractéristiques de nos plantes, nous savons lesquelles sont toxiques et lesquelles ne le sont pas, nous verrons si les prédictions du modèle sont en accord ou non avec ces données.

Si une plante est toxique, est que le modèle nous retourne une probabilité que celle-ci soit toxique à 90%, alors il est lui-même vraisemblable à 80%.

Pour calculer la vraisemblance du modèle, nous allons faire la somme de toutes les probabilités que le modèle nous retourne, en utilisant la loi de Bernoulli que nous avons vue.

$$L = \prod_{i=1}^{m} ai^{yi} \times (1 - ai)^{1-yi}$$

Si ce résultat, est de 100%, alors notre modèle est en accord parfait avec nos données, à l'inverse, un résultat proche de 0, alors notre modèle sera considéré comme fortement invraisemblable, cela signifiera que toutes les données dont nous disposons sont fausses.

Cependant, la vraisemblance cause un problème, car comme on effectue un produit de probabilités, cela signifie que, plus on a de données, plus le résultat sera proche de 0, évidemment ceci n'est pas un problème quand la quantité de données dont nous disposons n'est pas importante, mais si l'on travaille sur des milliers voir des dizaines de milliers de données, notre résultat sera tellement proche de 0, que la machine ne pourra même plus stocker ce chiffre dans sa mémoire.

Il faudra donc trouver une solution qui va permettre de réaliser ces calculs sans pour autant, fausser nos calculs et converger vers 0, une solution est d'utiliser le logarithme, car le logarithme d'un produit est égal à la somme des logarithmes[25] :

$$\log(ab) = \log(a) \times \log(b)\,.$$

Donc :

$$\log(L) = \log\left(\prod_{i=1}^{m} ai^{yi} \times (1 - ai)^{1-yi}\right)$$

$$L = \prod_{i=1}^{m} ai^{yi} \times (1 - ai)^{1-yi}$$

[2525] Pour les formules mathématiques, voir, introduction au Deep Learning Eugene Chatniak.

$$\log(L) = \log(L1) + \log(L2) + \cdots \ldots \ldots + \log(Lm)$$

On pourrait penser que ceci va complètement déformer nos calculs, mais dans la réalité, ce n'est pas le cas, car comme la fonction logarithme est une fonction monotone croissante, elle conserve l'ordre de nos termes, c'est-à-dire, si a<b, alors log(a)<log(b).

Cela signifie, que chercher le maximum de notre vraisemblance, revient à chercher le maximum du log de notre vraisemblance.

$$LL = \log\left(\prod_{i=1}^{m} ai^{yi} \times (1 - ai)^{1-yi}\right)$$

$$LL = \sum_{i=1}^{m} \log\left(ai^{yi} \times (1 - ai)^{1-yi}\right)$$

$$LL = \sum_{i=1}^{m} \log(ai^{yi}) + \log\left((1 - ai)^{1-yi}\right)$$

$$LL = \sum_{i=1}^{m} \log(ai^{yi}) + \log\left((1 - ai)^{1-yi}\right)$$

$$LL = \sum_{i=1}^{m} \text{yi}\log(ai) + (1 - yi)\log\left((1 - ai)\right)$$

Cette fonction ressemble de très près à notre fonction coût, celle que l'on cherche à avoir, la seule différence, est qu'il manque le facteur $\frac{-1}{m}$ au début, ce qui est normal, car jusque-là nous avons cherché à calculer le log de notre vraisemblance, qu'on cherche à maximiser pour avoir le meilleur modèle possible, or en mathématiques les algorithmes de maximisation n'existent simplement pas, ce qu'on utilise à la place sont des algorithmes de minimisation, mais ceci n'est pas un problème car :

$$maximiser\ f(x) = minimiser - f(x)$$

C'est pourquoi, pour maximiser la vraisemblance de notre modèle, ce qu'on fait, c'est minimiser sa fonction négative, d'où la présence du -1 au début de l'expression, le facteur 1/m lui n'est là que pour nous aider à normaliser notre résultat.

Maintenant, que nous avons notre fonction coût, on pourra s'en servir pour minimiser les erreurs de notre modèle, et pour cela nous allons utiliser l'algorithme de la **descente de gradient.**

## 2.4. La descente de gradient [26]:

Celui-ci est l'un des algorithmes les plus utilisés en Machine Learning et en Deep Learning, il consiste à ajuster les paramètres W et b de façon à, minimiser les erreurs du modèle, c'est-à-dire, à minimiser la fonction cout, pour ça, il faudra déterminer la variation de celle-ci en fonction des différents paramètres, est-ce qu'elle diminue quand W augmente ou l'inverse, pour répondre à cette question on calcule se qu'on appelle un gradient(une dérivée), la dérivée de notre fonction cout, la dérivée d'une fonction nous indique comment celle-ci varie.

Si $\frac{\partial L}{\partial W} < 0$, la fonction coût diminue quand W augmente, il faudra augmenter W si l'on veut réduire nos erreurs.

Si $\frac{\partial L}{\partial W} > 0$, la fonction coût augmente quand W augmente, il faudra donc diminuer W si l'on veut réduire nos erreurs.

Pour faire cela, nous allons utiliser la formule suivante[27] :

$$W_{t+1} = W_t - a\frac{\partial L}{\partial W}$$

$W_{t+1}$: $paramètre\ W$ à $instant\ t+1$.

$W_t$: $paramètre\ W$ à $l'instant$.

$a$: $pas\ d'apprentissage\ positif$.

$\frac{\partial L}{\partial W}$: $gradient$ à $l'instant$.

[26] Algorithme de descente du gradient stochastique, Laura GAY, Université Claude Bernard Lyon 1.
Descente de gradient Bruno Bouzy 5 octobre 2005.
Apprendre le Machine Learning en une semaine, Guillaume Saint-Cirgue.

De cette manière, si le gradient est négatif, alors W va augmenter grâce à notre formule, s'il est positif, alors W va diminuer, en répétant cette fonction en boucle, on est ainsi capable d'atteindre le minimum de notre fonction coût en descendant progressivement sa courbe, d'où le terme de descente de gradient.

Un critère doit être respecté pour que cela fonctionne, notre fonction doit être convexe, c'est-à-dire ne présente pas de minimum local sur lequel notre algorithme pourrait rester bloqué.

## 3. Application du Machine Learning en finance [28]:

Nous l'avons vue dans la première section, les domaines d'applications du Machine Learning sont multiples, dans cette section, nous allons voir son application en finance.

### 3.1. La gestion des risques :

La gestion des risques est un domaine très important pour les entreprises et les institutions financières, celle-ci a un impact direct sur les différentes décisions prises et sur l'orientation stratégique de l'entreprise, en raison des différents concurrents sur le marché, et de la réglementation des politiques gouvernementales.

Dans le domaine de la finance, nous disposons d'énormes quantités de données des clients, cours boursiers…. Et autres données financières, en utilisant des modèles de Machine Learning et de Deep Learning, nous pouvons apprendre à mieux prévoir, et à prendre de meilleures décisions en matière de risque.

### 3.2. La détection de fraudes :

Assurer la sécurité de ses clients est d'une importance capitale pour toutes les institutions financières, les experts en Machine Learning sont capables de créer des algorithmes capables de détecter et de trouver des anomalies ou des fraudes, la détection de fraudes peut être notamment utilisée dans le domaine bancaire à cette fin.

Les études de fraudes concernent aussi les marchés financiers. On peut distinguer différents types de fraudes dont les fraudes de rendement (e.g. chaines de ponzi, frais de courtages injustifiés), les fraudes de brokers (deals non autorisés à partir des titres de clients, deals

[28] Machine learning en finance : vers de nouvelles stratégies ? Vincent Bouchet, archives-ouvertes.fr.

réalisés en dehors des heures de marché) ainsi que les manipulations de marché (Golmohammadi et Zaiane, 2012) Les données utilisées sont à la fois variées (articles, messageries de traders, bases de données répertoriant les opérations) et volumineuses (plus de 5000 transactions par seconde sur le NASDAQ, Golmohammadi and Zaiane (2012)).

### 3.3. Le trading algorithmique :

Le trading algorithmique, aussi appelé trading automatisé ou trading automatique, à un impact sur les données financières en temps réel. Celles-ci impliquent des données structurées et non structurées, de nombreuses sociétés et institutions financières ont pris des décisions utiles et bénéfiques à partir de données en temps réel, les sociétés financières engagent des experts en Machine Learning et en Deep Learning qui sont chargés d'utiliser ces données en temps réel pour créer des modèles prédictifs pour prévoir les opportunités du marché financier.

De cette manière, les traders peuvent définir des instructions préprogrammées liées à certaines variables (telles que le temps, le prix, le volume). Ces instructions sont basées sur leur stratégie de trading. Lorsque les conditions du marché sont dans les paramètres programmés, l'algorithme suit cette stratégie et gère les transactions entrées sorties sur le compte du trader.

### 3.4. Gestion des risques d'un portefeuille d'actifs financier :

Une théorie qui s'applique à la gestion de portefeuille appelé CAPM (capital asset pricing model), modèle d'évaluation des actifs financiers, l'idée est d'utiliser des données historiques pour définir le risque systématique d'une action ou d'un actif financier au rapport au marché[29].

Afin de mesurer ce risque, il faut définir la notion du rendement d'une action et d'un coefficient beta :

Rendement : Le rendement d'une action est le rapport, exprimé en pourcentage, entre le dividende et le cours de Bourse. C'est donc le revenu annuel que procure, à un instant

[29]Le modèle CAPM, célèbre technique d'évaluation : https://www.abcbourse.com/apprendre/19_capm.html

donné, une action à son détenteur, en supposant que le dividende soit maintenu. A dividendes constants, plus les cours montent, plus le rendement diminue. (Source BFM Bourse)

Formule mathématique :

$$R_i = \frac{P_i - P_{i-1}}{P_{i-1}} 30$$

$R_i$: $rendement\ a\ l'instant\ i.$

$P_i$: $prix$ à $l'instant\ i.$

$P_{i-1}$: $prix$ à $l'instant\ i-1.$

a) Le coefficient beta :

Le Bêta est un outil de mesure du risque d'un actif notamment utilisé dans le modèle d'évaluation du CAPM. On l'utilisera entre autres pour mettre en place des stratégies de limitation des risques.

Le principe de cet outil est de comparer les mouvements effectués par un actif par rapport à son marché de référence, ce qui permet de déterminer son niveau de risque par rapport aux autres actifs de référence. La mesure est effectuée en comparant la rentabilité de l'actif à celle du marché. On peut aussi réduire le marché à un indice boursier (CAC 40, Dow Jones, etc.).

Pour le calculer mathématiquement :

Il se définit comme le rapport de la covariance de la rentabilité de l'actif avec celle du marché à la variance de la rentabilité du marché. (Source ABC Bourse.[31])

$$\beta = \frac{Cov(r_p, r_m)}{Var(r_m)}$$

Avec la série temporelle de l'évolution de l'indice de marché, on peut déterminer un rendement, ce rendement est appelé le rendement de marché, en faisant la même chose pour l'évolution de l'action de l'entreprise, ceci va permettre de déterminer le rendement de

[30]RENDEMENT, BFM Bourse : https://www.tradingsat.com/lexique-boursier/definition-rendement-192.html
[31] Voir : https://www.tradingsat.com/lexique-boursier/definition-rendement-192.html

l'action, ces calculs vont nous permettre de réaliser une régression linéaire simple dont la pente est déterminée par le coefficient bêta.

Ces instruments sont utilisés par les investisseurs pour mesurer les risques systématiques d'une action ou d'un actif financier par rapport au marché.

## Conclusion :

Le Deep Learning est un sous-domaine de l'apprentissage machine (Machine Learning), qui est particulièrement efficace sur des données représentées ou stockées sans format prédéfini et non structuré, les méthodes d'apprentissage neurones sont utilisées pour analyser des images ou du texte en grande quantité, comme en disposent les GAFA (Google, Amazon, Facebook, Apple) pour identifier des objets et des visages dans les photos ou du sentiment ou des sujets dans du texte.

- **Application programme apprentissage supervisé sur R :**

Nous allons voir sur cette application comment construire un modèle de régression, pour prédire la valeur d'une variable numérique, nous verrons comment diviser nos données en données d'entraînement pour entraîner notre algorithme, et en données test pour évaluer sa performance en calculant l'erreur quadratique moyenne.
Nous allons utiliser une base de données de différents appartements dans la banlieue de Boston(https://www.kaggle.com/puxama/bostoncsv).

Information sur la base de données utilisée :

**Source :** Université Carnegie Mellon, bibliothèque StatLib,

**Créateur :** Harrison, D. et Rubinfeld, DL « Prix hédoniques et le demande d'air pur », J. Environ. Economie & Gestion, vol.5, 81-102, 1978.

**Date :** 7 juillet 1993

**Utilisation passée :**

- Utilisé dans Belsley, Kuh & Welsch, 'Regression diagnostics...', Wiley, 1980. NB Diverses transformations sont utilisées dans le tableau sur pages 244-261.

- Quinlan, R. (1993). Combiner l'apprentissage basé sur les instances et basé sur les modèles. Dans Actes de la dixième conférence internationale de la machine Apprentissage, 236-243, Université du Massachusetts, Amherst. Morgan Kaufmann.

**Description de la base :**

Concerne la valeur des logements dans la banlieue de Boston.

Nombre d'instances : 506 Nombre d'attributs, 13 attributs continus (y compris « classe » attribut "MEDV"), 1 attribut à valeur binaire.

**Informations sur les attributs :**

1. Taux de criminalité par habitant du CRIM par ville
2. Proportion ZN des terrains résidentiels zonés pour les lots de plus de 25 000 pi.ca.
3. Proportion INDUS d'acres commerciales non commerciales par ville
4. Variable muette CHAS Charles River (= 1 fleuve ; 0 sinon)
5. Concentration d'oxydes nitriques NOX (parties pour 10 millions)
6. RM nombre moyen de pièces par logement
7. Proportion d'ÂGE des logements occupés par leur propriétaire construits avant 1940
8. Distances pondérées par DIS à cinq centres d'emploi de Boston
9. Indice RAD d'accessibilité aux autoroutes radiales
10. TAXE au taux d'impôt foncier sur la valeur totale par tranche de 10 000 $
11. Ratio élèves-enseignant PTRATIO par ville
12. B 1000(Bk - 0.63) ^2 où Bk est la proportion de noirs par ville
13. LSTAT % statut inférieur de la population
14. MEDV Valeur médiane des logements occupés par leur propriétaire en milliers de dollars

Pour pouvoir réaliser ce programme d'apprentissage automatique, nous allons utiliser le **Library(caret),** (Classification And Régression Training), librairie qui couvre une large fraction du Machine Learning, notamment en analyse prédictive (classement et régression).

```
library(caret)
#importation de données#
boston <- read.csv('https://raw.githubusercontent.com/JosueAfouda/Machine-Learning-par-la-pratique-avec-Python/master/Boston.csv')
head(boston)
#structure de nos données
str(boston)

#résumé statistique
summary(boston)

#mélange aléatoire des données

seed <- 123
set.seed(seed)

rows<-sample(nrow(boston))

boston_mel<-boston[rows, ]
head(boston_mel)

split<-round(nrow(boston_mel)*0.80)
```

Nous avons d'abord importé nos données grâce à la fonction Library, et nous avons aussi importé nos données, la fonction **head** permet d'afficher les observations.

```
     CRIM ZN INDUS CHAS   NOX    RM  AGE    DIS RAD TAX PTRATIO LSTAT MEDV
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

La fonction **str(boston)** nous permet de voir la structure de nos données structure de nos données :

```
> str(boston)
'data.frame':   506 obs. of  13 variables:
 $ CRIM   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ ZN     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ INDUS  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ CHAS   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ NOX    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ RM     : num  6.58 6.42 7.18 7 7.15 ...
 $ AGE    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ DIS    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ RAD    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ TAX    : int  296 242 242 222 222 222 311 311 311 311 ...
 $ PTRATIO: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ LSTAT  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ MEDV   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Nous avons 506 observations et 13 variables, chaque variable représente un attribue de cette maison, La variable MEDV est notre variable d'intérêt. Elle indique le prix des maisons en millier de dollars.

La fonction **summary(boston),** nous donne une description statistique de la table de donnée, la fonction renvoie 5 valeurs : le minimum (Min.), le premier quartile (1st Qu.), la médiane (Median), la moyenne (Mean), le troisième quartile (3rd Qu.) et le maximum (Max).

```
> summary(boston)
      CRIM                ZN              INDUS            CHAS              NOX              RM              AGE        
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000   Min.   :0.3850   Min.   :3.561   Min.   :  2.90  
 1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000   1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02  
 Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000   Median :0.5380   Median :6.208   Median : 77.50  
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917   Mean   :0.5547   Mean   :6.285   Mean   : 68.57  
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000   3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08  
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000   Max.   :0.8710   Max.   :8.780   Max.   :100.00  
      DIS              RAD              TAX             PTRATIO          LSTAT            MEDV      
 Min.   : 1.130   Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 1.73   Min.   : 5.00  
 1st Qu.: 2.100   1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95   1st Qu.:17.02  
 Median : 3.207   Median : 5.000   Median :330.0   Median :19.05   Median :11.36   Median :21.20  
 Mean   : 3.795   Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :12.65   Mean   :22.53  
 3rd Qu.: 5.188   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95   3rd Qu.:25.00  
 Max.   :12.127   Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :37.97   Max.   :50.00  
```

Nous pouvons noter que d'après nos résultats, il n'y a pas de valeurs manquantes dans la data frame.

Une fois que nous avons chargé notre Library caret, importé nos données, et vérifié leurs structures, nous avons lancé un processus aléatoire pour mélanger nos données, et défini un

seed afin de spécifier la valeur de l'amorce du générateur aléatoire, ce qui est utile si on veut répéter une simulation absolument à l'identique.

```
#mélange aléatoire des données

seed <- 123
set.seed(seed)

rows<-sample(nrow(boston))

boston_mel<-boston[rows, ]
head(boston_mel)

split<-round(nrow(boston_mel)*0.80)
```

Avec la fonction Head(boston_mel), nous pouvons voir que nos données ne sont plus dans le même ordre qu'au début.

```
> head(boston_mel)
        CRIM ZN INDUS CHAS   NOX    RM   AGE    DIS RAD TAX PTRATIO LSTAT MEDV
415 45.74610  0 18.10    0 0.693 4.519 100.0 1.6582  24 666    20.2 36.98  7.0
463  6.65492  0 18.10    0 0.713 6.317  83.0 2.7344  24 666    20.2 13.99 19.5
179  0.06642  0  4.05    0 0.510 6.860  74.4 2.9153   5 296    16.6  6.92 29.9
14   0.62976  0  8.14    0 0.538 5.949  61.8 4.7075   4 307    21.0  8.26 20.4
195  0.01439 60  2.93    0 0.401 6.604  18.8 6.2196   1 265    15.6  4.38 29.1
426 15.86030  0 18.10    0 0.679 5.896  95.4 1.9096  24 666    20.2 24.39  8.3
```

La fonction **Split** permet de diviser les données en groupes en fonction des niveaux de facteur, nous avons divisé en deux, données d'entrainement et d'évaluation.

```
# Train data

train <- boston_mel[1:split, ]

# Test data

test <- boston_mel[(split + 1):nrow(boston_mel), ]
# Dimension du train data

  dim(train)
```

La fonction **dim(train)** permet de vérifier la division de nos données, résultat 405, environ 80% comme définit au départ.

Une fois la division de nos données faite, nous allons passer maintenant à la construction du modèle et à son évaluation.

**Script R :**

```
#construction du modéle à partir des données d'entraînement

model <-lm(MEDV~., data = train)

model
# fonction d'évaluation du modèle
model_evaluation <- function (Model) {
  # Prédiction sur le train data

preds_train <- predict(Model, train)

# Prédictions sur le test data

preds_test <- predict(Model, test)

# Erreur sur le train data

error_train <- preds_train - train[, "MEDV"]

# Erreur sur le test data

error_test <- preds_test - test[, "MEDV"]

# RMSE sur les train data

rmse_train <- sqrt(mean(error_train ^ 2))
# RMSE sur le test data

rmse_test <- sqrt(mean(error_test ^ 2))

print(paste("RMSE sur le train data :", rmse_train))

print(paste('RMSE sur le test data :', rmse_test))
}
```

L'évaluation du modèle permet de voir si le modèle est performant sur des données qui n'ont pas servi à son entraînement, comme il s'agit d'une tâche d'évaluation, nous avons utilisé comme outil d'évaluation, l'erreur quadratique moyenne, ce qui s'appelle le RMSE, la variable MEDV est prédite à partir de toutes les autres variables de la data frame.

Nous avons calculé l'erreur sur les données d'entrainement et données du test, nous avons aussi calculé l'erreur quadratique moyenne sur chacune des données, le train data et test data, ce qui important et l'erreur sur les données du test, mais sur les données d'entrainement ne sont pas à exclure, car elles ont servi à construire le modèle, l'erreur des données test dépend des erreurs des données d'entrainement.

**Script R :**

```
# Evaluation du modèle 'model'

model_evaluation(model)
```

**Console :**

```
> model_evaluation(model)
[1] "RMSE sur le train data : 4.67603405287116"
[1] "RMSE sur le test data : 5.02971020858122"
```

Une fois le modèle évalué, nous pouvons voir qu'il n'y a pas une grande différence entre l'erreur du train et du test (pas d'erreur ajustement). Ce qui est aussi appelé overfiting.

Cependant, la métrique d'évaluation calculée par simple division aléatoire arbitraire n'est pas forcément représentative de la capacité du modèle à généraliser sur de nouvelles données.

Une meilleure approche est l'utilisation de la technique de la validation croisée (cross_validation). Celle-ci consiste à diviser nos données en plusieurs plis, en 5 plis, chaque pli sera considéré comme test. Ensuite, le premier pli est pris comme ensemble de tests et l'algorithme est entraîné avec les 4 plis restants. On fait les prédictions sur l'ensemble des tests puis on calcule la métrique d'évaluation du modèle. Le deuxième pli est maintenant utilisé comme ensemble de tests et les 4 autres plis comme données d'entraînements. On fait de même jusqu'à ce que chacun des 5 plis ait été utilisé comme ensemble de tests. Ainsi, on aura 5 métriques d'évaluations et on peut faire la moyenne de ces métriques d'évaluations. Donc la moyenne de l'erreur quadratique moyenne, celle-ci nous donnera véritablement une estimation de la performance du modèle.

Appliquons donc cette méthode :

**Script R :**

```
#modèle de régression avec validation croisée
model_cv <- train(MEDV ~ .,
                  data = train,
                  method = "lm",
                  trControl = trainControl(method = "cv",
                                           number = 5
                  )
)
# Affichage du model

model_cv
```

Voici notre modèle créé par validation croisé, nous allons maintenant l'évaluer :

**Script R :**

```
# Evaluation du modèle 'model_cv'

model_evaluation(model_cv)
```

**Console :**

```
> model_evaluation(model_cv)
[1] "RMSE sur le train data : 4.67603405287116"
[1] "RMSE sur le test data : 5.02971020858122"
```

Si l'on compare notre nouveau modèle à l'ancien, pas de grande différence entre les deux.

Nous allons maintenant faire une validation croisée répétée, qui est une validation croisée, répétée un certain nombre de fois.

**Script R :**

```
# Modèle de régréssion avec répétition du processus de validation croisée

model_cv2 <- train(MEDV ~ .,
                   data = train,
                   method = "lm",
                   trControl = trainControl(method = "repeatedcv",
                                            number = 5,
                                            repeats = 5
                   )
)

# Affichage du modèle

model_cv2
```

**Console:**

```
> model_cv2
Linear Regression

405 samples
 12 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 325, 325, 324, 323, 323, 323, ...
Resampling results:

  RMSE      Rsquared   MAE
  4.867522  0.7276673  3.416793

Tuning parameter 'intercept' was held constant at a value of TRUE
>
```

**Script R :**

```
# Evaluation du modèle 'model_repeatcv'

model_evaluation(model_cv2)
```

**Console :**

```
> model_evaluation(model_cv2)
[1] "RMSE sur le train data : 4.67603405287116"
[1] "RMSE sur le test data : 5.02971020858122"
>
```

Les données ne sont pas à la même échelle et cela pourrait causer une mauvaise performance du modèle, il faudrait les standardiser.

**Scripte R :**

```
# Standardisation des données avant modélisation

model_stand <- train(MEDV ~ .,
                     data = train,
                     method = "lm",
                     PreProcess = c("center", "scaler"),
                     trControl = trainControl(method = "cv",
                                              number = 5
                     )
)

# Affichage du modèle

model_stand
```

Nous avons ajouté la commande Prepocess, centre et réduit nos données pour les standardisés.

**Console :**

```
> model_stand
Linear Regression

405 samples
 12 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 324, 324, 324, 324, 324
Resampling results:

  RMSE      Rsquared   MAE
  4.841974  0.7341165  3.421173
```

**Script R :**

```
# Evaluation du modèle 'model_stand'

model_evaluation(model_stand)
```

**Console :**

```
> model_evaluation(model_stand)
[1] "RMSE sur le train data : 4.67603405287116"
[1] "RMSE sur le test data : 5.02971020858122"
```

Nous pouvons aussi réaliser une analyse par composante principale et entrainer notre algorithme dessus avant de modéliser et standardiser nos données.

**Script R :**

```
# Standardisation et ACP des données avant modélisation

model_pca <- train(MEDV ~ .,
                   data = train,
                   method = "lm",
                   PreProcess = c("center", "scaler", "pca"),
                   trControl = trainControl(method = "cv",
                                            number = 5
                   )
)

# Affichage du modèle

model_pca
# Evaluation du modèle 'model_pca'

model_evaluation(model_pca)
```

**Console:**

```
> model_pca
Linear Regression

405 samples
 12 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 324, 324, 325, 324, 323
Resampling results:

  RMSE      Rsquared   MAE
  4.909677  0.7154149  3.465592

Tuning parameter 'intercept' was held constant at a value of TRUE
> # Evaluation du modèle 'model_pca'
>
> model_evaluation(model_pca)
[1] "RMSE sur le train data : 4.67603405287116"
[1] "RMSE sur le test data : 5.02971020858122"
>
```

# Conclusion générale :

Le machine Learning a connu une évolution exponentielle et peut se vanter d'avoir atteint un niveau quasi humain pour des taches spécifiques comme la régression ou la classification d'image, mais on ne peut dire qu'il est automatisé au sens propre du terme, car cela signifierait que l'ordinateur est capable de se programmer lui-même, sans que l'homme ne le fasse, or ce n'est pas le cas, son apprentissage a besoin d'être guidé par l'homme, un peu comme un enfant qui à besoins d'être guidé par son père dans ses premiers pas.

Le Machine Learning et le Deep Learning se distinguent par leurs propres avantages et leurs domaines d'application. Les algorithmes de Machine Learning sont capables d'analyser et d'apprendre et prendre une décision finale à partir des données fournies, avec tout de même une assistance humaine, alors qu'un algorithme de Deep Learnning n'a pas besoin de beaucoup d'aide en raison de l'émulation de base du flux de travail du cerveau humain et compréhension du contexte.

Le choix entre une méthode d'apprentissage profond plutôt qu'une méthode d'apprentissage automatique dépend de la quantité et des types de données dont on dispose, un algorithme d'apprentissage profond sera plus efficace quand la quantité de données dont on dispose est de l'ordre du million, ces quantités de données concernent particulièrement GAFA (Google, Apple, Facebook et Amazon). L'option Deep Learning, dépend davantage du type de données : le Deep Learning sera particulièrement intéressant sur des données non structurées telles que les données-images, ou sons.

Nous avons vu que l'apprentissage machine peut réaliser de grandes choses, effectuer les activités qu'un cerveau humain peut faire, de façon plus efficace et plus rapide, Aujourd'hui, nous avons vu que les machines peuvent battre des champions humains dans des jeux tels que les échecs, AlphaGO, qui sont considérés comme très complexes.

Comme nous, le machine Learning a de nombreux avantages, mais connaît des limites, il doit sans cesse être adapté par son programmeur avant tout se procurer l'échantillon le plus représentatif s'il veut obtenir des résultats pertinents et un modèle vraisemblable. Le plus représentatif possible. Il doit également choisir la fonction la plus adaptée à son modèle.

La croissance et la maturation de l'écosystème d'outils et de processus de l'intelligence artificielle nous laissent cependant voir que l'apprentissage machine est loin d'avoir atteint son apogée, celui-ci est même l'un des domaines les plus mouvementés ces dernières années.

## *Bibliographie :*


- ✓ Stuart Russell et Peter Norvig. (2010). Intelligence artificielle. 3eme edition. PEARSON. (pp 735-902).
- ✓ Massih-Reza AMINI. (2020). Machine Learning. 2eme edition. EDITIONS EYROLLES. (pp 7-27 et 81-111).
- ✓ Chloé-Agathe Azencott. (2019). Introduction au Machine Learning. DUNOD. (pp1-32 et 92-106).
- ✓ Chloé-Agathe Azencott. Introduction au Deep Learning, Chloé-Agathe Azencott. DUNOD. (pp 3-25).
- ✓ John Paul Mueller et Luca Massaron. (2019). Intelligence artificielle pour les nuls. FIRST. (pp 155-175).
- ✓ Apprendre le Machine Learning en une semaine. Guillaume Saint-Cirgue. (pp 8-18, 47-58, 61-69).
- ✓ Romain Rissoan, Romain Jouin. (2018). La boîte à outils de la stratégie big data. 10-35. https://www.cairn.info/la-boite-a-outils-de-la-strategie-big-data--9782100778980.htm
- ✓ Aurélien Garivier. (2013). Apprentissage supervisé. Université de Toulouse. https://www.math.univ-toulouse.fr/~agarivie/mydocs/apprentissageSupervise.pdf
- ✓ Ricco Rakotomalala. Deep Learning. Perceptrons simples et multicouches. Université Lumière Lyon 2. https://eric.univ-lyon2.fr/~ricco/cours/slides/reseaux_neurones_perceptron.pdf
- ✓ Définition du rendement. BFM Bourse. Consulté le 10 juin 2021 à 17h32. https://www.tradingsat.com/lexique-boursier/definition-rendement-192.html
- ✓ Le modèle CAPM, célèbre technique d'évaluation. Abc bourse. Consulté le 10 juin 2021 à 18h07. https://www.abcbourse.com/apprendre/19_capm.html
- ✓ Vincent Bouchet. (2017). Machine Learning en finance : vers de nouvelles stratégies ?,archives-ouvertes. 21-27. https://dumas.ccsd.cnrs.fr/dumas-01706572
- ✓ J. Rynkiewicz. (2020). Introduction au Deep Learning Présentation et histoire du Deep Learning. Université Paris 1. http://samm.univ-paris1.fr/IMG/pdf/coursdeep1-2.pdf
- ✓ Gilbert Saporta. (2018). Une brève histoire de l'apprentissage. Archives-ouvertes. 2-17. https://hal-cnam.archives-ouvertes.fr/hal-02470857/document
- ✓ Claude Touzet. (1992). LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME. Archives-ouvertes. 6-21. https://hal-amu.archives-ouvertes.fr/hal-01338010/file/Les_reseaux_de_neurones_artificiels.pdf
- ✓ Philippe Thomas, André Thomas. (2008). Elagage d'un perceptron multicouche : utilisation de l'analyse de la variance de la sensibilité des paramètres. Archives-ouvertes. 1-8. https://hal.archives-ouvertes.fr/hal-00320832/document
- ✓ Laura GAY. (2015). Algorithme de descente du gradient stochastique.Université Claude Bernard Lyon 1. 5-10. https://www-ljk.imag.fr/membres/Marianne.Clausel/Fichiers/ProjectGay.pdf
- ✓ Bruno Bouzy. (2005). Descente de gradient. http://helios.mi.parisdescartes.fr/~bouzy/Doc/AA1/DescenteGradient.pdf

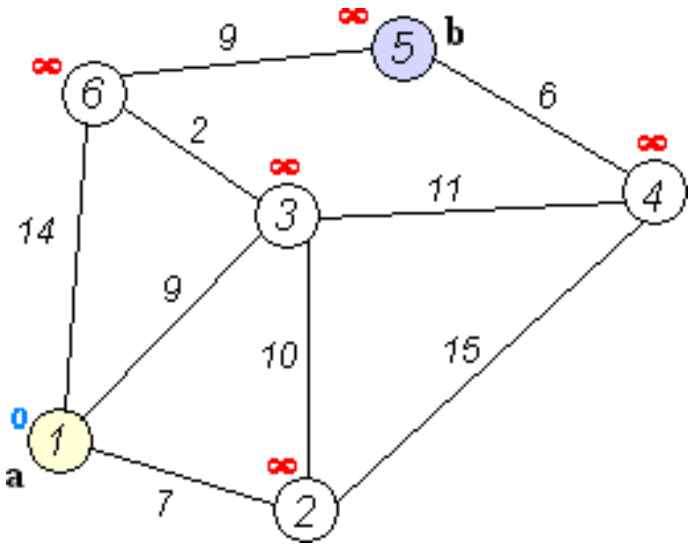
✓ BRECHET Thomas et TRIBINO Julien. 4 avril 2008. Le Pendule inverseEt son application en robotique. https://docplayer.fr/21890200-Le-pendule-inverse-et-son-application-en-robotique-brechet-thomas-et-tribino-julien.html

# *Annexes :*

## **Annexe 1 :** Algorithme Djikstra.

L'algorithme de Dijkstra peut être considéré comme l'un des algorithmes de chemin le plus court les plus célèbres. Par exemple, il permet de déterminer un trajet plus court d'une ville à une autre tout en connaissant le réseau routier d'une région , Ce dernier se base sur le chemin de largeur afin de trouver le chemin le plus court dans le graphe pondéré positivement (c'est très important, mais nous y reviendrons plus en détail plus tard).

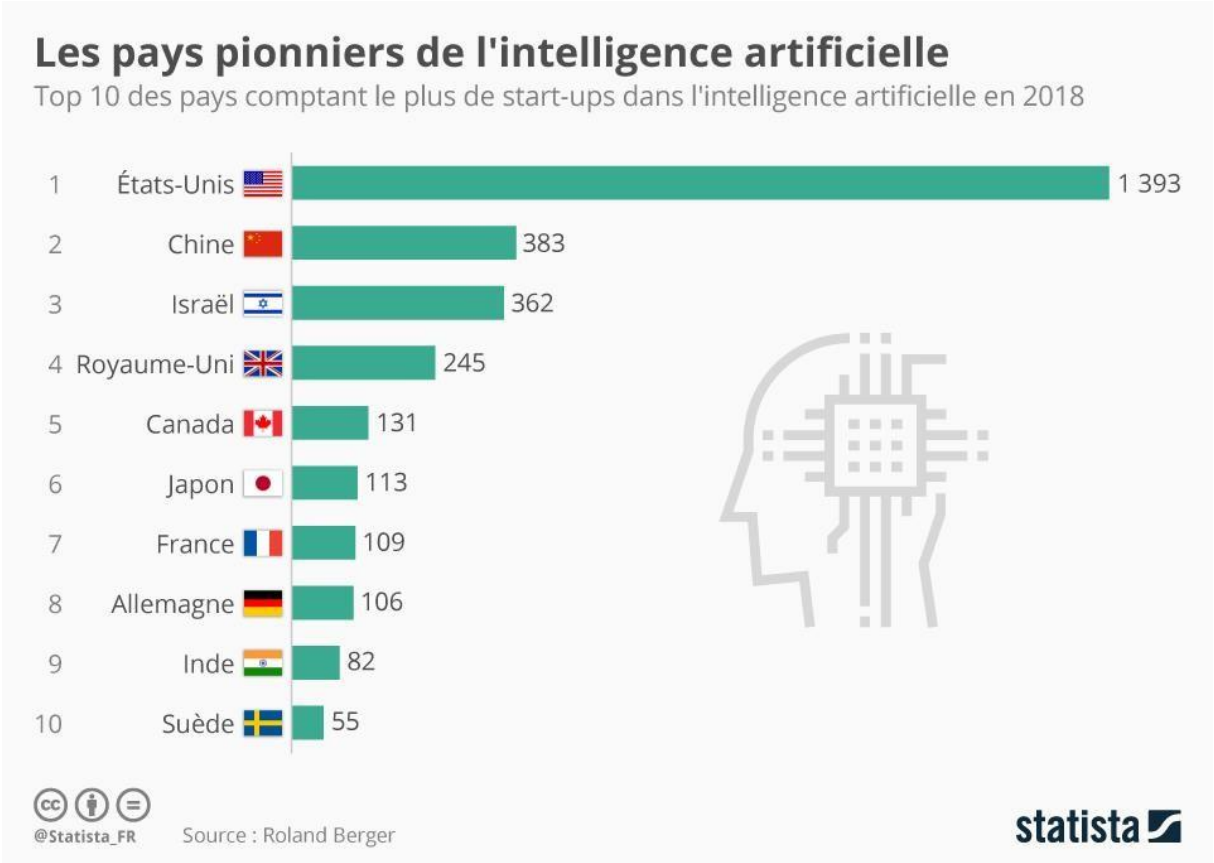
Nous savons que le parcours en largeur peut trouver le chemin le plus court sur un graphe non pondéré (nous associons chaque arc à 1 unité de n'importe quelle longueur), et il utilise des files d'attente pour stocker les éléments afin de les parcourir un par un en profondeur. Même idée, mais il n'utilisera pas une file d'attente simple (car tous les arcs n'ont pas le même poids), mais une file d'attente prioritaire. Ce dernier permet d'avoir toujours un nœud avec la plus petite distance du point de départ en tête de file, et ensuite de s'assurer que l'on trouve le chemin le plus court lorsque l'on atteint le nœud de sortie.




Source : https://commons.wikimedia.org/wiki/File:Dijkstra_Animation.gif.

## **Annexe 1 :** Avis des francais sur le développement de la robotique dans plusieurs domaines.

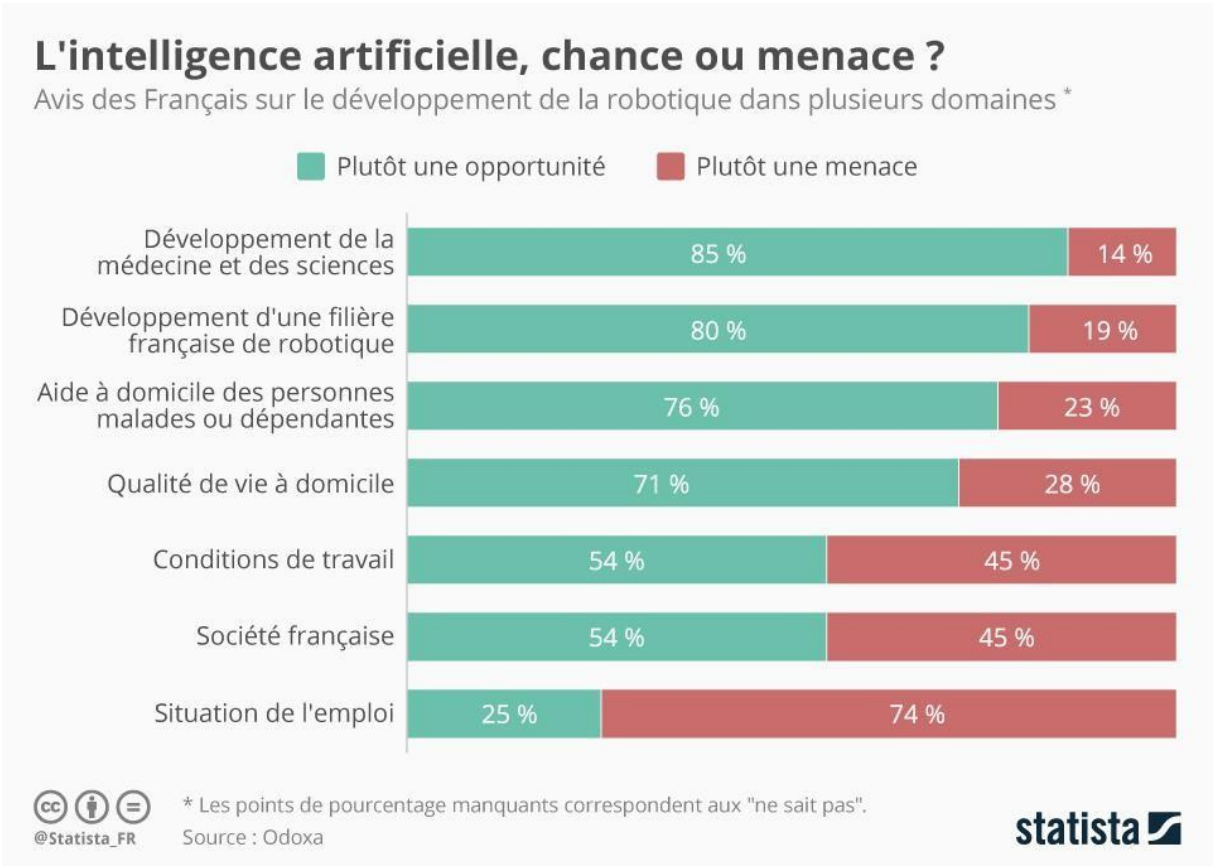
Les Français aiment l'intelligence artificielle autant qu'ils la craignent, au regard des résultats d'un récent sondage Odoxa. Ainsi, une grande majorité reconnait en le développement de la robotique une opportunité pour le monde scientifique (85 %) et la filière française (80 %).
La robotique, grâce à l'intelligence artificielle, améliorerait également le quotidien de personnes dépendantes ou malades selon 76 % des personnes interrogées, et même la qualité de vie à domicile en générale pour 71 % d'entre eux.
Mais plus épineux sont les domaines du travail et de l'emploi. À peine un peu plus de la moitié des répondants considèrent que ces avancées amélioreraient les conditions de travail. Ils sont encore plus méfiants en ce qui concerne l'emploi : près des trois quarts voient dans la robotique une menace pour la situation de l'emploi.




Source : https://fr.statista.com/infographie/9813/lintelligence-artificielle-chance-ou-menace/

## **Annexe 2 :** Les pays pionniers de l'intelligence artificielle

Le Président de la République, qui a l'ambition de faire de la France une « start-up nation », peut se réjouir du nombre de jeunes pousses évoluant dans le domaine de l'intelligence artificielle, marché d'avenir : selon un récent rapport Roland Berger, l'Hexagone en compte 109, se plaçant ainsi au 7ème rang des pays comptant le plus de start-ups dans l'IA.
La France fait donc mieux que l'Allemagne, l'Inde et la Suède, mais reste toutefois très loin des États-Unis, pays par excellence de l'intelligence artificielle. Près de 1 400 start-ups dans l'IA y ont en effet été recensées. Même la Chine, deuxième de ce classement, ne peut rivaliser avec ses 383 start-ups.




Source : https://fr.statista.com/infographie/14370/les-pays-pionniers-de-lintelligence-artificielle/