UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRESENTÉ À

L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR

ROKHAYA YADE

CLASSIFICATION BAYÉSIENNE, APPRENTISSAGE ET RÉSEAUX DE NEURONES : APPLICATION EN SCIENCE DES DONNÉES

DECEMBRE 2022

Université du Québec à Trois-Rivières

Service de la bibliothèque

Avertissement

L'auteur de ce mémoire, de cette thèse ou de cet essai a autorisé l'Université du Québec à Trois-Rivières à diffuser, à des fins non lucratives, une copie de son mémoire, de sa thèse ou de son essai.

Cette diffusion n'entraîne pas une renonciation de la part de l'auteur à ses droits de propriété intellectuelle, incluant le droit d'auteur, sur ce mémoire, cette thèse ou cet essai. Notamment, la reproduction ou la publication de la totalité ou d'une partie importante de ce mémoire, de cette thèse et de son essai requiert son autorisation.

Table des matières

Résumé **vi**

Abstract **vii**

Avant-propos **viii**

Introduction **1**

Classification bayésienne **3**

1.1 Classifieur optimal de Bayes 4

1.2 Classifieur naïf de Bayes 5

1.3 Classification sous la fonction de perte 7

1.3.1 Observation gaussienne univariée avec une variance commune . . . 9

1.3.2 Observations dépendantes avec covariance commune 11

1.3.3 Observations dépendantes dont les covariances diffèrent par un facteur d'échelle 11

1.4 Classification des données incertaines 12

1.5 La méthode Autoclass 16

Réseaux bayésiens **19**

2.1 Définition d'un réseau bayésien 20

2.2 Inférence 22

2.3 Apprentissage des paramètres 23

2.4 Etude pratique, avantages, inconvénients 24

Réseaux de neurones artificiels **28**

3.1 Le Perceptron multicouches (PMC) 30

3.2 Les réseaux réccurents 31

3.3 Apprentissage 33

3.4 Le Réseau de Kohonen 37

3.5 Réseaux de neurones convolutifs 38
3.5.1 Contexte et importance de la convolution 39
3.5.2 Définition de la convolution 41
3.5.3 Hyperparamètres 42
3.5.4 Architecture CNN 45
3.5.5 Exemples de réseaux convolutifs 47
3.6 Les auto-encodeurs 49
3.6.1 Architecture des auto-encodeurs 50
3.6.2 Applications des auto-encodeurs 53

Application en science des données **54**
4.1 Description des données 54
4.2 Etude comparative : expérimentation et résultats 55

Conclusion et perspectives **57**

Bibliographie **63**

Annexes **64**

Table des figures

1 Structure d'un réseau bayésien 21

2 Réseau bayésien sigmoidal [51] 23

3 Modèle artificiel d'un neurone biologique [20]. 28

4 Perceptron Multicouches avec trois couches [58] 31

5 Réseaux réccurents ART-1 de Grossberg et Carpenter [13] 32

6 Réseaux RBF [12] 35

7 La fonction de base radiale 36

8 Structure d'un réseau de Kohonen [4] 37

9 Carte rectangulaire avec 42 neurones [43] 38

10 Un exemple de convolution 2-D. La sortie d'une image d'entrée et d'un noyau est plus petite que l'image d'entrée [36] 42

11 Etapes de Stride [51] 43

12 Etapes de la convolution[46] 44

13 Architecture du Réseau LeNet. 47

14 Architecture du Réseau AlexNet 48

15 Architecture d'un auto-encodeur[40] 51

16 Réduction de dimensionnalité linéaire et non linéaire [42] 52

Liste des tableaux

1 Récapitulatif des F-score selon les méthodes et les temporalités de prédiction[6] 26

2 Tableau comparatif de la performance Classifieur Naïf de Bayes, Réseaux bayésiens, PMC et Réseaux de neurones convolutifs sur le jeu de données MNIST et COVID 56

Résumé

La classification est une technique d'apprentissage automatique qui facilite le traitement de l'information permettant de faire le regroupement des données en différentes classes. On compte à nos jours diverses méthodes de classification statistiques ou neuronales et qui peuvent être supervisées ou non supervisées. Dans ce mémoire de recherche, nous nous penchons d'une part sur la méthode statistique probabiliste dite classification bayésienne. D'autre part, nous aborderons les algorithmes de classification en lien avec les réseaux de neurones. Concernant la classification bayésienne, nous présenterons l'aspect formel de quelques méthodes à savoir la classification optimale de Bayes, la méthode dite Naïve qui est la plus populaire et les réseaux bayésiens. Pour les réseaux de neurones artificiels, le perceptron multicouches (PMC) qui est un classique sera traité ; les réseaux de neurones convolutifs conçus pour la classification d'images et les réseaux auto-encodeurs seront également vus. Toutes les méthodes citées seront appliquées sur deux jeux de données différents afin d'évaluer leur performance et établir une comparaison. Nous travaillerons principalement sur le jeu de données MNIST, la base d'images des chiffres manuscrits et celui traitant un jeu de données de la Covid19 afin de prédire le risque de mortalité des patients covid-positifs présentant certaines comorbidités.

Abstract

Classification is a machine learning technique that facilitates the processing of information allowing the grouping of data into different classes. Nowadays, there are various statistical or neural classification methods that can be supervised or unsupervised. In this research paper, we focus on the probabilistic statistical method called Bayesian classification. On the other hand, we will discuss classification algorithms in relation to neural networks. Concerning the Bayesian classification, we will present the formal aspect of some methods, namely the optimal classification of Bayes, the so-called Naïve method which is the most popular and the Bayesian networks. For artificial neural networks, the multilayer perceptron (MLP) which is a classic will be treated ; convolutional neural networks designed for image classification and auto-encoding networks will also be seen. All the mentioned methods will be applied on two different datasets in order to evaluate their performance and establish a comparison. We will mainly work on the MNIST dataset, the handwritten digit image database and the one processing a Covid19 dataset to predict the mortality risk of covid-positive patients with certain comorbidities.

Avant-propos

Ce mémoire de recherche s'intéresse principalement à la classification bayésienne, aux réseaux de neurones artificiels et à leur application en science des données. La méthode bayésienne est une méthode statistique classique basée sur le théorème de Bayes avec l'usage de probabilité conditionnelle. Elle fait partie des premières formes d'apprentissage automatique et est encore largement utilisée. Les méthodes de classification neuronales quant à elles ont émergé dans les années 50 mais n'ont connu un important essor que dans les années 80 avec la découverte de l'algorithme de rétropropagation. Dans notre étude, nous commencerons par faire une revue de littérature de ces algorithmes de classification et, par la suite, nous étudierons leurs performances en les comparant sur deux jeux de données distincts.

Je tiens à remercier tout d'abord ma directrice de recherche, Mme Nadia Ghazzali, pour tout le soutien et la confiance qu'elle m'a accordés durant tout mon parcours à l'UQTR. Son esprit critique et sa rigueur m'ont permis de bonifier ce document. Je lui serai éternellement reconnaissante pour toute les connaissances et expériences qu'elle a partagées avec moi tout au long de la maîtrise ainsi que de son soutien financier. Mes remerciements vont aussi à l'endroit de mon codirecteur, M.Mhamed Mesfioui, pour son soutien pédagogique et financier. Je remercie aussi l'Institut des Sciences Mathématiques du Québec (ISM), le laboratoire d'intelligence Artificielle Appliquée (LI2A) de l'UQTR et le gouvernement du Québec pour leur appui financier. Je remercie tout le personnel de l'UQTR, particulièrement celui du département de mathématiques et d'informatique pour tous les services rendus, leur réactivité et leur orientation qui a beaucoup facilité mon intégration.

Je remercie également mes parents et mes frères qui m'ont toujours encouragés, accompagnés et motivés durant tout mon cursus scolaire. Je remercie aussi tous mes amis particulièrement Birama Fall et mes camarades qui ont lu et apporté leur contribution à la rédaction de ce mémoire.

Introduction

La classification est une méthode très utilisée en analyse des données. Son objectif est de regrouper les objets similaires d'un ensemble de données. Dans le processus, on vise à former des classes ou groupes homogènes de sorte que les objets qui se ressemblent le plus soient dans la même classe (homogénéité) et que les classes soient distinctes entre elles (hétérogénéité). Les méthodes de classification peuvent être classées en 2 principales catégories : les méthodes statistiques et les méthodes neuronales. Ces méthodes peuvent aussi être regroupées selon l'approche adoptée. On distingue l'approche supervisée et celle non supervisée.

L'apprentissage supervisé est l'approche la plus courante. Il consiste à apprendre comment associer à des données d'entrée des étiquettes cibles (targets) connues étant donné un ensemble d'exemples d'apprentissage [17].

L'apprentissage non supervisé contrairement à l'apprentissage supervisé consiste à rechercher des transformations interessantes des données d'entrée sans l'aide d'aucune étiquette cible [17].

On parle aussi d'apprentissage auto-supervisé qui est un apprentissage supervisé sans étiquettes annotées par des humains. Les auto-encodeurs en sont un exemple. On a aussi l'apprentissage par renforcement qui intéresse actuellement la communauté de la recherche et qui a été longtemps négligée. Dans cette approche, un agent reçoit des informations sur son environnement et il apprend à choisir des actions qui maximisent sa récompense [17].

Dans notre étude, nous nous intéressons à la classification bayésienne, qui est une méthode statistique probabiliste, et aux méthodes neuronales. Ce document est constitué de quatre chapitres. Dans le premier chapitre, on présentera des techniques de classification bayésienne à savoir la classification optimale de Bayes et la méthode naïve. On abordera aussi la classification bayésienne sous l'angle de la fonction de perte. Le chapitre 2 évoquera sommairement les réseaux bayésiens afin de compléter la présentation de l'approche bayésienne. Le chapitre 3 sera consacré aux réseaux de neurones précisément le perceptron multicouches, les réseaux de Kohonen, les réseaux convolutifs et les auto-encodeurs. Le

chapitre 4 détaillera et comparera les résultats obtenus sur deux jeux de données à savoir le MNIST avec les chiffres manuscrits et celui traitant de la covid19 pour prédire le risque de mortalité en fonction de certaines variables cliniques.

Chapitre 1

Classification bayésienne

La classification bayésienne est l'une des premières méthodes statistiques pour l'analyse des données. Elle est basée sur la théorie de Bayes dont les premiers travaux datent du 17 ième siècle. Cependant, on voit que depuis une décennie cette méthode s'applique dans de nombreux domaines tel que les sciences de la santé, la chimie, la bioinformatique, la thermodynamique, les sciences cognitives, l'intelligence artificielle, la robotique, etc. L'outil fondamental utilisé par la méthode bayésienne pour traiter l'information est la notion de probabilité. Les données sont classifiées en se basant sur l'inférence bayésienne. On dispose d'un échantillon X de données dont la classe est inconnue et qu'on veut déterminer. On définit H une hypothèse (X appartient à la classe C par exemple) et on cherche à déterminer $P(H|X)$ la probabilité de vérification de H après l'observation de X. On appelle $P(H|X)$ la probabilité a posteriori après la connaissance de X et $P(H)$ la probabilité a priori représentant la probabilité de vérification de H pour n'importe quel exemple de données. La probabilité postérieure est calculée selon le théorème de Bayes de la manière suivante :

$$P(H|X) = \frac{P(X|H).P(H)}{P(X)} \tag{1}$$

où $P(H|X)$ est la probabilité conditionnelle.

La méthode bayésienne pour résoudre un problème d'analyse de données commence par la formulation de notre connaissance a priori du problème en établissant un modèle M probabiliste de la génération des données. Si ce modèle possède des paramètres libres Θ, on traduit ces connaissances en une distribution de probabilités sur ces paramètres. Après la collecte des données, on utilise la formule de Bayes pour mettre à jour les connaissances

afin de tirer des conclusions à travers des prédictions. Par la suite, on vérifie la pertinence des résultats. Si la vraisemblance du modèle est trop faible ou si les performances prédictives sont mauvaises ou si les conclusions sont manifestement fausses, on retourne à la formulation a priori du problème.

Pour construire un classifieur de Bayes à partir d'estimations de densité on commence par séparer l'ensemble d'entraînement en m sous-ensembles contenant chacun tous les points d'une même classe. Ensuite, on entraîne un estimateur de densité sur chacun et on détermine les probabilités a priori de chaque classe avant d'appliquer la règle de Bayes pour obtenir les probabilités a posteriori.

Le but est de minimiser le taux d'erreur. La classification bayésienne réunit une famille d'algorithmes qui vise à prédire la valeur des parmètres du modèle en termes de probabilité [44]. Pour ce faire, il faut commencer par lui fournir une distribution a priori sur les paramètres (i.e. reflétant les hypothèses que l'on a, a priori, sur le modèle ; cette distribution est appelée «prior»). L'algorithme apprend la distribution (appelée «posterior») via la règle de Bayes, en utilisant les données [44]. Dans le cadre de l'apprentissage bayésien, on note différents types de classifieurs : le classifieur optimal de Bayes, le classifieur Bayes naïf et les réseaux bayésiens.

1.1 Classifieur optimal de Bayes

Dans l'optique de la minimisation du taux d'erreur, le principe fondamental des méthodes bayésiennes est le classifieur optimal de Bayes où sont considérées deux situations pour classer un évènement :

-Premièrement, un évènement e est affecté à une classe C_i si ses occurences sont dans cette dernière. Dans ce cas, aucune ambiguité ne se pose.

-Deuxièmement, si les occurrences de l'évènement sont dans des classes différentes, alors la meilleure prédiction de classe est celle qui minimise le taux d'erreur, on choisit donc la classe qui apparaît le plus, pour les différentes occurrences. Ainsi la règle de la décision de classer l'événement e dans la classe C_i peut être formulée comme suit :

$$P(e|C_i)P(C_i) \geq P(e|C_j)P(C_j) \tag{2}$$

pour chaque $i \neq j$.

Cette méthode tire son nom par le principe de maximisation de la probabilité d'appartenance des occurrences d'un évènement à une classe donnée. La difficulté de ce classifieur réside du fait de la détermination de la probabilité conditionnelle qui n'est pas facile à estimer dans la pratique. Cependant, l'application de cette méthode peut s'avérer impossible si les variables sont dépendantes et la population infinie ou non dénombrable [7].

1.2 Classifieur naïf de Bayes

Cette technique est applicable uniquement en cas de vérification de l'indépendance entre les variables aléatoires qui est une hypothèse forte d'où le nom de "naïf". Elle est contrôlée par la matrice de corrélation et ses valeurs propres. Les attributs ou variables aléatoires doivent suivre une loi normale. C'est une technique simple et moins couteuse en temps de calcul. Son avantage est que l'arrivée d'une nouvelle information ne nécessite pas la reprise de tous les calculs pour la prendre en considération.

Pour construire un classifieur naïf de Bayes, on suppose que, pour chaque classe $c \in \{1, .., m\}$, étant donné c, les composantes de X sont indépendantes. On modélise donc les densités univariées. On construit ensuite un classifieur de Bayes à partir des estimateurs ainsi obtenus. Lorsqu'on n'a pas beaucoup de classes, on peut construire un classifieur de Bayes en estimant la probabilité conjointe et en calculant la probabilité conditionnelle de la classe C [48, 26]. On dispose de J variables et K classes sur un échantillon d'apprentissage de n individus. Soient $N = (X_1, X_2, ..., X_J)$, l'ensemble des variables décrites (descripteurs), Y la variable cible (l'attribut ou classe comportant K modalités). L'objectif est de prédire l'appartenance d'un individu à une catégorie k selon l'information sur les j descripteurs $j = 1, 2, ..., J$. Cela se fait par la règle bayésienne d'affectation optimale. Ainsi, classer un individu i revient à maximiser la probabilité a posteriori d'appartenance

aux classes $k = 1, 2, ..., K$:

$$\hat{y}(\omega) = y_{k^*} \Leftrightarrow y_{k^*} = \arg\max_k \mathrm{P}\left[\mathrm{Y} = y_k | N(\omega)\right] \tag{3}$$

Cette maximisation est une estimation viable de la probabilité conditionnelle $\mathrm{P}[\mathrm{Y}|X]$

$$\mathrm{P}\left[\mathrm{Y} = y_k | N(\omega)\right] = \frac{\mathrm{P}\left[\mathrm{Y} = y_k\right] \mathrm{P}\left[N(\omega) | \mathrm{Y} = y_k\right]}{\mathrm{P}[N(\omega)]} \tag{4}$$

Etant donné que $P[N(\omega)]$ est indépendant de k donc la maximasition de la probablité a posteriori donne :

$$\hat{y}(\omega) = y_{k^*} \Leftrightarrow y_{k^*} = \arg\max_k \mathrm{P}\left[\mathrm{Y} = y_k\right] * \mathrm{P}\left[N(\omega) | \mathrm{Y} = y_k\right] \tag{5}$$

$\mathrm{P}\left[\mathrm{Y} = y_k\right]$ est la proportion de chaque modalité de la variable cible.

En pratique, on utilise la «m probabilité» estimate pour lisser l'effet des petits effectifs.

Ainsi pour un échantillon de n individus, cette probabilité est donnée par :

$$\widehat{P}\left[\mathrm{Y} = y_k\right] = p_k = \frac{n_k + m}{n + m * K} \tag{6}$$

pour $k = 1, 2, ..., K$.

Usuellement, on fixe $m = 1$ et on obtient l'estimateur Laplacien des probabilités [7].

Par indépendance des descripteurs nous avons :

$$\mathrm{P}\left[N(\omega) | \mathrm{Y} = y_k\right] = \prod_{j=1}^{J} \mathrm{P}\left[Xj | \mathrm{Y} = y_k\right] \tag{7}$$

Lorsque le nombre de variables est élevé, nous passons au logarithme pour éviter les débordements de capacité surtout avec la présence de valeurs inférieures à 1. Donc la règle d'affectation devient :

$$\hat{y}(\omega) = y_{k^*} \Leftrightarrow y_{k^*} = \arg\max_k (\log \mathrm{P}\left[\mathrm{Y} = y_k\right] + \sum_{j=1}^{J} \log \mathrm{P}\left[X_j(\omega) | \mathrm{Y} = y_k\right]). \tag{8}$$

La classification bayésienne peut être traité sous l'angle de la fonction de perte $0/1$.

1.3 Classification sous la fonction de perte

Dans les problèmes de classification, on suppose que la nature prend des valeurs sur un ensemble discret S et que le but est de décider quel est le véritable état de la nature (c'est-à-dire que nous prenons $A = S$), étant donné l'observation x [14] .

On connaît également $f_X(x \mid s)$, pour $s \in S$, ce qui dans ce contexte est généralement appelé le modèle d'observation conditionnel à la classe. Les observations réelles sont un échantillon de la vraie, c'est-à-dire de $f_X(x \mid s \text{ vraie})$). L'antériorité est ici une fonction de masse de probabilité (puisque S est discret), $p_S(s)$, pour $s \in S$, et le dénominateur du théorème de Bayes apparaît alors dans sa version discrète :

$$f_X(x) = \sum_{s \in S} f_X(x \mid s) p(s) \tag{9}$$

On note que ceci est vrai, lorsque $f_X(x)$ et $f_X(x \mid s)$ soient des densités de probabilité (X continu) ou des fonctions de masse (X discret).

La fonction de perte 0/1, pour les problèmes de classification, attribue un coût nul à toute décision correcte, et un coût unitaire à toute décision erronée. Elle s'écrit comme suit :

$$L(s, a) = \begin{cases} 1 & \text{si s } \neq a \\ 0 & \text{si s } = a \end{cases} \tag{10}$$

En insérant cette définition dans l'expression générale de la perte attendue a posteriori puis à rechercher la décision optimale de Bayes nous obtenons :

$$\begin{aligned} \delta(x) &= \arg\min_{d \in A} \sum_{s \in S} L(s, d) p_S(s \mid x) \\ \delta(x) &= \arg\min_{d \in A} \left(\sum_{S \in S} p_S(s \mid x) - p_S(d \mid x) \right) \\ \delta(x) &= \arg\min_{S \in S} \left(1 - p_S(s \mid x)\right) \\ \delta(x) &= \arg\max_{s \in S} p_S(s \mid x) \equiv \delta_{MAP}^{(x)} \end{aligned} \tag{11}$$

que l'on appelle le classifieur a posteriori maximum (MAP). En d'autres termes, la

règle de décision optimale de Bayes consiste à choisir la classe présentant la probabilité a posteriori maximale, compte tenu de l'observation particulière en question.

Étant donné que, pour une observation donnée x, la marge $f_X(x)$ au dénominateur du théorème de Bayes est une constante, le critère MAP peut être simplifié comme suit :

$$\delta_{MAP}(X) = \arg\max_{s \in S} f_X(x \mid s) p_S(s)$$

qui est équivalent au maximiseur de la fonction de probabilité conjointe soit :

$$\delta_{MAP}(X) = \arg\max_{S \in S} f_{X,S}(X, S) \tag{12}$$

Il est également fréquent de voir la version logarithmique du critère MAP :

$$\delta_{MAP}(X) = \arg\max_{s \in S} \{\log f_X(x \mid s) + \log p_S(s)\} \tag{13}$$

Cette forme, probablement meilleure que I'équation précédente, montre clairement que la règle de décision MAP tente d'atteindre un compromis entre les attentes a priori portées par $p_S(s)$ et l'évidence fournie par les données via la fonction de vraisemblance $f_X(x \mid s)$.

Dans les problèmes binaires (à deux classes), lorsque A = S = $\{s_1, s_2\}$, il est possible d'aller un peu plus loin que la forme générale du classifieur MAP et d'écrire :

$$\delta_{MAP}(x) = \begin{cases} s_1, l(x) \geq t \\ s_2, l(x) < t \end{cases} \tag{14}$$

Avec

$$l(x) = \frac{f_X\left(x \mid s_1\right)}{f_X\left(x \mid s_2\right)}$$

$$t = \frac{p_S\left(s_2\right)}{p_S\left(s_1\right)}$$

où $l(x)$ est appelé rapport de vraisemblance et t est un seuil de décision.

Un exemple classique de décision binaire, dans lequel la fonction de perte "0/1" est utilisée est le problème de détection binaire, tel qu'il est connu dans la littérature sur les communications. Dans sa forme la plus simple, un émetteur émet une des deux valeurs constantes, disons s_0 et s_1, représentant respectivement les chiffres binaires "0" et "1" ; leurs probabilités a priori (connues) sont respectivement p_0 et $p_1 = 1 - p_0$. Chaque chiffre est transmis par un canal bruyant qui lui ajoute un échantillon d'une variable aléatoire gaussienne de moyenne nulle et de variance σ^2 ; c'est le modèle le plus simple, mais de loin le plus courant, du bruit de canal dans les communications numériques. Les conditionnels de classe correspondants de la valeur reçue x, sont alors $f_X(x \mid s_0) = \mathrm{N}(x \mid s_0, \sigma^2)$ et $f_X(x \mid s_1) = \mathrm{N}(x \mid s_1, \sigma^2)$. En raison de la nature exponentielle de la fonction de densité de probabilité gaussienne, il est plus pratique de travailler ici avec la version logarithmique qui, après de simples manipulations, donne :

$$\delta_{MAP}(x) = \begin{cases} s_0, l(x) \geq t \\ s_1, l(x) < t \end{cases} \tag{15}$$

avec

$$l(x) = (x - s_1)^2 - (x - s_0)^2$$

$$t = 2\sigma^2 \log\left(\frac{p(s_1)}{p(s_2)}\right)$$

On note que $l(x)$ mesure la différence entre les distances au carré de la valeur observée à s_1 et s_2.

1.3.1 Observation gaussienne univariée avec une variance commune

Une famille de problèmes de classification exhaustivement étudiée est celle où les conditionnels de classe (les vraisemblances) sont des densités gaussiennes (éventuellement multivariées).

Pour un problème à M classes, avec $S = \{s_1, s_2, \ldots, s_M\}$, si les observations sont des vecteurs à n dimensions (i.e., $X = I\mathbb{R}^n$), les vraisemblances gaussiennes sont les suivantes :

$$f_X\left(x \mid s_i\right)=\frac{1}{\sqrt{(2\pi)^n \det\left(C_i\right)}} \exp\left\{-\frac{1}{2}\left(\mathrm{x}-\mu_i\right)^T C_i^{-1}\left(\mathrm{x}-\mu_i\right)\right\} \tag{16}$$

$$\delta_{MAP}(x)=\begin{cases} s_0, l(x) \geq t \\ s_1, l(x) < t \end{cases}$$

$$l(x)=\left(x-s_1\right)^2-\left(x-s_0\right)^2$$

$$t=2\sigma^2 \log\left(\frac{p\left(s_1\right)}{p\left(s_2\right)}\right)$$

Dans l'équation T désigne la transposition vectorielle et det (C_i) représente le déterminant d'une matrice ; chaque μ_i est le vecteur moyen à n dimensions correspondant à la classe S_i,

$$\mu_i=E_X\left[X \mid s_i\right]$$

et chaque C_i est la matrice de covariance associée aux observations de la classe s_i, définie comme

$$C_i=E_X\left[\left(\mathrm{x}-\mu_i\right)\left(\mathrm{x}-\mu_i\right)^T \mid s_i\right]$$

Cette forme particulière permet de réécrire le classifieur MAP de l'équation, en utilisant les logarithmes naturels et en supprimant les constantes, comme suit :

$$\delta_{MAP}(X)=\arg\max_{S \in S}\left\{\log p_S\left(s_i\right)-\log\det\left(C_i\right)-\left(\mathrm{x}-\mu_i\right)^T C_i^{-1}\left(\mathrm{x}-\mu_i\right)\right\} \tag{17}$$

Cette forme peut être encore simplifiée dans les deux cas particuliers suivants.

Un scénario où seule la moyenne est supposée différer d'une classe à I'autre, et que les observations sont indépendantes avec une variance égale à σ^2, est décrit en prenant $C_i=\sigma^2$, pour i $=1,2,\ldots,$ M (où I désigne une matrice d'identité). On remarque qu'un tel modèle résulte si nous supposons que, étant donné certaines classes s_i, le modèle d'observation est $X=\mu_i+N$, où N est un vecteur n-dimensionnel de gaussiens indépendants de moyenne nulle. II est clair que les termes log det (C_i) sont tous égaux et peuvent être

supprimés de l'équation. De plus, $C^{-1} = 1/\sigma^2$, ce qui donne le classifieur plus simple suivant :

$$\delta_{MAP}(X) = \arg\max_{s_i \in S} \left\{ 2\sigma^2 \log p_S(s_i) - (\mathrm{x} - \mu_i)^T (\mathrm{x} - \mu_i) \right\}$$
$$\delta_{MAP}(X) = \arg\min_{s_i \in S} \left\{ -2\sigma^2 \log p_S(s_i) + \|\mathrm{x} - \mu_i\|^2 \right\}$$

où $\|\mathrm{x} - \mu_i\|^2$ désigne le carré de la distance euclidienne entre x et μ_i. Cette forme montre clairement que le classifieur trouve un compromis entre la sélection de la classe dont la moyenne μ_i est la plus proche de l'observation x et la probabilité a priori de cette classe. Nous pouvons alors réduire le classifieur MAP à sa forme la plus simple.

$$\delta_{MAP}(X) = \arg\max_{s_i \in S} \left\{ \mathrm{X}^T + \sigma^2 \log p_S(s_i) + \|\mu_i\|^2 / 2 \right\} \tag{18}$$

1.3.2 Observations dépendantes avec covariance commune

Dans cette prochaine configuration la plus simple, on suppose que bien que les observations ne soient pas indépendantes les unes des autres, toutes les matrices de covariance sont les mêmes, $C_i = \mathrm{C}$, $i = \{1, 2, \ldots \mathrm{M}\}$.

Puisque $C_i = \mathrm{C}$, il est encore possible de laisser tomber le terme (constant) $\log \det(C_i)$ et d'écrire :

$$\begin{aligned} \delta_{MAP}(X) &= \arg\max_{S \in S} \left\{ 2 \log p_S(s_i) - (\mathrm{x} - \mu_i)^T C^{-1} (\mathrm{x} - \mu_i) \right\} \\ \delta_{MAP}(X) &= \arg\min_{S \in S} \left\{ -2 \log p_S(s_i) + \|\mathrm{x} - \mu_i\|_C^2 \right\} \end{aligned} \tag{19}$$

Avec $\|\mathrm{x} - \mu_i\|_C^2 = (\mathrm{x} - \mu_i)^T C^{-1} (\mathrm{x} - \mu_i)$ appelée la distance de Mahalanobis au carré.

1.3.3 Observations dépendantes dont les covariances diffèrent par un facteur d'échelle

II s'agit d'une légère variation du dernier cas. Considérons que les matrices de covariance des classes ne diffèrent les unes des autres que par un certain facteur d'échelle, c'est-à-dire que nous pouvons écrire $C_i = \gamma_i \mathrm{C}$.

Dans ce cas, on a :

$$\log\det(C_i) = \log\det(\gamma_i \mathrm{C}) = \mathrm{n}\log\det\gamma_i + \log\det(\mathrm{C}) \tag{20}$$

Puisque log det(C) est une constante, nous obtenons toujours un classifieur linéaire de la forme :

$$\begin{gathered} \delta_{MAP}(X) = \arg\max_{s_i \in S} \left\{ \mathrm{x}^T \alpha_i + \beta_i \right\}. \\ \alpha_i = C^{-1}\mu_i \\ \beta_i = \gamma_i \log p_S(s_i) - \frac{1}{2}\|\mu_i\|_C^2 - \frac{1}{2} n\gamma_i \log\gamma_i \end{gathered} \tag{21}$$

1.4 Classification des données incertaines

De nombreux travaux ont été réalisés pour la classification des données de manière générale. La littérature nous montre que diverses méthodes ont été proposées comme l'arbre de décision [9], les classifications basées sur des règles [19] etc. Cependant, la classification des données incertaines aussi appelées données symboliques [15, 39] reste un défi. C'est dans ce cadre que cette étude sur la construction d'un classifieur des données incertaines a été menée.

On note la méthode proposée par l'article de Biao Qin et al.[11] qui propose une nouvelle méthode pour classer et prédire directement les données dites incertaines. On appelle données incertaines des données découlant des erreurs de mesure, de décision, et de transmission peu fiable des données. La méthodologie consiste à calculer de manière différente les probabilités conditionnelles afin que le classifieur naïf de Bayes puisse traiter les données incertaines. Les résultats expérimentaux montrent que le nouveau classifieur classe les données incertaines avec des précisions potentiellement plus élevées que le classifieur naïf de Bayes. Dans ce cadre aussi, on utilise les distributions de probabilité pour calculer la distance entre deux objets incertains. Cormode et al [5] proposent une version pondérée sur des données avec incertitude. Xia et al [54] présentent un nouvel algorithme de regroupement conceptuel spécifiquement pour les données catégorielles incertaines.

Concernant, la classification par arbre de décision, en présence de valeurs manquantes ou bruitées, la stratégie utilisée par Dayanik [22] consiste à présenter des algorithmes d'apprentissage d'intervalles de caractéristiques qui représentent des valeurs multidimensionnelles. Ces algorithmes d'apprentissage d'intervalles de caractéristiques représentent des descriptions multi-concepts sous la forme d'intervalles de caractéristiques disjoints. L'incertitude n'est pas représentée par des valeurs manquantes ou erronées, mais par des intervalles incertains avec des fonctions de distribution de probabilité [18]. Tsang et al [52] et Qin et al [45] ont développé indépendamment des classifications par arbre de décision pour les données incertaines. En effet, Tsang et al [52] ont converti chaque valeur numérique en un ensemble de s points d'échantillonnage entre l'intervalle incertain [a_j,b_j] avec la valeur associée f(x), en approximant effectivement chaque f(x) par une distribution discrète. Qin et al.[45] ont également proposé une classification basée sur des règles. Le problème clé de l'apprentissage des règles est d'identifier efficacement les points de coupure optimaux à partir des données d'apprentissage.

Le classifieur bayésien naïf estime la probabilité conditionnelle de classe en supposant que les attributs sont conditionnellement indépendants, étant donné l'étiquette de classe C_k. Supposons qu'il y ait n classes, $C_1, C_2, \ldots, C_n$, l'hypothèse d'indépendance conditionnelle peut être formellement énoncée comme suit :

$$P\left(X \mid C_k\right) = \prod_{i=1}^{m} P\left(X_i \mid C_k\right) \qquad k = 1, ..., n \tag{22}$$

où chaque ensemble d'attributs X = $\{X_1, X_2, \ldots, X_m\}$ est constitué de m attributs.

Avec l'hypothèse d'indépendance conditionnelle, au lieu de calculer la probabilité conditionnelle de classe pour chaque combinaison de X, nous devons seulement estimer la probabilité conditionnelle de chaque X_i étant donné C_k. Cette dernière approche est plus pratique car elle ne nécessite pas un très grand ensemble d'apprentissage pour obtenir une bonne estimation de la probabilité [18]. Pour classifier un enregistrement de test, le classifieur bayésien naïf calcule la probabilité postérieure pour chaque classe C_k comme suit :

$$P\left(C_k \mid X\right) = \frac{P\left(C_k\right) \prod_{i=1}^{m} P\left(X_i \mid C_k\right)}{P(X)} \tag{23}$$

Étant donné que $P(X)$ est fixe pour chaque C_k, il suffit de choisir la classe qui maximise le terme du numérateur, $P\left(C_k\right) \prod_{i=1}^{m} P\left(X_i \mid C_k\right)$. Notons que dans chaque classe C_k, les valeurs des attributs numériques sont normalement distribuées. Avec les données incertaines, une nouvelle approche de calcul des probabilités conditionnelles est adoptée. Lorsque la valeur d'un attribut numérique est incertaine, l'attribut est appelé attribut numérique incertain [18]. Notons A_{ij} la j ième instance de A_i où A_i qui est représentée comme une plage ou un intervalle [14, 22, 39] et la fonction de distribution de probabilité notée $f(x)$ sur cette plage [19]. A_i est traitée comme une variable aléatoire continue. La fonction $f(x)$ peut être liée d'un attribut si toutes les instances ont la même distribution, ou liée à chaque instance si chaque instance a des distributions différentes. Pour une valeur d'attribut numérique incertaine $[A_{ij} \cdot \mathrm{a}, A_{ij}.\mathrm{b}]$, μ_j peut être estimé comme (a+b)/2 puisqu'il n'y a pas d'erreur biaisée ; et σ_j peut être estimé comme (b-a)/6 puisque la probabilité qu'une valeur se situe dans la moyenne plus ou moins 3 écarts-types est supérieure à 99% pour une variable aléatoire normale [56]. Un intervalle incertain de A_{ij}, noté A_{ij}.U, est un intervalle $[A_{ij}.\ \mathrm{a},\ A_{ij} \cdot \mathrm{b}]$ où $A_{ij} \cdot \mathrm{b} \geq A_{ij}.\mathrm{a}$. Une fonction de distribution de probabilité incertaine de A_{ij} est noté $A_{ij}.f(x)$. On suppose qu'une variable aléatoire d'intervalle Z a m instances et que chaque instance peut être observée avec la même probabilité $\frac{1}{m}$, la fonction de densité empirique de Z est :

$$f(x) = \frac{1}{m} \sum_{j=1}^{m} f_j(x) \tag{24}$$

où $f_j(x)$ désigne la fonction de densité de la j ième instance de Z.

Si un attribut A est numérique, on suppose souvent qu'il a une distribution gaussienne pour la classification bayésienne naïve on a :

$$P\left(A \mid C_k\right) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{\left(x-\mu_k\right)^2}{2\left(\sigma_k\right)^2}} \tag{25}$$

Nous supposons qu'une variable aléatoire Z à valeur d'intervalle suit une distribution

gaussienne ainsi que chaque instance z_j dans son propre intervalle. Le théorème suivant traite de la manière de calculer les paramètres de la distribution gaussienne de la variable aléatoire Z basée sur un intervalle.

Théorème 1 : Pour une variable aléatoire Z à valeurs d'intervalle, ses valeurs observées sont $[a_j,\, b_j\,]$ pour $j = 1, \ldots, m$.

Soit A et B des variables aléatoires représentant les valeurs minimales et maximales de l'intervalle d'échantillonnage ; alors Z suit la distribution de :

$$P(Z \mid (A, B)) \sim N\left(\mu_{\frac{A+B}{2}}, S^2_{\frac{A+B}{2}} + \Delta\right) \tag{26}$$

$$\text{Avec } \Delta = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{b_j - a_j}{6}\right)^2$$

Pour chaque variable incertaine Z, sa valeur observée est de la forme $[a_j, b_j]$. Supposons que sa vraie valeur soit z_j, alors $a_j \leq z_j \leq b_j$. Nous désignons l'erreur de gauche (erreur négative) par ϵ_{1j} et l'erreur de droite (erreur positive) avec ϵ_{2j} et l'erreur globale par ϵ_j. Comme les erreurs courantes, telles que l'imprécision des mesures, peuvent généralement être approximées par des distributions gaussiennes. Supposons que l'erreur de gauche est une distribution gaussienne de moyenne μ et de variance σ_1^2 c'est-à-dire $\epsilon_{1j} \sim (\mu, \sigma_1^2)$. De même, nous supposons que l'erreur de droite est une gaussienne de moyenne μ et de variance $\sigma_2^2, \varepsilon_{2j} \sim (\mu, \sigma_2^2)$, et que l'erreur globale est une distribution gaussienne de moyenne 0 et de variance $\sigma^2, \epsilon \sim (0, \sigma^2)$. Nous nous concentrons ici sur l'erreur statistiquement aléatoire, qui est le type d'erreur le plus courant dans la pratique ; nous supposons donc que l'erreur de gauche et l'erreur de droite ont la même moyenne μ, et que l'erreur globale a une moyenne 0. Si l'erreur est statistiquement biaisée négativement ou positivement, les hypothèses et les calculs peuvent être facilement ajustés. Le théorème suivant traite de la manière de calculer les paramètres de la distribution gaussienne de la variable aléatoire Z basée sur l'erreur [11].

Théorème 2 : Supposons qu'un attribut numérique incertain Z satisfasse la distribution gaussienne. Soit A et B la variable aléatoire représentant les valeurs minimales et

maximales de l'intervalle d'échantillonnage respectivement ; $\epsilon_{1j}, \epsilon_{2j}$ et ϵ désignent l'erreur de gauche (erreur négative), l'erreur de droite (erreur positive) et l'erreur globale [11].

Supposons que $\epsilon_{1j} \sim N(\mu, \sigma_1^2), \epsilon_{2j} \sim N(\mu, \sigma_2^2), \epsilon \sim N(0, \sigma^2), \epsilon_{1j}$ et ϵ_{2j} sont indépendants, alors Z satisfait à la distribution de :

$$P(Z \mid (A, B)) \sim N\left(\mu_{\frac{A+B}{2}}, S^2_{\frac{A+B}{2}} - \Delta\right) \tag{27}$$

$$\text{Avec } \Delta = \frac{S^2_{B-A}}{2}$$

1.5 La méthode Autoclass

AutoClass détermine le nombre le plus probable de classes présentes dans les données, les descriptions les plus probables de ces classes et la probabilité d'appartenance de chaque objet à chaque classe. AutoClass est un programme de classification automatique qui recherche des classes dans les données en utilisant des techniques de statistiques bayésiennes. Il définit les classes non pas comme des partitions des données mais comme des descriptions probabilistes des processus représentés dans les données. A partir de ces descriptions, on peut déterminer la probabilité que chaque objet soit membre de chaque classe. AutoClass détermine automatiquement le nombre le plus probable de classes. Les classes trouvées représentent la structure réelle des données. Pour des données aléatoires, AutoClass découvre une seule classe. Le théorème de Bayes est tout ce qui est nécessaire pour effectuer la classification. Aucune mesure de similarité ad hoc, aucune règle d'arrêt ou aucun critère de qualité de regroupement n'est nécessaire. La théorie de la décision s'applique directement aux distributions de probabilité calculées par AutoClass. La classification est probabiliste. Les descriptions de classes et les affectations d'objets aux classes sont données sous forme de distributions de probabilités. Les classes «floues» qui en résultent illustrent mieux qu'une classification catégorielle la notion de sens commun de l'appartenance à une classe. Les attributs à valeurs réelles et discrètes peuvent être librement mélangés, et toutes les valeurs d'attributs peuvent être manquantes. Les

attributs «à valeur arborescente» peuvent également être facilement incorporés dans le modèle AutoClass. Les classifications sont invariantes aux changements d'échelle ou d'origine des données. Lors de la classification d'une base de données, AutoClass calcule plutôt des descriptions probabilistes de classes qui rendent compte des données observées. Pour trouver des classes sur un jeu de données, des déclarations explicites sur la distribution des données sous forme de modèle de classe probabiliste paramétrée est fournie. Ainsi, nous pouvons trouver la probabilité que les données proviennent d'un tel ensemble de classes à l'aide de formules de probabilité simples. La classification a longtemps été étudiée en ces termes comme la théorie des mélanges finis. AutoClass est une implémentation de la solution bayésienne au problème des mélanges finis. Nous commençons par une distribution de probabilité antérieure non formative sur les paramètres de classification, puis nous mettons à jour cette distribution en utilisant les informations de la base de données pour calculer la distribution de probabilité postérieure des paramètres. Cette distribution postérieure nous permet de déterminer à la fois les paramètres de classification les plus probables pour un nombre donné de classes ainsi que le nombre le plus probable de classes présentes dans les données. A partir de ces informations, il est également possible de calculer la probabilité que chaque objet soit un membre de chaque classe. Le programme AutoClass recherche des classes dans lesquelles les attributs varient indépendamment au sein d'une classe. Il modélise les attributs à valeurs réelles par des distributions de probabilité gaussiennes et les attributs discrets par des listes de probabilités de résultats. AutoClass utilise une variante bayésienne de l'algorithme EM de Dempster et Laird [10] pour rechercher le maximum de la distribution postérieure des paramètres de classification et approximer la distribution autour de ce maximum. AutoClass inclut également des techniques heuristiques pour éviter les maxima locaux dans la recherche. Bien que ces maxima posent un problème difficile dans la pratique, il s'agit d'un problème algorithmique qui ne nécessite aucune théorie additionnelle. AutoClass utilise une antériorité informative faible qui introduisent peu de biais qui pourrait être facilement étendu pour inclure des connaissances préalables fortes, si elles sont disponibles, alors que de nombreuses approches non bayésiennes auraient des difficultés à incorporer de telles connaissances sans heurts. Cette méthode peut être uti-

lisée pour apprendre à partir d'exemples. Si le programme reçoit un ensemble d'objets pré-classifiés par un enseignant, il peut former des descriptions des classes spécifiées et les utiliser pour classifier de nouveaux objets. En outre, il peut estimer les valeurs des paramètres manquants de sa classification en se basant sur les valeurs présentes. Ainsi, l'apprentissage supervisé peut être combiné avec l'apprentissage non supervisé dans le même système, en utilisant la même théorie. Le programme fonctionne aussi bien, voire mieux, que d'autres systèmes de classification automatique lorsqu'il est exécuté sur les mêmes données. AutoClass a classé des données fournies par des chercheurs actifs dans divers domaines et a donné des résultats fascinants. Par exemple, les données de Fisher sur trois espèces d'iris constituent un test classique pour les systèmes de classification. AutoClass trouve les trois classes présentes dans les données avec un degré de confiance très élevé [21].

Chapitre 2

Réseaux bayésiens

Les réseaux bayésiens ont été développés au début des années 1980 pour tenter de résoudre certains problèmes de prédiction et d'abduction, courants en intelligence artificielle. Ils s'appuient sur le théorème de Bayes qui est un résultat de base en théorie des probabilités issu des travaux du révérend Thomas Bayes (1702-1761), présenté à titre posthume en 1763. Les réseaux bayésiens également appelés réseaux de croyance sont des modèles probabilistes graphiques. Ces derniers sont composés de nœuds qui représentent les variables aléatoires et d'arcs aditionnels qui sont les corrélations entre ces variables. Elle est également la représentation graphique de la distribution conjointe. Deux variables aléatoires sont reliées par un arc si l'un cause ou influe sur l'autre : le prédécesseur est la cause et le successeur est l'effet. Les réseaux bayésiens sont appliqués pour le diagnostic médical et industriel, l'analyse de risques, la détection de spams, le datamining, la détection de fraudes, l'exploitation du retour d'expérience, la modélisation et la simulation de systèmes complexes, etc.

Selon le type d'application, l'utilisation pratique d'un réseau bayésien peut être envisagée au même titre que celle d'autres modèles : réseau de neurones, système expert, arbre de décision, modèle d'analyse de données (régression linéaire), arbre de défaillances, modèle logique. Le choix de ces réseaux est motivé par certains aspect comme la facilité, le coût et le délai de mise en oeuvre d'une solution. En outre, les critères suivants font que les réseaux bayésiens sont préférables à d'autres modèles :

— Acquisition des connaissances

La possibilité de rassembler et de fusionner des connaissances de diverses natures

dans un même modèle : retour d'expérience (données historiques ou empiriques), expertise (exprimée sous forme de règles logiques, d'équations, de statistiques ou de probabilités subjectives).

— Représentation des connaissances.

La représentation graphique d'un réseau bayésien est explicite, intuitive et compréhensible par un non spécialiste, ce qui facilite à la fois la validation du modèle, ses évolutions éventuelles et surtout son utilisation.

— Utilisation des connaissances.

Un réseau bayésien est polyvalent : on peut se servir du même modèle pour évaluer, prévoir, diagnostiquer, ou optimiser des décisions, ce qui contribue à rentabiliser l'effort de construction du réseau bayésien.

2.1 Définition d'un réseau bayésien

Un réseau bayésien est défini par :

— un graphe acyclique dirigé (DAG en anglais) G = (V, E), où X est l'ensemble des nœuds de G, et E l'ensemble des arcs de G ;

— un espace probabilisé fini (Ω,Z, p) ;

— un ensemble de variables aléatoires associées aux nœuds du graphe et définies sur (Ω,Z, p), tel que :

On peut exprimer en écrivant la distribution conjointe comme suit :

$$\boxed{p(X_1, X_2, ..., X_l) = \prod_{i=1}^{l} = p(X_i|Pa_i)} \tag{28}$$

où Pa_i désigne le sous-ensemble de variables associé à la variable aléatoire X_i.

Prenons l'exemple suivant :

$$p(X_1, X_2, X_3, X_4, X_5, X_6) = p(X_6|X_4)p(X_5|X_3, X_4)p(X_4|X_1, X_2)p(X_3|X_1)p(X_2)p(X_1). \tag{29}$$

Alors $Pa_6 = \{X_4\}$, $Pa_5 = \{X_3, X_4\}$, $Pa_4 = \{X_1, X_2\}$, $Pa_3 = \{X_1\}$, $Pa_2 = \varnothing$, $Pa_1 = \varnothing$.

Les variables de l'ensemble Pa_i sont définies comme les parents de X_i, et d'un point de vue statistique, cela signifie que X_i est statistiquement indépendante de toutes les variables étant donné les valeurs de ses parents. Chaque $p(X_i|Pa_i)$ exprime une relation d'indépendance conditionnelle et impose une structure probabiliste qui sous-tend notre ensemble multivarié.

Un réseau bayésien est représenté par un graphe acyclique orienté (DAG) et par une table de probabilité conditionnelle pour chaque nœud X_i. Le réseau bayésien correspondant à l'équation 29 peut être représenté graphiquement de la sorte :

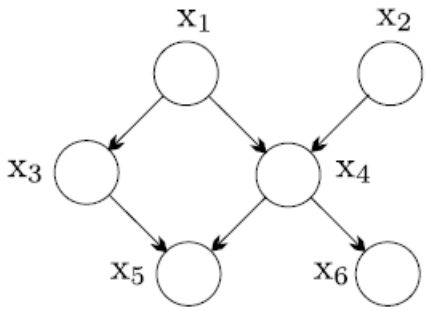

FIGURE 1 – Structure d'un réseau bayésien

Dans ce réseau, chaque nœud représente une variable aléatoire discrète. Les arcs indiquent une corrélation entre les 2 variables. Un réseau bayésien (BN) est une paire (G, p), où la distribution p factorise le DAG G en termes d'un ensemble de distributions de probabilités conditionnelles, associées aux nœuds de G. En d'autres termes, un BN est associé à une distribution spécifique. L'intérêt particulier des réseaux bayésiens est de tenir compte simultanément de connaissances a priori d'experts (dans le graphe) et de l'expérience contenue dans les données.
La construction d'un réseau bayésien revient donc en résumé à définir le graphe du modèle appelé "structure" du modèle et les tables de probabilités de chaque variable qui sont les paramètres.

2.2 Inférence

Dans ce type de tâche, il est nécessaire de trouver une interprétation cohérente des observations avec les données connues a priori. L'inférence probabiliste signifie donc le calcul de $P(Y|X)$ où X est un ensemble d'observations et Y un ensemble de variables décrivant le problème et qui sont jugées importantes pour la prédiction ou le diagnostic.

Lorsque les variables aléatoires impliquées sont discrètes, les probabilités conditionnelles $P(x_i|Pa_i), i = 1, ...l$, associées aux nœuds d'une structure de graphe bayésien doivent être apprises à partir des données d'apprentissage. Si le nombre d'états possibles et/ou le nombre de variables dans Pa_i sont suffisamment grands, cela équivaut à un grand nombre de probabilités qui doivent être apprises. Ainsi, un grand nombre de points d'apprentissage est nécessaire pour obtenir de bonnes estimations. Il est possible de remédier à ce problème en exprimant les probabilités conditionnelles sous une forme paramétrique :

$$P(x_i|Pa_i) = P(x_i|Pa_i; \theta_i), \quad i = 1, 2, ..., l \tag{30}$$

θ_i représente la probabilité du nœud X_i conditionnellement à l'état de ses parents.
Dans le cas de variables à valeurs binaires, une forme fonctionnelle courante consiste à considérer P comme un modèle de régression logistique.

$$P(x_i = 1|Pa_i; \theta_i) = \sigma(t_i) = \frac{1}{1 + \exp(-t_i)}, \quad i = 1, 2, ..., l \tag{31}$$

$$t_i = \theta_i + \sum_{k:x_k \in Pa_i} \theta_{ik} x_k. \tag{32}$$

Cela réduit le nombre de vecteurs de paramètres à apprendre à O(l). Le nombre exact de paramètres dépend de la taille des ensembles de parents. En supposant que le nombre maximum de parents pour un nœud est K, le nombre inconnu de paramètres à apprendre à partir des données d'apprentissage est inférieur ou égal à lk. En tenant compte de la nature binaire des variables, nous pouvons écrire :

$$P(x_i = 1|Pa_i;\theta_i) = x_i\sigma(t_i) + (1 - x_i)(1 - \sigma(t_i)), \tag{33}$$

Ces modèles sont également connus sous le nom de réseaux bayésiens sigmoïdaux et ont été proposés comme un type de réseau neuronal.

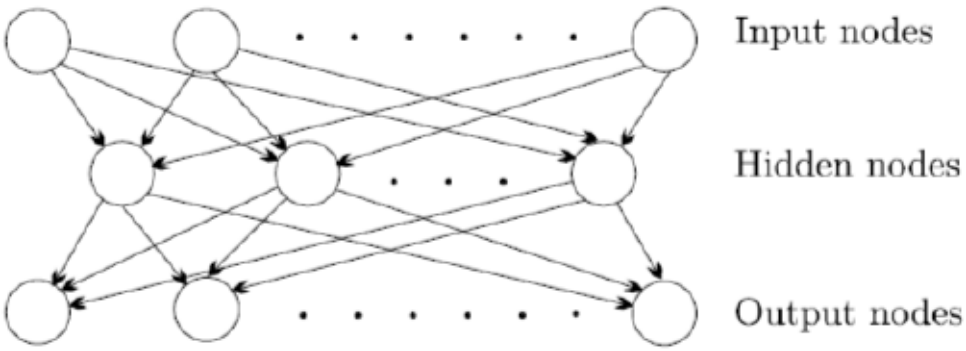

FIGURE 2 – Réseau bayésien sigmoidal [51]

La figure 2 présente la structure graphique d'un tel réseau. Le réseau peut être traité comme une structure de réseau bayésien en associant une variable binaire à chaque nœud et en interprétant les activations des nœuds comme des probabilités, comme le dicte l'équation (33). La réalisation de l'inférence et de l'apprentissage des paramètres dans de tels réseaux n'est pas une tâche facile. Nous devons recourir à des approximations.

2.3 Apprentissage des paramètres

Dans le cadre des RB, l'apprentissage des paramètres consiste à rechercher le meilleur jeu de paramètres qui sont les différentes probabilités conditionnelles utilisées dans le graphe pour rendre compte des données observées.

Soit un réseau bayésien constitué des nœuds $\{X_1, X_2, ..., X_n\}$, chaque nœud est supposé prendre des valeurs discrètes. Soit également une base d'exemples D constituée de la mesure de chacune des X_i pour un certain nombre d'exemples N. Si X_i est un nœud, on

note :

— r_i le nombre de ses états possibles ;

— C_i l'ensemble de ses parents, dont l'ensemble des états possibles est indexé par j

On note également θ_{ijk} la probabilité pour que X_i soit dans l'état k, conditionnellement au fait que l'ensemble de ses parents soit dans l'état j. Si nous supposons de plus les hypothèses fortes suivantes :

— La base d'exemples D est effectivement produite par un réseau de structure donnée, notée B_s.

— Les paramètres θ_{ijk} sont indépendants entre eux, et ils sont distribués suivant une loi de Dirichlet. On peut alors montrer que :

$$E(\theta_{ijk}|D, B_s) = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}} \tag{34}$$

où :

N_{ijk} est le nombre d'exemples dans la base D, tels que X_i est dans l'état k alors que ses parents sont dans l'état j.

N_{ij} est le nombre d'exemples dans la base D, tels que les parents du nœud X_i sont dans l'état j, indépendamment de l'état de ce nœud.

α_{ijk} est l'exposant du paramètre θ_{ijk} dans la distribution de Dirichlet initiale, et $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$.

2.4 Etude pratique, avantages, inconvénients

Pour l'aspect pratique, nous nous pencherons sur l'article de Charon et al. [6] qui propose une classification pour la prédiction d'évènements de santé défavorable en établissement d'hébergement pour personnes agées dépendantes.

Cette étude a été réalisée avec la librairie Python pyAgrum, une boîte à outils pour construire des modèles et des algorithmes pour les modèles graphiques probabilistes [23]. Trois jeux de données ont été utilisés et proviennent de 3000 établissements d'hébergement et renseignent avec différentes temporalités pour les cas positifs 1 mois avant le premier

épisode d'escarre, 2 mois avant et 3 mois avant. Le jeu de données principal est constitué de 116 702 résidents avec 11 934 porteurs d'escarres.

L'objectif est de prédire le risque de survenance de l'escarre qui est une maladie de la peau dûe à une mauvaise irrigation du sang.

La méthode d'apprentissage utilisée est la méthode dite Tabu [27]. Cette dernière est une stratégie de résolution de problèmes d'optimisation combinatoire dont les applications vont de la théorie des graphes et des matrices aux problèmes généraux de programmation en nombres entiers purs et mixtes. C'est une procédure adaptative capable d'utiliser de nombreuses autres méthodes, telles que les algorithmes de programmation linéaire et les heuristiques spécialisées, qu'elle dirige pour surmonter les limites de l'optimalité locale [29]. Le choix est porté sur cette méthode car elle confère les meilleurs resultats avec les données de cette étude. La performance du modèle a été évaluée avec le F-score. Il est calculé avec la moyenne pondérée des scores de précision et de sensibilité [16].

Le score est obtenu avec la formule suivante :

$$Fscore = 2 \times \frac{precision \times sensibilite}{precision + sensibilite}$$

$$precision = \frac{vp}{vp + fp}$$

$$sensibilite = \frac{vp}{vp + fn}$$

vp =vrais positifs ;

fp=faux positifs ;

fn=faux négatifs

Dans cette étude, d'autres algorithmes ont été utilisés au delà des réseaux bayésiens. Nous avons l'algorithme des forêts aléatoires, la méthode Adaboost, les k-plus proches voisins, la régression logistique et les échelles d'évaluation du risque d'escarre de Braden et Norton. Les résultats de cette étude sont répertoriés dans le tableau 1 qui montre que pour le classifieur du réseau bayésien, plus on avance dans le temps, plus il est difficile de prédire correctement la maladie. Comparativement aux autres algorithmes d'apprentissage, le

pouvoir prédictif du classifieur BN est meilleur excepté les forêts aléatoires qui a un F-score légèrement supérieur.

F-Score	1 mois	2 mois	3 mois
BN Classifier	0,70	0,69	0,67
Random Forest	0,72	0,69	0,70
AdaBoost	0,69	0,67	0,69
Nearest Neighbors	0,55	0,55	0,56
Logistic Regression	0,32	0,36	0,42
Braden	0,32	-	-
Norton	0,29	-	-

TABLE 1 – Récapitulatif des F-score selon les méthodes et les temporalités de prédiction[6]

Ces réseaux sont caractérisés par un aspect qualitatif et quantitatif. Le premier facilite la compréhension de la décomposition et des relations entre les variables et le second définit la relation entre les nœuds dits parents avec la table de probabilités associée.
Les résesaux bayésiens sont plus précis que les autres méthodes d'apprentissage. Les probabilités peuvent être modifiées à l'arrivée de chaque nouvelle information. La représentation de la prédiction sous forme de distribution permet ainsi une prise de décision tenant compte des probabilités de chaque valeur de la variable à prédire. Cela signifie qu'un problème comportant plusieurs solutions, qui est insoluble avec de nombreuses méthodes d'apprentissage statistique (selon le concept du problème bien posé), pourra être abordé par un réseau bayésien, qui présentera les différentes solutions avec leurs probabilités associées. Ainsi, un réseau bayésien sera apte à inférer des valeurs pour des problèmes dits « directs», dont la solution est unique, mais également les problèmes « inverses », comportant plusieurs solutions.
Le principal point faible des réseaux bayésiens est qu'en présence de variables continues, on ne peut déterminer les tables de probabilités [25]. En sus de cela, lorsque les réseaux sont complexes, l'inférence bayésienne peut se révéler très coûteuse en termes de temps de calcul. Et dans ce cas, on fait recours à des méthodes d'inférence approximative [8].
Quelques travaux ont été menés récemment sur les réseaux bayésiens[47, 41, 50]. Ces der-

niers cherchent à optimiser le temps d'entrainement tout en gardant la précision. Parmi ces études, on a la combinaison des réseaux bayésiens avec les SVM ou avec les méthodes à noyaux afin d'estimer les paramètres d'apprentissage et produire un seul critère de classification plus précis. Ces réseaux sont appliqués dans de nombreux domaines notamment la santé et l'éducation. Dans le domaine éducatif, une méthode appelée EDM (Educationnal Data Mining) a été développée. Il s'agit de découvrir des formes pouvant transmettre de l'information sur la façon dont les étudiants apprennent, à partir des bases de données des universités ou des systèmes scolaires [47]. L'EDM porte un intérêt particulier à l'analyse prédictive, notamment la prédiction du succès des étudiants en cours de formation, ainsi que la prédiction des échecs et des abandons, à partir des données disponibles sur le processus de formation en cours. Cette méthode s'intéresse aussi à la modélisation des comportements d'étudiants, notamment la performance lors des examens. Ils sont utilisés pour prédire la réussite des étudiants [47]. Haddawy et al. [41] ont utilisé les réseaux bayésiens pour prédire la valeur de la moyenne cumulée d'étudiants internationaux postulant pour rentrer à l'Asian Institute of Technology. Sharabiani [50] les a appliqués pour prédire les notes des étudiants à des cours clefs du programme d'ingénierie de l'université de Chicago, en se basant encore sur des facteurs sociaux et personnels.

Chapitre 3

Réseaux de neurones artificiels

Les réseaux de neurones artificiels inspirés des neurones biologiques sont un modèle informatique du cerveau humain. Les premiers travaux sur les réseaux de neurones remontent à 1943 avec les travaux de Warren McCulloch et Walter Pitts[43] qui ont développé un modèle computationnel pour le neurone de base. De plus, ils ont fourni des résultats qui lient la neurophysiologie à la logique mathématique. Ils ont montré qu'avec un nombre suffisant de neurones et en ajustant de manière appropriée les liens synaptiques, chacun associé à un poids, on peut calculer, en principe, n'importe quelle fonction calculable [51]. McCulloch et Pitts ont publié les premières études systématiques des réseaux neuronaux artificiels [20]. Cette étude se présentait sous la forme d'un modèle computationnel de l'activité nerveuse des cellules du système nerveux humain. La plupart de leurs travaux sont axés sur le comportement d'un neurone simple, dont le modèle mathématique ou computationnel est présenté par la figure suivante avec $Z = \sum w_i x_i$:

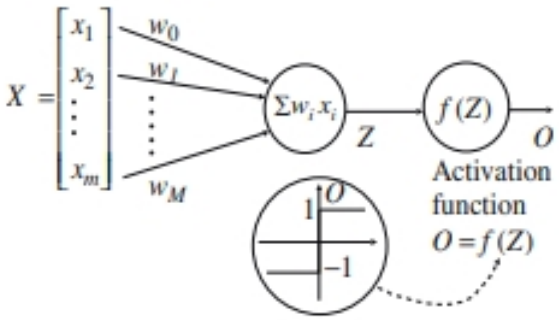

FIGURE 3 – Modèle artificiel d'un neurone biologique [20].

À l'intérieur du neurone artificiel, la somme de chaque entrée x_i multipliée par un

facteur d'échelle (ou poids w_i) est effectuée. Les entrées émulent les excitations reçues par les neurones biologiques. Les poids représentent la force de l'union synaptique : un poids positif représente un effet excitateur, et un poids négatif un effet inhibiteur. Si le résultat de la somme est supérieur à une certaine valeur seuil ou biais (représentée par le poids w_0), la cellule s'active en fournissant une valeur positive (normalement $+1$) ; dans le cas contraire, la sortie présente une valeur négative (normalement -1) ou zéro. Il s'agit donc d'une sortie binaire. Comme on peut l'observer, la relation entre l'entrée et la sortie suit une fonction non linéaire appelée fonction d'activation. Dans le premier modèle présenté dans la figure 3, l'activation est une fonction à seuil dur. Le modèle suit le comportement neurobiologique : les cellules nerveuses produisent des réponses non linéaires lorsqu'elles sont excitées par une certaine entrée. En particulier, McCulloch et Pitts ont proposé une fonction d'activation, qui représente la non-linéarité du modèle, appelée fonction de seuil dur. Cependant, ce premier modèle ne peut réaliser que des tâches très simples, et la potentialité des systèmes neuronaux réside essentiellement dans l'interconnexion entre les neurones pour former des réseaux.

Ainsi, en 1957, Franck Ronsenblatt [20] crée un réseau découlant de l'idée d'un modèle de neurones, tel que suggéré par McCulloch et Pitts pour construire une véritable machine d'apprentissage qui apprend à partir d'un ensemble de données d'apprentissage. Ce type de réseaux neuronaux est appelé Perceptron multicouches (PMC).

Le modèle mathématique d'un réseau de neurone peut être représenté par la figure 3. Le neurone est constitué des caractéristiques d'entrée qui effectue la somme pondérée de ces derniers. Le terme de biais b est ensuite ajouté à leur combinaison linéaire et le résultat est poussé à travers la non-linéarité.

$$f = w_{j1}x_1 + w_{ji}x_i + w_{jn}x_n + b \tag{35}$$

On notera la sortie y_j du neurone qui est donné par l'équation suivante :

$$y_j = f\left(\sum w_{ji}x_i\right) \tag{36}$$

Chaque neurone j dans la couche cachée fonctionne selon le modèle de la figure 3. La fonction d'activation f peut être une fonction linéaire, à seuil simple, sigmoïdale, tangente hyberbolique ou base radiale ou relu.

— Fonction linéaire

$f(x) = x$ tel que $x \in]-\infty, +\infty[$

— Fonction à seuil simple

$$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ 1 \text{ si } x \geq 0 \end{cases} \text{ tel que } x \in \{0, 1\}$$

— Fonction sigmoïdale

$f(x) = \frac{1}{1+\exp(-x)}$ tel que $x \in]0, 1[$

— Fonction tangente hyperbolique

$f(x) = \tanh x = \frac{2}{1+\exp(-2x)} - 1$ tel que $x \in]-1, 1[$

— Fonction relu

$$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases} \text{ tel que } x \in [0, \infty[$$

Les réseaux de neurones peuvent être classés en fonction de leur structure et de leur algorithme d'apprentissage. On a les réseaux à action directe (feedforward network) et les réseaux récurrents (recurrent networks).

3.1 Le Perceptron multicouches (PMC)

Le réseau neuronal feedforward le plus populaire est le perceptron multicouches (PMC) : tous les signaux circulent dans une seule direction de l'entrée vers la sortie du réseau. La figure suivante montre un PMC avec trois couches : une couche d'entrée, une couche de sortie et une couche intermédiaire ou cachée.

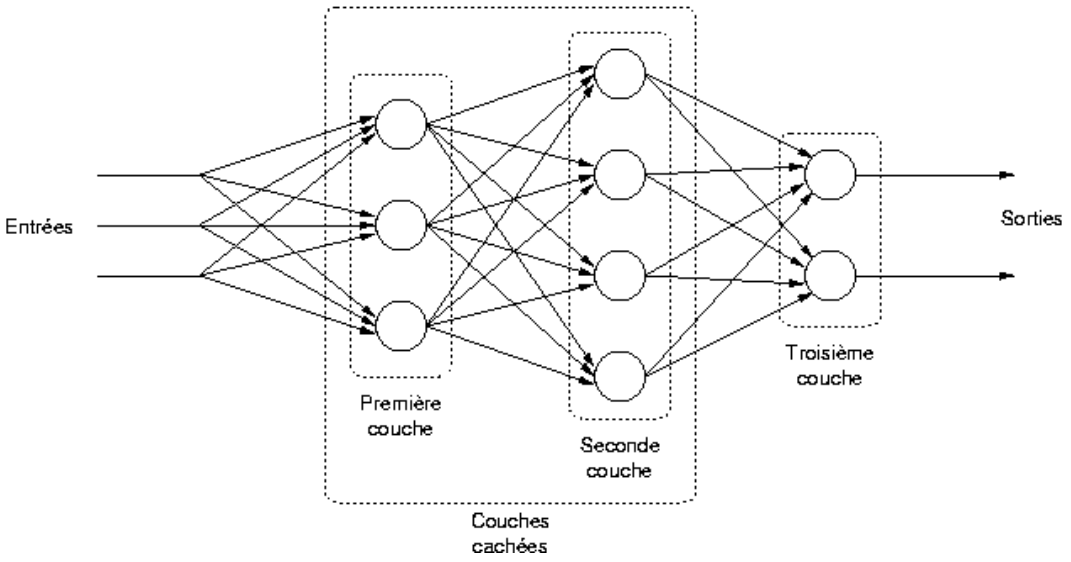

FIGURE 4 – Perceptron Multicouches avec trois couches [58]

Les neurones de la couche d'entrée servent uniquement de tampons pour distribuer les signaux d'entrée aux neurones de la couche cachée. On peut citer parmi les réseaux à action directe : le réseaux de quantification vectorielle d'apprentissage (LVQ) [35], le réseau de contrôle de l'articulation du modèle cérébelleux (CMAC) [2] et le réseau de traitement des données par la métode de groupe (GMDH) [28].

3.2 Les réseaux réccurents

Les réseaux récurrents sont des réseaux où les sorties de certains neurones sont renvoyées aux mêmes neurones ou aux neurones des couches précédentes. Les signaux peuvent donc circuler dans les deux sens, vers l'avant et vers l'arrière. On dit des réseaux récurrents qu'ils ont une mémoire dynamique : la sortie de ces réseaux à un instant donné reflète l'entrée actuelle ainsi que les entrées précédentes et les signaux [3]. Voici quelques exemples de réseaux récurrents le réseau de Hopfield [32], le réseau d'Elman [24] et le réseau de Jordan [33]. La figure 5 montre un réseau neuronal récurrent simple et bien connu, le réseau ART-1 de Grossberg et Carpenter[13] :

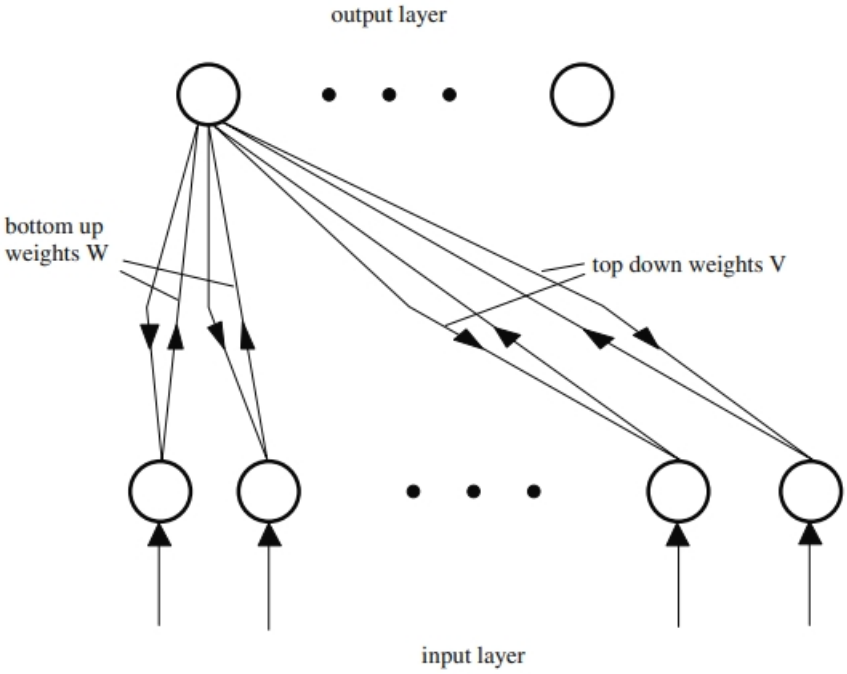

FIGURE 5 – Réseaux réccurents ART-1 de Grossberg et Carpenter [13]

Ce réseau comporte deux couches, une couche d'entrée et une couche de sortie. Les deux couches sont entièrement interconnectées, les connexions sont à la fois vers l'avant et vers l'arrière. Elles sont à la fois dans le sens direct (ou ascendant) et dans le sens de la rétroaction (ou descendant). Le vecteur $W = W_i$ des poids des connexions ascendantes vers un neurone de sortie i forme un exemplaire de la classe qu'il représente. L'ensemble des vecteurs $W = W_i$ constitue la mémoire à long terme du réseau. Ils sont utilisés pour sélectionner le neurone gagnant, ce dernier étant encore une fois le neurone dont le vecteur W_i est le plus similaire au modèle d'entrée actuel. Le vecteur V_i des pondérations des connexions descendantes d'un neurone de sortie i est utilisé pour le test de vigilance, c'est-à-dire pour déterminer si un modèle d'entrée est suffisamment proche d'un modèle stocké. Les vecteurs de vigilance V_i constituent la mémoire à court terme du réseau, V_i et W_i sont liés dans le sens que W_i est une copie normalisée de V_i, c'est-à-dire qu'il s'agit d'une copie de V_i.

$$W_i = \frac{V_i}{\epsilon + \sum V_{ji}} \tag{37}$$

ϵ est une constante et V_{ji}, la jième composante de V_i (c'est-à-dire le poids de la connexion du neurone de sortie i au neurone d'entrée j).

En 1972, Teuvo Kohonen a developpé de nouveaux réseaux pouvant servir de mémoires associatives [35]. Il a proposé un algorithme pour ajuster les poids d'un réseau dont l'entrée est un vecteur de N composantes et la sortie un autre vecteur de dimension différente $M(M < N)$. De cette façon, la dimension du sous-espace d'entrée est réduite regroupant physiquement les données.

3.3 Apprentissage

On distingue deux principaux types d'algorithmes d'apprentissage : les algorithmes d'apprentissage supervisés et les algorithmes d'apprentissage non supervisés. On parle aussi d'algorithme par renforcement qui peut être considéré comme un cas particulier de l'apprentissage supervisé.

— L'apprentissage supervisé nécessite un enseignant ou un superviseur pour fournir des signaux de sortie souhaités ou cibles. Parmi les exemples d'algorithmes d'apprentissage supervisé, nous citons la règle du delta [55], la règle du delta généralisée ou l'algorithme de rétropropagation [49] et l'algorithme LVQ [35].

— Les algorithmes d'apprentissage non supervisés n'exigent pas que les résultats souhaités soient connus. Pendant la formation, seuls les modèles d'entrée sont présentés au réseau neuronal, qui adapte automatiquement les poids de ses connexions pour regrouper les modèles d'entrée en groupes présentant des caractéristiques similaires. Parmi les exemples d'algorithmes d'apprentissage non supervisés on inclut les algorithmes d'apprentissage compétitifs de Kohonen [35] et de la théorie de la résonance adaptative (ART) de Carpenter-Grossberg [13].

— Comme mentionné auparavant, l'apprentissage par renforcement est un cas particulier d'apprentissage supervisé. Au lieu d'utiliser un enseignant pour donner des sorties cibles, un algorithme d'apprentissage par renforcement emploie un seuil critique uniquement pour évaluer la qualité de la sortie du réseau neuronal cor-

respondant à une entrée donnée. Un exemple d'algorithme d'apprentissage par renforcement est l'algorithme génétique (AG) [30]. L'algorithme de la rétropropagation et l'algorithme de la descente de gradient sont les plus adaptés concernant le perceptron multicouches. Il donne la variation Δw_{ji} du poids d'une connexion entre les neurones i et j comme suit :

$$\Delta w_{ji} = \eta \delta_j x_i \tag{38}$$

avec x_i les entrées, η le taux d'apprentissage et δ_j est un facteur qui dépend de j tel que j est un neurone de sortie ou un neurone caché.

Pour les neurones de sortie,

$$\delta_j = \left(\frac{\partial f}{\partial net_j}\right)\left(y_j^{(t)} - y_j\right) \tag{39}$$

avec y_j les sorties et pour les neurones de cachés,

$$\delta_j = \left(\frac{\partial f}{\partial net_j}\right)\sum_q w_{qj}\delta_q \tag{40}$$

net_j est la somme pondérée totale des signaux d'entrée au neurone j et $y_j^{(t)}$ est la sortie cible du neurone j.

Comme il n'existe pas de sorties cibles pour les neurones cachés, dans l'équation (40), la différence entre la sortie cible et la sortie réelle d'un neurone caché j est remplacée par la somme pondérée des q termes déjà obtenus pour les neurones q connectés à la sortie de j. Ainsi, de manière itérative, en commençant par la couche de sortie, le terme est calculé pour les neurones de toutes les couches et les mises à jour des poids sont déterminées pour toutes les connexions. Le processus de mise à jour des poids peut avoir lieu après la présentation de chaque modèle d'apprentissage (apprentissage basé sur les modèles) ou après la présentation de chaque modèle d'apprentissage. Dans les deux cas, on dit qu'une époque d'apprentissage est terminée lorsque tous les motifs d'apprentissage ont été présentés une fois au

PMC.

Plusieurs époques sont nécessaires pour que le PMC soit correctement formé. Une méthode couramment adoptée pour accélérer l'apprentissage consiste à ajouter un terme de "momentum" à l'équation de variation qui permet effectivement à la modification de poids précédente d'influencer le nouveau changement de poids.

$$\Delta w_{ji}(k+1) = \eta \delta_j x_i + \mu \Delta w_{ji}(k) \tag{41}$$

où $\Delta w_{ji}(k+1)$ et $\Delta w_{ji}(k)$ sont les changements de poids pour les époques $(k+1)$ et k respectivement et μ est le momentum, η le taux d'apprentissage et δ_j le facteur. L'apprentissage des grands réseaux de perceptrons multicouches (PMC) prend beaucoup de temps. Cela a conduit la construction de réseaux alternatifs tels que le réseau RBF (Radial Basis Function) [53]. Le réseau RBF est le réseau le plus utilisé après les PMC. La figure suivante montre la structure d'un réseau RBF qui se compose de trois couches :

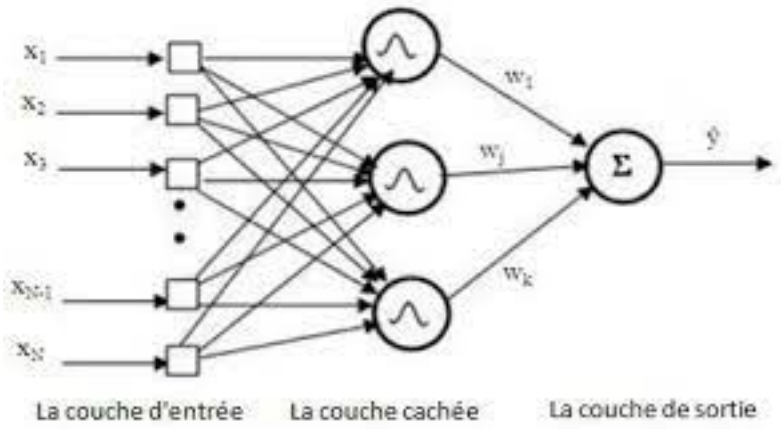

FIGURE 6 – Réseaux RBF [12]

En effet, les neurones de la couche d'entrée reçoivent les entrées $x_1, ..., x_N$. Les neurones de la couche cachée fournissent un ensemble de fonctions d'activation qui constituent une "base" arbitraire pour les modèles d'entrée dans l'espace d'entrée à étendre dans l'espace caché par le biais d'une transformation non linéaire. À l'entrée de chaque neurone caché, la distance entre le centre de chaque activation

ou fonction de base et le vecteur d'entrée est calculée. L'application de la fonction de base à cette distance produit la sortie du neurone caché. La sortie $\widehat{y}$ du réseau RBF est formée par le neurone de la couche de sortie comme une somme pondérée de l'activation des neurones de la couche cachée. La fonction de base est généralement choisie pour être une fonction standard qui est positive en son centre $x = 0$, puis décroît uniformément jusqu'à zéro de part et d'autre, comme le montre la figure 7 suivante :

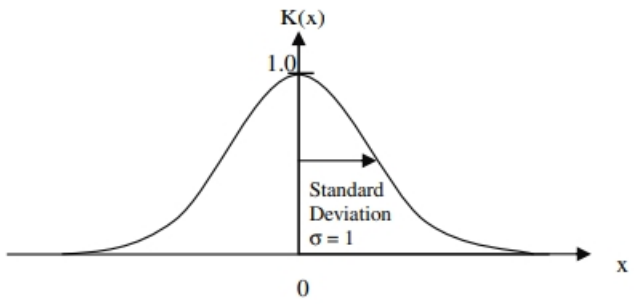

FIGURE 7 – La fonction de base radiale

Un choix courant est la fonction de distribution gaussienne :

$$K(x) = \exp\left(-\frac{x^2}{2}\right) \tag{42}$$

Cette fonction peut être décalée vers un centre arbitraire, $x = c$, et étirée en faisant varier son écart-type comme suit :

$$K\left(\frac{(x-c)}{\sigma}\right) = \exp\left(-\frac{(x-c)^2}{2\sigma^2}\right) \tag{43}$$

La sortie du réseau RBF est donnée par l'équation suivante :

$$y = \sum_{i=1}^{N} w_i K\left(\frac{\mid\mid x - c_i \mid\mid}{\sigma_i}\right) \tag{44}$$

w_i est le poids de la neurone i cachée, c_i le centre de la fonction de base i et σ_i l'écart-type de la fonction. $\mid\mid x - c_i \mid\mid$ est la norme de $(x - c_i)$. La façon la plus

courante pour calculer la norme est la distance euclidienne donnée par :

$$|| x - c_i ||= \sqrt{(x_1 - c_{i1})^2 + (x_2 - c_{i2})^2 + ... + (x_N - c_{iN})^2} \tag{45}$$

Cette norme donne la distance entre deux points x et c_i dans un espace à N dimensions.

3.4 Le Réseau de Kohonen

Le réseau de Kohonen ou carte auto-organisée de kohonen est un réseau non supervisé avec un apprentissage compétitif qui établit les critères de regroupement des données d'entrée, en extrayant les régularités et les corrélations. Lorsqu'un nombre suffisant de vecteurs d'entrée a été présenté, les poids sont auto-organisés. Ces neurones sont constitués d'un vecteur de poids dans l'espace des entrées et la carte définit des relations de voisinage entre les neurones. Ce type de réseau comporte toujours deux couches : celle d'entrée et celle de sortie.

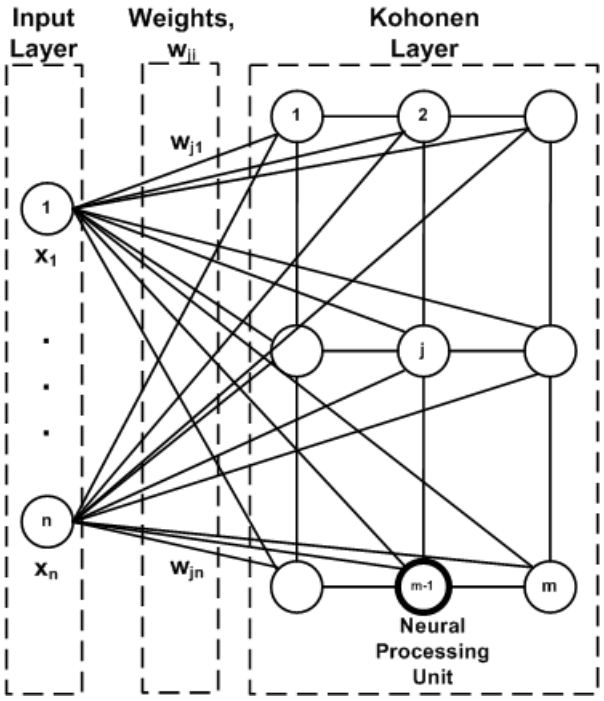

FIGURE 8 – Structure d'un réseau de Kohonen [4]

L'algorithme qui attribue des valeurs aux connexions des poids synaptiques est basé sur les concepts de voisinage et d'apprentissage compétitif. La distance entre l'entrée et les poids de tous les nœuds est calculée et le nœud le plus proche devient le nœud gagnant. La mise à jour des poids est effectuée pour ce nœud et les nœuds voisins. Le reste n'est pas actualisé, ce qui favorise une organisation physique concrète.

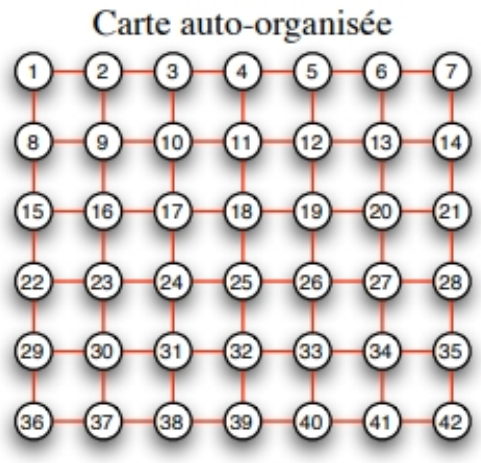

FIGURE 9 – Carte rectangulaire avec 42 neurones [43]

L'algorithme d'apprentissage du reseau de Kohonen de type compétitif met à jour les poids du neurone au temps t.

3.5 Réseaux de neurones convolutifs

Les réseaux de neurones ont dominé le domaine de l'apprentissage automatique jusqu'au milieu des années 1990. Aujourd'hui, les réseaux neuroneux avec de nombreuses couches appelés réseaux profonds ont pris le contrôle de l'apprentissage automatique. Des travaux antérieurs, associés à ce que l'on appelle les réseaux convolutifs et les réseaux neuronaux récurrents, ont inspiré le domaine du deep learning qui est aujourd'hui en plein essor. Ils sont appliqués dans de nombreux domaines particulièrement dans la classification [51].

3.5.1 Contexte et importance de la convolution

Les réseaux convolutifs, également appelés réseaux neuronaux convolutifs ou CNN, sont un type de réseau neuronal spécialisé dans le traitement des données dont la topologie est connue et en forme de grille [37]. Ces réseaux ont connu un succès fulgurant dans le cadre de leur application pratique. Ils demeurent la plus grande réussite de l'intelligence artificielle d'inpiration biologique [57]. Les principes de ces réseaux découlent de la neuroscience. Les neurophysiologistes David Hubel et Torsten Wiesel ont collaboré pendant plusieurs années pour déterminer les faits les plus fondamentaux sur le fonctionnement du système de vision de mammifères [31]. Leurs travaux qui consistaient à observer comment les neurones du cerveau du chat réagissaient face aux images projetés à des endroits précis sur un écran leur a valu un prix Nobel. Ces réseaux utilisent une opération mathématique appelée "convolution" dans au moins une des couches à la différence des réseaux neuroneux qui utilisent la multiplication matricielle. La convolution est une opération sur deux fonctions d'un argument à valeur réelle. Elle exploite trois idées importantes qui peuvent aider à améliorer un système d'apprentissage automatique : interactions éparses, partage des paramètres et représentations équivoques. De plus, la convolution fournit un moyen de travailler avec des des entrées de taille variable. Par exemple, si nous utilisons une image bidimensionnelle I comme entrée, nous souhaitons probablement utiliser un noyau bidimensionnel K :

$$S(i,j) = (I * K)(I,J) = \sum_m \sum_n I(m,n)\, K(i-m, j-n) \tag{46}$$

Au lieu de cela, de nombreuses bibliothèques "packages" de réseaux neuronaux implémentent une fonction apparentée, appelée corrélation croisée, qui est identique à la convolution mais sans inverser le noyau :

$$S(i,j) = (I * K)(I,J) = \sum_m \sum_n I(i+m, j+n)\, K(m,n)\,. \tag{47}$$

Les réseaux de neurones convolutifs ont été introduits par Yann lecun en 1989 qui s'interessait au problème de la perception visuelle invariante et voir la façon dont les méthodes d'apprentissage pouvaient aider à le résoudre. Il a alors expérimenté des architectures simples de réseaux neuronaux connectés localement, et a démontré que les connexions locales et les poids partagés amélioraient clairement les performances des réseaux neuronaux sur des tâches simples de reconnaissance de formes [1].

En traitement d'images, travailler directement avec des données brutes rend la tâche tout simplement ingérable. Si on prend l'exemple d'une image de taille 256×256, en le vectorisant, on obtient un vecteur d'entrée $x \in \mathbf{R}^{\ell}$, où $l = 65536$. Soit le nombre de nœuds de la première couche soit $k_1 = 1000$ le nombre de paramètres impliqués θ_{jk}, $j = 1, 2, ..., 65536$, $k = 1, 2, ...1000$, reliant tous les nœuds d'entrée à tous les nœuds de la première couche dans un réseau entièrement connecté, serait de l'ordre de 65 millions. Ainsi, le nombre de paramètres explose lorsqu'on a à traiter des images de plus haute résolution. En outre, le nombre de paramètres devient encore plus grand s'il s'agit des images en couleur où la dimensionnalité de l'entrée est multipliée par trois pour un schéma de représentation des couleurs RVB (rouge-vert-bleu). L'augmentation du nombre de couches cachées également engendre l'augmentation des paramètres. On fait face ainsi aux problèmes de capacité de calculs. La formation de réseaux avec un grand nombre de paramètres met sérieusement à mal leur performance de généralisation. De tels réseaux nécessiteraient une énorme quantité de données d'apprentissage afin de faire face à la tendance au surapprentissage. La vectorisation d'un réseau d'images entraîne une perte d'informations, car nous éliminons des informations importantes sur la façon dont les pixels sont liés entre eux dans une zone de l'image. L'importance de la convolution réside sur le fait qu'elle répond aux problèmes de l'explosion des paramètres en extrayant des informations qui quantifient les corrélations ou autres dépendances statistiques reliant les valeurs des pixels dans l'image.

3.5.2 Définition de la convolution

La convolution peut être définie comme une opération sur deux fonctions d'un argument à valeur réelle. La convolution est une transformation mathématique sur deux fonctions u et v qui donne une troisième fonction. Cette dernière est normalement considérée comme une version transformée de l'une des fonctions initiales. Dans le cas de deux fonctions réelles ou complexes, u et v la convolution est une autre fonction, qui est généralement notée $u * v$ et définie par [48] :

$$u * v = \int_{-\infty}^{+\infty} u(x-t)v(t)dt \tag{48}$$

Elle satisfait certaines propriétés algébriques comme la commutativité, l'associativité, la distributivité et l'identité multiplicative afin de s'assurer que les caractéristiques de leurs images géométriques correspondantes sont préservées.
Dans un cadre plus général, l'opération de convolution entre deux matrices, $H \in \mathbf{R}^{\mathbf{m}\times\mathbf{m}}$ et $I \in \mathbf{R}^{\ell\times\ell}$, est une autre matrice, définie comme suit :

$$O(i,j) = \sum_{t=1}^{m}\sum_{r=1}^{m} h(t,r)I(i+t-1, j+r-1) \tag{49}$$

où, dans notre cas, $m < \ell$. En d'autres termes, $O(i,j)$ contient des informations dans une zone de fenêtre du tableau d'entrée. Selon la définition de l'équation (49), l'élément $I(i,j)$ est l'élément supérieur gauche de cette zone de fenêtre. La taille de la fenêtre dépend de la valeur de m. La taille de la matrice de sortie dépend des hypothèses que l'on adopte sur la manière de traiter les pixels aux limites de I. Dans le jargon du traitement du signal, l'équation précédente est appelée opération de corrélation croisée.

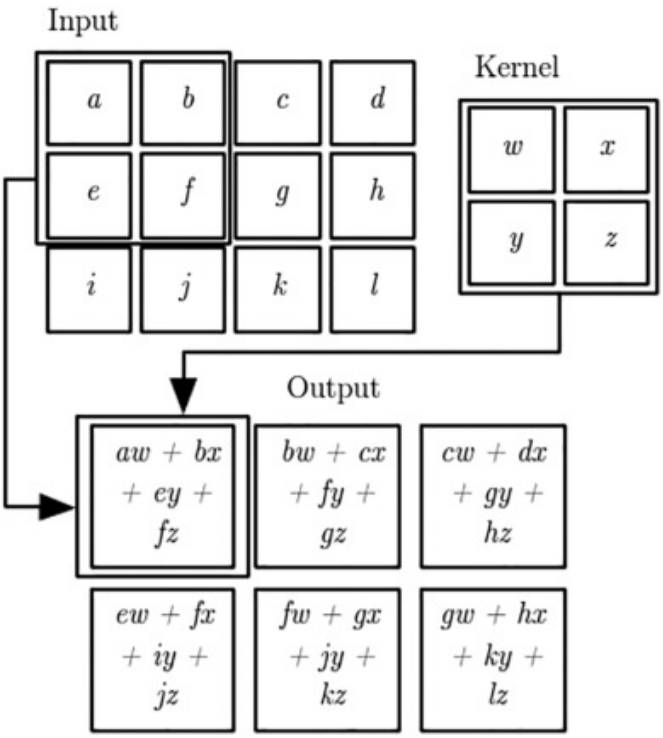

FIGURE 10 – Un exemple de convolution 2-D. La sortie d'une image d'entrée et d'un noyau est plus petite que l'image d'entrée [36]

La figure 10 décrit l'opération de convolution d'une image de taille $4X \times 3$ (input) avec le noyau de taille 2×2 (Kernel). On a un calcul de moyenne pondérée sur les pixels de la fenêtre de l'image avec la matrice noyau ou le filtre. La matrice de sortie est généralement désignée comme le tableau de la carte des caractéristiques. Chaque pixel d'une carte de caractéristiques de sortie est une moyenne pondérée des pixels situés dans une zone spécifique de l'image d'entrée (ou de la sortie de la couche précédente). La zone spécifique qui correspond à un pixel est appelée champ récepteur.

3.5.3 Hyperparamètres

En apprentissage automatique, on appelle hyperparamètre, un paramètre dont la valeur nous permet de contrôler le processus d'apprentissage. Dans les CNN, on distingue principalement 4 hyperparamètres :

— Le nombre de filtres

La profondeur d'une couche est le nombre de matrices de filtre qui sont em-

ployées dans cette couche. Elle est différentes de la profondeur du réseau, qui correspond au nombre total de couches cachées utilisées.

— La taille des filtres

— Le pas ou stride

En pratique, au lieu de faire glisser la matrice de filtrage d'un pixel à la fois, on peut la faire glisser de, disons s pixels. Cette valeur est connue sous le nom de stride. Pour des valeurs de $s > 1$, on obtient des tableaux de cartes de caractéristiques de plus petite taille. Ceci est illustré par la figure suivante :

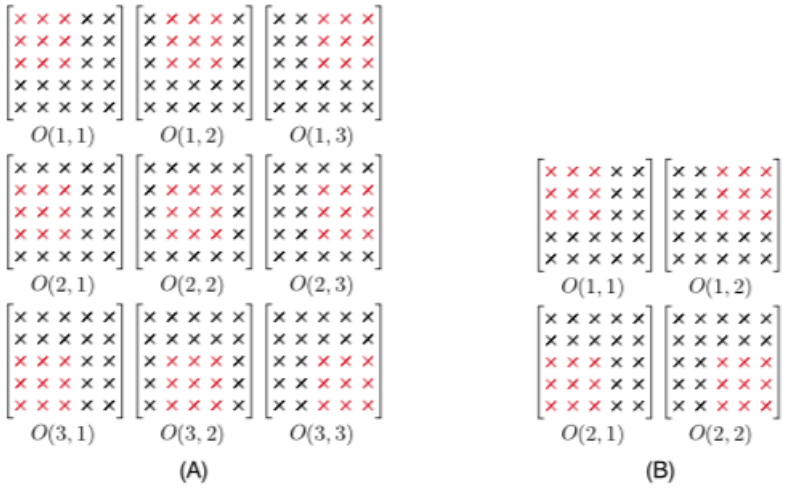

FIGURE 11 – Etapes de Stride [51]

La figure présente le cas d'une matrice d'entrée de taille 5×5 et d'une matrice de filtrage 3 ×3. Dans (A), le pas est égal à $s = 1$ et dans (B) il est égal à $s = 2$. Dans (A) la sortie est une matrice de taille 3×3 et dans (B) de taille 2×2.

— Le zéro padding

Parfois, des zéros sont utilisés pour remplir la matrice d'entrée autour des pixels de bordure. De cette façon, la dimension de la matrice augmente. Si la matrice d'origine a des dimensions $l \times l$, après l'avoir étendue avec p colonnes et lignes, les nouvelles dimensions deviennent $(l + 2p) \times (l + 2p)$. La raison pour laquelle le remplissage peut être utilisé est que les pixels de la bordure de l'image d'entrée contribuent moins à la sortie, par rapport aux pixels qui sont situés à l'intérieur de l'image.

Pour voir comment la convolution fonctionne, commençons par inspecter une image de dimension 7×7 en noir et blanc codé pour notre cas 0 pour la couleur noire et 1 pour le blanc. La matrice de gauche contient les nombres 0 et 1 qui correspondent chacun à la luminosité d'un pixel.

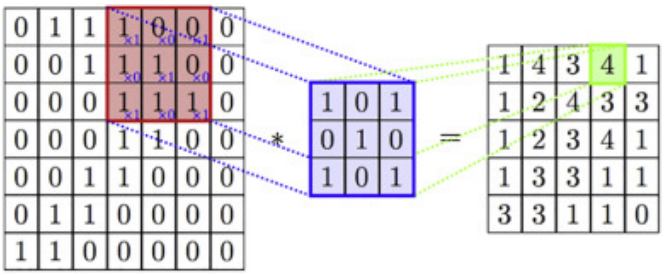

FIGURE 12 – Etapes de la convolution[46]

L'opération de convolution se fait en ajoutant chaque élément de l'image à ses voisins locaux, pondérés par le noyau de dimension 3×3. Les valeurs d'un pixel donné dans l'image de sortie sont déterminées en multipliant chaque valeur de noyau par les valeurs de pixel d'image d'entrée correspondantes.

(A) (B)

Le filtre appliqué sur l'image ci-dessus est connu comme détecteur de bords. La convolution d'un tableau d'images, I , avec la matrice précédente, H , permet de détecter les bords d'une image. La Figure (A) montre l'image du bateau où les bords ont été extraits et la figure B montre le résultat après le filtrage de l'image de gauche avec la matrice de filtre H ci-dessus.

$$H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & 1 & -1 \end{bmatrix}$$

De plus, en modifiant de manière appropriée les valeurs de H , on peut détecter les bords dans différentes orientations, par exemple, diagonale, verticale, horizontale ; en d'autres termes, en modifiant les valeurs de H , on peut générer différents types de caractéristiques. En pratique, le choix du filtre est adapté aux types de données que l'on traite.

3.5.4 Architecture CNN

La forme typique d'un réseau convolutif complet consiste en une séquence de couches convolutives, chacune comprenant les trois étapes de base, à savoir la convolution, la non-linéarité et la mise en commun.

— **Couche de convolution**

C'est la première et la plus importante couche du réseau convolutif. Cette couche permet d'extraire les caractéristiques de l'image. Dans cette couche, un CNN utilise différents filtres pour convoluer l'image entière ainsi que les cartes de caractéristiques intermédiaires, générant diverses cartes de caractéristiques. On a le filtre moyenneur qui calcule pour chaque pixel la moyenne du pixel avec ses 8 proches voisins. Ce filtre réduit la netteté de l'image. Comparé au filtre moyenneur de taille 5×5, le filtre gaussien 7×7 réduit autant le bruit sans altérer autant la netteté. De manière générale, le filtre gaussien est le meilleur filtre de lissage car sa réponse fréquentielle est aussi une gaussienne. Il existe également d'autres filtres comme le filtre réhausseur qui renforce les contours et le filtre laplacien qui détecte les contours. La profondeur du volume de sortie contrôle le nombre de neurones de la couche qui se connectent à la même région du volume d'entrée.

— **Couche de pooling**

Sa tâche consiste à simplifier ou à réduire les dimensions spatiales de l'information dérivée des cartes d'entités. Nous avons trois types de pooling : le premier est le pooling moyen, le second est le pooling de norme L2 et enfin l'utilisation la plus populaire est le pooling max [15] en raison de sa rapidité et de sa convergence améliorée. Cela prend essentiellement un filtre (normalement de taille 2×2) . Il l'applique ensuite au volume d'entrée et produit le nombre maximum dans chaque sous-région qui a le filtre qui tourne autour.

— **Couche ReLU**

C'est la couche d'unités linéaires rectifiées. C'est une couche de neurones qui applique la fonction de non-linéarité non saturante ou fonction de perte :

$$f(x) = max(0, x) \tag{50}$$

Il donne les propriétés non linéaires de la fonction de décision et du réseau global sans affecter les champs récepteurs de la couche de convolution. La couche ReLU se traduit par un réseau de neurones qui s'entraîne plusieurs fois rapidement, sans faire de différence significative dans la précision de la généralisation.

— **Couche entièrement connectée**

L'entrée de cette couche est un vecteur de nombres. Chacune des entrées est connectée à chacune des sorties d'où le terme "entièrement connecté". C'est la dernière couche, en général après la dernière couche de pooling dans le processus CNN. Les couches entièrement connectées fonctionnent comme un réseau de neurones classique et contiennent environ 90 % des paramètres de CNN. Cette couche prend essentiellement en entrée la sortie de la dernière couche de pooling et génère un vecteur de dimension N où N est le nombre de classes parmi lesquelles le programme doit choisir.

3.5.5 Exemples de réseaux convolutifs

Il existe un certain nombre de variantes autour de l'architecture des CNN.

LeNet-5 : C'est un exemple typique de la première génération de CNN et il a été construit pour reconnaître les chiffres des nombres . Pour des raisons historiques. L'entrée du réseau est constituée d'images en niveaux de gris de taille $32 \times 32 \times 1$. Le réseau utilise deux couches de convolution. Dans la première couche, le volume de sortie a une taille de $28 \times 28 \times 6$, qui après regroupement devient $14 \times 14 \times 6$. Dans la deuxième couche, les dimensions du volume sont de $10 \times 10 \times 16$ et, après regroupement, de $5 \times 5 \times 16$. La non-linéarité utilisée à l'époque était de type sigmoïde. Observez que la hauteur et la largeur des volumes diminuent et que la profondeur augmente, comme indiqué précédemment. Le nombre d'éléments du dernier volume est égal à 400. Ces éléments sont empilés dans un vecteur et alimentent les nœuds d'entrée correspondants d'un réseau entièrement connecté. Ce dernier est constitué de deux couches cachées avec 120 nœuds dans la première et 84 nœuds dans la seconde. Il y a 10 nœuds de sortie, un par chiffre, utilisant une non-linéarité softmax. Le nombre total de paramètres impliqués est de l'ordre de 60.000.

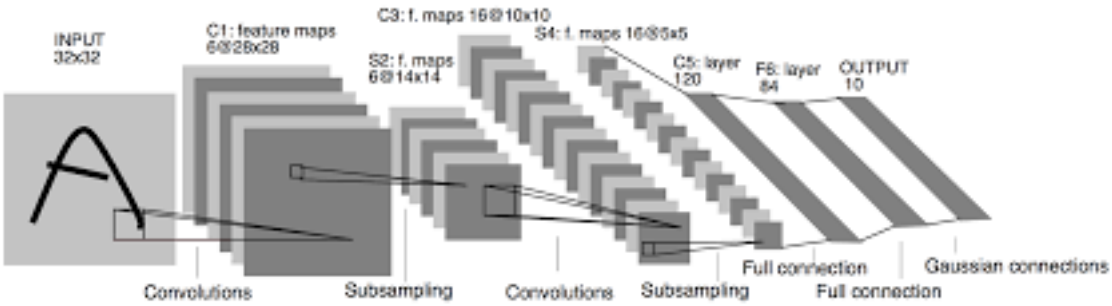

FIGURE 13 – Architecture du Réseau LeNet.

AlexNet : Ce réseau est également historique car il a démontré que le point crucial pour faire fonctionner de grands réseaux est la disponibilité de grands ensembles d'entraînement. L'article en question est celui qui a vraiment ramené les CNN sur la scène et a servi de catalyseur pour leur adoption bien au-delà de la tâche de

reconnaissance des chiffres. Alexnet est un développement de LeNet-5, mais il est beaucoup plus grand et implique environ 60 millions de paramètres. Les entrées du réseau sont des images RVB de taille $227 \times 227 \times 3$. Il comprend cinq couches cachées et le volume final se compose de 9216 éléments qui alimentent un réseau entièrement connecté avec deux couches cachées de 4096 unités chacune. La sortie consiste en 1000 nœuds softmax (un par classe) pour reconnaître les images du jeu de données ImageNet pour la reconnaissance d'objets. La ReLU a été utilisée comme non-linéarité dans les couches cachées.

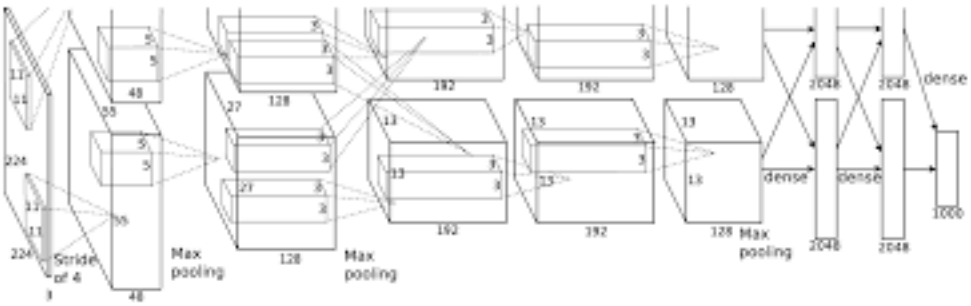

FIGURE 14 – Architecture du Réseau AlexNet

VGG-16 : Ce réseau est beaucoup plus grand qu'AlexNet. Il implique un total d'environ 140 millions de paramètres. La principale caractéristique de ce réseau est sa régularité. Il implique 3×3 filtres pour effectuer les mêmes convolutions en utilisant le padding et le stride $s = 1$ et 2×2 fenêtres pour le maxpooling avec le stride $s = 2$. A chaque fois que le pooling intervient, la hauteur et la largeur des volumes sont divisées par deux et à chaque fois la profondeur est augmentée de deux. En partant de $224 \times 224 \times 3$ images d'entrée et après 13 couches, le volume final a une taille de $7 \times 7 \times 512$, soit un total de 7168 éléments, qui après sa vectorisation est alimenté à un réseau entièrement connecté avec 2 couches cachées, chacune comprenant 4096 nœuds. Les 1000 nœuds de sortie sont construits autour de la non-linéarité softmax et ReLU a été utilisé pour les unités cachées dans tout le réseau.

GoogleNet et le réseau Inception : L'architecture utilisée dans ce réseau s'écarte de l'architecture "archétypale" classique. Au cœur de ce réseau se trouve le module dit d'initiation . Un module de démarrage est constitué de filtres de tailles et de profondeurs différentes ainsi que d'un chemin de mise en commun différent. L'un d'eux implique une convolution 1×1 qui agit sur la profondeur du volume d'entrée. Un autre chemin effectue une mise en commun, puis alimente une convolution 5×5. Deux chemins alimentent séparément deux étapes de convolution différentes, l'une basée sur un filtre 3×3 et l'autre sur un 5×5. Avant les convolutions, une couche de goulot d'étranglement, via une convolution 1×1, est utilisée pour réduire la charge de calcul correspondante. Les volumes de sortie de toutes ces chemins sont ensuite concaténés pour former la sortie finale de cette étape. L'idée derrière le module de création est de laisser le réseau, pendant la phase de formation, "décider" des opérations qui conviennent le mieux aux différentes couches et entrées. Par exemple, les caractéristiques de plus haute abstraction sont traitées par des couches plus proches de la sortie du réseau. Par conséquent, on s'attend à ce que la concentration spatiale des informations respectives diminue. Cela suggère que le rapport entre les convolutions 3×3 et 5×5 devrait augmenter à mesure que l'on passe aux couches supérieures. Le nombre de couches du réseau était de 22 et le nombre total de paramètres rapportés dans l'article était de l'ordre de 6 millions.

3.6 Les auto-encodeurs

Les auto-encodeurs font partie de la famille des réseaux de neurones ayant une configuration particulière et appartenant aux algorithmes d'apprentissage non supervisé. Ces réseaux ont le même nombre de neurones sur leur couche d'entrée et leur couche de sortie. Le but pour un auto encodeur est d'avoir une sortie la plus proche de l'entrée.

3.6.1 Architecture des auto-encodeurs

Un auto-encodeur est composé de 3 parties :

— l'encodeur

— l'espace latent et

— le décodeur

L'architecture d'un auto-encodeur peut être assimilé à un entonnoir avec l'encodeur à gauche et le décodeur à droite. Entre ces deux derniers, on a l'espace latent. L'encodeur permet de condenser l'information disponible initialement (image, texte, audio, etc.) en extrayant des features qui caractérisent du mieux possible l'information initiale. L'encodeur compresse donc l'entrée dans une représentation moins coûteuse. Le vecteur qui résulte de l'encodeur est de taille beaucoup plus petite que le vecteur initial.

L'encodeur transforme l'entrée en une représentation dans un espace de dimension plus faible appelé espace latent. Dans cet espace, on condense uniquement les informations les plus importantes qui sont contenues dans nos données d'entraînement, tout en filtrant le bruit, c'est le principe de la features extraction. Lorsque l'espace latent est bien construit, il préserve les éventuelles similarités entre les données et permet d'avoir une structure continue des données.

Le décodeur se charge de reconstruire l'information de départ, à partir du vecteur condensé. Par exemple lorsque l'on travaille sur des images, quand il est bien entraîné, un auto-encodeur est capable de prendre une image en entrée, de la condenser dans un vecteur de petite taille via l'encodeur, puis de la recréer uniquement grâce à ce vecteur de petite taille, via le décodeur. Ce dernier est chargé de reconstruire à l'aide de l'espace latent une sortie la plus fidèle à l'entrée.

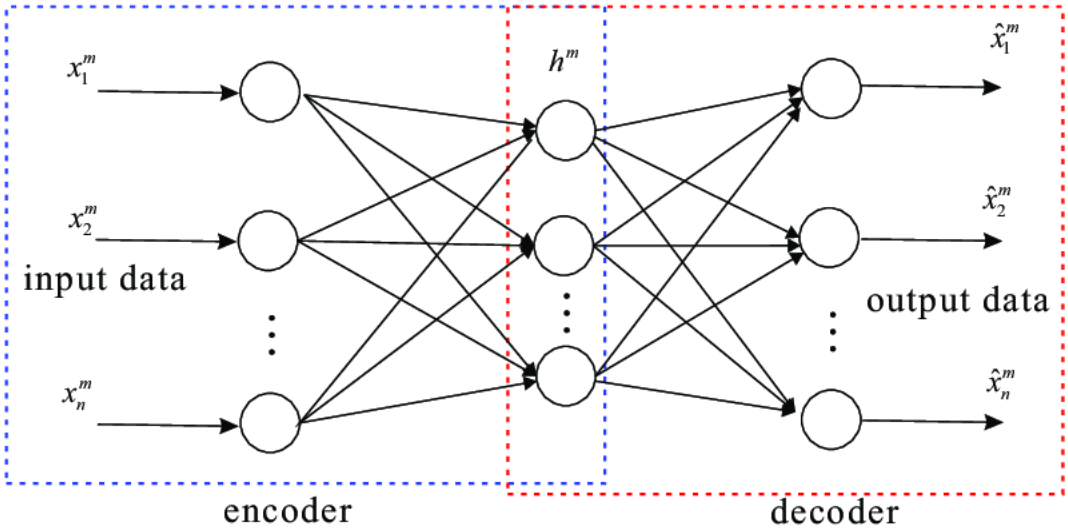

FIGURE 15 – Architecture d'un auto-encodeur[40]

Lors de l'apprentissage, l'auto encodeur va donc apprendre à chercher à garder le plus d'information possible entre l'entrée et l'espace latent afin que le décodeur ait les informations les plus essentielles pour reconstruire l'image. Ainsi, l'encodeur apprend les composantes les plus importantes d'une entrée pour avoir la meilleure compression possible.

À première vue, cela ressemble fortement à l'analyse en composantes principales (ACP). Cependant, les auto encodeurs permettent d'ajouter des non-linéarités grâce aux fonctions d'activation et à leur structure de réseau de neurones, ce qui distingue un auto encodeur simple d'une ACP. La différence entre ces 2 approches de réduction de la dimensionnalité est ilustrée par cette figure suivante :

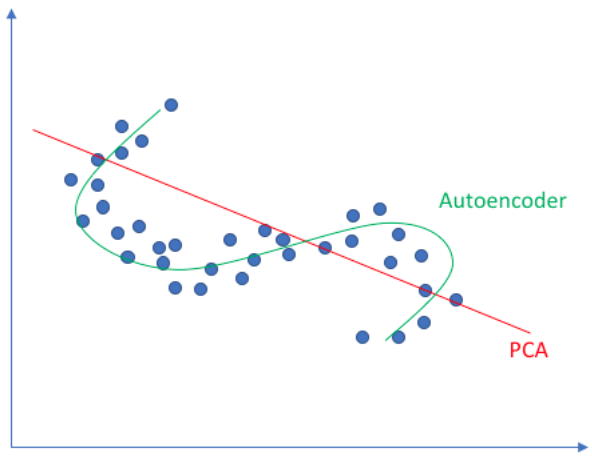

FIGURE 16 – Réduction de dimensionnalité linéaire et non linéaire [42]

Le processus d'apprentissage est décrit simplement comme la minimisation d'une fonction de perte :

$$L\left(x, g(f(x))\right) \tag{51}$$

où L est une fonction de perte pénalisant $g(f(x))$ pour être dissemblable de x, telle que l'erreur quadratique moyenne [7].

Pour entraîner un auto-encodeur, on lui fournit des données en entrée qu'il doit être capable d'encoder dans un espace d'une dimension fixée, puis de décoder à partir de l'encodage obtenu. Pendant l'entraînement on s'attend à ce que le modèle nous donne en sortie ce qu'on lui a fournit en entrée.

Pour suivre l'entraînement on fixe une perte qui repose sur le calcul de la distance entre la donnée fournie en entrée et celle reçue en sortie, on parle de reconstruction loss.

Les autres étapes de l'entraînement des auto-encodeurs sont similaires à ceux des architectures de réseaux de neurones classiques. On a une fonction d'optimisation et des paramètres à calibrer.

3.6.2 Applications des auto-encodeurs

Les applications des auto-encodeurs sont nombreuses. Avant que les réseaux adverses génératifs (GAN) ne deviennent incontournables pour générer de nouvelles images, les auto-encodeurs permettaient déjà de le faire. Pour cela on n'a qu'à utiliser le décodeur pour décoder des vecteurs de l'espace latent qui ne sont pas présents dans notre dataset initialement. On peut faire des opérations sur nos données ou même de l'interpolation entre deux observations.

Les auto-encodeurs peuvent aussi servir à débruiter des images. Le bruit dans les images englobe toutes les imperfections et les pixels mal positionnés qui peuvent être corrigés. Pour entraîner un auto-encodeurs de ce genre une option possible est de simuler un bruitage de données sur les données d'entraînement, de fournir les données bruitées en entrée de l'auto-encodeur et d'avoir les images débruitées en sortie. Ce principe d'auto encodeur a été généralisé à des tâches plus variées, en particulier dans les tâches de débruitage (on parle de denoising AE en anglais). Après avoir appris comment représenter de manière optimale les entrées, en fournissant à l'auto encodeur des entrées bruitées, il est capable de donner en sortie des données débruitées qui sont très proches de l'entrée originale. Un auto-codeur de débruitage (DAE) minimise plutôt :

$$L(x, g(f(\tilde{x}))) \tag{52}$$

où $\tilde{x}$ est une copie de x qui a été corrompue par une forme de bruit. En pratique, les meilleures solutions pour générer ou débruiter des images qui reposent sur des auto-encodeurs fonctionnent avec des auto-encodeurs variationnels (VAE). Les VAE apprennent la distribution de probabilité des données et permettent donc d'assurer la continuité des encodages obtenus.

Chapitre 4

Application en science des données

Dans cette partie, nous appliquerons les modèles d'apprentissage supervisés développés dans les chapitres précédents sur deux jeux de données. On utilisera la classification bayésienne précisemment le Classifieur Naïf dans la première partie. Ensuite, nous verrons les classifieurs avec les réseaux de neurones notamment le perceptron multicouches et les réseaux de neurones convolutifs avec la classification d'images. Nous étudierons la performance de chaque classifieur et on fera une étude comparative pour chaque jeu de données distinct.

4.1 Description des données

Le premier jeu de données provient d'un ensemble de données épidémiologiques de cas positifs et négatifs de COVID-19 du Mexique. Il a été rapporté par la Direction générale de l'épidémiologie, Secrétariat de la santé au Mexique et mis à disposition sur leur site Web officiel [59]. Ces données sont obtenues à partir d'informations du système de surveillance épidémiologique des maladies respiratoires virales, qui ont été signalées par les 475 unités de surveillance de ces au Mexique. L'ensemble de données contient les résultats des tests de réaction en chaîne par polymérase par transcription inverse (RT-PCR) du laboratoire pour les cas de COVID-19 au Mexique. Il compte 1.056.915 individus positifs ou négatifs à la Covid décrit par 35 variables. Sur ces données, nous allons essayer de faire l'analyse de la comorbidité associée au coronavirus en nous référant de l'article de Kernel Pietro [34]. Dans cet article, l'auteur a utilisé plusieurs classifieurs d'apprentissage supervisé (Arbres de

décision, K plus proches voisins, XGBoost, forêts aléatoires et le Classifieur naïf de Bayes) pour prédire la mortalité chez les patients atteints de COVID-19 qui présentent des comorbidités liées au COVID-19. L'objectif de cette étude était de développer des modèles de COVID-19 en utilisant l'apprentissage automatique pour prédire avec précision à la fois la demande de soins hospitaliers et la mortalité en fonction des patients qui présentent des maladies telles que l'hypertension, l'obésité, le diabète et le tabagisme. Nous allons donc orienter notre analyse dans ce sens en retenant les variables de comorbidités mentionnés sur l'ensemble des cas positifs qui est de 462.690 individus.

Pour le second jeu de données on aura la base d'images MNIST des chiffes manuscrits qui contient 60.000 images destinées à l'apprentissage et 10.000 images surlesquelles on fera notre test [38]. Ce jeu de données est un sous-ensemble d'une plus grande base de données spéciale de l'institut National des normes et de la technologie des chiffres écrits par des employés du Bureau du recensement des États-Unis et d'une base de données spéciale de chiffres écrits par des élèves du secondaire qui contiennent des images monochromes de chiffres manuscrits. Il s'agit d'un ensemble d'images de taille 28×28 pixels en niveaux de gris. Les valeurs des pixels de chaque image sont des entiers compris entre le noir (0) et le blanc(255). On y appliquera le classifieur naive de Bayes, les réseaux de neurones convolutifs et le perceptron multicouches.

4.2 Etude comparative : expérimentation et résultats

Plusieurs critères peuvent être considérés pour comparer les performances des méthodes de classification. Dans notre étude, nous nous interessons à l'accuracy qui est le taux de bon classement ou taux de bonne prédiction. Cet indicateur est calculé comme suit :

$$Accuracy = \frac{\text{vrais positifs} + \text{vrais négatifs}}{total}$$

avec $total$ = vrais positifs + vrais négatifs + faux positifs + faux négatifs

Modèles	MNIST	COVID
Classifieur naïf de Bayes	0,843	0.88
Réseaux bayésiens	0.87	-
PMC	0.9734	0.89
Réseaux de neurones convolutifs	0.9869	-

TABLE 2 – Tableau comparatif de la performance Classifieur Naïf de Bayes, Réseaux bayésiens, PMC et Réseaux de neurones convolutifs sur le jeu de données MNIST et COVID

Le tableau ci-dessus résume les résultats obtenus pour les différents classifieurs pour nos deux jeux de données respectifs. On remarque que pour le jeu de données MNIST, les méthodes neuronales sont en tête avec une prédiction avoisinnant les 97%. La performance des réseaux convolutifs vient confirmer de nombreuses études qui ont été réalisées avec ce même jeu de données. Ces réseaux donnent de meilleures performances que les deux autres modèles concernant la classification d'images avec un taux d'erreur de moins de 2% bien que son temps d'exécution est plus long comparativement aux autres méthodes. Les réseaux bayésiens devancent aussi le Classifieur naïf de Bayes de moins de 3% avec un "accuracy" de 87%. Toutefois, la mise en oeuvre des réseaux bayéiens est beaucoup plus complexe et prend plus de temps pour l'exécution. Pour le jeu de données de la covid, seul le Classifieur naïf de Bayes et le Perceptron multicouches ont été appliquées. Nous ne disposions pas des images pour appliquer les réseaux convolutifs. Concernant les réseaux bayésiens, le processus est assez long et nécessite des ressources dont on ne disposait pas. On remarque que les résultats sont sensiblement proches. De plus, l'exécution de l'algorithme PMC est plus long avec les nombreux paramètres à prendre en considération.

Conclusion et perspectives

Dans ce document, différentes méthodes de classification statistiques et neuronales ont été étudiées et chacune a été appliquée dans l'un des deux jeux de données proposés. Nous avons fait dans un premier temps une revue de littérature de ces méthodes et avons établi une étude comparative pour évaluer leur performance. L'application de ces méthodes a été faite sur le jeu de données MNIST, la base de données d'images des chiffres manuscrits et sur un jeu de données de la COVID. Ainsi, cette analyse a permis de montrer que les méthodes neuronales sont bien plus performantes que les méthodes statistiques probabilistes bien que leur temps d'exécution est plus long et donc nécessite beaucoup plus de ressources. Pour la classification d'images, la performance des réseaux de neurones convolutifs dépasse celle des autres méthodes. Ceci vient confirmer ce que la revue de littérature révèle. Concernant la COVID, on note une légère différence (1%) au niveau des deux méthodes appliquées. Toutefois, il faut noter que les résutats de tous ces algorithmes sont satisfaisants et donnent une prédiction d'au moins 84%.

Dans la poursuite de cette étude, nous pourrons essayer l'approche mixte en intégrant l'inférence bayésienne dans les réseaux de neurones pour voir comment se comporteront les résultats. Cela nous donnera l'occasion d'étudier les réseaux de neurones bayésiens à savoir leur architecture, leur fonctionnement et leur performance en science des données.

Références

[1] Dr D. Abdelhamid. *Classification Bayésienne*. 2017-2018.

[2] J.S. Albus. *A new approach to manipulator control : The cerebellar model articulation controller (CMAC)*. 1975.

[3] D. Andina et D. Pham. *Intelligence computationnelle : pour l'ingénierie et la fabrication*. Sous la dir. de Springer. 2007.

[4] E. Araujo, C.R. Silva et D. Sampaio. « Video target tracking by using competitive neural networks ». In : *WSEAS Transactions on Signal Processing* 4.8 (2008), p. 420-431.

[5] D. Aynur. « Feature interval learning algorithms for classification ». In : *Knowledge-Based Systems* 23.5 (2010), p. 402-417. ISSN : 0950-7051. DOI : https://doi.org/10.1016/j.knosys.2010.02.002. URL : https://www.sciencedirect.com/science/article/pii/S0950705110000249.

[6] J. Belmin, C. Charon et P. H.Wuillemin. *Réseaux Bayésiens pour la Prédiction d'évènements de Santé Défavorables en EHPAD établissement d'Hébergement pour Personnes Agées Dépendantes)*. Octobre 2021.

[7] L. Benabbou, A. Guitouni et P. Lang. *Méthodes de classification : revue de la littérature, essai de caractérisation et de comparaison*. Université Laval, Faculté des sciences de l'administration, 2004.

[8] P. Bessiere et al. *Programmation bayésienne*. Sous la dir. de CRC presse. 2013.

[9] Q. Biao, X.Yuni et L.Fang. « A Bayesian classifier for uncertain data ». In : *Proceedings of the ACM Symposium on Applied Computing* (jan. 2010), p. 1010-1014. DOI : 10.1145/1774088.1774301.

[10] Q. Biao et al. « A novel Bayesian classification for uncertain data ». In : *Knowl.-Based Syst.* 24 (déc. 2011), p. 1151-1158. DOI : 10.1016/j.knosys.2011.04.011.

[11] H. H. Bock et E. Diday. *Analysis of symbolic data. Exploratory methods for extracting statistical information from complex data*. Springer, 2004.

[12] B. Boudebbouz et al. « Utilisation des reseaux de neurones artificiels de type rbf pour la modelisation du regime normal a point de fonctionnement variable d'une installation industrielle ». In : *European Scientific JOURNAL* 11.18 (2015).

[13] A.G. Carpenter et S. Grossberg. « L'ART de la reconnaissance adaptative de formes par un réseau de neurones auto-organisé ». In : *Ordinateur* 21 (1988). Sous la dir. d'IEEE, p. 77-88.

[14] P. Cheeseman et J.Stutz. « Bayesian Classification(AutoClass) :Theory and Results ». In : *Advances in Knowledge Discovery and Data Mining* (mai 1997).

[15] R. Cheng, V. Dmitri et s. Prabhakar. « Évaluer les requêtes probabilistes sur des données imprécises ». In : (2003). Sous la dir. d'Association pour les machines informatiques, 551–562. doi : 10.1145/872757.872823. url : https://doi.org/10.1145/872757.872823.

[16] N. Chinchor. *Métriques d'évaluation MUC-4 dans Proc. de la Quatrième Conférence de Compréhension du Message 22–29*. Sous la dir. de Morgan Kaufmann. 1992.

[17] F. Chollet. *L'apprentissage profond avec Python*. «Les essentiels de l'IA». Saint-Cyr-sur-Loire : machinelearning.fr, DL 2020. isbn : 978-2-4916-7400-7.

[18] W. Cohen. « Fast Effective Rule Induction ». In : (1995), p. 115-123.

[19] G. Cormode et A. McGregor. « Approximation algorithm for clustering uncertain data ». In : (2000), p. 191-199.

[20] J.D. Cowan. « Discussion : McCulloch-Pitts et les réseaux de neurones associés de 1943 à 1989 ». In : *Bulletin de biologie mathématique* 52 (1990). Sous la dir. de Springer, p. 73-97.

[21] A.P. Dempster, N.M. Laired et D.B. Rubin. *Maximum likelihood from incomplete Data via the EM algorithm.* JOURNAL of the royal Statistic Society, Series B (Methodoligical), Vol.39, 1977.

[22] E. Diday et M. Noirhomme-Fraiture. *Symbolic Data Analysis and the SODAS software.* Jan. 2008. isbn : 9780470018835. doi : 10.1002/9780470723562.

[23] G. Ducamp, C. Gonzales et P.H. Wuillemin. *aGrUM/pyAgrum : a toolbox to build models and algorithms for Probabilistic Graphical Models in Python.*

[24] J.L. Elman. « Trouver la structure dans le temps ». In : *Sciences cognitives* 14 (1990), p. 179-211.

[25] N. Fenton, M. Neil et D. Marquez. « Utilisation des réseaux bayésiens pour prédire les défauts et la fiabilité des logiciels ». In : *Actes de l'Institution of Mechanical Engineers, Part O : JOURNAL of Risk and Reliability* 222 (2008), p. 701-712.

[26] M.A.T Figueiredo. « Lecture notes on Bayesian estimation and classification ». In : *Instituto de Telecomunicacoes-Instituto Superior Tecnico* 60 (2004).

[27] K. G. Fred, P. James et M. Laguna. *L'approche OptQuest de l'optimisation de la simulation Crystal Ball.* Sous la dir. de Citeseer. 1998.

[28] R. Hecht-Nielsen. « Sur la structure algébrique des espaces de poids du réseau feedforward ». In : *Ordinateurs neuronaux avancés.* Sous la dir. d'Elsevier. 1990, p. 129-135.

[29] A. Hertz et D. Werra. *La métaheuristique de recherche tabou :comment nous l'avons utilisée.* T. 1. Springer Pays-Bas, p. 111-121.

[30] P. Hollande et R. Rescorla. « L'effet de deux manières de dévaluer le stimulus inconditionné après un conditionnement appétitif de premier et de second ordre. » In : *JOURNAL of Experimental Psychology : Animal Behavior Processes* 1 (1975). Sous la dir. d'Association américaine de psychologie.

[31] D. Hubel, T. Wiesel et S. LeVay. *Functional architecture of area 17 in normal and monocularly deprived macaque monkeys*. T. 40. 1976, p. 581-589.

[32] J.J. Hopfield John. « Réseaux de neurones et systèmes physiques dotés de capacités de calcul collectives émergentes. » In : *Actes de l'académie nationale des sciences* 79 (1982), p. 2554-2558.

[33] J. Jordan et V. Stone. *Le sens de la mutualité*. T. 23. Stone Center for Developmental Services et Studies, Wellesley College, 1986.

[34] P. Kernel. « Current forecast of COVID-19 in Mexico : A Bayesian and machine learning approaches ». In : *PloS one* 17.1 (2022), e0259958.

[35] T. Kohonen. « Self-organizing feature maps ». In : *Self-organization and associative memory*. Springer, 1989, p. 119-157.

[36] B. Latosinska. « Machine learning-based object detection and user behavior classification using virtual reality landscape images and head motion data ». Thèse de doct. Mai 2020. DOI : 10.13140/RG.2.2.31619.53288.

[37] Y. LeCun et al. « Backpropagation applied to handwritten zip code recognition ». In : *Calcul neuronal* 1 (1989), p. 541-551. DOI : 10.1162/neco.1989.1.4.541.

[38] Y. LeCun et al. « Gradient-based learning applied to document recognition ». In : *Proceedings of the IEEE* 86.11 (1998), p. 2278-2324.

[39] P. Liu et al. « Research on the multi-attribute decision-making under risk with interval probability based on prospect theory and the uncertain linguistic variables ». In : *Knowl.-Based Syst.* 24 (mai 2011), p. 554-561. DOI : 10.1016/j.knosys.2011.01.010.

[40] T. Ma et al. « A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks ». In : *Sensors* 16.10 (2016). ISSN : 1424-8220. DOI : 10.3390/s16101701. URL : https://www.mdpi.com/1424-8220/16/10/1701.

[41] T.H. Nguyen et P. Haddawy. *Un système d'aide à la décision pour l'évaluation des candidatures d'étudiants internationaux*. 2007, F2A-1.

[42] H. Nugroho et al. *Fully Convolutional Variational Autoencoder For Feature Extraction Of Fire Detection System*. 2020.

[43] M. Parizeau. *Réseaux de neurones artificiels*. 2020.

[44] Data analytics post. *Classifieur bayésien*. 2021 (consulté le 19 juillet 2021).

[45] B. Qin, Y. Xia et F. Li. *DTU : A decision tree for uncertain data*. 2009, p. 4-15.

[46] Z. Rezgui. *Rapport de stage : Détection et classification de visages pour la description de l'égalité femme-homme dans les archives télévisuelles*. Nov. 2019. doi : 10.13140/RG.2.2.25957.76005.

[47] C. Romero et al. *HandBOOK of educational data mining*. CRC press, 2010.

[48] R.Rakotomalala. *Naive Bayes classifier explained*.

[49] D.E. Rumelhart, G. HintonE et J.L. McClelland. « The appeal of parallel distributed processing ». In : *Traitement distribué parallèle : Explorations dans la microstructure de la cognition* 1 (1986). Sous la dir. de MA : MIT Press Cambridge.

[50] A. Sharabiani et al. *Un modèle de réseau bayésien amélioré pour la prédiction des performances académiques des étudiants dans les programmes d'ingénierie*. 2014, p. 832-837.

[51] S.Theodoris. *Machine Learning (Second Edition)*. Second Edition. Academic Press, 2020, p. iv. isbn : 978-0-12-818803-3. doi : https://doi.org/10.1016/B978-0-12-818803-3.00003-9. url : https://www.sciencedirect.com/science/article/pii/B9780128188033000039.

[52] P. Tan et al. *Introduction to Data Mining*. Mai 2005.

[53] P.D. Truong, M. Packianather M. et A. Afify. *Réseaux de neurones artificiels*. Sous la dir. de Springer. 2007, p. 67-92.

[54] S. Tsang et al. « Arbres de décision pour les données incertaines ». In : *IEEE Transactions on Knowledge and Data Engineering* 23 (2011), p. 64-78. doi : 10.1109/TKDE.2009.175.

[55] B. Widrow et E. H. Marcian. *Circuits de commutation adaptatifs*. Rapp. tech. Stanford Univ Ca Stanford Electronics Labs, 1960.

[56] X. Yuni X. et X. Bowei. « Conceptual Clustering Categorical Data with Uncertainty ». In : *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)* 1 (2007), p. 329-336.

[57] K. Xu et al. *Show, attend and tell : Neural image caption generation with visual attention*. 2015, p. 2048-2057.

Webographie

[58] https://www.becoz.org/these/memoirehtml/ch06s04.html, consulté le 19 juillet 2022.

[59] https://coronavirus.gob.mx/datos/, consulté le 15 janvier 2022.

Annexe A

```
Etude comparative sur MNIST

1.1 IMPORTATION DES MODULES

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

import sys,os

from skimage.io import imread, imshow

from skimage import exposure

from skimage.color import rgb2gray

from importlib import reload


1.2 Importation des donnée avec Keras

(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()


x_train = x_train.reshape(-1,28,28,1)

x_test  = x_test.reshape(-1,28,28,1)


print("x_train : ",x_train.shape)

print("y_train : ",y_train.shape)

print("x_test  : ",x_test.shape)

print("y_test  : ",y_test.shape)


1.3 Preparation des données

xmax=x_train.max()

x_train = x_train / xmax

x_test  = x_test  / xmax
```

1.4 Représentation

```
import matplotlib.pyplot as plt
plt.figure(figsize=(7.195, 3.841), dpi=100)
for i in range(200):
  plt.subplot(10,20,i+1)
  plt.imshow(x_train[i,:].reshape([28,28]), cmap='gray')
  plt.axis('off')
plt.show()
```

2. NAIVE BAYES

2.1 Modèle

```
from sklearn.naive_bayes import BernoulliNB
model_BernoulliNB =BernoulliNB()
#model_BernoulliNB =BernoulliNB(priors =None , var_smoothing=0.1)
```

2.2 Apprentissage du modèle

```
x_train = x_train.reshape((60000,784))
x_test = x_test.reshape((10000,784))

#training
model_BernoulliNB.fit(x_train, y_train)
BernoulliNB()
```

2.3 Prédiction et évaluation du modèle

```
#prédiction
prediction = model_BernoulliNB.predict(x_test)
```

```
#evaluation du modèle
from sklearn.metrics import accuracy_score
precision = accuracy_score(y_test, prediction)*100
print(precision)

cm = confusion_matrix(y_test, prediction)
#cm # X-axis Predicted vs Y-axis Actual Values
matplot.subplots(figsize=(10, 6))
sb.heatmap(cm, annot = True, fmt = 'g')
matplot.xlabel("Predicted")
matplot.ylabel("Actual")
matplot.title("Confusion Matrix")
matplot.show()

#Evaluation
from sklearn.metrics import r2_score,mean_squared_error,accuracy_score,
classification_report
print(classification_report(y_test,prediction))
```

3. Perceptron multicouches (PMC)

3.1 Modèle

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Flatten(input_shape=[28,28]))
model.add(tf.keras.layers.Dense(256, activation="relu"))
model.add(tf.keras.layers.Dense(128, activation="relu"))
model.add(tf.keras.layers.Dense(10, activation="softmax"))
model.summary()
```

```
model.compile(
    loss="sparse_categorical_crossentropy",
    optimizer="adam",
    metrics=["accuracy"]
)
```

3.2 Entrainement du modèle

```
history = model.fit(x_train,
                    y_train,
                    epochs=10,
                    validation_split=0.2)
```

4. Réseaux de neurones convolutifs

4.1 Modèle et paramètres

```
model = keras.models.Sequential()

#Couche d'entrée
model.add( keras.layers.Input((28,28,1)) )

#couche convolutive
model.add( keras.layers.Conv2D(8, (3,3),  activation='relu') )
model.add( keras.layers.MaxPooling2D((2,2)))
model.add( keras.layers.Dropout(0.2))

#couche convolutive
model.add( keras.layers.Conv2D(16, (3,3), activation='relu') )
model.add( keras.layers.MaxPooling2D((2,2)))
model.add( keras.layers.Dropout(0.2))
```

```
#couches cachées
model.add( keras.layers.Flatten())
model.add( keras.layers.Dense(100, activation='relu'))
model.add( keras.layers.Dropout(0.5))

model.add( keras.layers.Dense(10, activation='softmax'))
model.summary()

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

4.2 Entrainement du modèle

```
batch_size  = 512
epochs      =  16

history = model.fit(x_train, y_train,
                    batch_size      = batch_size,
                    epochs          = epochs,
                    verbose         = 1,
                    validation_data = (x_test,y_test))
```

4.3 Evaluation du modèle

```
score = model.evaluate(x_test, y_test, verbose=0)

print(f'Test loss     : {score[0]:4.4f}')
```

```
print(f'Test accuracy : {score[1]:4.4f}')
Test loss     : 0.0380
Test accuracy : 0.9869
```

5. Réseaux bayésiens

5.1 Modèle et paramètres

```
pip install pyAgrum

#Importation des packages
import pyAgrum as gum
import pyAgrum.lib.notebook as gnb
from pyAgrum.skbn import BNClassifier
import matplotlib.pyplot as plt
import sklearn
from sklearn import datasets, metrics
from sklearn.model_selection import train_test_split
import pandas as pd
```

5.2 Entrainement du modèle

```
clf = BNClassifier(learningMethod='MIIC', aPriori='Smoothing',
aPrioriWeight=1,discretizationNbBins=3,discretizationStrategy=
"kmeans",discretizationThreshold=10)

clf.fit(x_train, y_train)
predicted = clf.predict(x_test)

disp = metrics.plot_confusion_matrix(clf, x_test, y_test)
disp.figure_.suptitle("Matrice de confusion")
```

```
plt.show()
print(f"Classification report for classifier {clf}:\n"
f"{metrics.classification_report(y_test, predicted)}\n")
gnb.show(clf.bn,size="8!")
```

Annexe B

Etude comparative sur les données de la covid

1. Préparation et analyse des données

1.1 IMPORTATION DES MODULES

```
from sklearn.preprocessing import MinMaxScaler,
StandardScaler, PolynomialFeatures

from sklearn.model_selection import train_test_split,
cross_val_score,GridSearchCV
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.feature_selection import SelectKBest, chi2,RFECV

from sklearn.metrics import r2_score,mean_squared_error,
accuracy_score,classification_report
from sklearn.metrics import confusion_matrix,plot_confusion_matrix,
roc_curve,auc,fbeta_score

from sklearn.pipeline import make_pipeline

from sklearn.naive_bayes import BernoulliNB,
MultinomialNB,CategoricalNB,GaussianNB,ComplementNB

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set();
```

```
import warnings
warnings.filterwarnings("ignore")



import numpy as np
import pandas as pd

%matplotlib inline
import matplotlib.pyplot as plt
from scipy.optimize import least_squares

import requests, zipfile, io
from datetime import date, timedelta

from matplotlib import rcParams
rcParams['figure.figsize'] = (10, 6)
rcParams['legend.fontsize'] = 16
rcParams['axes.labelsize'] = 16
```

1.2 IMPORTATION DES DONNEES

```
df  = pd.read_csv('220306COVID19MEXICO.csv',
                  encoding='latin1', low_memory=False)
df = df[(df["RESULTADO"] == 1)]
df.shape
```

1.3 APUREMENT ET TRAITEMENT DES DONNEES

```
print(df.columns.values)
['FECHA_ACTUALIZACION' 'ID_REGISTRO' 'ORIGEN' 'SECTOR' 'ENTIDAD_UM' 'SEXO'
 'ENTIDAD_NAC' 'ENTIDAD_RES' 'MUNICIPIO_RES' 'TIPO_PACIENTE'
 'FECHA_INGRESO' 'FECHA_SINTOMAS' 'FECHA_DEF' 'INTUBADO' 'NEUMONIA' 'EDAD'
 'NACIONALIDAD' 'EMBARAZO' 'HABLA_LENGUA_INDIG' 'DIABETES' 'EPOC' 'ASMA'
 'INMUSUPR' 'HIPERTENSION' 'OTRA_COM' 'CARDIOVASCULAR' 'OBESIDAD'
 'RENAL_CRONICA' 'TABAQUISMO' 'OTRO_CASO' 'RESULTADO' 'MIGRANTE'
 'PAIS_NACIONALIDAD' 'PAIS_ORIGEN' 'UCI']
#df = df0[(df0["RESULTADO"] == 1)]
df.shape
(462690, 35)
print(df['NEUMONIA'].unique())
print(df['DIABETES'].unique())
print(df['EPOC'].unique())
print(df['ASMA'].unique())
print(df['INMUSUPR'].unique())
print(df['CARDIOVASCULAR'].unique())
print(df['OBESIDAD'].unique())
print(df['RENAL_CRONICA'].unique())
print(df['TABAQUISMO'].unique())

df['FECHA_DEF'] = (df['FECHA_DEF'] == '9999-99-99').astype(int)
df
FECHA_ACTUALIZACION ID_REGISTRO ORIGEN SECTOR ENTIDAD_UM SEXO ENTIDAD_NAC
ENTIDAD_RES MUNICIPIO_RES TIPO_PACIENTE ... CARDIOVASCULAR
OBESIDAD RENAL_CRONICA TABAQUISMO OTRO_CASO RESULTADO MIGRANTE
PAIS_NACIONALIDAD PAIS_ORIGEN UCI

462690 rows × 35 columns
```

```
missing_neumonia       = df["NEUMONIA"].isin([99]).sum()
missing_diabetes       = df["DIABETES"].isin([98]).sum()
missing_epoc           = df["EPOC"].isin([98]).sum()
missing_asma           = df["ASMA"].isin([98]).sum()
missing_inmusupr       = df["INMUSUPR"].isin([98]).sum()
missing_hipertension   = df["HIPERTENSION"].isin([98]).sum()
missing_cardiovascular = df["CARDIOVASCULAR"].isin([98]).sum()
missing_obesidad       = df["OBESIDAD"].isin([98]).sum()
missing_renal_cronica  = df["RENAL_CRONICA"].isin([98]).sum()
missing_tabaquismo     = df["TABAQUISMO"].isin([98]).sum()
print(missing_neumonia)
print(missing_diabetes)
print(missing_epoc)
print(missing_asma)
print(missing_inmusupr)
print(missing_hipertension)
print(missing_cardiovascular)
print(missing_obesidad)
print(missing_renal_cronica)
print(missing_tabaquismo)

pos_diabetes       = df["DIABETES"].isin([1]).sum()
pos_epoc           = df["EPOC"].isin([1]).sum()
pos_asma           = df["ASMA"].isin([1]).sum()
pos_inmusupr       = df["INMUSUPR"].isin([1]).sum()
pos_hipertension   = df["HIPERTENSION"].isin([1]).sum()
pos_cardiovascular = df["CARDIOVASCULAR"].isin([1]).sum()
pos_obesidad       = df["OBESIDAD"].isin([1]).sum()
```

```
pos_renal_cronica  = df["RENAL_CRONICA"].isin([1]).sum()
pos_tabaquismo     = df["TABAQUISMO"].isin([1]).sum()
print(pos_diabetes)
print(pos_epoc)
print(pos_asma)
print(pos_inmusupr)
print(pos_hipertension)
print(pos_cardiovascular)
print(pos_obesidad)
print(pos_renal_cronica)
print(pos_tabaquismo)

clean_df=df.drop(df[ (df["NEUMONIA"] == 99)
                    | (df["DIABETES"] == 98 ) | (df["HIPERTENSION"] == 98)
                    | (df["OBESIDAD"] == 98) | (df["TABAQUISMO"] == 98)
                   ].index)
missing_neumonia       = clean_df["NEUMONIA"].isin([99]).sum()
missing_diabetes       = clean_df["DIABETES"].isin([98]).sum()
missing_epoc           = clean_df["EPOC"].isin([98]).sum()
missing_asma           = clean_df["ASMA"].isin([98]).sum()
missing_inmusupr       = clean_df["INMUSUPR"].isin([98]).sum()
missing_hipertension   = clean_df["HIPERTENSION"].isin([98]).sum()
missing_cardiovascular = clean_df["CARDIOVASCULAR"].isin([98]).sum()
missing_obesidad       = clean_df["OBESIDAD"].isin([98]).sum()
missing_renal_cronica  = clean_df["RENAL_CRONICA"].isin([98]).sum()
missing_tabaquismo     = clean_df["TABAQUISMO"].isin([98]).sum()
print(missing_neumonia)
print(missing_diabetes)
print(missing_epoc)
```

```
print(missing_asma)
print(missing_inmusupr)
print(missing_hipertension)
print(missing_cardiovascular)
print(missing_obesidad)
print(missing_renal_cronica)
print(missing_tabaquismo)

clean_df.rename(columns={'FECHA_DEF':'SURVIVED'}, inplace=True)
print(clean_df['SURVIVED'].unique())
print(len(clean_df[clean_df['SURVIVED'] == 0] ) )
print(len(clean_df[clean_df['SURVIVED'] == 1] ) )
```

```
[1 0]
50051
410554
```

1.4 ANALYSE DESCRIPTIVE

```
plot = clean_df['SURVIVED'].value_counts().plot(kind='pie', autopct='%.2f',
                                                figsize=(6, 6),
                                                title='MORTALITE')
```

```
<Figure size 432x432 with 1 Axes>
```

```
cormob_df=clean_df[["HIPERTENSION", "OBESIDAD","DIABETES","TABAQUISMO",
"SURVIVED"]]
cormob_df
```

```
HIPERTENSION OBESIDAD DIABETES TABAQUISMO SURVIVED
```

```
cormob_df.replace(to_replace = 2, value = 0, inplace = True)
```

```
cormob_df
```

HIPERTENSION OBESIDAD DIABETES TABAQUISMO SURVIVED

```
print(clean_df['SURVIVED'].unique())
print(len(clean_df[clean_df['SURVIVED'] == 0] ) )
print(len(clean_df[clean_df['SURVIVED'] == 1] ) )
```

[1 0]

50051

410554

1.4 MATRICE DE CORRELATION

```
corr_mtx=cormob_df.corr(method='spearman')
mask_ut=np.triu(np.ones(corr_mtx.shape)).astype(np.bool)
sns.heatmap(corr_mtx,
            mask=mask_ut,
            annot=True,
            cmap=sns.color_palette("coolwarm", 20))
```

<AxesSubplot:>

<Figure size 720x432 with 2 Axes>

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.pipeline import make_pipeline

import seaborn as sns; sns.set()
import matplotlib.pyplot as plt
import numpy as np
```

```
import pandas as pd

std_scl = StandardScaler()
poly = PolynomialFeatures(3)

X = cormob_df.iloc[:,0:4]
y = cormob_df['SURVIVED']



X_train_scl = std_scl.fit_transform(X) #Standardize and assign it
to a variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=123)
2. NAIVES BAYES
clf = BernoulliNB()
model_NB = clf.fit(X_train, y_train)
y_hat = model_NB.predict(X_test) #gnb
y_hat
array([1, 1, 1, ..., 1, 1, 1])
print("Prediction avec Naive Bayes:")
print()
print(classification_report(y_test, y_hat))
print(accuracy_score(y_test, y_hat))
Prediction avec Naive Bayes:
2. PERCEPTRON MULTICOUCHES
#initialisation du classifieur
from sklearn.neural_network import PMCClassifier
rna = PMCClassifier(hidden_layer_sizes=(2,),activation="relu",solver="sgd")
```

```
#apprentissage
rna.fit(X_train,y_train)
PMCClassifier(hidden_layer_sizes=(2,), solver='sgd')
#affichage des coefficients
print(rna.coefs_)
print(rna.intercepts_)
[array([[-0.27318772,  1.10260704],
       [ 0.52024543, -0.02520664],
       [ 0.63941357,  1.06724319],
       [-0.45979971,  0.23801983]]), array([[ 0.16874788],
       [-0.91542349]])]
[array([-1.03917417,  0.00132649]), array([2.63921959])]
#prédiction sur l'échantillon test
pred = rna.predict(X_test)
print(pred)
[1 1 1 ... 1 1 1]
print("Perceptron multicouches")
print()
print(classification_report(y_test, pred))
print(accuracy_score(y_test, pred))
Perceptron multicouches
```