Université Nouveaux Horizons

Invariance des transformations géométriques et qualités des objets dans la detection des plaques d'immatriculation

Auteur :
TSHELEKA KAJILA Hassan

Directeur :
Prof. MASAKUNA Jordan

Mémoire présenté à la Faculté des Sciences Informatiques en vue de l'obtention du grade de Licencié en informatique.

en

Calcul Scientifique

7 juin 2022

RÉSUMÉ

Au cours de la dernière décennie, la taille des données a augmenté plus rapidement que la vitesse des processeurs. Dans ce contexte, faire un traitement de reconnaissance des formes dans des images et vidéos, les ensembles de données d'entraînement pour les problèmes de détection d'objets sont généralement très volumineux et les capacités des méthodes d'apprentissage automatique statistique sont limitées par le temps de calcul plutôt que par la taille de l'échantillon.

Le cas des problèmes d'apprentissage à grande échelle implique la complexité de calcul de l'algorithme d'optimisation sous-jacent de manière non triviale. Des algorithmes d'optimisation improbables tels que la **descente de gradient stochastique** (en anglais : **Stochastic Gradient Descent** ou SGD) montre des performances étonnantes pour les problèmes à grande échelle, lorsque l'ensemble d'apprentissage est volumineux.
En particulier, les variants du SGD n'utilisent qu'un seul nouvel échantillon d'apprentissage à chaque itération, sont asymptotiquement efficaces après un seul passage sur l'ensemble d'apprentissage.

Ce travail vise à proposer une méthode intelligente, basée sur l'intelligence artificielle, qui permet aux ordinateurs et aux systèmes informatiques de dériver des informations significatives à partir d'images numériques, de vidéos et d'autres entrées visuelles, avec un coût plus bas que possible. Dans notre contexte la reconnaissance des plaques d'immatriculation des véhicules à l'aide d'un classificateur de la famille de descente de gradient stochastique. Pour minimiser la **fonction coût** du classificateur, la SGD adopte un modèle d'optimisation convexe. De plus, pour augmenter la vitesse de convergence du classificateur, la descente de gradient stochastique, à chaque étape, elle tire un échantillon aléatoire de l'ensemble des fonctions (f_i), de la fonction objectif, constituant la somme.

Mots clés : Apprentissage supervisé, vision par ordinateur, Descente de gradient stochastique, Adaline, ALPR.

ABSTRACT

Over the past decade, data size has grown faster than processor speeds. In this context, doing pattern recognition processing in real-time videos, training datasets for object detection problems are usually very large, and the capabilities of statistical machine learning methods are limited by computation time rather than sample size.

The case of large scale learning problems involves the computational complexity of the underlying optimization algorithm in a nontrivial way.
Improbable optimization algorithms such as **Stochastic Gradient Descent** (SGD) show amazing performance for large scale problems, when the training set is bulky.
In particular, SGD variants use only one new training sample at each iteration, are asymptotically efficient after a single pass over the training set.

This work aims to provide an intelligent method, based on artificial intelligence, that allows computers and computer systems to derive meaningful information from digital images, videos and other visual inputs, with a lower cost. as possible. In our context the recognition of vehicle license plates using a classifier of the family of stochastic gradient descent. To minimize the **cost function** of the classifier, the SGD adopts a convex optimization model. Moreover, to increase the speed of convergence of the classifier, the stochastic gradient descent, at each step, it draws a random sample from the set of functions (f_i), of the objective function, constituting the sum.

Key words : Supervised learning, computer vision, Stochastic gradient descent, Adaline, ALPR.

TABLE DES MATIÈRES

LISTE DES ACRONYMES

ML Machine Learning

CV Computer Vision

OCR Optical character recognition

ANPR Automatic number-plate recognition

ALPR Automatic license plate recognition

GD Gradient Descent

SGD Stochastic Gradient Descent

ADALINE ADAptative LInear NEuron

API Application Programming Interface

UML Unified Modeling Language

$\mathbb{N}$	Ensemble des entiers naturels
$\mathbb{R}^n$	Ensemble des réels ou Espace euclidien de dimension n
$\mathbb{B}^n = \{0,1\}^n$	Espace booléen de dimension n
$\mathcal{O}(\cdot)$ ou $\Omega(\cdot)$	L'ordre de grandeur maximal de complexité d'un algorithme
$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$	Un vecteur
$x = (x_1, \vdots, x_n)^T$	Un vecteur
$x^T = (x_1, \vdots, x_n)^T$	Un vecteur transposé
$\langle xy \rangle = x^T y$	Le produit vectoriel
$\parallel x \parallel$	La norme du vecteur
M^{-1}	La matrice inverse d'une matrice M
M^T	La matrice transposée
$\frac{\partial}{\partial x} f(x,y)$	La dérivée partielle par rapport à x de al fonction f des deux variable x et y
$\nabla_A J(A,B)$	Le vecteur dérivé par rapport au vecteur A de la fonctionnelle J des deux vec

Les éléments en apprentissage

$\mathcal{S}$	L'échantillon d'apprentissage (un ensemble ou une suite d'exemple)
$\hat{y}$	valeurs prédite après l'entrainement d'un modèle d'apprentissage automatiqu
$\mathcal{H}$	
$h \in \mathcal{H}$	
$y = h(x)$	
$\ell(f(x), h(x))$	

0 INTRODUCTION

0.1 CHOIX ET INTÉRÊT DU SUJET

L'intelligence désigne communément le potentiel des capacités mentales et cognitives d'un individu, animal ou humain, lui permettant de résoudre un problème ou de s'adapter à son environnement. L'intelligence nous fait ressentir ce besoin d'apprendre pour arriver à nos fins, extresinquement l'intelligence c'est l'apprentissage. Pour que nous puissions dire qu'une machine est intelligente, premièrement elle doit passer par une phase d'apprentissage. Apprendre à résoudre des problèmes ou à réaliser des tâches par lui-même d'une façon autonome. Dans le IA nous parlons de l'apprentissage automatique (en anglais : machine Learning, ML), nous utilisons plusieurs paradigmes d'apprentissage automatique : apprentissage supervisé, apprentissage non supervisé, apprentissage par renforcement, apprentissage en profondeur.

L'apprentissage supervisé représente une grande partie de l'activité de recherche en apprentissage automatique et de nombreuses techniques de ce paradigme ont trouvé une application dans le traitement de contenu multimédia [9]. La caractéristique qui définit ce type d'apprentissage est la disponibilité de données d'apprentissage annotées.
Les algorithmes d'apprentissage supervisé font l'expérience d'un ensemble de données contenant des caractéristiques, et chaque exemple est également associé à une étiquette ou à une cible [14].

L'application de cette étude dans l'apprentissage supervisé est orientée vers la reconnaissance automatique des plaques d'immatriculation (en anglais : automatic number plate recognition, ANPR) dans les images. Une des applications intéressantes parmi tant d'autres dans l'intelligence artificielle. Nous présentons une étude approfondie sur les algorithmes de minimisation de la fonction coût (en anglais : loss function) d'un modèle d'apprentissage appliqué à l'ANPR.

Lorsque nous voulons faire une application dans le traitement de reconnaissance des formes dans des vidéos, les ensembles de données d'entraînement pour les problèmes de détection d'objets sont généralement très volumineux et les capacités des méthodes d'apprentissage automatique statistique sont limitées par le temps de calcul plutôt que par la taille de l'échantillon [5]. Par exemple, pour entraîner une machine à reconnaître des plaques d'immatriculation de voiture, elle doit recevoir de grandes quantités d'images de plaques d'immatriculation et d'éléments liés aux plaques pour apprendre les différences et reconnaître une plaque, en particulier la voiture qui porte une plaque sans défaut. Plus nous avons des données, plus nous gagnons en précision et plus la complexité en

temps augmente.
Des contraintes d'exploitation découlent des observations citées ci-dessus, parmi lesquelles nous citerons celles qui sont liées à la reconnaissance des objets dans les vidéos et images. Par exemple, de nos jours, un très grand nombre de caméras est déployé exclusivement pour la surveillance vidéo [1] . Souvent, le contenu de ces vidéos est interprété par des opérateurs humains qui engendrent des coûts exorbitants pour le suivi et l'analyse du contenu, sans mentionner les erreurs qui peuvent être induites par la fatigue et l'inattention humaine.

0.2 PROBLÉMATIQUE

La complexité de calcul de l'algorithme d'apprentissage devient le facteur limitant critique lorsque l'on envisage de très grands ensembles de données. C'est à ce point critique qu'entre en jeu cette étude, la minimisation des erreurs sans alourdir la complexité en temps et espace de l'algorithme d'apprentissage.

Les ensembles de données d'entraînement pour les problèmes de détection d'objets dans des images sont généralement très volumineux. Minimiser les erreurs dans ces modèles d'apprentissage est une tâche très importante pour renforcer la fiabilité de notre *modèle entraîné* [15].
Établir un algorithme d'apprentissage qui s'adapte au mieux à notre modèle, selon la nature du problème métier traité, il existe différentes approches qui varient selon le type et le volume des données.

L'un des piliers de l'apprentissage automatique est l'optimisation mathématique [7], qui, dans ce contexte, implique le calcul numérique de minimisation des paramètres d'un système conçu pour prendre des décisions en fonction des données disponibles. Ces paramètres sont choisis pour être optimaux par rapport à notre problème d'apprentissage.

Dans l'ensemble, ce document tente d'apporter des réponses aux questions suivantes.

1. Comment les problèmes de minimisation surviennent-ils dans les applications d'apprentissage automatique ?
2. Quelles ont été les méthodes de minimisation les plus efficaces pour les ensembles des données d'apprentissage supervisé à grande échelle et pourquoi ?
3. Comment des algorithmes d'apprentissage supervisé arrivent-t-ils résoudre le problème de la reconnaissance automatique d'objet ?
4. Quelles avancées récentes ont été réalisées dans la conception d'algorithmes de minimisation des erreurs dans l'apprentissage et quelles sont les questions ouvertes dans ce domaine de recherche ?

0.3 HYPOTHÈSE

Le cas des problèmes d'apprentissage à grande ou à petite échelle implique la complexité de calcul de l'algorithme d'optimisation sous-jacent de manière non triviale.
En effet, dans ce travail, nous discutons des algorithmes de descente de gradient stochastique parce qu'ils montrent des performances d'optimisation incroyables pour les problèmes à grande échelle [5].

Le travail de Léon Bottou et al (e. g., dans [5] [26] [6]), présente *la descente de gradient stochastique comme un algorithme d'apprentissage fondamental.*
Une analyse plus précise révèle des compromis qualitativement différents pour le cas des problèmes d'apprentissage à grande échelle [7]. Des algorithmes d'optimisation improbables tels que la **descente de gradient stochastique** (en anglais : **Stochastic Gradient Descent** ou SGD) montre des performances étonnantes pour les problèmes à grande échelle, lorsque l'ensemble d'apprentissage est volumineux. En particulier, le gradient stochastique du second ordre et le gradient stochastique moyennée sont asymptotiquement efficaces après un seul passage sur l' ensemble d'entraînement [5]. Les optimiseurs SGD n'utilisent qu'un seul nouvel échantillon d'apprentissage à chaque itération.

0.4 MÉTHODE ET TECHNIQUE UTILISÉE

0.5 OBJECTIFS ET DIVISION DU TRAVAIL

Nous nous proposons dans ce mémoire d'aborder sur l'utilisation des algorithmes d'optimisation numérique, précisément de minimisation. Appliquée à l'apprentissage automatique qui permet aux ordinateurs et aux systèmes informatiques de dériver des informations significatives à partir d'images numériques et/ou d'autres entrées visuelles, avec un coût plus bas que possible.

En fait, nous faisons la reconnaissance des plaques d'immatriculation des véhicules à l'aide d'un classificateur de la famille de descente de gradient stochastique (SGD). Pour minimiser la fonction de coût du classificateur, la SGD adopte un modèle d'optimisation convexe [11]. De plus, pour augmenter la vitesse de convergence du classificateur, la descente de gradient stochastique, à chaque étape, tire un échantillon aléatoire de l'ensemble des fonctions (f_i), qui est notre fonction objectif, constituant une somme.
Pour chaque algorithme, nous examinons l'efficacité et comparons le score pour différents cas.

En dehors de cette introduction, la partie conclusive et l'annexe, ce mémoire est organisé en quatre chapitres comme suit.

> ***Chapitre 1*** est consacré à quelques rappels des matières sur lesquels je me base pour constituer l'ensemble de ce travail. Nous traitons des considéra-

tions de méthodes numériques impliquées dans la résolution de problèmes de minimisation des erreurs d'apprentissage. Certaines discussions sur les modèles de régression linéaire convexe et de classification dans d'apprentissage supervisé. Nous discutons également du réseau neuronal convolutif le plus adapté pour analyser l'imagerie visuelle.

Chapitre 2 explore une méthodologie parmi tant d'autres, pour entraîner les modèles d'apprentissage automatique de façon optimale, qui nous permettra par la suite de faire une classification d'images pour reconnaissance automatique de plaque d'immatriculation.
Pour la minimisation de la fonction coût nous utilisons des algorithmes comme ASGD, ADAM, ADADELTA, NAG. Puis faire une étude comparative de leurs performances.

Chapitre 3, Ici nous construirons des modèles à partir d'une base de données annotée pour l'apprentissage et pour les tests de reconnaissance d'objets. Les résultats concluants de cette étude pourront conduire à un déploiement de notre système dans les domaines comme celui de la surveillance vidéo de voitures dans une entrée de parking. Des métriques connues pour mesurer les erreurs et en déduire le score du classificateur seront utilisées pour évaluer la qualité de la reconnaissance automatique des plaques d'immatriculation (ANPR) par notre approche.

1

CONCEPTS ET ÉLÉMENTS MATHÉMATIQUES DE L'APPRENTISSAGE PROFOND

1.1 LES BASES D'OPTIMISATION NUMÉRIQUE ET STATISTIQUE

1.1.1 *Éléments de calcul différentiel*

A. *Convexité*

DÉFINITION : (ENSEMBLE CONVEXE) Une partie $\mathcal{C} \subset \mathbb{R}^n$ est dite convexe si et seulement si pour tout $(x, y) \in \mathcal{C}^2$, et pour tout $\alpha \in [0, 1]$, $\alpha x + (1 - \alpha)y \in \mathcal{C}$ combinaison convexe [19].

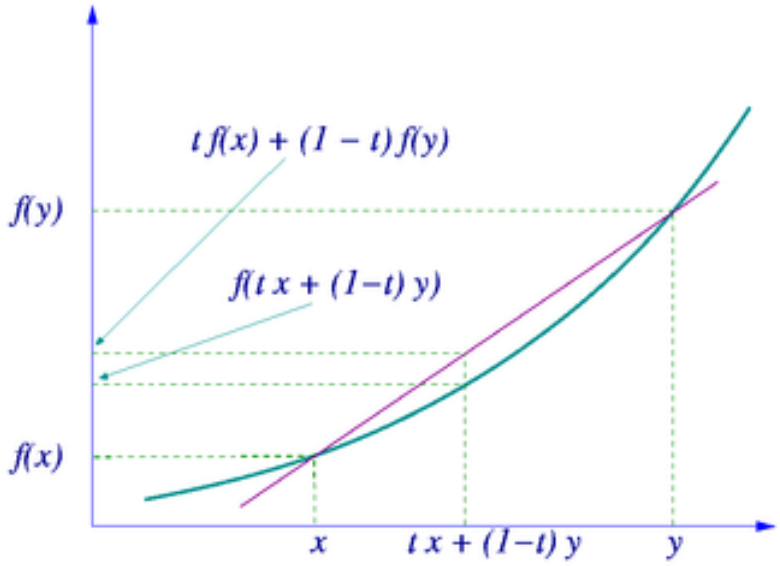


FIGURE 1 : Illustration fonction convexe [image de Wikipédia]

DÉFINITION : (FONCTION CONVEXE) Une fonction f d'un intervalle réel $I \in \mathcal{C}$ est dite fonction convexe lorsque, $\forall (x, y)$ de I tel que $(x, y) \in \mathcal{C}^2$ et tout $\alpha \in [0, 1]$ on a :

$$f(\alpha x + (1 - \alpha)y) \leqslant \alpha f(x) + (1 - \alpha)f(y) \tag{1}$$

et si

$$f(\alpha x + (1 - \alpha)y) < \alpha f(x) + (1 - \alpha)f(y) \tag{2}$$

on dit que la fonction est strictement convexe dans $\mathcal{C}$, [19]

Exemple :

PROPRIÉTÉ D'UNE FONCTION DÉRIVABLE : (Extremum local) Parmi les propriétés de dérivabilité il existe une qui est mise en relation avec l'effect qu'une fonction doit être convexe. énoncé ci-dessous [8, p. 212].
Soit $I \to \mathbb{R}$ une fonction et a un point de I.

+ On dit que m est un **minimum local** de f s'il existe $a > 0$ tel que m soit le minimum de f restreinte à $I \cap]a - \alpha, a + \alpha[$.
+ On dit que M est un **maximum local** de f s'il existe $a > 0$ tel que M soit le maximum de f restreinte à $I \cap]a - \alpha, a + \alpha[$.

Donc nous pouvons dire qu'une fonction convexe à un unique point minimum.

B. *Développement limité*

En physique et en mathématiques, un développement limité (noté DL) d'une fonction en un point est une approximation polynomiale de cette fonction au voisinage de ce point, c'est-à-dire l'écriture de cette fonction sous la forme de la somme d'une fonction polynomiale et d'un reste négligeable au voisinage du point considéré [8].

Soit f une fonction à valeurs réelles définie sur un intervalle I, et $x_0 \in I$. On dit que f admet un développement limité d'ordre n^2 (abrégé par DL_n) en x_0, s'il existe $n+1$ réels $a_0, a_1, \ldots, a_n$ tels que la fonction $R : I \to \mathbb{R}$ définie par :

$$f(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \cdots + a_n(x - x_0)^n + R(x) = \sum_{i=0}^{n} a_i(x - x_0)^i + R(x)$$

vérifie : $R(x)$ tend vers 0 lorsque x tend vers x_0, et ce plus rapidement que le dernier terme de la somme, c'est-à-dire que :

$$\lim_{x \to x_0} \frac{R(x)}{(x - x_0)^n} = 0.$$

La fonction reste $R(x)$ vérifiant ceci est notée $o((x–x0)^n)$ (selon la notation de Landau). On écrit donc :

$$f(x) = \sum_{i=0}^{n} a_i(x - x_0)^i + R(x) = \sum_{i=0}^{n} a_i(x - x_0)^i + o((x - x_0)^n)$$

Il est fréquent d'écrire un développement limité en posant $x = x0 + h$ on aura :

$$f(x_0 + h) = \sum_{i=0}^{n} a_i h^i + o(h^n)$$

CONSÉQUENCES IMMÉDIATES

- Si f admet un DL_0 en x_0, alors $a_0 = f(x_0)$. [8]
- Si f admet un DL_n en x_0, alors elle admet un DL_k en x_0 pour tout entier $k < n$ [8].
- Une condition nécessaire et suffisante pour que f admette un DL_n en x_0 est l'existence d'un polynôme P tel que $f(x) = P(x) + o((x–x_0)^n)$ [8]. S'il existe un tel polynôme P, alors il en existe une infinité d'autres, mais un seul d'entre eux est de degré inférieur ou égal à n : le reste de la division euclidienne de $P(X)$ par $(X–x_0)^{n+1}$. On l'appelle la partie régulière, ou partie principale, du DL_n de f en x_0.

Le théorème de Taylor-Young assure [8] qu'une fonction f dérivable n fois au point x_0 (avec $n \geqslant 1 n \geqslant 1$) admet un DLn en ce point :

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + o((x - x_0)^n)$$

soit en écriture abrégée

$$f(x) = \sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i + o((x - x_0)^n)$$

Le développement d'ordre 0 en x_0 revient à écrire que f est continue en x_0 :

$$f(x) = f(x_0) + o((x - x_0)^0) = f(x_0) + o(1)$$

Le développement limité d'ordre 1 en x_0 revient à approcher une courbe par sa tangente en x_0 on parle aussi d'approximation affine :

$$f(x) = f(x_0) + f'(x_0) \cdot (x - x_0) + o(x - x_0)$$

c. *Gradient*

DÉFINITION : Le gradient d'une fonction de plusieurs variables en un certain point est un vecteur qui caractérise la variabilité de cette fonction au voisinage de ce point. Défini en tout point où la fonction est différentiable, il définit un champ de vecteurs, également dénommé gradient. Le gradient est la généralisation à plusieurs variables de la dérivée d'une fonction d'une seule variable.

DÉFINITION MATHÉMATIQUE : Dans un système de coordonnées cartésiennes, le gradient d'une fonction $f(x_1, x_2, \ldots, x_n)$ est le vecteur de composantes

$\partial f/\partial x_i$ $(i = 1, 2, \ldots, n)$, c'est-à-dire les dérivées partielles de f par rapport aux coordonnées [19].

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \in \mathbb{R}^n$$

GRADIENT SOUS FORME DE DÉVELOPPEMENT LIMITÉ : *Si une application admet un gradient en un point, alors on peut écrire ce développement limité du premier ordre (voir le point b.).*

$$f(x + h) = f(x) + \langle \nabla f(x) \mid h \rangle + o(h)$$

ou

$$f(x - h) = f(x) - \langle \nabla f(x) \mid h \rangle + o(h)$$

Numériquement, il est très intéressant de faire ensuite la demi-différence des deux développements pour obtenir la valeur du gradient et on note que celui-ci ne dépend pas en fait de la valeur de la fonction au point $x : f(x)$. *Cette formule a l'avantage de tenir compte des gradients du 2e ordre et est donc beaucoup plus précise et numériquement robuste. L'hypothèse est, en pratique, de connaitre les valeurs "passé" et "futur" de la fonction autour d'un petit voisinage du point* x[19].

DÉFINITION NUMÉRIQUE : Une fonction multivariée (à variable vectorielle) $f(x) : \mathbb{R}^n \to \mathbb{R} : x \to f(x)$ définie sur un ouvert $O \in \mathbb{R}^n$ est dite dérivable (au sens de Fréchet, voir le point **??**) en x ssi il existe un vecteur noté $\nabla f(x) \in \mathbb{R}^n$ tel que

$$f(x + h) = f(x) + \nabla f(x)^T h + o(||h||) \tag{3}$$

$\nabla f(x) \in \mathbb{R}^n$ et où l'on a posé que le reste $o(||h||) = ||h||\epsilon(h) \in \mathbb{R}^n$, avec $h \in \mathbb{R}^n$

$$\epsilon(h) : \mathbb{R}^n \to \mathbb{R}, \qquad \lim_{||h|| \to 0} \epsilon(h) = 0.$$

Le vecteur $\nabla f(x)$ est unique et nommé **gradient** de $f(x)$ en x. Le gradient s'adresse aux fonctions scalaires à variables vectorielles.

A PROPOS DE LA NOTATION $o(||h||)$: La notation de Bachmann-Landau $o(||h||)$ traduit le comportement d'une fonction de h qui tend vers 0 d'un ordre de grandeur plus vite que $||h||$.

Elle est infiniment plus petit que h dans le voisinage de 0

D. *Hessienne*

DÉFINITION MATHÉMATIQUE : Étant donnée une fonction f à valeurs réelles

$$f : \mathbb{R}^n \to \mathbb{R}; (x_1, ..., x_n) \mapsto f(x_1, ..., x_n)$$

dont toutes les dérivées partielles secondes existent, le coefficient d'indice i, j de la **matrice hessienne**[1] $H(f)$ vaut $H_{ij}(f) = \frac{\partial^2 f}{\partial x_i \partial x_j}$.
Autrement dit,

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial {x_1}^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial {x_2}^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial {x_n}^2} \end{bmatrix}.$$

DÉFINITION NUMÉRIQUE : Supposons que $f : \mathbb{R}^n \to \mathbb{R}$ définie sur un ouvert $\mathcal{O} \in \mathbb{R}^n$. La fonction $f(x)$ est dite 2 fois continûment dérivable (au sens de Fréchet ? ?) si en tout $x \in \mathcal{O}$ on a

$$f(x + h) = f(x) + \nabla f(x)^T h + \frac{1}{2} h^T \nabla^2 f(x) h + o(||h||^2) \tag{4}$$

avec$\nabla f(x) \in \mathbb{R}^{n \times n}$ et où on a posé que le reste $o(||h||^2) = ||h|| \epsilon(h) \in \mathbb{R}$ avec $\lim_{||h|| \to 0} \epsilon(h) = 0$ La matrice carrée symétrique $\nabla^2 f(x)$ appelée **Hessien** de $f(x)$ en x.
Remarque :

$$\lim_{||h|| \to h} \frac{o(||h||^2)}{||h||} = 0 \in \mathbb{R}$$

La Hessienne s'adresse aux fonctions scalaires à variables vectorielles.

1.1.2 *Échantillonnage (statistique)*

En statistiques, l'échantillonnage est la sélection d'un sous-ensemble (un échantillon statistique) d'individus au sein d'une population statistique pour estimer les caractéristiques de l'ensemble de la population.

Sur un échantillon, on peut calculer différents paramètres statistiques de position (moyenne, etc.) ou de dispersion (écart type, etc.) issus de la statistique descriptive, de la même manière que l'on peut déterminer des paramètres statistiques d'une population par son recensement exhaustif.

On peut également déduire des propriétés de la population à partir de celles de l'échantillon par inférence statistique. D'après la loi des grands nombres, plus la taille de l'échantillon augmente, plus ses propriétés seront proches de celle de la population. En particulier, on peut estimer une probabilité sur les individus

1 En mathématiques, la matrice hessienne (ou simplement la hessienne) d'une fonction numérique f est la matrice carrée, notée $H(f)$, de ses dérivées partielles secondes.

d'une population par la fréquence observée sur un échantillon si sa taille est suffisamment grande.

Cette méthode présente plusieurs avantages : une étude restreinte sur une partie de la population, un moindre coût, une collecte des données plus rapide que si l'étude avait été réalisé sur l'ensemble de la population, la réalisation de contrôles destructifs, etc.

On peut procéder de différentes manières pour collecter les données de l'échantillon, il existe en effet plusieurs méthodes d'échantillonnage [21] :

- **Échantillonnage aléatoire et simple** : le tirage des individus de l'échantillon est aléatoire, c'est-à-dire que chaque individu a la même probabilité d'être choisi, et simple, c'est-à-dire que les choix des différents individus sont réalisés indépendamment les uns des autres.
- **Échantillonnage systématique** : le premier individu est choisi de manière aléatoire, puis les suivants sont déterminés à intervalle régulier. Par exemple, dans un verger, on choisit au hasard le 7e pommier, puis les 27e, 47e, 67e, etc.
- **Échantillonnage stratifié** : on subdivise la population en plusieurs parties avant de prendre l'échantillon1.
- **Échantillonnage par quotas** : la composition de l'échantillon doit être représentative de celle de la population selon certains critères jugés particulièrement importants. On utilise cette méthode pour réaliser les sondages d'opinions.

A. *La collecte de données*

La collecte de données est le processus de collecte et de mesure des informations sur des variables ciblées dans un système établi, qui permet ensuite de répondre aux questions pertinentes et d'évaluer les résultats.

Une bonne collecte de données implique :

- Suivre le processus d'échantillonnage défini
- Garder les données dans l'ordre du temps
- Noter les commentaires et autres événements contextuels
- Enregistrement des non-réponses

ERREUR D'ÉCHANTILLONNAGE : Dans les statistiques, les erreurs d'échantillonnage se produisent lorsque les caractéristiques statistiques d'une population sont estimées à partir d'un sous-ensemble, ou échantillon, de cette population. Étant donné que l'échantillon n'inclut pas tous les membres de la population, les statistiques de l'échantillon (souvent appelées estimateurs), telles que les moyennes et les quartiles, diffèrent généralement des statistiques de l'ensemble

de la population (appelées paramètres). La différence entre la statistique d'échantillon et le paramètre de population est considérée comme l'erreur d'échantillonnage [21].

1.2 CONCEPTS DE LA MODÉLISATION ET CLASSIFICATION DES DONNÉES

1.2.1 *Introduction*

A. *Les ingrédients d'apprentissage*

Résoudre un problème d'apprentissage, c'est d'abord le comprendre, c'est-à-dire discuter longuement avec les experts du domaine concerné pour identifier quelles sont les "entrées", les "sorties" ou résultats désirés, les connaissances disponibles, les particularités des données, par exemple : valeurs manquantes, taux de bruit dans les mesures des attributs de description, proportions des classes, stationnarité ou pas de l'environnement. C'est aussi réaliser un gros travail de *préparation des données* : nettoyage, ré-organisation, enrichissement, intégration avec d'autres sources de données, etc.Ces étapes de compréhension du problème, de préparation des données, de mise au point du protocole d'apprentissage et des mesures d'évaluation des résultats, prennent, et de loin, la plus grande partie du temps pour (tenter de) résoudre un problème d'apprentissage [3]. Nous avons toujours tendance à largement sous-estimer ces étapes et à vouloir se concentrer uniquement sur la phase excitante de l'essai de méthodes d'apprentissage sur des données supposées bonnes à la consommation.

B. *Concepts de la modélisation*

La modélisation est la conception et l'utilisation d'un *modèle*. Selon son objectif et les moyens utilisés, la modélisation est dite mathématique, géométrique, 3D, empirique, etc. En informatique, la modélisation permet de concevoir l'architecture globale d'un système d'information, ainsi que l'organisation des informations à l'aide de la modélisation des données ;

MODÈLE (INFORMATIQUE) : En informatique, un modèle a pour objectif de structurer les informations et activités d'une organisation : données, traitements, et flux d'informations entre entités.

MODÈLE (MATHÉMATIQUE) : Un modèle mathématique est une description d'un système utilisant des concepts et un langage mathématiques.

Un modèle peut aider à expliquer un système et à étudier les effets de différents composants, et à faire des prédictions sur le comportement.

Modèles non paramétriques

E. g. : Prenons l'exemple de données décrites dans l'espace d'entrée $\mathcal{X} = \mathbb{R}^n$ avec n variables réelles et supposons-les étiquetées par $\times$ ou par $\bullet$. On cherche

donc une fonction de décision h, appelée hypothèse ou modèle, telle qu'elle soit capable d'étiqueter toute entrée $x \in \mathcal{X}, h : x \to \{\times, \bullet\}$. Reste à définir l'espace des hypothèses ou modèles $\mathcal{H}$ que l'on est prêt à considérer.

Toujours en considérant le problème de prédiction basique (présenté ci-dessus), on pourrait définir une hypothèse par une procédure qui examine les trois plus proches voisins du point à étiqueter x et qui choisit l'étiquette majoritaire parmi ces trois points pour étiqueter x. Il n'y a évidemment plus de paramètres pour définir les modèles possibles [3].

Un **modèle non paramétrique** est construit selon les informations provenant des données. Dans [3, 4] il est expliqué que : La régression non paramétrique exige des tailles d'échantillons plus importantes que celles de la régression basée sur des modèles paramétriques parce que les données doivent fournir la structure du modèle ainsi que les estimations du modèle.

Un **modèle paramétrique** est, s'il est approximativement valide, plus puissant qu'un modèle non paramétrique, produisant des estimations d'une fonction de régression qui ont tendance à être plus précises que ce que nous donne l'approche non paramétrique [17]. Cela devrait également se traduire par une prédiction plus précise.

Selon [3], nous pouvons construire un modèle d'apprentissage, ou l'espaces des hypothèses d'apprentissage, par :

- La classification
- La régression
- Les distributions de probabilités
- Les arbres de décisions
- Les réseaux bayésiens
- Etc.

La table suivante présente d'abord les qualités des différentes représentations des hypothèses en fonction des critères cités ci-dessus.

	Fonctions séparatrices	Distributions de probabilités	Arbres de décision	Hiérarchies de concepts	Réseaux bayésiens	Chaînes de Markov
Concept	√	√	√	√	–	–
Classes multiples	√	√	√	√	–	–
Ontologies	–	–	√	√	–	–
Régression	–	√	√	–	–	–
Évolutions temporelles	–	√	–	–	–	√
Apprentissage non supervisé	√	√	√	√	–	–
Données continues	√	√	√	–	–	√
Connaissances relationnelles				√	√	–
Degré de certitude	–	√	–	–	√	√
Degré d'imprécision	–	√	–	–	√	–
Transparence, intelligibilité	–	–	–	√	√	√

Entraînement du modèle

Tout modèle, où toutes les informations nécessaires ne sont pas disponibles, contient certains paramètres qui peuvent être utilisés pour adapter le modèle au système qu'il est censé décrire. Si la modélisation est effectuée par un réseau de neurones artificiels ou un autre apprentissage automatique, l'optimisation des paramètres est appelée **entraînement** (en anglais : **training**), tandis que l'optimisation des hyperparamètres du modèle est appelée **réglage** (en anglais : **tuning**) et utilise souvent la validation croisée [14]. Dans une modélisation plus conventionnelle à travers des fonctions mathématiques explicitement données, les paramètres sont souvent déterminés par ajustement de courbe (voir le point ??).

Une partie cruciale du processus de modélisation consiste à évaluer si oui ou non un modèle mathématique donné décrit un système avec précision. Il peut être difficile de répondre à cette question car elle implique plusieurs types d'évaluation différents.

1.2.2 *Les problèmes de régressions*

L'algorithme d'apprentissage automatique est défini comme un algorithme capable d'améliorer les performances d'un programme informatique à certaines tâches via l'expérience est quelque peu abstraite. Pour rendre cela plus concret, Une des méthode d'apprentissage automatique basique est *la régression linéaire*[14].

Dans la modélisation statistique, l'analyse de régression est un ensemble de processus statistiques permettant d'estimer les relations entre une variable dépendante et une ou plusieurs variables indépendantes [??].
En statistique, la régression linéaire est une approche linéaire pour modéliser (voir le point b.) la relation entre une réponse scalaire et une ou plusieurs variables explicatives (également appelées variables dépendantes et indépendantes). Le cas d'une variable explicative est appelé régression linéaire simple ; pour plus d'un, le processus est appelé régression linéaire multiple.

Dans la régression linéaire, les relations sont modélisées à l'aide de *fonctions prédictives*[2] linéaires dont les paramètres de modèle inconnus sont estimés à partir des données [17]. De tels modèles sont appelés modèles linéaires.

La régression linéaire a de nombreuses utilisations pratiques. Si l'objectif est la prédiction, la prévision ou la réduction des erreurs, la régression linéaire peut être utilisée pour ajuster un modèle prédictif à un ensemble de données observées de valeurs de la réponse et de variables explicatives [10]. Après avoir développé un tel modèle, si des valeurs supplémentaires des variables explicatives sont collectées sans valeur de réponse d'accompagnement, le modèle ajusté peut être utilisé pour faire une prédiction de la réponse.

Dans ce type de tâche, le programme informatique est invité à prédire une valeur numérique à partir d'une entrée donnée. Pour résoudre cette tâche, l'algorithme d'apprentissage est invité à sortir une fonction $f : \mathbb{R}^n \to \mathbb{R}$. Ce type de tâche est similaire à la **classification**, sauf que le format de sortie est différent [14].

A. *Le cas de la régression linéaire*

On appelle problèmes de régression de tels problèmes, dans lesquels la sortie est numérique, généralement un vecteur de réels, supposé dépendre de la valeur d'un certain nombre de facteurs en entrée[17].

Le vecteur d'entrée $x = (x_1, x_2, ..., x_n)^T$ est souvent appelé variable indépendante, tandis que le vecteur de sortie y est appelé variable dépendante. On formalise le problème en supposant que la sortie résulte de la somme d'une fonction déterministe f de l'entrée et d'un bruit aléatoire :

$$y = f(x) + \epsilon \tag{5}$$

2 En statistique et en apprentissage automatique, une fonction de prédicteur linéaire est une fonction linéaire d'un ensemble de coefficients et de variables explicatives, dont la valeur est utilisée pour prédire le résultat d'une variable dépendante.

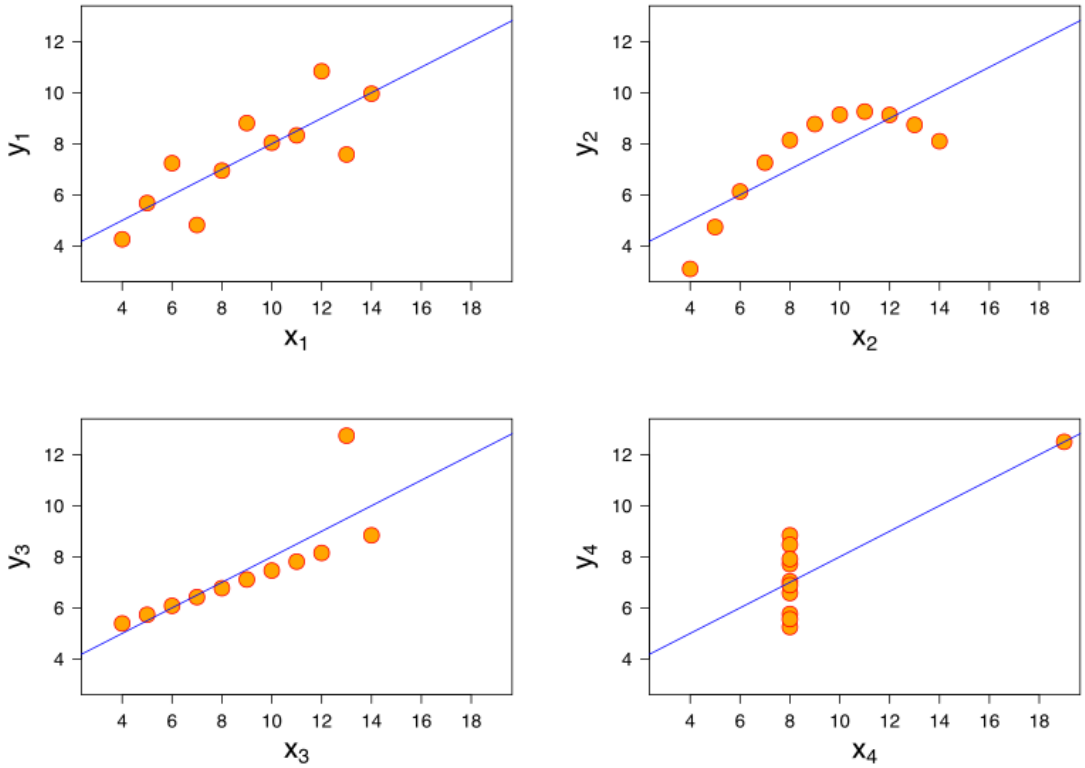


FIGURE 2 : Images illustrant l'efficacité de la régression linéaire sur plusieurs type de modèle [image Wikipédia]

où $f(x)$ est la fonction inconnue que nous souhaitons approcher par un estimateur $h(x|w)$, où h est défini à l'aide d'un vecteur w de paramètres[2].
Si l'on suppose que le bruit ϵ est nulle et de variance constante σ^2, c'est-à-dire $\epsilon = \mathcal{N}(0, \sigma^2)$, alors, en plaçant notre estimateur $h(\cdot)$ à la place de la fonction inconnue, on devrait avoir la densité conditionnelle réelle $p(y|x)$ vérifiant :

$$p(y|x) = \mathcal{N}(h(x|w), \sigma^2) \tag{6}$$

On peut estimer le vecteur de paramètres w grâce au principe de maximisation de la vraisemblance. On suppose que les couples (x_t, y_t) de l'échantillon d'apprentissage sont tirés par tirages indépendants d'une distribution de probabilités jointes inconnue $p(x, y)$, qui peut s'écrire :

$$p(y|x) = p(y|x)p(x)$$

où $p(y|x)$ est la probabilité de la sortie étant donnée l'entrée et $p(x)$ est la densité de probabilité sur les entrées [17].

Étant donné un échantillon d'apprentissage $S = \langle (x_t, y_t) \rangle_{1 \leqslant t \leqslant m}$ supposé tiré de manière indépendante et identiquement distribuée. Maximiser l'expression résultante revient alors à minimiser la somme de carrés des erreurs (SCE) :

$$SCE(w|S) = \frac{1}{2} \sum_{t=1}^{m} [y_t - h(x_t|w)]^2 \tag{7}$$

B. *Le cas de la régression générale*

La plupart des modèles de régression proposent que Y_i est une fonction de X_i et w, avec ϵ_i représentant un terme d'erreur additif ou bruit statistique aléatoire qui peut remplacer des déterminants non modélisés de Y_i :

$$Y_i = f(X_i, w) + \epsilon_i \tag{8}$$

L'objectif est d'estimer la fonction $f(X_i, w)$ qui correspond le mieux aux données.

Pour effectuer une analyse de régression, la forme de la fonction f doit être spécifié. Parfois, la forme de cette fonction est basée sur la connaissance de la relation entre Y_i et X_i. Si ces connaissances ne sont pas disponibles, un formulaire souple ou pratique pour f est choisi. Par exemple, une simple régression univariée peut proposer

$$f(X_i, w) = w_0 + w_1 X_i$$

ou

$$Y_i = w_0 + w_1 X_i + e_i$$

être une approximation raisonnable du processus statistique générant les données.

Différentes formes d'analyse de régression fournissent des outils pour estimer les paramètres. w. Par exemple, les moindres carrés trouvent la valeur de w qui minimise la somme des carrés des erreurs [11]

$$\sum_i (Y_i - f(X_i, w))^2$$

Étant donné un ensemble de données $\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^n$ de n unités statistiques, un modèle de régression linéaire suppose que la relation entre la variable dépendante y et le vecteur p des régresseurs x est linéaire. Cette relation est modélisée par un terme de perturbation ou une variable d'erreur ϵ : une variable aléatoire non observée qui ajoute du "bruit" à la relation linéaire entre la variable dépendante et les régresseurs. Ainsi le modèle prend la forme

$$y_i = w_0 + w_1 x_{i1} + \cdots + w_n x_{in} + \varepsilon_i = \mathbf{x}_i^\top \boldsymbol{w} + \varepsilon_i, \qquad avec \quad i = 1, \ldots, n,$$

Souvent, ces n équations sont empilées et écrites en notation matricielle comme

$$\mathbf{y} = X\boldsymbol{w} + \boldsymbol{\varepsilon},$$

où

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\mathcal{W}} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

$\mathbf{y}$ est un vecteur de valeurs observées y_i $(i = 1, \ldots, n)$ de la variable appelée variable mesurée ou variable dépendante.

X peut être vu comme une matrice de vecteurs-lignes $\mathbf{x}_i$ ou de vecteurs-colonnes à n dimensions X_j, appelées régresseurs, variables explicatives, variables d'entrée, variables prédictives ou variables indépendantes. La matrice X est parfois appelée la matrice de conception.

$\boldsymbol{w}$ est un vecteur de paramètre de dimension $(p + 1)$, où w_0 est le terme d'interception, s'il n'est pas inclus dans le modèle $\boldsymbol{w}$ est de dimension p. Ses éléments sont appelés coefficients de régression [3]. En régression linéaire simple, $p = 1$, et le coefficient est appelé **pente** de régression.
L'estimation statistique et l'inférence dans la régression linéaire se concentrent sur w. Les éléments de ce vecteur de paramètres sont interprétés comme les dérivées partielles de la variable dépendante par rapport aux différentes variables indépendantes [10].

En définissant les vecteurs et matrice ci dessous, $\mathbf{X}$, $\boldsymbol{w}$ et $\mathbf{y}$ (avec $S_y = y$) [3] ; le critère de la somme des carrées des erreurs s'écrit alors :

$$SCE(w|S) = \frac{1}{2}(\mathbf{S}_y - \mathbf{X}\boldsymbol{w})^T(\mathbf{S}_y - \mathbf{X}\boldsymbol{w}) \tag{9}$$

Il suffit de prendre la dérivée de la somme des carrés des erreurs (équation 7) par rapport à w, qui est maintenant remplacer par w, pour obtenir les équations :

$$\frac{\partial SCE}{\partial w} = -\mathbf{X}^T(\mathbf{S}_y - \mathbf{X}\boldsymbol{w})$$

$$\frac{\partial^2 SCE}{\partial^2 w \partial^2 w^T} = -\mathbf{X}^T\mathbf{X}$$

En supposant que la matrice X est non singulière, et donc que X^TX est positive définie, et en posant que la dérivée première est nulle, on obtient :

$$X^TXw = X^TS_y \tag{10}$$

à partir de quoi on peut calculer l'unique solution par :

$$\hat{w} = (X^TX)^{-1}X^TS_y \tag{11}$$

La valeur $\hat{y}$ prédite pour une entrée x_n est donc :

$$\hat{y} = \hat{w} \cdot x_n = (X^TX)^{-1}X^TS_yx_n$$

La régression linéaire multiple est une généralisation de la régression linéaire simple au cas de plus d'une variable indépendante, et un cas particulier des modèles linéaires généraux, limités à une variable dépendante. Le modèle de base de la régression linéaire multiple est
???

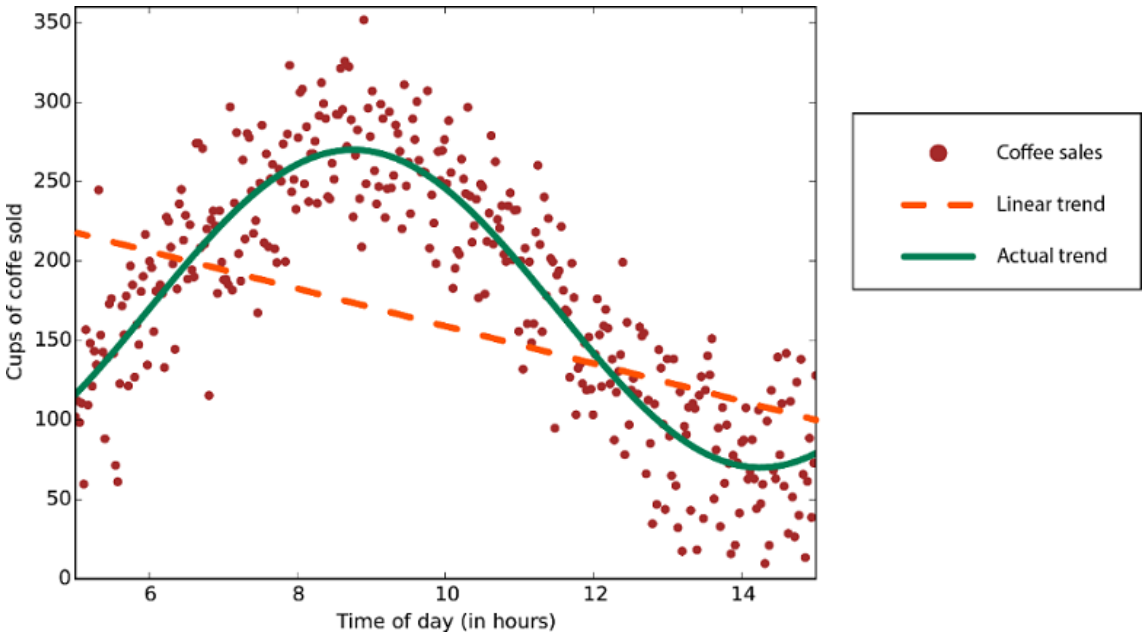


FIGURE 3 : L'efficacité de la régression non linéaire par rapport à une régression linéaire..

1.2.3 *Les problèmes de classifications*

En apprentissage automatique, les classifieurs linéaires sont une famille d'algorithmes de classement statistique. Le rôle d'un classifieur est de classer dans des groupes (des classes) les échantillons qui ont des propriétés similaires, mesurées sur des observations. Un classifieur linéaire est un type particulier de classifieur, qui calcule la décision par combinaison linéaire des échantillons [3].

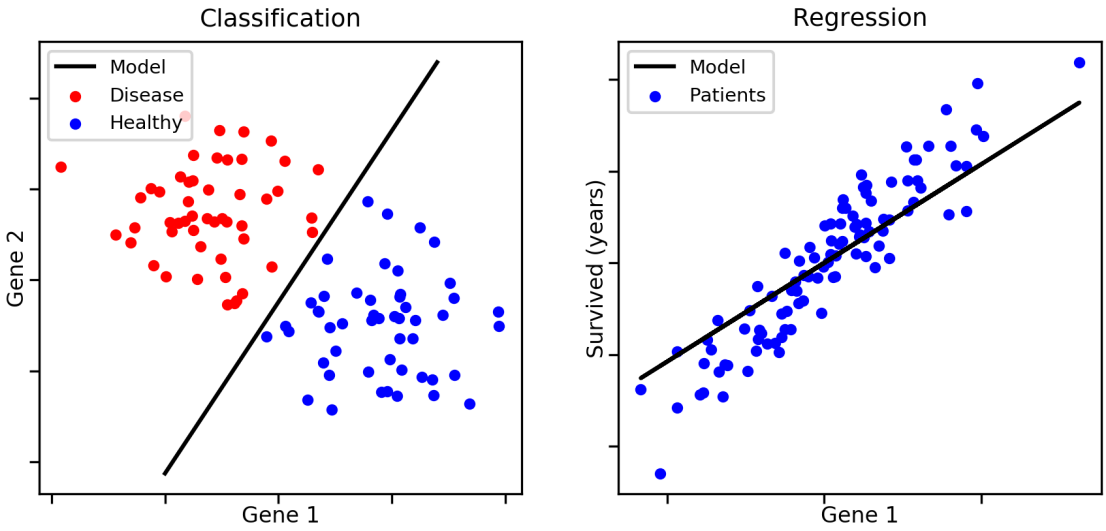


FIGURE 4 : Classification vs régression [22]

Nous nous plaçons dans le cadre où la variable dépendante ou à prédire prend ses valeurs dans un ensemble fini que l'on associe généralement à un ensemble

de classes. A la différence de la régression linéaire où l'ensemble de valeurs à prédire est infini.

Lorsque l'on se place dans un espace de représentation euclidien, on peut librement faire des hypothèses sur la géométrie des classes ou sur celles de leurs surfaces séparatrices. La plus simple d'entre elles est de supposer que deux classes peuvent être séparées par une certaine surface, définie par une équation ; les paramètres qui régissent cette équation sont alors les variables à apprendre.

Le nombre de paramètres à calculer est minimal si l'on suppose cette surface linéaire ; aussi est-ce l'hypothèse qui prévaut souvent, en particulier lorsque l'échantillon de données est de taille réduite par rapport à la dimension de l'espace d'entrée, d'autant qu'elle permet de mener des calculs faciles et de visualiser précisément le résultat obtenu [20].

Dans $\mathbb{R}^n$, une surface linéaire est un hyperplan A, défini par l'équation :

$$a_0 + a^T x = 0$$

avec a vecteur de dimension n et a_0 scalaire. Si deux classes $\mathcal{C}_1$ et $\mathcal{C}_2$ sont *séparables* par A, tous les points de la première classe sont par exemple tels que :

$$x \in \mathcal{C}_1 \implies a_0 + a^T x > 0 \tag{12}$$

et ceux de la seconde vérifient alors :

$$x \in \mathcal{C}_2 \implies a_0 + a^T x \leqslant 0 \tag{13}$$

Dans un espace de dimension $d = 1$, une séparation linéaire se réduit à la comparaison à un seuil. Prenons ce cas particulier pour donner deux exemples où un problème de discrimination à deux classes ne peut pas en pratique être complètement résolu par une séparatrice linéaire.

SÉPARATRICE LINÉAIRE : On appelle hyperplan séparateur ou séparatrice linéaire un hyperplan qui sépare parfaitement deux classes, c'est-à-dire qui vérifie les équations 12 et 13 ; en particulier, il sépare parfaitement leurs points d'apprentissage. Un hyperplan discriminant est un classificateur linéaire pour deux classes qui ne sont pas linéairement séparables. [3]

A. *Le cas non séparable*

B. *Le modèle de la régression logistique*

Ce qu'il est convenu d'appeler *régression logistique* concerne en fait une méthode de classification binaire, à l'instar du perceptron (voir le point 1.3.1). A la différence

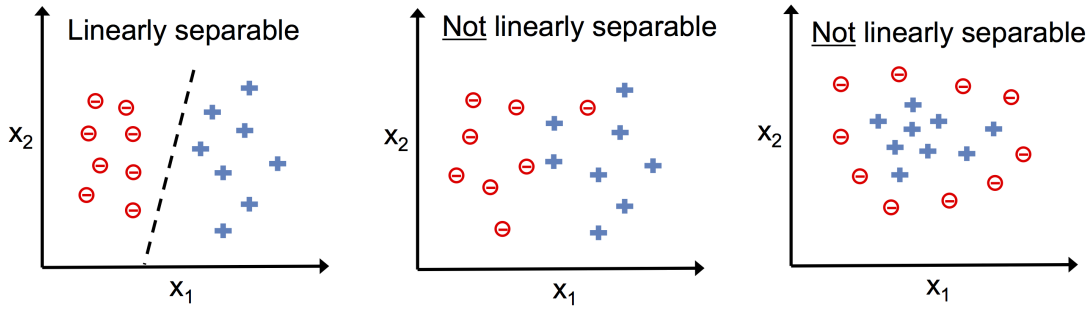


FIGURE 5 : Classes linéairement séparables [image de 22, p. 48]

du perceptron, cependant, nous allons chercher à apprendre une hypothèse h définie de $\mathbb{R}^n$ dans [0,1], et non pas dans 0,1, une motivation étant d'interpréter $h(x)$ comme étant la probabilité que l'entrée x appartienne à la classe d'intérêt que nous notons $\mathcal{C}_1$. [3]

Dans le cas à deux classes, nous sommes intéressés par le rapport de probabilités conditionnelles :

$$\frac{\mathbf{P}(y = \mathcal{C}_1|\mathbf{x})}{\mathbf{P}(y = \mathcal{C}_2|\mathbf{x})} = \frac{\mathbf{P}(y = \mathcal{C}_1)}{\mathbf{P}(y = \mathcal{C}_2)} \frac{\mathbf{p}(\mathbf{x}|y = \mathcal{C}_1)}{\mathbf{p}(\mathbf{x}|y = \mathcal{C}_2)} \tag{14}$$

Bien entendu, on affecte l'entrée x à la classe $\mathcal{C}_1$ si le rapport 14 est > 1, et à la classe $\mathcal{C}_1$ sinon [3].

Le terme $\frac{\mathbf{p}(\mathbf{x}|y=\mathcal{C}_1)}{\mathbf{p}(\mathbf{x}|y=\mathcal{C}_2)}$ est n'est facile à estimer à partir des fréquences mesurées des classes $\mathcal{C}_1$ et $\mathcal{C}_2$. Pour estimer ce terme, il faut faire des hypothèses sur sa forme. Dans la régression logistique, on fait l'hypothèse que le logarithme du rapport est de forme linéaire :

$$\log \left\{ \frac{\mathbf{p}(\mathbf{x}|y = \mathcal{C}_1)}{\mathbf{p}(\mathbf{x}|y = \mathcal{C}_2)} \right\} = w^T x + b_0$$

Il est possible de ré-exprimer l'équation 14 en utilisant les propriétés logarithmique, rappel :

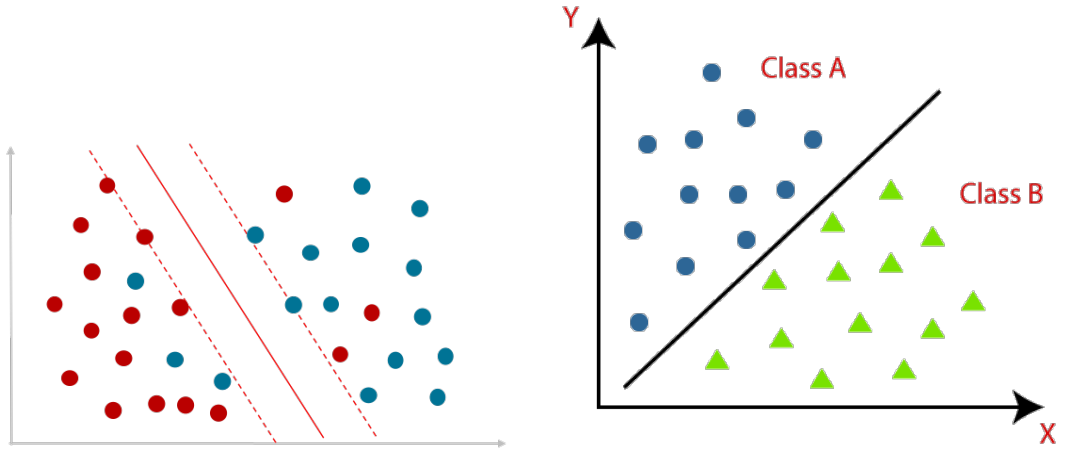
$$\log(a.b) = \log(a) + \log(b) \qquad \log(x)^n = n \log(x)$$

et en utilisant la règle de Bayes, l'équation 14 deviendra :

$$\log \frac{\mathbf{P}(\mathcal{C}_1|\mathbf{x})}{1 - \mathbf{P}(\mathcal{C}_1|\mathbf{x})} = \log \frac{\mathbf{P}(\mathcal{C}_1)}{\mathbf{P}(\mathcal{C}_2)} + \log \frac{\mathbf{p}(\mathbf{x}|y = \mathcal{C}_1)}{\mathbf{p}(\mathbf{x}|y = \mathcal{C}_2)} = w^T x + b \tag{15}$$

La fonction de la droite séparatrice comme l'illustre la figure 6a et 6b s'écrit :

$$z = w_1 x_1 + \cdots + w_n x_n + b \tag{16}$$



(a) La zone de décision et la droite séparatrice.

(b) Classification linaire de deux classes A et B

FIGURE 6 : Exemple de la classification

avec $i = 1, \ldots, n$,

$$\begin{cases} \hat{y} = 0 \quad (y \in \mathcal{C}_1) & \text{si } z < 0 \\ \hat{y} = 1 \quad (y \in \mathcal{C}_2) & \text{si } z \geqslant 0 \end{cases}$$

La fonction logistique est une fonction sigmoïde , qui prend n'importe quelle entrée réelle t, et renvoie une valeur comprise entre zéro et un [22]. La fonction logistique standard $\sigma : \mathbb{R} \to (0, 1)$ est défini comme suit :

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{17}$$

Supposons que t est une fonction linéaire (comme la droite de la formule 16) $t = z$. Et la fonction logistique générale $p : \mathbb{R} \to (0, 1)$ peut maintenant l'écrire :

$$p(x) = \sigma(t) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(w_1 x_1 + \cdots + w_n x_n + b)}} \tag{18}$$

Dans le modèle logistique, $p(x)$ est interprété comme la probabilité de la variable dépendante Y équivalant à un succès/cas-oui plutôt qu'à un échec/non-cas. Il est clair que les variables de réponse Y_i ne sont pas identiquement répartis :$P(Y_i = 1 \mid X)$ diffère d'un point de données X_i à l'autre, bien qu'ils soient indépendants étant donné la matrice de conception X et paramètres partagés w [3].

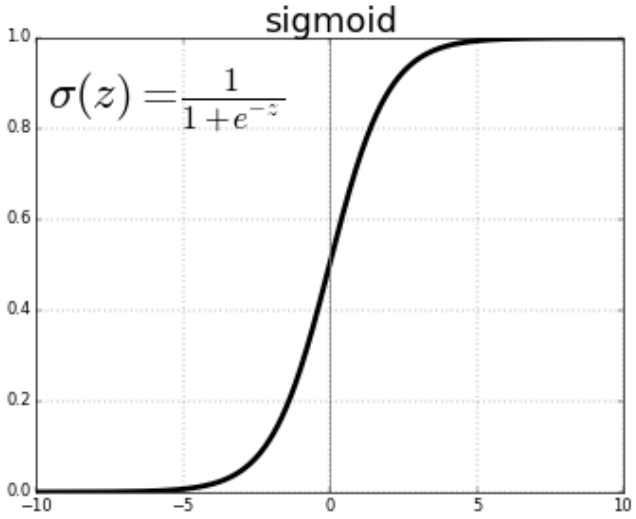


FIGURE 7 : Graphique représentant fonction sigmoïde logistique ajustée aux données (x_n, y_n). [22]

LA VRAISEMBLANCE : Indique la plausibilité du modèle vis-a-vis du vraies données. Soit l'échantillon $\mathcal{S} = (\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)$, avec $y_i \in \{\mathcal{C}_1, \mathcal{C}_2\}, \forall_i \in (1, ..., m)$. Sa vrai semblance en fonction des paramètres à apprendre s'écrit :

$$L = \prod_{i=1}^{m} p_i{}^{y_i}(1 - p_i)^{1-y_i} \tag{19}$$

où m est le nombre d'exemples d'apprentissage appartenant à la classe.

Dans [3], il est montré que ces paramètres peuvent être obtenus par maximisation de la vraisemblance des paramètres conditionnellement aux exemples. Il a été de plus montré que, sous des conditions très générales [20], le maximum de L est unique. La maximisation de la vraisemblance se fait sois en passant par le logarithme, pour obtenir la log-vraisemblance :

$$\begin{aligned} \log(L) &= \log(\prod_{i=1}^{m} p_i{}^{y_i}(1 - p_i)^{1-y_i}) \\ &= \sum_{i=1}^{m} \log(p_i{}^{y_i}) + \log((1 - p_i)^{1-y_i}) \\ &= \sum_{i=1}^{m} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \end{aligned} \tag{20}$$

Comme en Machine Learning on est plus apte à minimiser qu'à maximiser et Maximiser une fonction $f(\cdot)$ consiste à Minimiser $-f(\cdot)$ alors le log-vraisemblance s'écrira :

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \tag{21}$$

avec le terme $\frac{1}{m}$ pour augmenter la précision. Avec cette fonction, nous allons maximiser la vraisemblance L en minimisant $-\log(L)$.

1.3 RÉSEAU DE NEURONES, APPRENTISSAGE EN PROFONDEUR

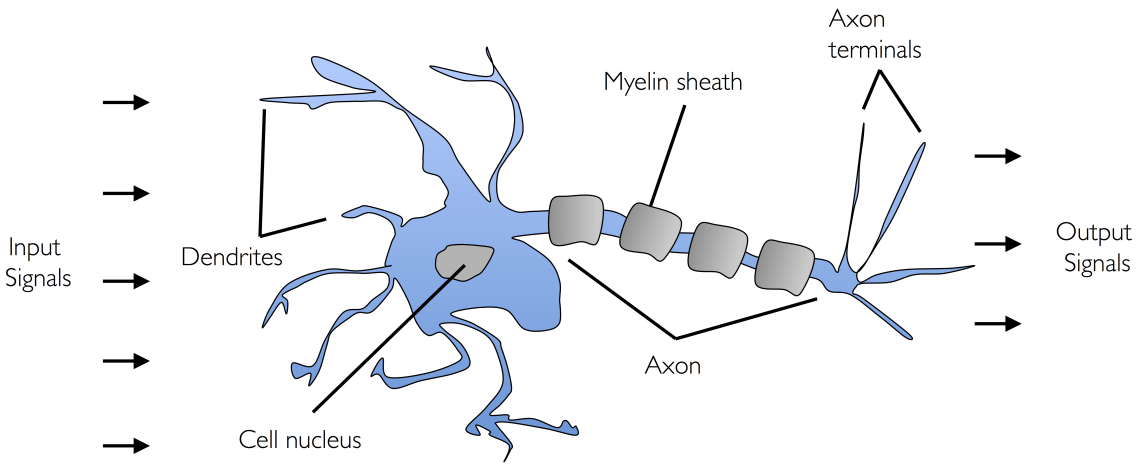


FIGURE 8 : Neurone biologique [22]

1.3.1 *Perceptron*

Le perceptron est un modèle simplifié d'un neurone biologique. Alors que la complexité des modèles de neurones biologiques est souvent nécessaire pour bien comprendre le comportement neuronal, la recherche suggère qu'un modèle linéaire de type perceptron peut produire certains comportements observés dans de vrais neurones.

Un perceptron est donc une unité de réseau neuronal, l'élément de traitement de base, qui effectue certains *calculs* pour détecter des caractéristiques ou une intelligence économique dans les données d'entrée.
Le perceptron, à l'instar du neurone artificiel classique, est aussi un algorithme d'apprentissage supervisé de classificateurs binaires. Un classificateur binaire est une fonction qui peut décider si une entrée, représentée par un vecteur de nombres, appartient ou non à une classe spécifique [??]. Il s'agit d'un type de classificateur linéaire , c'est-à-dire un algorithme de classification qui fait ses prédictions sur la base d'une fonction prédictive linéaire combinant un ensemble de poids avec le vecteur de caractéristiques.

Le perceptron a des entrées qui peuvent provenir de l'environnement ou peuvent être les sorties d'autres perceptrons. Associé à chaque entrée, $x_j \in \mathbb{R}, j = 1, 2, \ldots, n$, est un *poids de connexion, ou poids* synaptique (voir le point 1.3.2) $w_j \in \mathbb{R}$, et la sortie, y, dans le cas le plus simple est une somme pondérée des entrées [2].

$$y = \sum_{j=1}^{n} w_j x_j + w_0$$

w_0 est la valeur d'interception pour rendre le modèle plus général, il est généralement modélisé comme la pondération provenant d'une unité de biais supplémentaire, x_0 , qui est toujours +1. Nous pouvons écrire la sortie du perceptron sous la forme d'un produit scalaire.

$$y = x^T w$$

Pendant le test, avec des poids donnés, w, pour l'entrée x, nous calculons la sortie y. Pour implémenter une tâche donnée, nous avons besoin d'apprendre les poids w, les paramètres du système, de sorte que des sorties correctes soient générées compte tenu des entrées.

A. *Calcul avec l'algorithme du perceptron*

Le premier concept de règle d'apprentissage du perceptron, a été publié par Frank Rosenblatt [??], basé sur le modèle neuronal MP Neuron (McCulloch-Pitts Neuron).

Avec sa règle de perceptron, Il a proposé un algorithme qui apprendrait automatiquement les coefficients de poids optimaux qui sont ensuite multipliés par les caractéristiques d'entrée afin de décider si un neurone se déclenche ou non. Dans le cadre de l'apprentissage supervisé et de la classification, un tel algorithme pourrait alors être utilisé pour prédire si un échantillon appartient à une classe ou à l'autre [22].

Il est important de noter que la convergence du perceptron n'est garantie que si les deux classes sont linéairement séparables et que le taux d'apprentissage est suffisamment faible. Si les deux classes ne peuvent pas être séparées par une limite de décision linéaire, nous pouvons définir un nombre maximum de passages sur l'ensemble de données d'apprentissage (époques) et/ou un seuil pour le nombre d'erreurs de classification tolérées - le perceptron n'arrêterait jamais de mettre à jour les poids autrement : ???

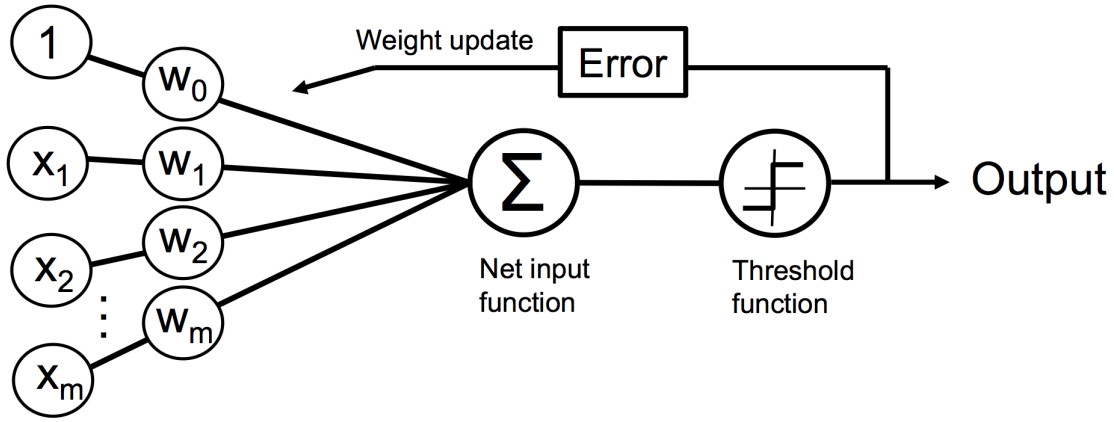


FIGURE 9 : Neurone artificiel modèle : Perceptron [22]

B. *Le réseau de neurones artificiels (Perceptron Multicouche)*

Les réseaux de neurones ont été introduits pour la première fois comme méthode d'apprentissage par Frank Rosenblatt, bien que le modèle d'apprentissage appelé perceptron soit différent des réseaux de neurones modernes, nous pouvons toujours considérer le perceptron comme le premier réseau de neurones artificiels [20].

Le perceptron multicouche (en anglais : multilayer perceptron MLP) est un type de réseau neuronal artificiel organisé en plusieurs couches, où les informations ne circulent que de la couche d'entrée à la couche de sortie. Il s'agit donc d'un réseau à propagation directe (feed forward), autrement dit le réseau profond à action directe. Un perceptron multicouche est juste une fonction mathématique mappant un ensemble de valeurs d'entrée à des valeurs de sortie. La fonction est formée en composant de nombreuses fonctions plus simples. Nous pouvons considérer chaque application d'une fonction mathématique différente comme fournissant une nouvelle représentation de l'entrée [3, 14].

Les perceptrons à une seule couche ne sont capables d'apprendre que des motifs linéairement séparables. Pour une tâche de classification avec une fonction d'activation d'étape, un seul nœud aura une seule ligne divisant les points de données formant les motifs. Plus de nœuds peuvent créer plus de lignes de division, mais ces lignes doivent en quelque sorte être combinées pour former des classifications plus complexes. Une deuxième couche de perceptrons, voire de nœuds linéaires, suffit à résoudre de nombreux problèmes autrement non séparables [3].

Les réseaux de neurones artificiels (ANN) fonctionnent vaguement sur le principe de l'apprentissage d'une distribution distribuée de données. L'hypothèse sous-jacente est que les données générées sont le résultat d'une combinaison non linéaire d'un ensemble de facteurs latents et si nous sommes capables d'apprendre cette représentation distribuée, nous pouvons alors faire des prédictions précises sur un nouvel ensemble de données inconnues. Le réseau de neurones le plus simple aura une couche d'entrée, une couche cachée (résultat de l'application d'une transformation non linéaire aux données d'entrée) et une couche de sortie. Les paramètres du modèle ANN sont les poids de chaque connexion qui existent dans le réseau et parfois un paramètre de biais [20].

1.3.2 *Fonctions d'activation, poids et biais*

A. *Poids et biais*

B. *Fonctions d'activation tangente (Sigmoïde et hyperbolique)*

La fonction d'activation est responsable de la transformation de l'entrée pondérée sommée du nœud en activation du nœud ou de la sortie pour cette entrée. Pour un nœud donné, les entrées sont multipliées par les poids d'un nœud et

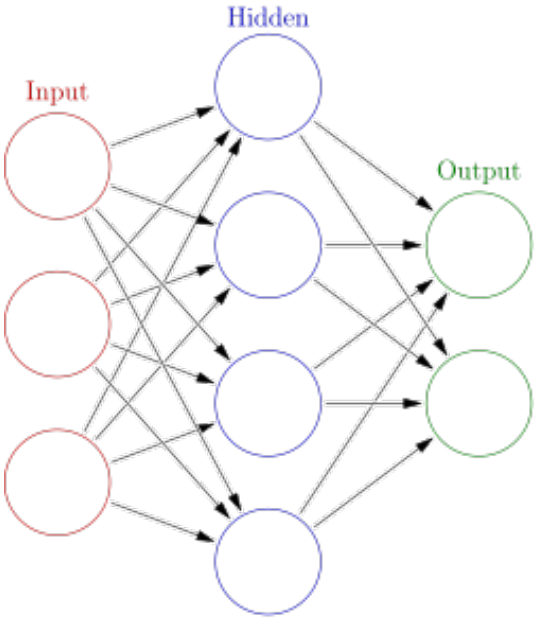


FIGURE 10 : Un réseau de neurones artificiels est un groupe de nœuds inter-connectés, inspiré d'une simplification des neurones d'un cerveau. Ici, chaque nœud circulaire représente un neurone artificiel et une flèche représente une connexion de la sortie d'un neurone artificiel à l'entrée d'un autre.

additionnées. Cette valeur est appelée activation sommée du nœud. L'activation sommée est ensuite transformée via une fonction d'activation et définit la sortie spécifique ou « activation » du nœud [22].
La fonction sigmoïde (aussi appelé fonction logistique voir le point 1.2.3 b.) est utilisée ici comme une fonction d'activation.

La fonction d'activation la plus simple est appelée activation linéaire, où aucune transformation n'est appliquée. Un réseau composé uniquement de fonctions d'activation linéaires est très facile à former, mais ne peut pas apprendre des fonctions de cartographie complexes. Les fonctions d'activation linéaires sont toujours utilisées dans la couche de sortie pour les réseaux qui prédisent une quantité (par exemple, les problèmes de régression, voir le point 1.2.2) [**geron2017hands**, 16].

Les fonctions d'activation non linéaires sont préférées car elles permettent aux nœuds d'apprendre des structures plus complexes dans les données. Traditionnellement, deux fonctions d'activation non linéaires largement utilisées sont les fonctions d'activation tangente sigmoïde et hyperbolique [14].

La fonction **d'activation sigmoïde**, est traditionnellement une fonction d'activation très populaire pour les réseaux de neurones. L'entrée de la fonction est transformée en une valeur comprise entre 0,0 et 1,0. Les entrées qui sont beaucoup plus grandes que 1,0 sont transformées à la valeur 1,0, de même, les valeurs beaucoup plus petites que 0,0 sont alignées sur 0,0. La forme de la fonction pour toutes les entrées possibles est une forme en S de zéro jusqu'à 0,5 à 1,0. Pendant longtemps, jusqu'au début des années 1990, c'était l'activation par défaut utilisée sur les réseaux de neurones [16].

La fonction **tangente hyperbolique**, ou **tanh** en abrégé, est une fonction d'activation non linéaire de forme similaire qui génère des valeurs comprises entre -1,0 et 1,0. À la fin des années 1990 et au cours des années 2000, la fonction tanh a été préférée à la fonction d'activation sigmoïde car les modèles qui l'utilisaient étaient plus faciles à entraîner et avaient souvent de meilleures performances prédictives [14]. la fonction d'activation tangente hyperbolique fonctionne généralement mieux que la sigmoïde logistique.

Limitations des fonctions d'activation sigmoïde et tanh

Cela signifie que pour tanh et sigmoïde, les grandes valeurs sont alignées sur 1,0 et les petites valeurs sont alignées sur -1 ou 0. De plus, la fonction n'est vraiment sensible qu'aux changements proches du point médian de l'entrée. Par exemple, 0,5 pour les sigmoïdes et 0,0 pour la tanh.

Les unités sigmoïdales saturent sur la majeure partie de leur domaine - elles saturent à une valeur élevée lorsque z est très positif, saturent à une valeur faible lorsque z est très négatif et ne sont fortement sensibles à leur entrée que lorsque z est proche de 0 [22].

La sensibilité et la saturation limitées de la fonction se produisent indépendamment du fait que l'activation additionnée du nœud fourni en entrée contient des informations utiles ou non. Une fois saturé, il devient difficile pour l'algorithme d'apprentissage de continuer à adapter les poids pour améliorer les performances du modèle.

Les couches profondes des grands réseaux utilisant ces fonctions d'activation non linéaires ne reçoivent pas d'informations de gradient utiles. L'erreur est rétropropagée (voir le point ? ? ?) sur le réseau et utilisée pour mettre à jour les pondérations. La quantité d'erreur diminue considérablement avec chaque couche supplémentaire à travers laquelle elle se propage, compte tenu de la dérivée de la fonction d'activation choisie. C'est ce qu'on appelle le problème du gradient de fuite et empêche les réseaux profonds (multicouches) d'apprendre efficacement [**geron2017hands**].

Bien que l'utilisation de fonctions d'activation non linéaires permette aux réseaux de neurones d'apprendre des fonctions de cartographie complexes, elles empêchent efficacement l'algorithme d'apprentissage de fonctionner avec des réseaux profonds.

c. *Fonction d'activation ReLU*

Dans le domaine des réseaux de neurones artificiels, ReLU (Rectified Linear Unit) ou fonction d'activation du redresseur est une fonction d'activation définie comme la partie positive de son argument [14].

$$f(x) = x^+ = \max(0, x)$$

ReLU est une fonction linéaire par morceaux qui produira l'entrée directement si elle est positive, sinon, elle produira zéro. C'est devenu la fonction d'activation par défaut pour de nombreux types de réseaux de neurones, car un modèle qui l'utilise est plus facile à former et atteint souvent de meilleures performances [**geron2017hands**].

La conception d'unités cachées est un domaine de recherche extrêmement actif et ne dispose pas encore de nombreux principes directeurs théoriques définitifs. Les fonctions d'activations ReLU sont un excellent choix par défaut d'unité cachée. De nombreux autres types d'unités cachées sont disponibles. Il peut être difficile de déterminer quand utiliser quel type, bien que les unités linéaires rectifiées soient généralement un choix acceptable [14].

???

Pour un exemple de la façon dont ReLU peut résoudre le problème des gradients de fuite, consultez le didacticiel

1.3.3 *Réseau neuronal convolutif (CNN)*

Le réseau neuronal convolutif est un type de réseau neuronal artificiel qui utilise plusieurs perceptrons qui analysent les entrées d'image et ont des poids et des bases apprenables sur plusieurs parties d'images et capables de se séparer les unes des autres [25].

???

L'un des avantages de l'utilisation du réseau de neurones convolutifs est qu'il exploite l'utilisation de la cohérence spatiale locale dans les images d'entrée, ce qui leur permet d'avoir moins de poids car certains paramètres sont partagés [25].

??? [23]

A. *La convolution*

La convolution est une opération mathématique simple généralement utilisée pour le traitement et la reconnaissance d'images. Sur une image, son effet s'assimile à un filtrage dont voici le fonctionnement

???

B. *Les différents couches d'un CNN*

???

c. *L'architecture VGGNet*

VGG signifie Visual Geometry Group, il s'agit d'une architecture standard de réseau de neurones à convolution profonde (CNN) à plusieurs couches.

L'architecture VGG est la base d'un modèle innovant de reconnaissance d'objets. Développé en tant que réseau neuronal profond, VGGNet va au-delà d'ImageNet et dépasse la ligne de base de nombreuses tâches et ensembles de données. De plus, c'est toujours l'une des architectures de reconnaissance d'images les plus populaires [3, 25].

Les VGGNet sont basés sur les caractéristiques les plus essentielles des réseaux de neurones convolutifs (CNN). Le graphique suivant montre le concept de base du fonctionnement d'un CNN.

Un bref coup d'œil à l'architecture de VGG :

- **Entrée** : Le VGGNet prend une taille d'entrée d'image de 224×224. Pour le concours ImageNet, les créateurs du modèle ont recadré le patch central 224 × 224 dans chaque image pour conserver la cohérence de la taille d'entrée de l'image [24].
- **Couches convolutives** : Les couches convolutives de VGG tirent parti d'un champ de réception minimal, c'est-à-dire 3 × 3, la plus petite taille possible qui capture toujours haut/bas et gauche/droite. De plus, il existe également des filtres de convolution 1 × 1 agissant comme une transformation linéaire de l'entrée. Vient ensuite une unité ReLU, qui est une énorme innovation d'AlexNet qui réduit le temps de formation. ReLU signifie fonction d'activation d'unité linéaire rectifiée ; c'est une fonction linéaire par morceaux qui produira l'entrée si elle est positive ; sinon, la sortie est nulle. La foulée de convolution est fixée à 1 pixel pour conserver la résolution spatiale préservée après la convolution (la foulée est le nombre de décalages de pixels sur la matrice d'entrée) [16, 25].
- **Couches cachées** : Toutes les couches cachées du réseau VGG utilisent ReLU. VGG n'utilise généralement pas la normalisation de la réponse locale (LRN) car elle augmente la consommation de mémoire et le temps de formation. De plus, il n'apporte aucune amélioration à la précision globale [25].
- **Couches entièrement connectées** : Le VGGNet a trois couches entièrement connectées. Sur les trois couches, les deux premières ont 4096 canaux chacune et la troisième a 1000 canaux, 1 pour chaque classe [25].

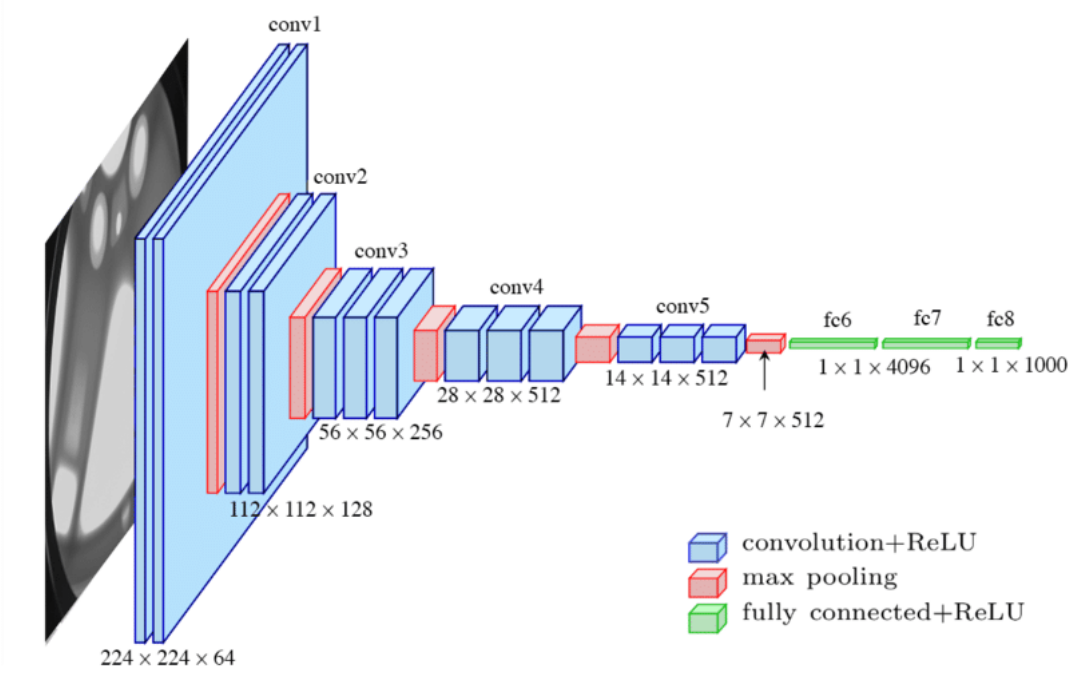


FIGURE 11 : CNN : architecture VGG-16 [22]

ANNEXES ET BIBLIOGRAPHIES

BIBLIOGRAPHIE

[1] Yaovi Ahadjitse. "Reconnaissance d'objets en mouvement dans la vidéo par description géométrique et apprentissage supervisé". Thèse de doct. Université du Québec en Outaouais, 2013.

[2] E. Alpaydin. *Introduction to Machine Learning*. Adaptive computation and machine learning. MIT Press, 2010. isbn : 9780262012430.

[3] Vincent Barra Antoine Cornuéjols Laurent Michet. *Apprentissage artificiel : Deep leaning, concepts et algorithmes*. 3rd. Eyrolles, 2018, p. 239-263.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. First. Springer-Verlag New York, 2006, p. 179-195.

[5] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In : *Proceedings of COMPSTAT'2010*. Springer, 2010, p. 177-186.

[6] Léon Bottou. "Stochastic gradient descent tricks". In : *Neural networks : Tricks of the trade*. Springer, 2012, p. 421-436.

[7] Léon Bottou, Frank E Curtis et Jorge Nocedal. "Optimization methods for large-scale machine learning". In : *Siam Review* 60.2 (2018), p. 223-311.

[8] F. Coulombeau, G. Debeaumarché, B. David, F. Dorra, S. Dupont et M. Hochart. *Mathématiques MPSI-PCSI : Programme 2013 avec algorithmique en Scilab*. Cap Prépa. Pearson, 2013. isbn : 9782744076527.

[9] Pádraig Cunningham, Matthieu Cord et Sarah Jane Delany. "Supervised learning". In : *Machine learning techniques for multimedia*. Springer, 2008, p. 21-49.

[10] R.B. Darlington et A.F. Hayes. *Regression Analysis and Linear Models : Concepts, Applications, and Implementation*. Methodology in the Social Sciences. Guilford Publications, 2016. isbn : 9781462521135.

[11] Natarajan Deepa, B Prabadevi, Praveen Kumar Maddikunta, Thippa Reddy Gadekallu, Thar Baker, M Ajmal Khan et Usman Tariq. "An AI-based intelligent system for healthcare analysis using Ridge-Adaline Stochastic Gradient Descent Classifier". In : *The Journal of Supercomputing* 77 (2021), p. 1998-2017.

[12] Kary Främling. "Scaled Gradient Descent Learning Rate". In : *Reinforcement Learning With Light-Seeking Robot, Proceedings of ICINCO* (2004), p. 1-8.

[13] Yoav Freund et Robert E Schapire. "Large margin classification using the perceptron algorithm". In : *Machine learning* 37.3 (1999), p. 277-296.

[14] I. Goodfellow, Y. Bengio et A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. isbn : 9780262035613.

[15] Daniel Kirsch JUDITH HURWITZ. *Machine Learning For Dummies*. IBM Limited Edition. John Wiley et Sons, Inc., 2018.

[16] Alex KRIZHEVSKY, Ilya SUTSKEVER et Geoffrey E HINTON. "Imagenet classification with deep convolutional neural networks". In : *Advances in neural information processing systems* 25 (2012).

[17] N. MATLOFF. *Statistical Regression and Classification : From Linear Models to Machine Learning*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2017. ISBN : 9781351645898.

[18] Praneeth NETRAPALLI. "Stochastic gradient descent and its variants in machine learning". In : *Journal of the Indian Institute of Science* 99.2 (2019), p. 201-213.

[19] Jorge NOCEDAL et Stephen J WRIGHT. *Numerical optimization*. T. 35. 1999.

[20] D. SARKAR, R. BALI et T. SHARMA. *Practical Machine Learning with Python : A Problem-Solver's Guide to Building Real-World Intelligent Systems*. Apress, 2017. ISBN : 9781484232071.

[21] Carl-Erik SÄRNDAL, Bengt SWENSSON et Jan WRETMAN. *Model assisted survey sampling*. Springer Science & Business Media, 2003.

[22] Vahid Mirjalili SEBASTIEN RASCHKA. *Python Machine Learning and Deep Learning, with sckit-learn and Tensorflow*. 2nd. Packt, 2017, p. 17-139.

[23] Hoo-Chang SHIN, Holger R ROTH, Mingchen GAO, Le LU, Ziyue XU, Isabella NOGUES, Jianhua YAO, Daniel MOLLURA et Ronald M SUMMERS. "Deep convolutional neural networks for computer-aided detection : CNN architectures, dataset characteristics and transfer learning". In : *IEEE transactions on medical imaging* 35.5 (2016), p. 1285-1298.

[24] Karen SIMONYAN et Andrew ZISSERMAN. "Very deep convolutional networks for large-scale image recognition". In : *arXiv preprint arXiv :1409.1556* (2014).

[25] Srikanth TAMMINA. "Transfer learning using VGG-16 with deep convolutional neural network for classifying images". In : *International Journal of Scientific and Research Publications (IJSRP)* 9.10 (2019), p. 143-150.

[26] Rob GJ WIJNHOVEN et PHN de WITH. "Fast training of object detection using stochastic gradient descent". In : *2010 20th International Conference on Pattern Recognition*. IEEE. 2010, p. 424-427.