**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université Saâd Dahlab Blida**


**Faculté des sciences**

Département Informatique

## Mémoire de fin d'études

Pour l'obtention du diplôme de Master en informatique

**Option : Ingénierie de logiciel**

**Thème :**

# *Extraction des itemsets fréquents incertains en utilisant l'apprentissage profond*

**Réalisé par :**
**ABADI sara**

**Encadré par** Mme. F.Z. Zahra

**Présidé par** Mme. ARKAM

**Examiné par** Mme.

Promotion: 2018/20

**Résumé**

L'extraction des motifs fréquents basé sur les données incertainés est une tâche devenue fréquente dans le domaine de la fouille de données. De nombreux algorithmes ont été implémentés pour trouver ces motifs dans le but de rechercher les combinaisons fréquentes d'items, en utilisant, des méthodes et outils parallèles et distribués.

Notre travail consiste d'introduire une nouvelle approche pour l'extraction des items fréquents incertains en se basant sur une nouvelle technologie appelé Deep Learning ou apprentissage profond. Cette technique à la pointe a connu un grand succès dernièrement dans plusieurs domaines.

**Abstract**

Frequent pattern extraction based on uncertain data has become a common task in the field of data mining. Many algorithms have been implemented to find these patterns in order to find common combinations of items, using parallel and distributed methods and tools.

Our job is to introduce a new approach to extracting uncertain frequent items based on a new technology called Deep Learning. This trending technique has had great success lately in several fields.

# Table des matières

# INTRODUCTION GÉNÉRALE

## 1. Contexte

L'émergence des techniques d'Extraction de la Connaissance dans les Données (ECD) est le résultat de l'accroissement de la taille des bases de données. Les données traitées par jour dépassent le milliard avec une puissance de calcul toujours plus importante (loi de Moore).

Les entreprises dans un contexte de concurrence accrue qui stockent sur supports informatiques des données informationnelles sur leur client se sont donc rapidement intéressées aux outils utilisés en ECD. Dans un but économique, les entreprises cherchent à valoriser l'information potentielle et la connaissance qu'elles ont encore non exploitées. En règle générale, ces techniques visent à découvrir des corrélations existantes dans les jeux de données et à définir des motifs fréquents dans lesquels, il est possible de tirer de l'information pertinente. Cependant, il existe des situations dans lesquelles les données sont *incertaines*.

Découvrir des ensembles d'éléments fréquents sur des bases de données incertaines est devenu un sujet brûlant dans la communauté de l'exploration de données en raison de la large application des données incertaines.

L'extraction de motifs est un domaine essentiel de Data Mining qui consiste à découvrir des motifs intéressants, inattendus et utiles dans un ensemble de données.

Le Deep Learning quant à lui est un domaine de l'intelligence artificiel qui a littéralement exploser ces dernières années. C'est un domaine d'expertise assez vaste dont l'application est exploitée dans plusieurs industries.

## 2. Problématique et motivation

Au cours de ces dernières années, le Deep Learning a joué un rôle important dans le Data Mining. En effet, plusieurs techniques de Data Mining sont issues du domaine de l'apprentissage automatique.

Le problème des données imparfaites (manquantes, imprécises, inconsistantes et incertaines), particulièrement les données incertaines est un problème largement traité dans le domaine de la fouille de données et de l'apprentissage automatique. Particulièrement, L'extraction des itemsets fréquents à partir de données incertaines a attiré beaucoup l'intention des chercheurs de la communauté de Data Mining, et par conséquent un nombre important des algorithmes a été proposé dans la littérature. Cependant, le problème de passage à l'échelle est le problème commun de ces algorithmes

Motivés par le succès de Deep Learning dans ce domaine et l'efficacité de l'application de ses modèles dans plusieurs domaines, nous nous somme poser la question suivante :

" Peut-on adapter un des modèles de l'apprentissage profond à l'extraction des itemsets fréquents à partir de données incertaines ? "

## 3. Objectifs

Le but de notre travail est en premier lieu, l'exploration de cet axe de recherche et la détermination est ce que le Deep Learning est un une méthode appropriée pour l'extraction des motifs fréquents pour les données incertaines. Cette dernière étant une méthode descriptive de Data Mining; or, le Deep Learning est appliqué généralement dans les méthodes prédictives.

En deuxième lieu l'adaptation d'un des modèles de Deep Learning ou la proposition d'un nouveau modèle qui permet d'extraire des itemsets fréquents. Autrement dit, développer une nouvelle méthode d'extraction des itemsets fréquents incertains basée sur le Deep Learning.

## 4. Organisation du mémoire

Le mémoire se reparti en cinq chapitres

**Chapitre 1 : « Extraction des motifs fréquents à partir de données incertaines »**

Ce chapitre est consacré à la présentation des données incetaines, le datamining ainsi que des algorithmes pour l'extraction des items fréquents incertains les plus connues.

**Chapitre 2 : « Les réseaux de neurones »**

Dans ce chapitre nous allons présenter les réseaux de neurones, leurs architectures et fonctionnement.

**Chapitre 3 : « les auto-encodeurs »**

Dans ce chapitre nous présenterons l'une des variantes des réseaux de neurones les plus utilisés en ce moments : les auto-encodeurs.

**Chapitre 4 : « Approche proposée »**

Ce chapitre, sera réservé pour exposer notre solution proposée et ces résultats.

**Chapitre 5 : « Tests et Expérimentation »**Le dernier chapitre concrétise et valide la méthode adopté

# CHAPITRE 1 : *Extraction des motifs fréquents à partir de données incertaines*

## I. Introduction

L'extraction de connaissances peut prendre de multiples formes, permettant de délivrer à des experts divers types de connaissances. A cet égard, les règles d'association, leurs variantes étendues plus particulièrement et les motifs fréquents de données incertaines sont des modèles fréquemment fournis aux utilisateurs finaux.

Plusieurs algorithmes sont proposés pour exploiter des ensembles d'éléments fréquents sur des données certaines. Ces dernières années, des algorithmes basés sur des arbres ont été proposés pour extraire des ensembles d'éléments fréquents à partir de données incertaines.

Dans ce chapitre nous allons étudier les algorithmes les plus connus qui ont traité ce domaine.

## II. Extraction des connaissances

Le terme ECD désignant l'extraction des connaissances à partir des données (Knowledge Discovery in Databases : KDD) est né des travaux croisés des chercheurs en statistique, en intelligence artificielle, en reconnaissance des formes, et de visualisation. Ces travaux ont pour objectif l'automatisation de processus de découverte des connaissances pertinentes nouvelles et utiles dans les bases de données gigantesques.[5]

Une autre définition dans [ZIGH 2003] annonce que : « l'ECD vise à transformer des données (volumineuses, multiformes, stockées sous différents formats sur des supports pouvant être distribués) en connaissances. Ces connaissances peuvent s'exprimer sous forme d'un concept général qui enrichit le champ sémantique de l'usager par rapport à une question qui le préoccupe. Elles peuvent prendre la forme d'un rapport ou d'un graphique. Elles peuvent s'exprimer comme un modèle mathématique ou logique pour la prise de décision. Les modèles explicites quelle que soit leur forme, peuvent alimenter un système à base de connaissances ou un système expert ».[5]

Ce processus peut se décomposer en quatre phases : [5]

1. **La sélection des données** : Cette étape concerne le filtrage des données et La réduction de la dimensionnalité des données qui se fait par l'élimination d'attributs sans intérêt, ou ayant beaucoup de valeurs erronées ou manquantes
2. **Le prétraitement des données :** Le rôle de cette étape est de préparer les données afin qu'ils soient de meilleurs qualités afin d'arriver à des résultats de qualité. Le prétraitement des données concerne, entre autres, le nettoyage des données, c'est-à-dire l'élimination du bruit, ainsi que le traitement des valeurs manquantes, ou erronées
3. **Fouille de données (Data Mining)** : Dans cette étape, des méthodes intelligentes sont utilisées afin d'extraire des modèles ou patterns. Cette étape est aussi désignée comme l'étape cœur du processus d'ECD
4. **Evaluation et interprétation des connaissances :** Les connaissances obtenues devraient être interprétables, nouvelles, valides et utiles aux utilisateurs. Ces derniers peuvent les utiliser directement, ou les incorporer dans un système de gestion de connaissances

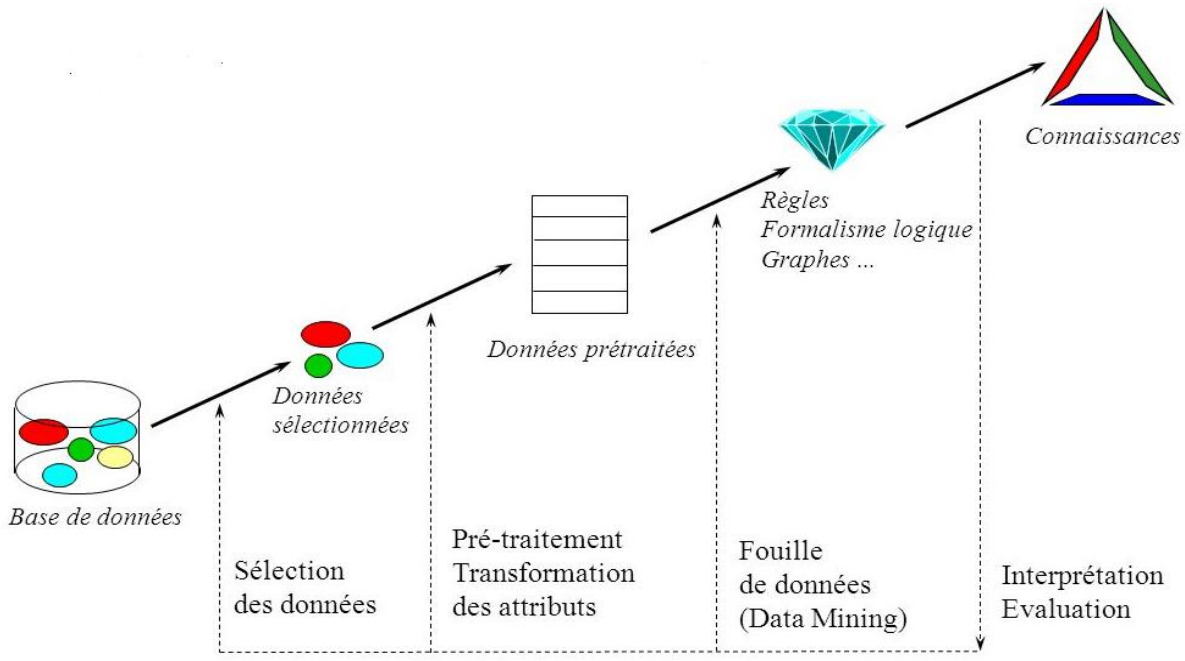



**Figure1.1. Etape du processus d'extraction des connaissances [17]**

## III. Fouille de données (Datamining)

Définition Le datamining (ou fouille de données) est l'une des étapes du processus d'extraction de Connaissances à partir de Données ECD.

Cette étape regroupe l'ensemble des méthodes et techniques destinées à l'exploration des bases de données de façon automatique, ou semi-automatique, pour détecter :

-des règles,

-des associations,

-des tendances nouvelles et imprévisibles,

-des structures particulières restituant l'essentiel de l'information utile.

Il s'agit d'un processus de sélection, exploration, modification et modélisation de grandes bases de données afin de découvrir des relations entre les données [2]

## IV. Concepts de base

Soit O un ensemble fini d'objets, P un ensemble fini d'éléments ou items, et R une relation binaire entre ces deux ensembles, le contexte formel est le triplet D = (O,P,R), ou La base de donnés D représente notre espace de travail [1]

Dans ce qui suit nous allons élucider notre travail avec un exemple d'un fichier qui comporte 10 observations (transactions) :

| S1 | S2 | S3 | S4 |
|---|---|---|---|
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 |

Tableau 1.1. exemple de fichier d'itms certains[1]

- **Transaction :** Soit T = {T1, T2, T3, …, Tn}, Ti ⊆ D. On appelle Ti un ensemble de lignes contenant les occurrences de la base de données D. Tous les Ti sont appelés des transactions. Dans l'exemple connu du panier de la ménagère, les transactions sont les tickets de caisse (c'est-à-dire les achats effectués par les clients) [1].

- **Uncertain transaction :** Une transaction incertaine t est une transaction qui contient des éléments incertains. Une base de données de transactions T contenant des transactions incertaines est appelée une base de données de transactions incertaine.

Une transaction incertaine t est représentée dans une base de données de transactions incertaines par les éléments x ∈ I associés à une valeur de probabilité existentielle 1 P (x ∈ t) ∈ [0, 1].[10]

- **Item :** On appelle item toute variable Xi représentant une occurrence de D [3]. Dans notre cas Nous avons 4 items (S1, S2, S3 et S4) dans notre fichier.

- **Itemset :** On appelle itemset, l'ensemble formé d'items. Exemple le singleton {X1} et la paire {X1, X2} sont des itemset. Un itemset de taille k est noté k-itemset[3]

**Exemple :** {S1,S2} est un itemset de cardinal CARD({S1, S2} =2.

- **Uncertain item :** Un élément incertain est un élément x ∈ I dont la présence dans une transaction t ∈ T est définie par une probabilité existentielle P (x ∈ t) ∈ (0, 1).[9]

- **Support :** Le support d'un item est égal au nombre de transactions dans lesquelles il apparaît. Par exemple, le support de {S1} est égal à 5. Le support peut être exprimé également en termes relatifs [1].

Dans ce cas, nous division le support (absolu) par le nombre total de transactions.

Pour S1, nous avons :

*Support({S1}) = 5 /10 = 20%, et Support({S1,S2}) = 3 /10 = 0.3.*

$$Support(Xi \rightarrow Xj) = \frac{Freq(Xi \cup Xj)}{card(T)} \quad (1,1)$$

ou Support (Xi →Xj) Nombre de fois où Xi et Xj apparaissent ensemble dans les transactions T.[1]

- **Confiance :** On appelle confiance le pourcentage de fois où la règle est vérifiée [1], c'est-à-dire

$$Confiance(Xi \rightarrow Xj) = \frac{Freq(Xi \cup Xj)}{Freq(Xi)} (1, 2)$$

Donc pour confiance ({S1,S2})= 3/5=0.6=60%

- **Un motif :** Un motif est un sous-ensemble de P. On dit qu'un motif P est inclus dans l'objet o (ou que o contient P) si P et o sont en relation : ∀p ∈ P, (o, p) ∈ R. Un motif de taille k est noté k-motif. Les motifs sont aussi appelés ensembles d'items (« itemsets » dans la littérature anglo-saxonne) [9]

- **Motif fréquent:** Un itemset est dit fréquent si son support est supérieur à un seuil défini à l'avance, paramètre de l'algorithme de recherche.

Dans notre exemple, en fixant le support minimum à 2 (ou 20% en relatif), nous observons dans le schéma suivant les itemsets fréquents (toutes les combinaisons non-grisées) [3].

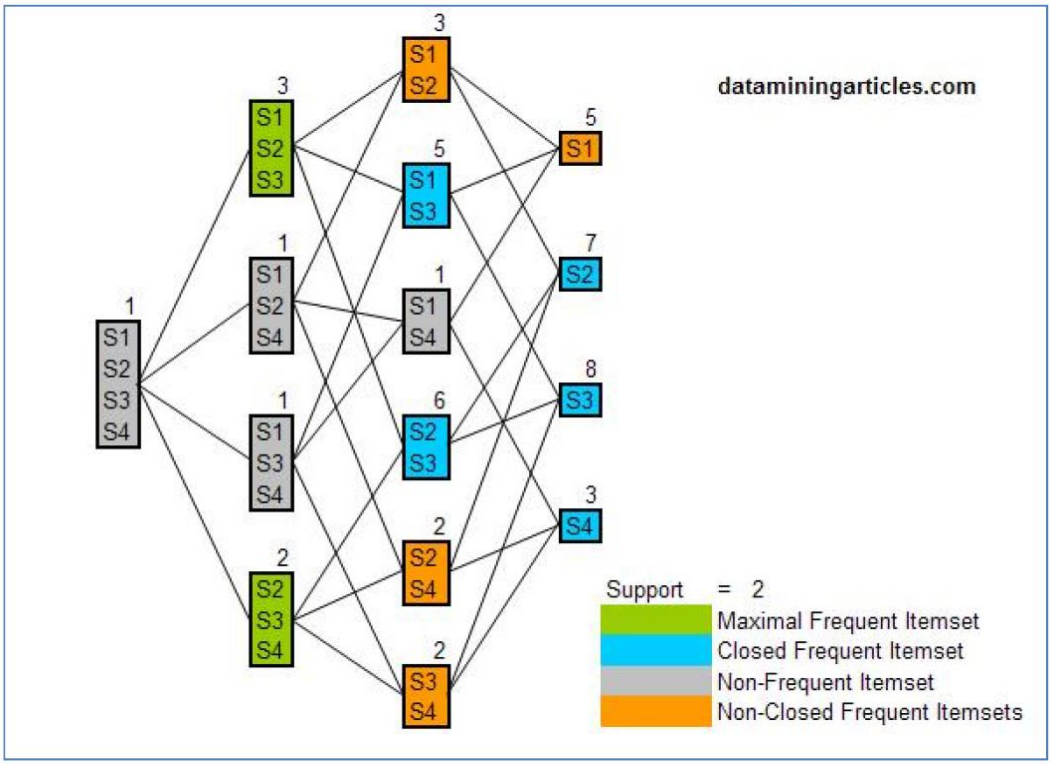



**Figure1.2. Extraction des itemsets fréquents[3]**

- **Superset :** Un superset est un itemset défini par rapport à un autre itemset, par exemple: {S1, S2, S3} est un superset de {S1, S2}.[3]

- **Itemset fermé (closed itemset) :** Un itemset fréquent est dit fermé si aucun de ses supersets n'a de support identique, autrement dit, tous ses supersets ont un support strictement plus faible [3].

Pour l'exemple ci-dessus, {*S1, S3} est fermé car aucun de ses supersets n'a de support égal à 5/10 : Support ({S1, S2, S3}) = 3/10, Support ({S1, S3, S4}) = 1/10.*

- **Maximal itemset :** Un itemset est dit maximal si aucun de ses supersets n'est fréquent [3].

Dans notre exemple ci-dessus, {S1, S2, S3} est maximal car son superset {S1, S2, S3, S4} (il n'y en a qu'un) n'est pas fréquent avec un support de 1/10.

- **Generator itemset** :Un itemset est générateur si tous ses sous-itemsets ont un support strictement supérieur [3].

Dans notre exemple :

*{S1, S2, S3} de support 4/10 n'est pas générateur puisqu'on trouve {S1, S2} avec un support identique.*

*En revanche, {S2, S4}, de support 2/10, est générateur car Support({S2}) = 7/10 et Support({S4}) = 3/10.*

# *V.* Données imparfaites

## 1. Définition

La notion d'imperfection dans les données fait appel à quatre Concepts : l'imprécision, l'incertitude, l'incomplétude et l'erreur.

Les quatre concepts d'imperfection sont définis comme ci-dessous :

**A. Imprécision :** Elle correspond à une difficulté dans l'énoncé de la donnée, soit parce que des données numériques sont mal connues causé par exemple par l'insuffisance des instruments d'observation, d'erreurs de mesure (poids à 1% près) ou encore de connaissances flexibles (la taille d'un adulte est environ entre 1.50 et 2 mètres).[7]

**B. Incertitude :** Parfois notre connaissance du réel ne peut pas être énoncée avec confiance ou garantie absolue. Car la donnée énoncée avec l'incertitude n'est pas incorrecte et ne compromet pas son intégrité. Bien que, l'âge d'une personne de 20 ou 24 ans est

imprécis. La donnée : l'âge est probablement 20 est une donnée incertaine, dans quelques cas, le degré de la certitude est donné, par exemple : l'âge est 32 ans avec une probabilité de 0.6 et 33 avec une probabilité de 0.4. [7]

**Remarque :** L'imprécision et l'incertitude sont deux notions très liées, on peut dans quelque cas modéliser l'imprécision par l'incertitude et vice versa. Plus la donnée est précise plus elle est certaine à titre d'exemple «je suis sûr que la note est entre 10 et 12 mais je ne suis pas certaine qu'elle soit 11 » ou bien « je suis certaine que je serais à l'université l'après-midi, mais je ne suis pas sûr que je serais là à 13h30min » si la valeur est précise mais pas certaine, elle est entourée par d'autres valeurs possibles ceci incrémente la certitude mais l'imprécision sera importante également.

C. **Incomplétude :** La donnée est dite incomplète si elle contient au moins une valeur manquante, dans ce cas on a uniquement une donnée partielle du réel perçu. Les incomplétudes sont :[7]

- Des absences de données ou des données partielles sur certaines caractéristiques de l'objet. Elles peuvent être dues à l'impossibilité d'obtenir certains renseignements (fichiers de malades dans lesquels certaines rubriques ne sont parfois pas remplies ) ou à un problème au moment de la capture de la donnée.
- Elles peuvent aussi être associées à l'existence de données générales sur l'état d'un système, habituellement vraies, soumises à des exceptions que l'on ne peut pas énumérer ou prévoir, selon les cas, (" généralement, Pierre est à son bureau tous les jours ", sauf s'il est malade ou si un événement grave survient dans sa famille).

**D. Erreur :** C'est le type le plus simple de donnée imparfaite. La donnée Stockée est incorrecte quand elle est différente de l'information vraie. Cependant d'après un balayage de la littérature il n'existe pas de représentation ou une modélisation précise pour les données erronées sauf que ce sont des données aberrantes. [7]

## 2. Les causes d'imperfection des données

L'imperfection des données est due à plusieurs raisons nous citons à titre d'exemple :

- L'obtention des données à partir du réel s'effectue en deux étapes : l'observation et la représentation. La première se produit à travers des intermédiaires (humains) qui sont généralement soumis à des erreurs, des imprécisions et des incertitudes. La seconde étape est celle de la représentation de ces données. Autant l'observation que la représentation entraîne une perte de données est d'autant plus grande que le système est complexe. [7][8]
- L'absence de rigueur ou de flexibilité inhérente au système lui-même et son fonctionnement, c'est le cas pour toutes les caractéristiques de phénomènes naturels tel que la durée de maturation d'un fruit, la taille d'un animal adulte, le passage progressif et non strict du jour à la nuit. [7][8]
- Certains systèmes artificiels tels que la charge maximale d'un ascenseur indiquée en kilogrammes dans un souci de simplicité mais à laquelle on peut ajouter quelques grammes sans problèmes majeur.[7][8]

## 3. Théorie des données imparfaites

Plusieurs théories traitent l'imperfection des données, et chaque théorie est plus ou moins adaptées à l'un des concepts vus précédemment, la figure ci-dessus représentent un récapitulatif de ses théories, mais vu que nous traitons dans notre étude les données incertaines alors nous allons utiliser le **modèle probabiliste** que nous allons détailler par la suite.

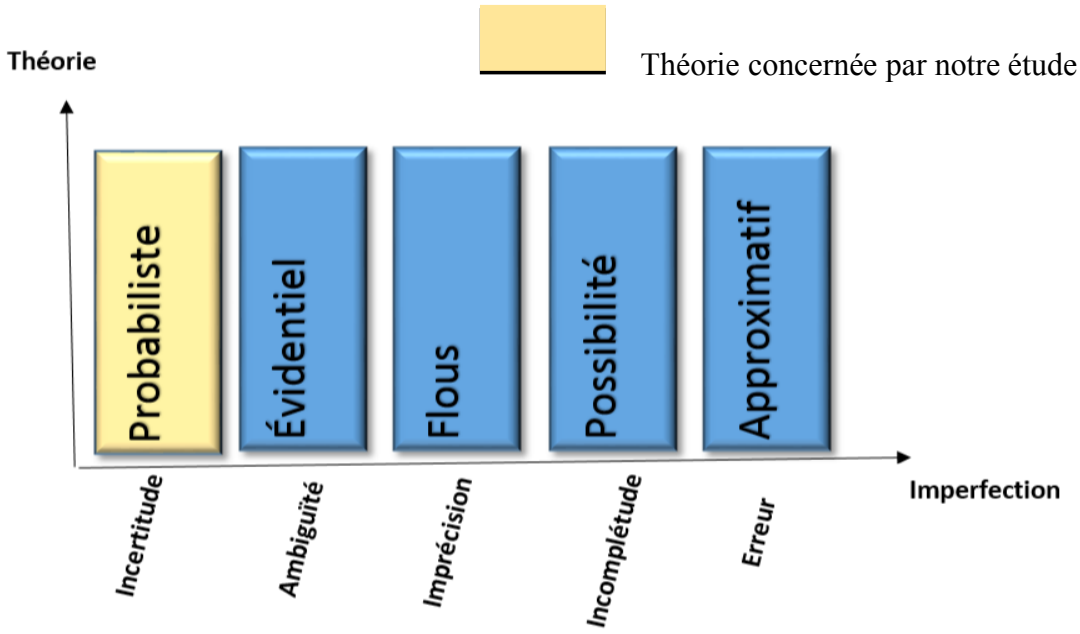



**Figure 1.3 :** Les imperfections en fonction des théories. **[8]**

## VI. Mesures de calcule des fréquences des itemsets incertains

Lors de l'extraction des motifs fréquents à partir de données incertaines, nous adoptons l'approche probabiliste, en effet dans un jeu de données probabilistes (*PD : probabilist data*) de données incertaines, l'utilisateur n'est pas certain de la présence ou l'absence d'un élément x dans une transition $t_j$. Cette incertitude de présence peut être exprimée en utilisant la probabilité existentielle $P(x, t_j)$, qui représente la probabilité que x soit présent dans PD pour une transaction $t_i$ . La valeur de $P(x, t_j)$ est comprise entre 0 et 1. [6]

Les algorithmes d'extraction des motifs fréquents à partir de données incertaines, utilisent trois mesures pour calculer les itemsets fréquent :

- Expected Support.
- Probabilistic Support.
- Evidential Support.

### 1. Expected support

Soit un ensemble de n éléments, l'**e**xpected support noté expSup (X, tj) de k-itemset (X est un sous ensemble de n), où X = {x1, x2, ..., xk}, est le produit de probabilité de tous les $P(x_i, t_j)$ pour k transactions.[6]

$$expSup(X, t_j) = \prod_{i=1}^{k} P(x_i, t_j) \quad (1, 3)$$

expSup (X) d'un ensemble probabiliste de n transactions de données incertaines peut être calculé en sommant expSup (X, tj) sur chaque transaction tj contenant X:[6]

$$expSup(X) = \sum_{j=1}^{n} expSup(X, t_j), \quad (1,4)$$

La formule la plus générale serait donc :

$$\text{expSup(X)} = \sum_{j=1}^{n} \prod_{i=1}^{k} P(x_i , t_j) \quad (1,5)$$

L'ensemble d'éléments X est considéré comme fréquent si son experted support expSup (X) dans l'ensemble de données incertain atteint ou dépasse le seuil minimum spécifié par l'utilisateur minsup.[6]

$$expSup(X) >= minsup \quad (1, 6)$$

Voici un exemple de calcule d'un expected support d'un ensemble{S1,S2}[15]

| | S1 | S2 | Weighted Support of {S1,S2} |
|---|---|---|---|
| Patient 1 | 90% | 80% | 0.72 |
| Patient 2 | 40% | 70% | 0.28 |
| Expected Support of {S1,S2} | | | 1 |

Tableau 1.2. Exemple de calcule de l'expected support [15]

## 2. Probabilistic support

Dans les bases de données de transactions incertaines, le support d'un ou plusieurs éléments ne peut pas être représenté par une valeur unique, mais plutôt, doit être représenté par une distribution de probabilité discrète. [7]

Soit T est la base de données de transaction et W est l'ensemble des éléments possibles de T, le Pi(X) est la probabilité de support d'un itemset X tel que X a le support i.

$$P(X) = \sum_{w,W,(S(X,wj)=i)} P(wj) \quad (1,7)$$

Où S (X, wj) est le support de X dans un élément wj.[12]

Soit I l'ensemble de tous éléments possibles et T représente la base de données incertaines, où une transaction tj ∈ T est un ensemble des éléments incertains, à savoir, tj ⊆ I. Contrairement à la base de données précise traditionnelle, chaque item xi ∈ tj est associé à une probabilité existentielle P (xi, tj) ∈ [0,1], qui dénote la probabilité que xi soit réellement présent dans tj. Pour un itemset X ⊆ tj, basé sur l'hypothèse commune que les éléments de X sont indépendants, la probabilité existentielle est alors : [11]

$$P (X \subseteq tj) = \prod x \in X \; P (x, tj). \quad (1,8)$$

Le support attendu (Expected support) de X est alors la somme de la probabilité existentielle sur toutes les transactions. [11].

$$expSup (X) = \sum tj \in T \; P (X \subseteq tj). \quad (1 ,9)$$

Pour une base de données de transactions incertaine T avec des transactions mutuellement indépendantes et tout 0 ≤ i ≤ | T |, le support probabilistic Pi (X) peut être calculée comme suit:[12]

$$Pi(X) = \sum_{\substack{s\subseteq T \\ |S|=i}} \left( \prod_{t\in S} P(X \subseteq t) \prod_{t\in T-S} (1 - P(X \subseteq t)) \right) \quad (1,10)$$

**Remarque :** Un itemset X est un fréquent probabiliste si son existence dans une transaction minsup est supérieure ou égale au seuil d'utilisateur spécifique minprob.[7]

$$P\ (sup(X) >= minsup) >= minprob \quad (1,11)$$

On donne :

(i) une base de données de transaction incertaine T.

(ii) un seuil minimum de support minsup.

(iii) un seuil minimum de la probabilité fréquente minprob, le problème d'extraction de motifs fréquents probabilistes est de trouver tous et seuls les motifs fréquents probabilistes ; à savoir, trouver tous les X tel que P (X) ≥ minprob.

## 3. Evidential support

Une base de données évidentielles (notée EDB) permet le stockage de données incertaines et imprécises qui sont modélisées à l'aide de la théorie de Dempster-Shafer. Une base de données évidentielles notée EDB contient n attributs et d lignes. Chaque attribut i (1 ≤ i ≤ n) a un domaine Θi de valeurs discrètes.

La fonction de masse élémentaire m d'une hypothèse X ⊆ Θ, notée m(X), représente la partie du degré de croyance placée sur X et qui n'a pas été distribuée aux sous-ensembles de X. La masse de croyance élémentaire, notée bba (bba : basic belief assignment) est définie par :[13]

$$m : 2^{\Theta} \rightarrow [0, 1]$$

où l'espace puissance 2 Ω = {∅, {ω1}, {ω2}, {ω1, ω2}, {ω3}, {ω1, ω3}, {ω2, ω3}, {ω1, ω2, ω3}, . . . , {ω1, . . . , ωK}} rassemble tous les sous-ensembles possibles formées des hypothèses et unions d'hypothèses de Ω.les éléments de m sont définis par :[14]

$$m_{ij} : 2^{\Theta_i} \rightarrow [0, 1] \; avec :$$

$$m_{ij}(\emptyset) = 0 \; et \sum_{x\subseteq\Theta_i} m_{ij}(x) = 1 \quad (1,12)$$

La fonction de croyance (bel) est définie à partir de la fonction de masse m, la somme des masses de tous les sous-ensembles de A.

$$bel(A) = \sum_{B \subseteq A} m(B) \quad (1,13)$$

Support d'un itemset évidentiel : Soit X un itemset évidentiel dans EDB. Le support de X est sa croyance dans la base.[13]

$$Support(X) = Bel_{EDB}(X) \quad (1,14)$$

Un itemset évidentiel fréquent est un itemset dont le support excède le seuil minimum de support. Soit X un itemset évidentiel et Θ le produit cartésien des domaines des attributs d'EDB. L'ensemble F est défini ainsi :[13]

$$F = \{X \subseteq \Theta / Support(X) \geq min_{sup}\} \quad (1,15)$$

## VII. Les algorithmes d'extraction des itemsets fréquents

Un survol des algorithmes d'extraction des motifs fréquents parus dans la littérature a montré que la majorité de ces algorithmes sont des extensions de trois méthodologies principales à savoir : Apriori, FP-growth et Eclat.

Nous avons recensé ces algorithmes avec les mesures de calcule vu précédemment pour faire l'extraction des itemsets fréquents à partir de données incertaines comme vu dans la figure 1.4.

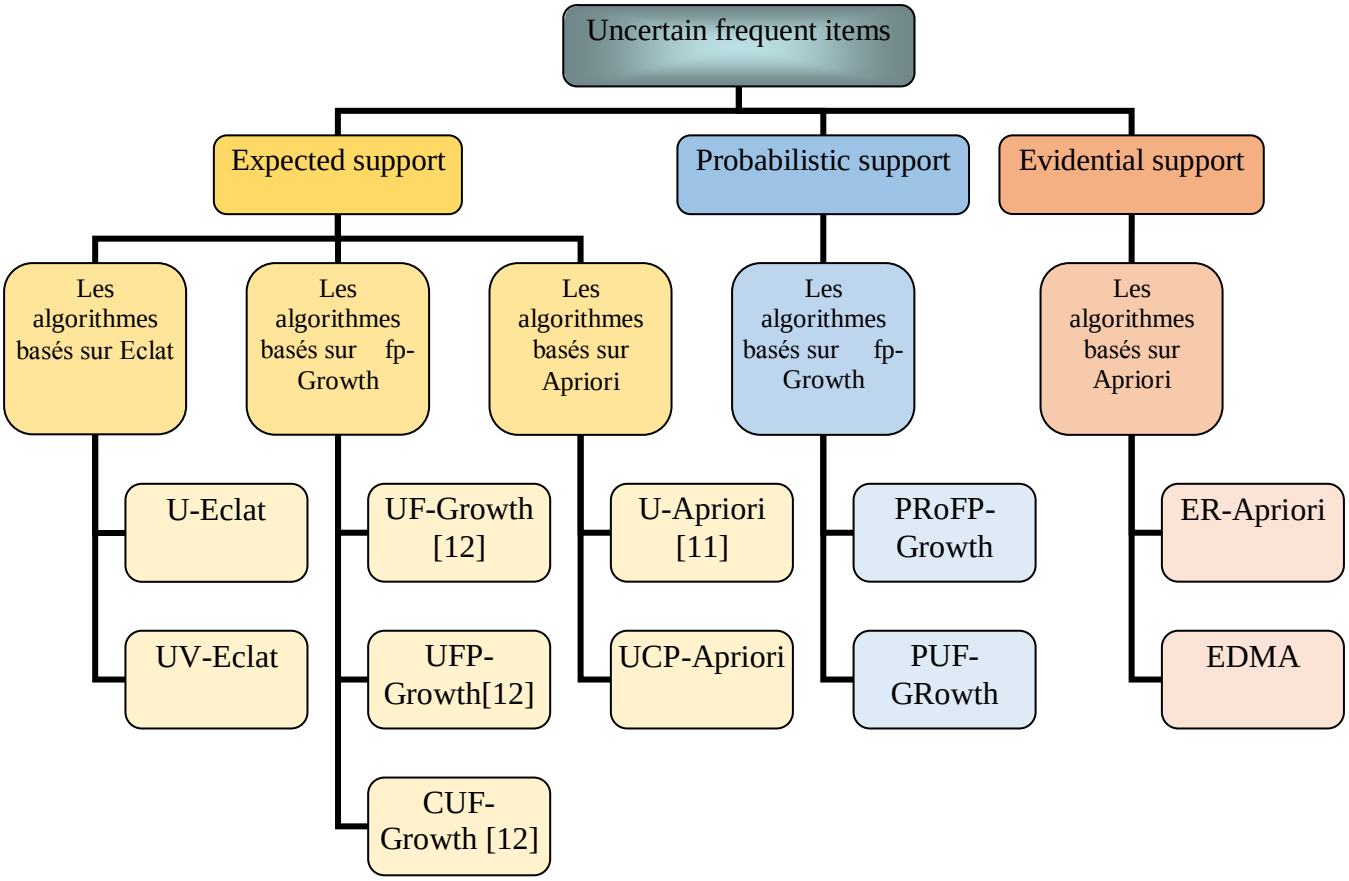



**Figure 1.4 :** Classification des algorithmes selon le support utilisé.

## 1. U- Apriori

U-Apriori est la version incertaine d'Apriori algorithme. Pour faire face au problème des fréquentes itemsets sous les données incertaines de l'Apriori l'algorithme a été modifié pour l'algorithme U-Apriori. R. Agrawal a été le premier à proposer cet algorithme U-Apriori[16]

L'algorithme U-Apriori, qui est une modification de l'algorithme Apriori. Plus précisément, au lieu d'incrémenter le support des modèles candidats, U-Apriori incrémente l'expected support.[11]

**U-Apriori algorithm :[12]**

L'algorithme prend en entrée un expected support minimum en entrée(MS)

**Étape -1**: Scannez la base de données de transactions pour obtenir l'expected support S de chaque jeu de 1 élément, comparez S avec le support minimum (MS) et obtenez un ensemble de jeux de 1 élément fréquents, L1.

**Étape -2**: utilisez Lk-1 pour générer un ensemble de condidats de k itemset. Et utilisez la propriété Apriori pour élaguer les ensembles d'éléments k non fréquentés de cet ensemble.

**Étape -3**: Scannez la base de données de transactions pour obtenir l'expected support S de chaque ensemble de k-articles candidat dans le jeu final, comparez S avec MS et obtenez un ensemble de k-itemsets fréquents, Lk.

**Étape -4**: pour chaque ensemble d'éléments fréquents l, générez tous sous-ensembles non vides de l.

**Étape -5**: pour chaque sous-ensemble non vide de l, sortez la règle "s => (l-s)",

U-**Apriori souffre des problèmes suivants:** [11]

(i) héridité :de l'algorithme Apriori, U-Apriori n'évolue pas bien lors de la manipulation de grandes quantités de données car il suit également un cadre de génération et de test de niveau.

(ii) Si les probabilités existentielles de la plupart des éléments d'un motif X sont petites, des incrémentations pour chaque transaction peut être insignifiante. Par conséquent, de nombreux candidats sont connu comme infréquent jusqu'à ce que la plupart (sinon la totalité) des transactions soient traitées.

## 2. UF-growth

Pour représenter efficacement les données incertaines, nous proposons un UF-tree qui est une variante de l'arbre uf-tree. Chaque nœud dans UF-tree stock un item (un article), son Expected support, et le nombre d'occurrences de ces expected support de cet article. [12]

**UF-Growth algorithm[12]**

Construction de l'UF-tree comme suit :

**Etape1 :** Il scanne la base de données une fois et accumule le support attendu (expected support) de chaque item. Par conséquent, il trouve tout articles fréquents (articles ayant un expected support ≥minsup).

**Etape2 :**Il trie ces éléments fréquents par ordre décroissant d'expected support

l'algorithme analyse ensuite la base de données la deuxième fois et insère chaque transaction dans UF-tree dans un,mode similaire à celui de la construction d'un FP-tree

**Etape3 :**Une fois l'arbre UF construit, l'algorithme extrait récursivement les motifs fréquents de cet arbre dans un mode comme dans FP-Growth à l'exception de ce qui suit:

La nouvelle transaction est fusionnée avec un nœud enfant de la racine de l'arbre UF que si le même élément et le même support attendu existe dans la transaction et les nœuds enfants. Avec une telle construction arborescente processus, UF-tree possède une belle propriété que le nombre d'occurrences d'un nœud est au moins la somme de nombre d'occurrences de tous ses nœuds enfants .

Pour mieux comprendre cet algorithme nous vous proposons l'exemple suivant d'une base de données incertaines :[11]

| Transactions | Contents |
|---|---|
| $t_1$ | $\{a$:0.9, $b$:0.8, $c$:0.7, $d$:0.6, $f$:0.8$\}$ |
| $t_2$ | $\{a$:0.9, $c$:0.7, $d$:0.6, $f$:0.1$\}$ |
| $t_3$ | $\{b$:0.9, $c$:0.5, $e$:0.4$\}$ |
| $t_4$ | $\{b$:0.9, $e$:0.2$\}$ |
| $t_5$ | $\{a$:0.9, $c$:0.7, $d$:0.6, $e$:0.3$\}$ |

***Tableau 1.3*** transactions avec données incertaines [11]

Apres applications de UF-trees nous aurons les résultats suivants :

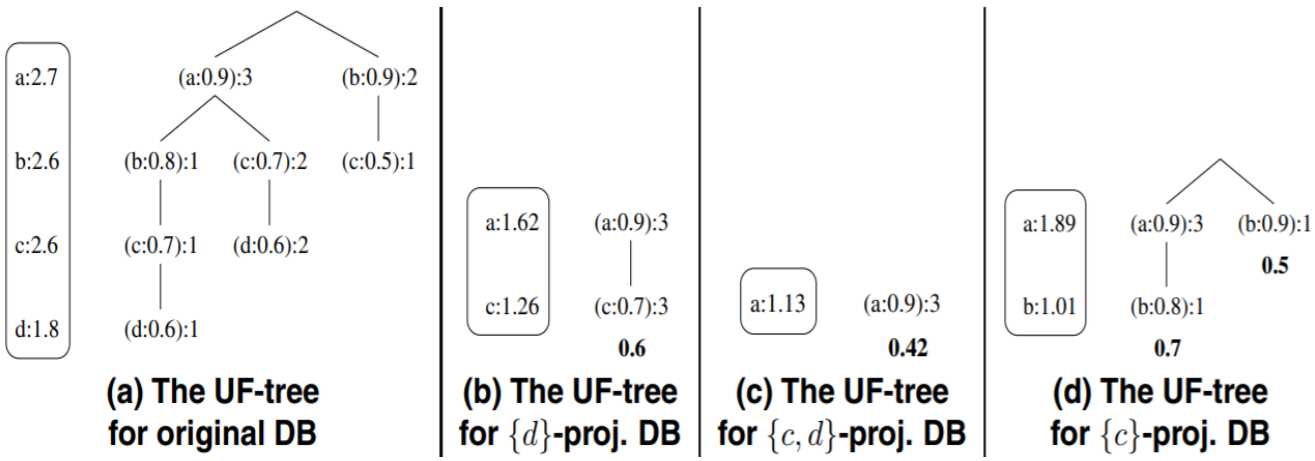



**Figure 1.5:** UF-trees [11]

En appliquant notre algorithme UF-Growth, à UF-tree (illustré à la figure précédente) qui capture le contenu de données incertaines dans l'exemple, nous avons trouvé des modèles fréquents suivant :{a}: 2,7, {b}: 2,6, {c}: 2,6, {d}: 1,8, {a, c}: 1,89, {a, d}: 1,62, {a, c, d}: 1,13,{b, c}: 1,01 et {c, d}: 1,26.

## 3. PUF-Growth

Pour réduire la taille des arbres UF et UFP, Leung a proposé **prefix-capped uncertain frequent pattern tree (PUF-tree)** dans laquelle des informations importantes sur les données incertaines sont capturé de sorte que les modèles fréquents peuvent être extraits de l'arbre.

PUF-tree est construit en considérant une borne supérieure de la valeur de probabilité existentielle pour chaque élément lors de la génération d'un ensemble de k éléments (où k>1). La limite supérieure d'un élément xr dans une transaction tj est le plafond d'élément (item cap) (préfixe) de xr en tj, tel que défini comme suit,[12]

**L'item cap (préfixe) :** $I^{cap}$ (xr, tj) d'un élément xr dans une transaction tj = x1,…, xr,…., xh}, où 1 <r <h, est défini comme le produit de P (xr, tj) et la valeur de probabilité existentielle la plus élevée M des éléments x1 à xr-1 en tj (c'est-à-dire dans le préfixe approprié de xr en tj)[12]

$$I^{cap}(x_r,t_j) = \begin{cases} P(x_r,t_j) \times M \; if \; h > 1 \\ P(x_1,t_j) \qquad if \; h = 1 \end{cases}, \text{ where M} = max_{1 \le q \le r-1} \, P(x_q,t_j) \quad (1, 16)$$

L'expected support $expSup^{cap}$ (X) d'un modèle X = {x1, ……, xk} (où k> 1) est défini comme la somme de tous les item caps de xk dans toutes les transactions contenant X:

$$expSup^{cap}(X) = \sum_{j=1}^{n} \{ (I^{cap}(x_k,t_j) | X \subset t_j) \} \quad (1, 17)$$

Pour mieux comprendre cette notions d'item cap on considère l'exemple vu précédemment (tableau 1.1) :

Item cap de c dans t1 est :

$$I^{cap}(c,t_1) = 0.7 \times \max\{P(a,t_1), P(b,t_1)\} = 0.7 \times \max\{0.2,0.2\} = 0.7 \times 0.2 = 0.14$$

Item cap de f dans t1 est :

$$I^{cap}(f,t_1) = 0.8 \times \max\{P(a,t_1), P(b,t_1), P(c,t_1)\} = 0.8 \times \max\{0.2,0.2,0.7\} = 0.8 \times 0.7 = 0.56$$

**PUF-tree Algorithm[12]**

**Etape 1 :** Lors de la première analyse de la base de données, recherchez éléments fréquents distincts dans la base de données

**Etape 2 :** construire une table d'en-tête appelée I-list pour stocker seuls les articles fréquents dans un ordre cohérent (par exemple, ordre canonique) pour faciliter la construction de l'arbre.

**Etape 3 :** L'arbre PUF réel est construit avec le deuxième scan (analyse) de base de données d'une manière similaire à celle de FP-tree.

**Etape 4 :** lors de l'insertion d'un élément de transaction, d'abord calculer l'item cap, puis l'insérer dans l'arbre PUF selon l'ordre de la liste I.

**Etape 5 :** Si ce nœud existe déjà dans le chemin, mettre à jour son item cap en ajoutant le l'item cap calculé à l'item cap existant.

Sinon, créez un nouveau nœud avec cette nouvelle valeur d'item cap.

Appliquons cet algorithme (puf-tree) à notre exemple (voir tableau 1.1)

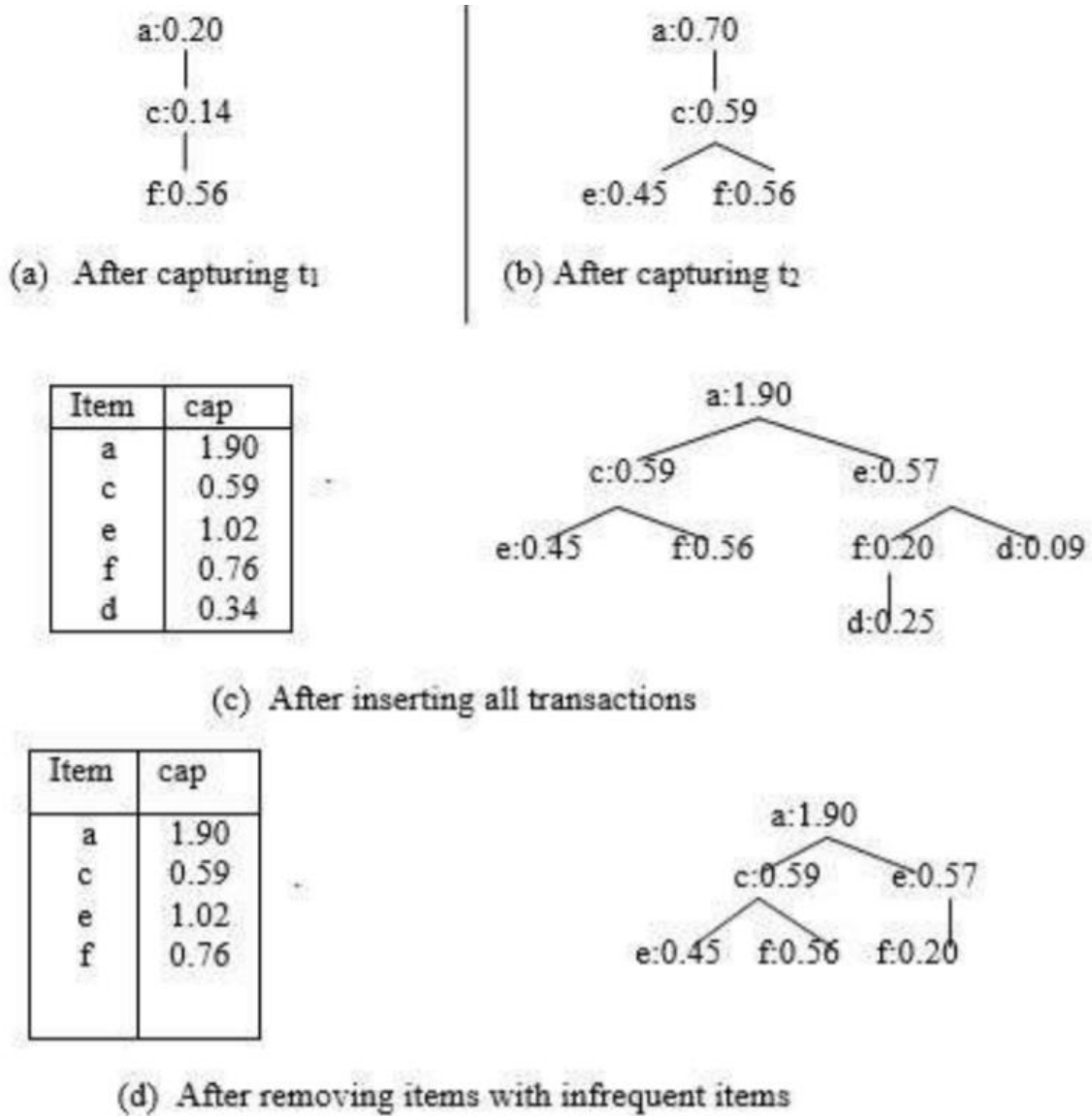



**Figure 1.6 :** PUF-tree avec minsup =0,5 [12]

## *VIII. Comparaisons entre algorithmes*

Dans le tableau suivant nous allons comparer les algorithmes vus précédemment [12]

| ALGORITHME | AVANTAGES | INCONVENIENT |
|---|---|---|
| **U-Apriori** | Cet algorithme réduit considérablement la taille de ensemble de candidats. | Il scanne la base de données plusieurs fois et donc la performance est affecté. |
| **UF-growth** | Cet algorithme utilise des arbres UF-tree pour exploiter fréquemment modèles de bases de données incertaines en scannant deux fois la base de données. | Il contient un chemin d'arbre distinct pour chaque article distinct (distinct item) |
| **PUF-growth** | Cet algorithme explore les modèles fréquents en construisant une base de données projetée pour chaque modèle fréquent potentiel et explore récursivement ses extensions fréquentes potentielles. | il crée des faux positifs |

## *IX. Conclusion*

Dans ce chapitre nous avons vu les concepts de données incertaines ainsi que les principaux algorithmes d'extractions d'itemsets fréquents à partir de données incertaines et leurs fonctionnements, et dans ce qui suit dans nous allons étudier les réseaux de neurones.

# I. Introduction

Les réseaux de neurones, inspirés de la structure du cerveau humain, sont au cœur des progrès récents de l'intelligence artificielle. Dotés de capacités impressionnantes, ils arrivent à reconnaître des images avec grande précision et sont utilisés dans les voitures autonomes. Ils peuvent lire et écrire et même jouer à des jeux vidéo

Ces réseaux utilisent des principes mathématiques relativement simples pour représenter ces connaissances pourtant complexes. Dans ce qui suit nous allons voir certains de ces concepts les plus importants.

# *II. Définitions*

## 1. Intelligence Artificiel

Créée en 1956 par John McCarthy, l'intelligence artificielle fait appel à des machines capables d'accomplir des tâches caractéristiques de l'intelligence humaine. Bien que cela soit plutôt général, cela inclut des choses comme la planification, la compréhension du langage, la reconnaissance des objets et des sons, l'apprentissage et la résolution de problèmes [43].

On peut classer l'IA en deux catégories, générale et étroite. L'IA générale aurait toutes les caractéristiques de l'intelligence humaine, y compris les capacités mentionnées ci-dessus. L'intelligence artificielle étroite présente certaines facettes de l'intelligence humaine, et peut très bien faire cette facette, mais elle fait défaut dans d'autres domaines. Une machine qui reconnaît très bien les images, mais rien d'autre, serait un exemple d'IA étroite [38].

## 2. Machine Learning

Au fond, l'apprentissage automatique n'est qu'un moyen d'atteindre l'IA. Arthur Samuel a inventé la phrase peu de temps après AI, en 1959, la définissant comme " la capacité d'apprendre sans être explicitement programmé ". Vous pouvez obtenir l'IA sans utiliser l'apprentissage automatique, mais cela nécessiterait la construction de millions de lignes de codes avec des règles et des arbres de décision complexes [4][38].

Au lieu de coder des logiciels avec des instructions spécifiques pour accomplir une tâche particulière, l'apprentissage automatique est une façon de " entraîner " un algorithme afin qu'il puisse apprendre comment. " L'entraînement " consiste à fournir d'énormes quantités de données à l'algorithme et à permettre à l'algorithme de s'ajuster et de s'améliorer [4][38], le volume des données nécessaires aux algorithmes de Machine Learning étant très grand, on associe souvent Machine Learning avec BigData.[19]

Autrement dit, la Machine Learning est une discipline consacrée à l'analyse des données. Le but de cette discipline est de créer de la connaissance de manière automatique à partir de données brutes. Cette connaissance (ou modèle) peut alors être exploitée pour prendre des décisions. On parle parfois de stratégie pilotée par les données (data-driven strategy) pour une entreprise[19].

## 3. Deep Learning

L'apprentissage profond ou le *deep learning* est une technique d'apprentissage machine qui vise à entraîner un système pour qu'il résolve des situations sans que tous les paramètres nécessaires à la résolution du problème n'aient été calculés par l'implémentateur. L'objectif est d'entraîner un algorithme à paramètres variables (une « boîte noire ») à prendre une décision correcte concernant une tâche donnée. L'apprentissage se fait en optimisant les paramètres variables pour améliorer les décisions prises.[20]

L'apprentissage profond est l'une des nombreuses approches de l'apprentissage automatique. D'autres approches comprennent l'apprentissage par arbre de décision, la programmation logique inductive, le clustering, l'apprentissage par renforcement et les réseaux bayésiens, entre autres [4][38].

L'apprentissage profond a été inspiré par la structure et la fonction du cerveau, à savoir l'interconnexion de nombreux neurones. Les réseaux de neurones artificiels (RNA) sont des algorithmes qui imitent la structure biologique du cerveau [4][38].

Dans les RNA, il y a des "neurones" qui ont des couches discrètes et des connexions à d'autres "neurones". Chaque couche choisit une caractéristique spécifique à apprendre, comme les courbes/bords dans la reconnaissance d'image. C'est cette superposition qui

donne à l'apprentissage profond son nom, la profondeur est créée en utilisant plusieurs couches plutôt qu'une seule couche [38]

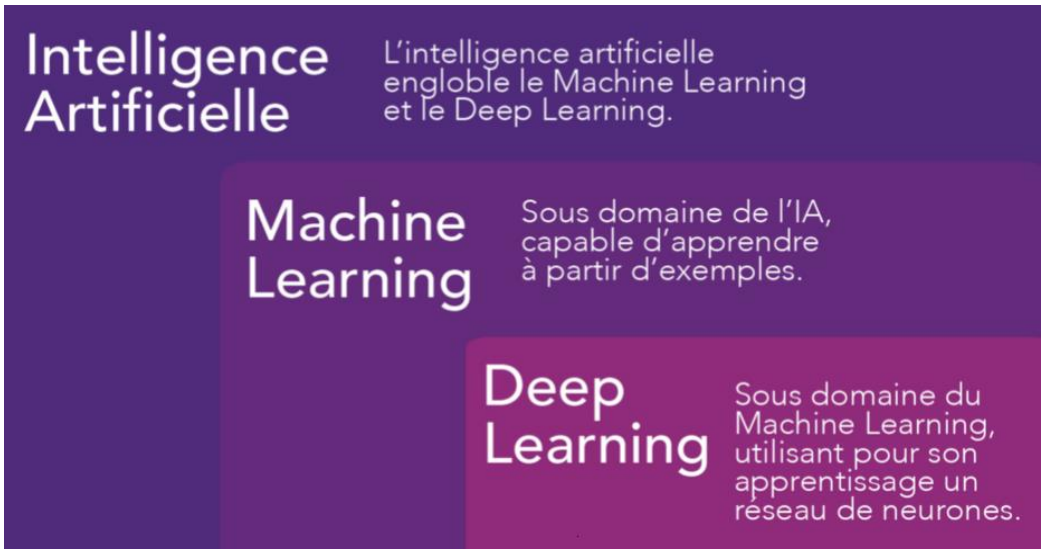



***Figure 2. 1 :*** *Différence entre intelligence artificiel deep learning et machine learning [39].*

# III. Types d'apprentissage automatique

## 1. L'apprentissage supervisée

L'apprentissage supervisé ajuste les paramètres du réseau par une comparaison directe entre la sortie réseau réelle et la sortie souhaitée. L'apprentissage supervisé est un système de rétroaction en boucle fermée, où l'erreur est le signal de rétroaction. La mesure d'erreur, qui montre la différence entre la sortie réseau et la sortie des échantillons de l'entrainement La mesure de l'erreur est habituellement définie par l'erreur quadratique moyenne (EQM).

$$E = \frac{1}{N} \sum_{p=1}^{N} \| \boldsymbol{y}_p - \hat{\boldsymbol{y}}_p \|^2 \quad (2,1)$$

Où N est le nombre de paires de motifs dans l'ensemble d'échantillons, $y_p$ est la partie de sortie de la p-éme paire de motifs, et $\hat{}y_p$ est la sortie réseau correspondant à la paire de motifs p. L'erreur E est calculée à nouveau après chaque époque. Le processus d'apprentissage se termine lorsque E est suffisamment petit ou lorsqu'un critère d'échec est satisfait [20][21][22].

Nous pouvons résumés l'apprentissage supervisé comme étant un processus en deux étapes :

1. Construction du modèle à partir de l'ensemble d'apprentissage (training data set).
2. Utilisation du modèle pour la classification de nouvelles données [23]

Voici quelques exemples populaires d'algorithmes d'apprentissage automatique supervisé:[24]

- Arbres de décision
- K Nearest Neighbours
- SVC linéaire (classificateur de vecteur de support)
- Régression logistique
- Naive Bayes
- Les réseaux de neurones
- Régression linéaire
- Régression vectorielle de support (SVR)
- Arbres de régression

## 2. Apprentissage non supervisé

L'apprentissage non supervisé (Unsupervised Learning) consiste à ne disposer que de données d'entrée (X) et pas de variables de sortie correspondantes. L'objectif de l'apprentissage non supervisé est de modéliser la structure ou la distribution sous-jacente dans les données afin d'en apprendre davantage sur les données.[24]

Nous pouvons résumer l'apprentissage non supervisé en deux points :

• On dispose d'éléments non classés *Exemple* : mots d'un texte

• On veut les regrouper en classes *Exemple* : si deux mots ont la même étiquette, ils sont en rapport avec une même thématique...[25]

Voici quelques exemples populaires d'algorithmes d'apprentissage automatique non supervisé:[23]

- K-means
- Expectation-maximization
- Hierarchical clustering
- Self organizing map
- Isodata

## 3. L’apprentissage semi-supervisé

Le contexte semi-supervisé qui se situe à l’intersection entre le contexte supervisé et le contexte non supervisé, est alors une solution alternative. Il se caractérise par la présence de quelques informations disponibles sur l’ensemble des données. Ces informations sont représentées soit sous la forme de quelques données labellisées, soit sous la forme de ressemblance ou dissemblance au sein de couples de données.

Le contexte semi-supervisé utilise des connaissances partielles qui sont soit incomplètes (par exemple, le cas où des relations entre certains individus sont connues) ou tout simplement les exemples étiquetés ne sont pas en quantité suffisante pour que l’on puisse appliquer des algorithmes supervisés.[26]

**Remarque : classification ou régression**

Une autre distinction qui aide dans le choix d'un algorithme de machine learning est le type de sortie que l'on attend de notre programme : est-ce une valeur **continue** (un nombre) ou bien une valeur **discrète** (une catégorie) ? Le premier cas est appelé une **régression**, le second une **classification** [34] .

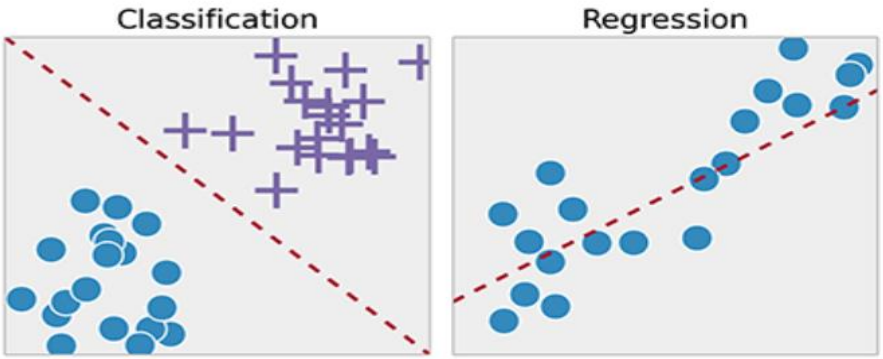



Figure 2.2 Illustration de la différence entre classification linéaire et régression linéaire[34]

## 4. L’apprentissage par renforcement

L’apprentissage par renforcement, au sens général, est un cadre formel qui modélise des problèmes décisionnels séquentiels. Au sein de ce cadre, un agent* apprend à prendre des décisions optimales en interagissant avec l’environnement. Lorsqu’il effectue une action, l’état du système change et l’agent reçoit une valeur scalaire, appelée récompense, qui encode les informations sur la qualité de la transition (voir Figure 2.2)[36].

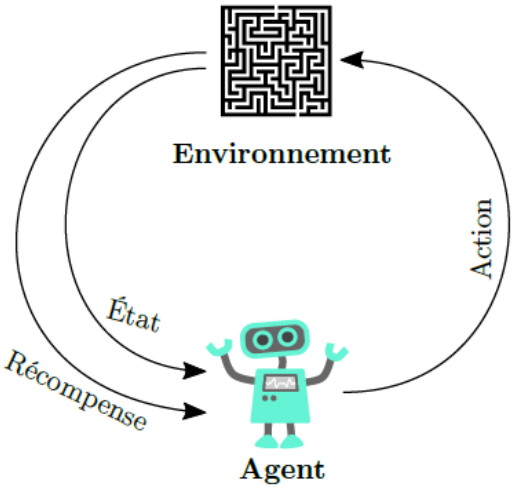



Figure 2.3 – Illustration du cadre général de l'apprentissage par renforcement.

L'apprentissage par renforcement a donc comme objectif d'entraîner un agent à se comporter de façon intelligente dans un environnement donné. Un agent interagit avec l'environnement en choisissant, à chaque temps donné, d'exécuter une action parmi un ensemble d'actions permises. Le comportement intelligent que doit apprendre cet agent est donné implicitement via un signal de renforcement ou récompense qui, après chaque décision de l'agent, indique s'il a bien ou mal agi.

L'agent doit donc se baser sur ce signal afin d'améliorer son comportement, qui est dicté par sa politique d'actions. À chaque temps donné, l'agent a normalement à sa disposition un ensemble de caractéristiques ou indicateurs d'entrée, décrivant l'environnement.

Les concepts d'action et de signal de renforcement sont probablement ceux qui distinguent le plus l'apprentissage par renforcement des apprentissages supervisé et non-supervisé [37] .

(*) : entité capable de percevoir son environnement grâce à des capteurs et d'agir sur celui-ci à travers des effecteurs afin de réaliser des buts

# IV. Les réseaux de neurones artificiels

## 1. Définition d'un neurone

Un neurone, est une fonction algébrique non linéaire et bornée, dont la valeur dépend de paramètres appelés coefficients ou poids. Les variables de cette fonction sont

habituellement appelées « entrées » du neurone, et la valeur de la fonction est appelée « sortie ». Un neurone est donc avant tout un opérateur mathématique, dont on peut calculer la valeur numérique par quelques lignes de programme informatique. Il est très rarement réalisé physiquement sous la forme d'un objet (circuit électronique par exemple). Il est cependant pratique de le représenter graphiquement [43].

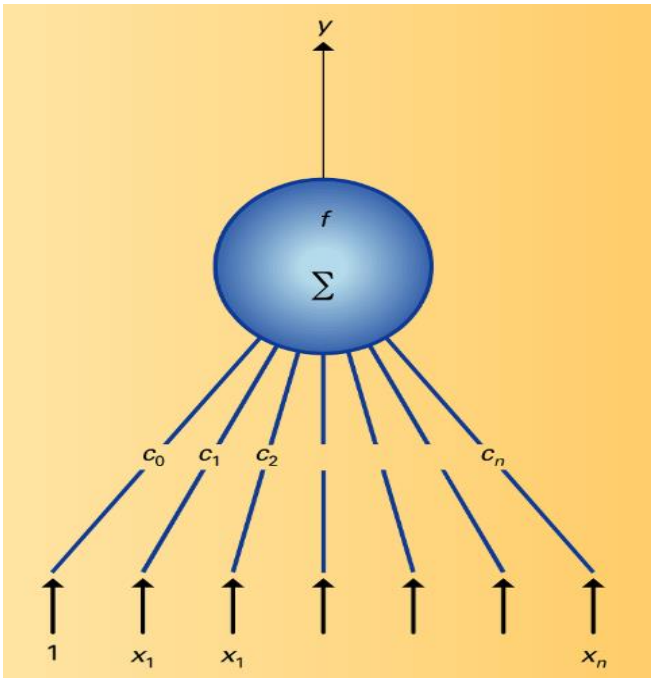



Figure 2.4 – Représentation graphique d'un neurone [43]

## 2. Présentation du perceptron

Le perceptron est un cas particulier de réseau de neurones, mais très souvent, quand on parle de réseau de neurones artificiels (artificial neural network) on fait référence au perceptron (cas le plus simple et le plus répandu). A l'image de la biologie, le perceptron est un ensemble de neurones organisés en couche. D'une couche à l'autre se propage le signal d'entrée jusqu'à la sortie, en activant ou non au fur et à mesure des neurones [40].

Le principe est de regarder la sortie par rapport à ce qui était attendu et de mettre à jour les liaisons entre les neurones (les renforcer ou les inhiber) pour améliorer notre résultat final, qui sera une prédiction de la part du réseau [40][41].

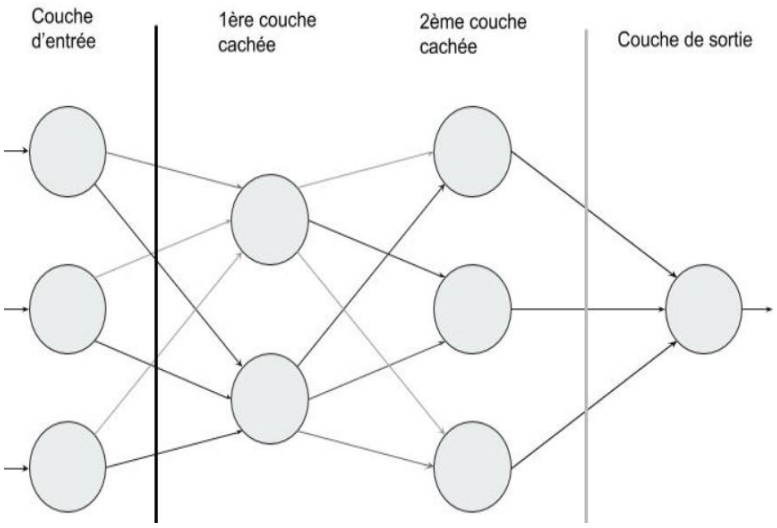



Figure 2.5 – Illustration des couches du perceptron[40]

Nous pouvons dès à présent remarquer que le perceptron est composé de trois couches de neurones [40][42]:

**La couche d'entrée (input layer)** : un ensemble de neurones qui portent le signal d'entrée.

**La couche cachée (hidden layer**) ou plus souvent LES couches cachées. Il s'agit du coeur de notre perceptron, là où les relations entre les variables vont être mises en exergue

**La couche de sortie (output layer)** : cette couche représente le résultat final de notre réseau, sa prédiction.

## 3. Fonctionnement d'un neurone

Des signaux $x_0$, $x_1$, $x_2$ arrivent à notre neurone (ils viennent de la couche précédente, donc on peut en déduire qu'elle contient 3 neurones). Chaque lien qui amène le signal est pondéré, respectivement $w_0$, $w_1$, $w_2$. C'est ce poids (weight) qui va être adapté tout au long de l'apprentissage pour permettre au réseau de prédire efficacement (en général il reste entre 0 et 1 ou -1 et 1)[40].

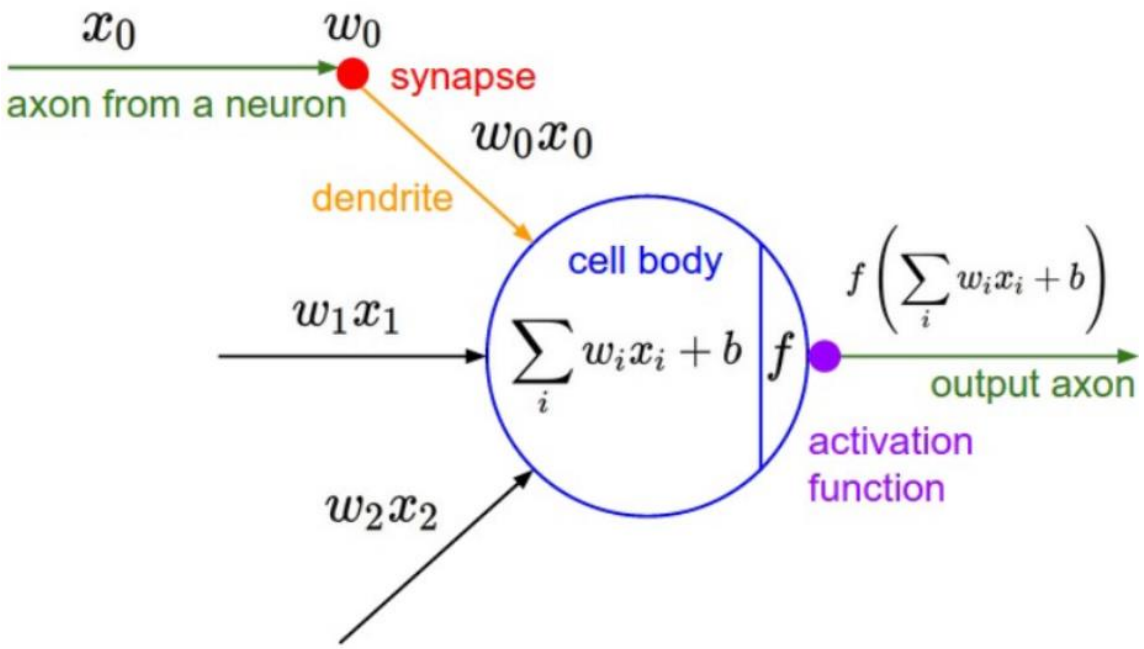



Figure 2.6 – zoom sur le fonctionnement d'un neurone[40]

- On calcule la somme de tous ces signaux pondérés $(\sum_{i=0}^{2} w_i \cdot x_i)$ et on ajoute un certain biais b. Ce biais (bias) peut être vu comme un neurone externe supplémentaire qui envoie systématiquement le signal 1 de poids b au neurone bleu. Grâce à lui, la fonction d'activation va être décalée et le réseau aura donc de plus grandes opportunités d'apprentissage[40].
- Une fois cette somme calculée, on applique une **fonction d'activation** (activation function) pour obtenir notre signal de sortie. Cette activation représente le seuil à partir duquel un neurone va émettre un signal (s'il a été suffisamment stimulé), et est donc lié au potentiel d'action en biologie.
- La formule de sortie d'un neurone caché sera donc toujours de la forme

$$y = f_{\text{activation}}\left(b + \sum_i w_i \cdot x_i\right) \quad (2,2)$$

  - Celle d'un neurone d'entrée sera y=x (en général on ne considère même pas qu'il y a un calcul)
  - Et celle d'un neurone de sortie sera $y = \sum_i w_i \cdot x_i$

Dans la pratique, les poids sont initialisés au hasard lorsqu'on crée le réseau de neurones. Il en va de même pour le biais. Au niveau de la fonction d'activation, pour les neurones

d'entrée il n'y a pas de somme ni d'activation (on peut considérer utiliser l'identité), pour les couches cachées on a plusieurs choix (sigmoïde, reLu, …) et pour la couche de sortie, on n'applique rien directement mais on peut appliquer en dehors « softmax » (ou d'autres fonctions)[40].

***Remarque :*** *Pourquoi avons-nous besoin de poids et de biais?[44]*

**-le Poids** montre la force du nœud particulier.

- **Une valeur de biais** vous permet de décaler la courbe de la fonction d'activation vers le haut ou le bas.

On peut dire aussi qu'UN NEURONE C'EST : UNE SOMME PONDÉRÉE DE SIGNAUX, AVEC UNE FONCTION PAR-DESSUS[40]

## 4. Fonction d'activation

Dans le cas des réseaux de neurones artificiels, il y existe plusieurs types de fonctions d'activation. Chacune d'elles est utilisé dans certaines circonstances. Aux débuts des réseaux de neurones artificiels l'une des premières fonction d'activation a été la fonction sigmoïd aussi connue sous le nom de fonction logistique. Elle prend une valeur réelle en entrée et la transforme en une valeur réelle de sortie comprise entre 0 et 1 [46].

L'intérêt de la fonction d'activation (ou la fonction de transfert dans certains ouvrages) est déterminé la sortie du réseau de neurones comme oui ou non. Il mappe les valeurs résultantes entre 0 et 1 ou -1 pour 1, etc. (en fonction de la fonction) [45].

Les fonctions d'activations sont deux classes linéaires et non linéaires

### A. Fonction d'activation linéaire

C'est une fonction simple de la forme: f(x) = ax ou f(x) = x. En gros, l'entrée passe à la sortie sans une très grande modification ou alors sans aucune modification. On reste ici dans une situation de proportionnalité[47].

Mais cela n'aide pas avec la complexité ou les divers paramètres des données habituelles qui sont introduites dans les réseaux de neurones [45].

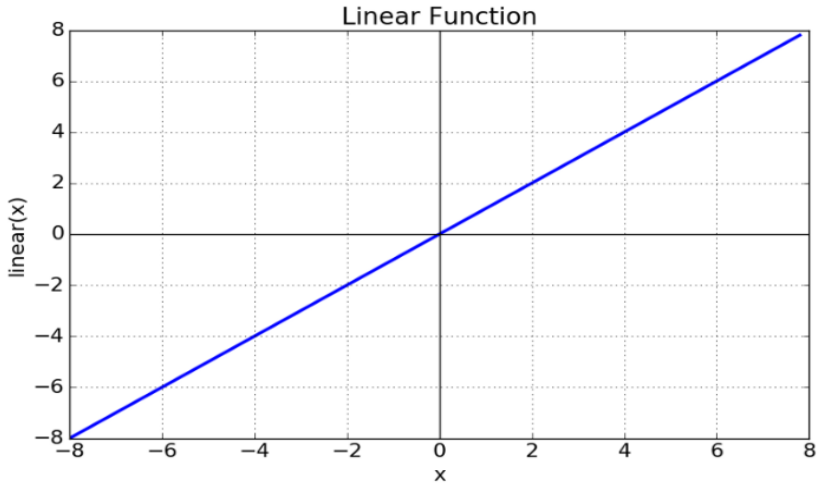



**Figure 2.7 :** Fonction d'activation linéaire [45].

## B. Fonction d'activation non linéaire

Les fonctions d'activation non linéaire sont les fonctions d'activation les plus utilisées. Il permet au modèle de généraliser ou de s'adapter à une variété de données et de différencier les résultats [45].

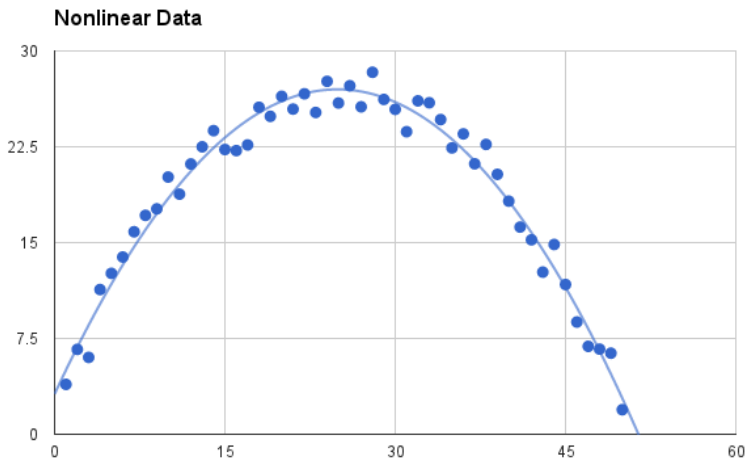



**Figure 2.8 :** Fonction d'activation non linéaire [45].

Il existe plusieurs types fonctions d'activation non linéaires, elles sont principalement divisées sur la base de leur **plage ou** de leurs **courbes** [45]

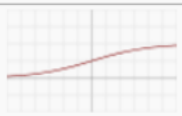
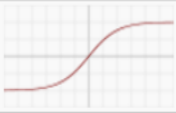
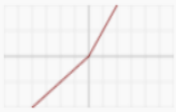
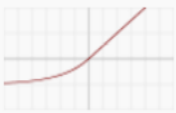
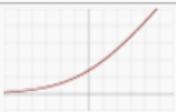
| Name | Plot | Equation | Derivative |
|---|---|---|---|
| Identity |  | $f(x) = x$ | $f'(x) = 1$ |
| Binary step |  | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$ |
| Logistic (a.k.a Soft step) |  | $f(x) = \frac{1}{1+e^{-x}}$ | $f'(x) = f(x)(1-f(x))$ |
| TanH |  | $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$ | $f'(x) = 1 - f(x)^2$ |
| ArcTan |  | $f(x) = \tan^{-1}(x)$ | $f'(x) = \frac{1}{x^2+1}$ |
| Rectified Linear Unit (ReLU) |  | $f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| Parameteric Rectified Linear Unit (PReLU) [2] |  | $f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| Exponential Linear Unit (ELU) [3] |  | $f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$ |
| SoftPlus |  | $f(x) = \log_e(1+e^x)$ | $f'(x) = \frac{1}{1+e^{-x}}$ |

**Figure 2.9 :** principales Fonctio*n d'activation non linéaire [45*].

## 5. Propriétés des Réseaux de Neurones

Un réseau de neurone est un ensemble d'éléments de traitement de l'information, avec une topologie spécifique d'interconnexions entre ces éléments et une loi d'apprentissage pour adapter les points de connexions. D'une manière générale, un réseau de neurones possède les propriétés suivantes [48]:

### A. Le parallélisme

Cette notion se situe à la base de l'architecture des réseaux de neurones considérés comme ensembles d'entrés élémentaires qui travaillent simultanément.

### B. La résistance aux pannes

A cause de l'abondance des entrées et la structure du réseau. Les données bruitées ou les pannes locales dans un certain nombre de ses éléments n'affectent pas ses fonctionnalités. Cette propriété résulte de fonctionnement collectif et simultané des neurones qui les composent.

### C. La capacité d'adaptation

Celle-ci se manifeste tout d'abord dans les réseaux de neurones par la capacité d'apprentissage qui permet au réseau de tenir compte des nouvelles contraintes ou des nouvelles données du monde extérieur. De plus, ils se caractérisent par leur capacité d'auto organisation qui assure leur stabilité en tant que système dynamique.

### D. La généralisation

La capacité de généralisation d'un réseau de neurone est son aptitude de donner une réponse satisfaisante à une entrée qui ne fait pas partie des exemples à partir desquels il a appris

### E. Structure de connexion

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elles sont très variées, le nombre de connexions étant énorme. Cette topologie fait apparaître une certaine régularité de l'arrangement des neurones.

## V. Topologie des réseaux de neurones

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité :

## 1. Réseau Feed-forward

C'est la propagation vers l'avant de l'information où les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches suivantes [49][50], , les informations passent directement de l'entrée aux nœuds de traitement puis aux sorties[42], ce type de réseau il est dit aussi statique

### A. Réseau multicouche

Chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) [49] il est dit aussi réseau multi-perceptron [50]

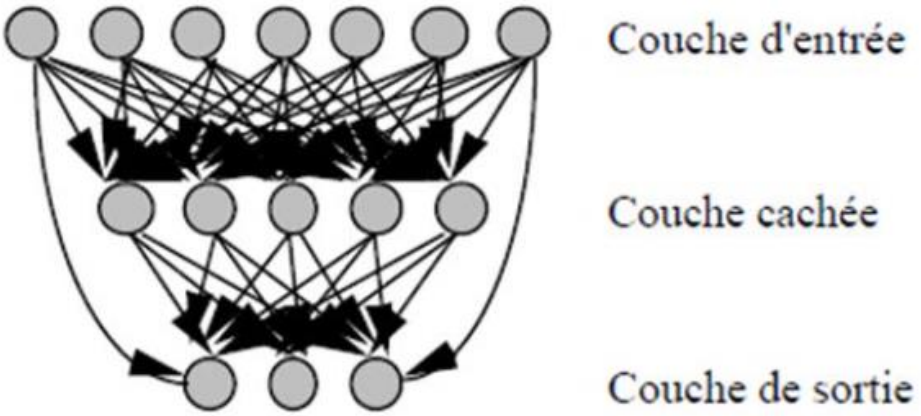



**Figure 2.10** : réseau multicouches [51]

### B. Réseau à connexions locales

Il s'agit d'une structure multicouche, mais qui à l'image de la rétine, conserve une certaine topologie. Chaque neurone entretien des relations avec un nombre réduit et localisé de neurones de la couche suivante (Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique) [49][51].

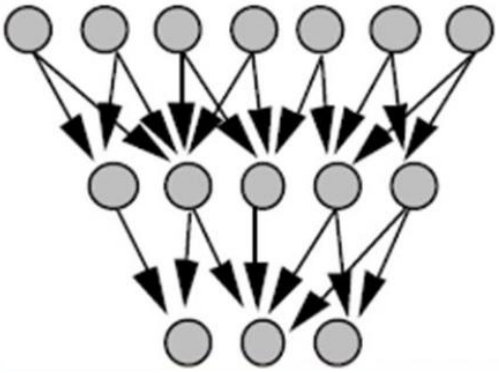


**Figure 2.10** : réseau à connexions locales [51]

## 2. Les réseaux de neurones récurrents

Les **réseaux de neurones récurrents (feedback network)** sauvegardent les résultats produits par les nœuds de traitement et nourrissent le modèle à l'aide de ces résultats. Ce mode d'apprentissage est un peu plus complexe [42]

### A. Réseau récurrents simples

Les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales [56].

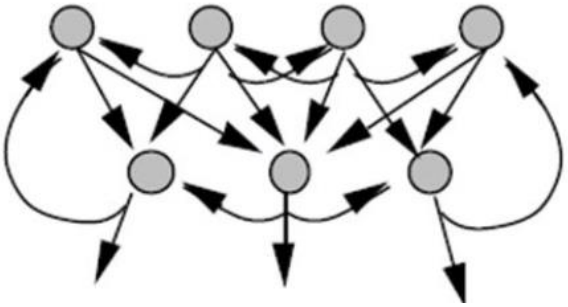


**Figure 2.11.** Réseau à connexions récurrentes simples[**56]**

### B. Réseau à connexion complète (récurent complet)

C'est la structure d'interconnexion la plus générale où Chaque neurone est connecté à tous les neurones du réseau [56]

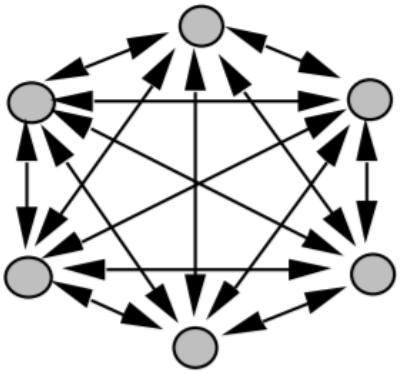


**Figure 2.12.** Réseau de neurones à connexions complète[**56]**

## *VI. Réseaux de neurones profonds*

L'apprentissage profond et les réseaux de neurones fournissent actuellement les meilleurs exemples de la qualité des données. Ils fournissent également les meilleures solutions possibles à divers problèmes dans différents domaines, y compris le traitement du langage naturel, la reconnaissance de la parole et de l'image, etc. .

Les réseaux de neurones sont conçus pour nous aider à classer et à regrouper une grande quantité de données. En d'autres termes, pensez aux réseaux de neurones comme couche de classification et de clustering qui sont placés au-dessus des données qui seront gérées et stockées. Les réseaux de neurones sont également conçus pour aider à regrouper diverses

données non étiquetées en fonction de certaines similitudes existant dans différentes collections de données.

On peut également dire que les réseaux de neurones extraient diverses caractéristiques de données qui sont transmises à différents algorithmes pour une classification et un regroupement plus poussés. Il faut penser au réseau de neurones en tant que divers composants de la plus grande famille de modèles et d'applications d'apprentissage automatique. Les réseaux de neurones impliquent différents algorithmes conçus pour la régression des données, la classification des données et le renforcement de l'apprentissage profond [52][53].

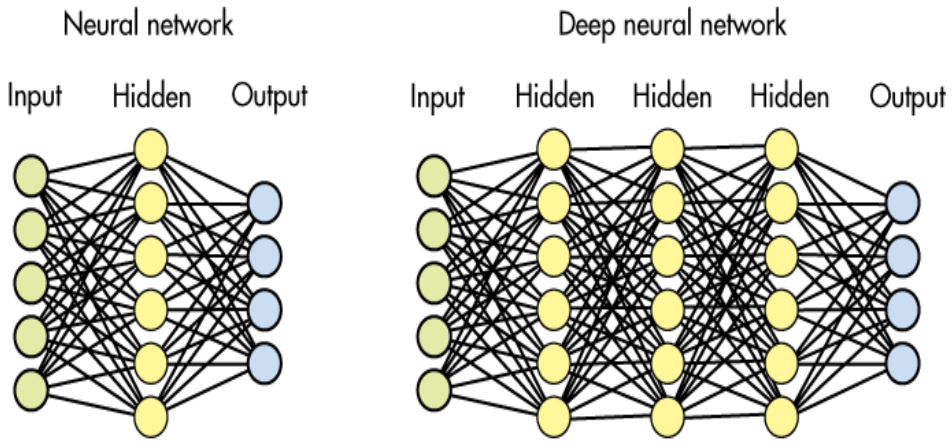



***Figure 2 .13:*** *Différence entre "Neural network" et "Deep neural network" [53].*

Il existe plusieurs types de réseaux de neurones profonds, nous citons à titre d'exemples :

## 1. Réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels (en anglais Convolutional neural networks), aussi connus sous le nom de CNNs, sont spécialement conçus pour traiter des images en entrée. Leur architecture est alors plus spécifique [54]: ui sont généralement composés des couches suivantes : La couche convolutionnelle et la couche de pooling et la couche Fully Connected[55]

### A. Caractéristiques

Le CNN compare les images fragment par fragment. Les fragments qu'il recherche sont appelés les caractéristiques. En trouvant des caractéristiques approximatives qui se ressemblent à peu près dans 2 images différentes, le CNN est bien meilleur à détecter des similitudes que par une comparaison entière image à image.

Chaque caractéristique est comme une mini-image — i.e. un petit tableau de valeurs en 2 Dimensions. Les caractéristiques rassemblent les aspects les plus communs des images[57].

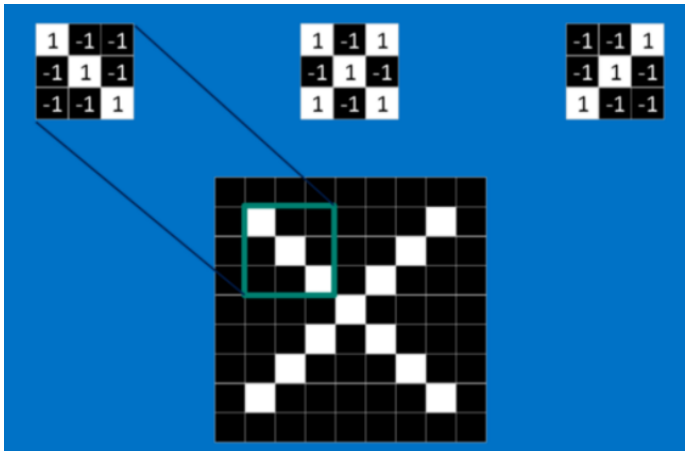



***Figure 2 .14:*** *image d'un « X » découpé en caractéristiques [57]*

## B. Couche convolution

Quand on lui présente une nouvelle image, le CNN ne sait pas exactement si les caractéristiques seront présentes dans l'image ou où elles pourraient être, il cherche donc à les trouver dans toute l'image et dans n'importe quelle position.

En calculant dans toute l'image si une caractéristique est présente, nous faisons un filtrage. Les mathématiques utilisés pour réaliser cette opération sont appelés une convolution, de laquelle les réseaux de neurones à convolution tiennent leur nom.

Pour calculer la correspondance entre une caractéristique et une sous-partie de l'image, il suffit de multiplier chaque pixel de la caractéristique par la valeur que ce même pixel contient dans l'image. Ensuite, on additionne les réponses et divise le résultat par le nombre total de pixels de la caractéristique. [57]

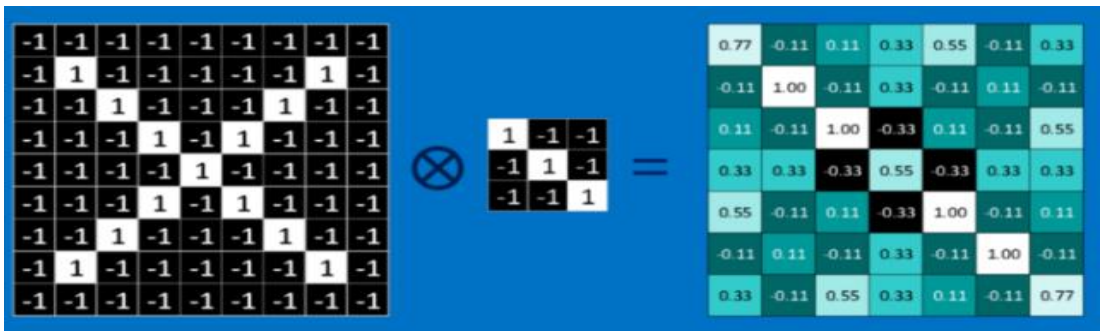



***Figure 2 .14:*** *exemple après convolution d'une caractéristique [57]*

## C. Non linéarité (Relu)

Une opération supplémentaire appelée ReLU est utilisée après chaque opération de convolution, ReLU signifie unité linéaire rectifiée et est une opération non linéaire.sa formule de sortie est : Sortie =Max (Zéro, Entré).

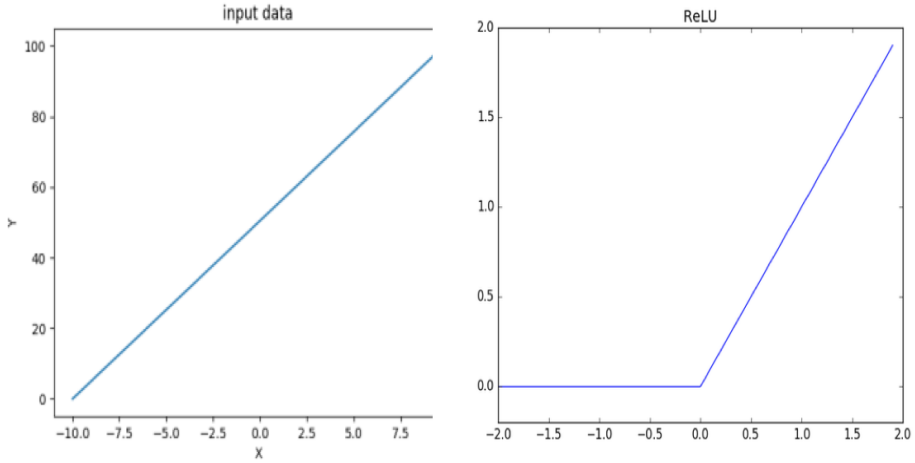



**Figure 2. 15 :** Relu Non-Linéarité [58].

ReLU est une opération appliquer par pixel et remplace toutes les valeurs de pixels négatifs dans la Feature map par zéro ().le but de ReLu est d'introduire la notion de non-linéarité, puisque la plupart des données du monde réel que nous voudrions que notre convolutional network apprenne serait non-linéaire [58].

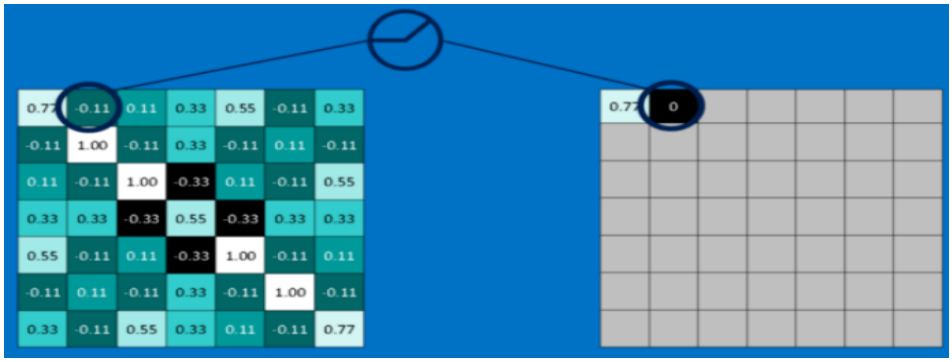


***Figure 2 .14:*** *exemple d'application de Relu sur l'image [57]*

## D. La couche Pooling

L'étape Pooling réduit la dimensionnalité de chaque Feature Map mais conserve les informations les plus importantes. Les deux formes les plus utiliser sont le 'Average Pooling' qui consiste à faire la moyenne des valeurs de la zone, ou bien 'Max Pooling' qui prend l'élément le plus grand de la Feature Map. Le Pooling rend les représentations

d'entrée plus petites et plus faciles à gérer. Il réduit aussi le nombre de paramètres et de calculs dans le réseau, contrôlant ainsi le sur-apprentissage [58].

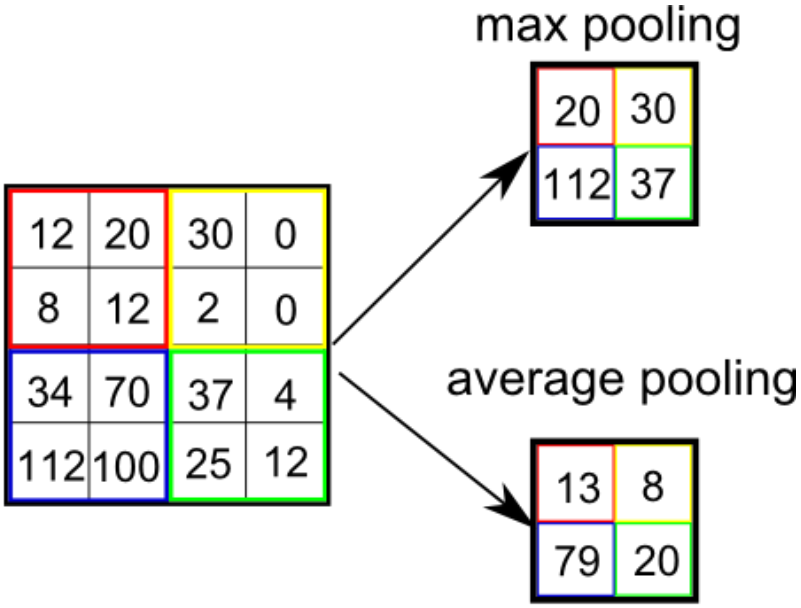



**Figure 2.15 :** Étape Max/average Pooling [58].

Après avoir procédé au pooling, l'image n'a plus qu'un quart du nombre de ses pixels de départ [57].

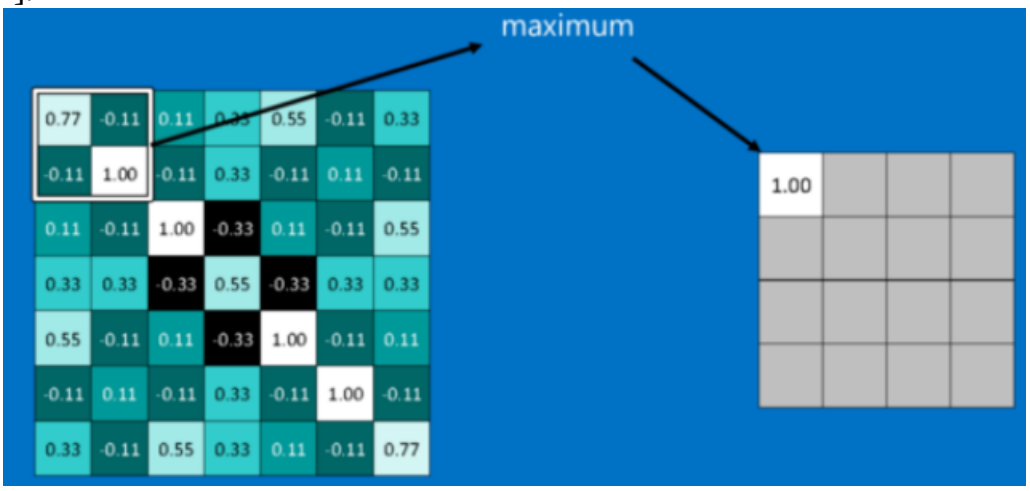



***Figure 2 .16:*** *exemple après un pooling [57]*

## E. Couche Fully Connected

La couche Fully Connected est configurée exactement comme son nom l'indique : elle est entièrement connectée à la sortie de la couche précédente. Une couche entièrement connectée prend tous les neurones de la couche précédente (qu'elle soit entièrement connectée, en pool ou convolutionnelle) et la connecte à chaque neurone qu'elle possède [58].

# 2. Réseau de neurones récurrents (RNN)

Les réseaux de neurones récurrents (en anglais recurrent neural networks), aussi appelés RNNs, sont une classe de réseaux de neurones qui permettent aux prédictions antérieures d'être utilisées comme entrées, par le biais d'états cachés (en anglais hidden states) [59].

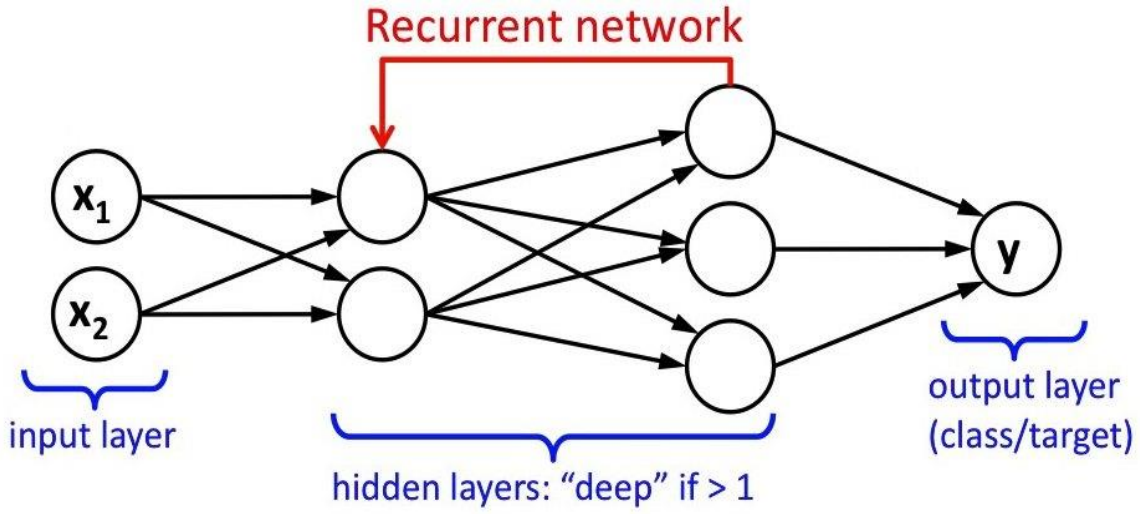



**Figure 2.17 :** Architecture d’un réseau de neurones récurrents [60].

Les avantages et inconvénients des architectures de RNN traditionnelles sont résumés dans le tableau ci-dessous [59] :

| Avantages | Inconvénients |
|---|---|
| • Possibilité de prendre en compte des entrées de toute taille<br>• La taille du modèle n'augmente pas avec la taille de l'entrée<br>• Les calculs prennent en compte les informations antérieures<br>• Les coefficients sont indépendants du temps | • Le temps de calcul est long<br>• Difficulté d'accéder à des informations d'un passé lointain<br>• Impossibilité de prendre en compte des informations futures un état donné |

## A. Réseaux récurrents à longue mémoire à court terme (LSTM)

Les LSTM (Long short-term memory (LSTM) ) sont inspirés des RNN, contrairement à eux, les LSTM implémentent spécifiquement Des "cellules de mémoire" qui ont la capacité de stocker de la mémoire pour de plus longues périodes de temps [4].

Les LSTM contiennent des informations en dehors du flux normal du réseau récurrent dans une cellule ‘gated’. Les informations peuvent être stockées, écrites ou lues dans une cellule, comme les données de la mémoire d’un ordinateur. La cellule prend des décisions sur ce qu’il faut stocker, et quand autoriser les lectures, les écritures et les effacements, via des portes qui s’ouvrent et se ferment. Contrairement au stockage numérique sur ordinateur, ces portes sont analogiques, implémentés avec une multiplication par sigmoids, qui sont toutes dans la plage de 0-1 [4][61].

Ces portes agissent sur les signaux qu’elles reçoivent et, comme les nœuds du réseau de neurones, elles bloquent ou transmettent des informations basées sur leur force et leur importation, qu’ils filtrent avec leurs propres ensembles de poids. Ces poids, comme les poids qui modulent les états d’entrée et les états cachés, sont ajustés via le processus d’apprentissage des réseaux récurrents. C’est-à-dire que les cellules apprennent à autoriser les données à entrer, à quitter ou à être supprimées au cours du processus itératif consistant à faire des suppositions, des erreurs de rétropropagation et à ajuster les pondérations [4][61].

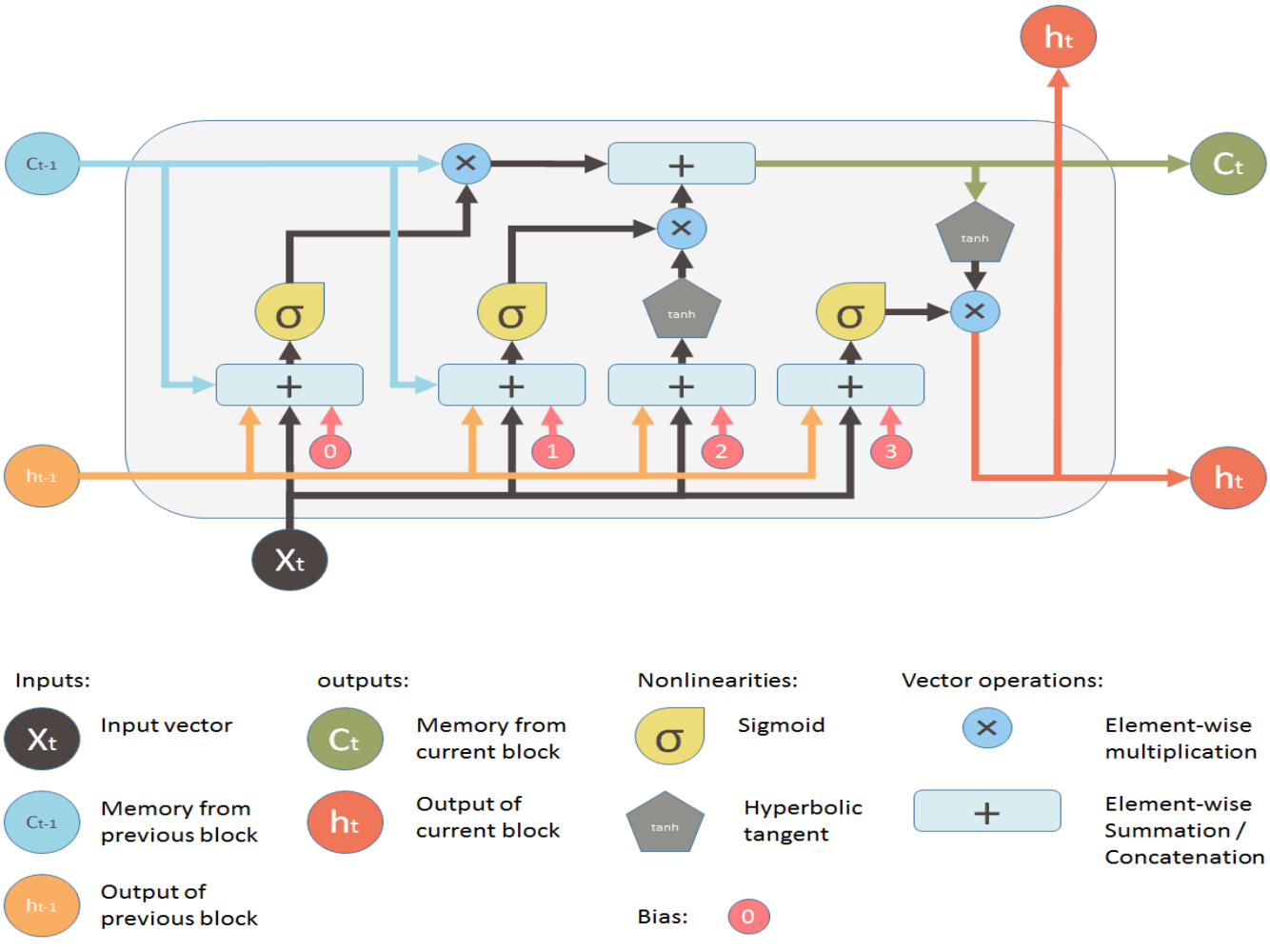


**Figure 2.18 :** Architecture d'un LSTM [61].

### *B. Réseau récurrent à portes (Gated Recurrent Unit -GRU-)*

Un réseau récurrent à portes est une variation d'un réseau de neurones récurent, il utilise également des unités de contrôle pour moduler le flux d'informations à l'intérieur de l'unité, mais sans cellules de mémoire, les portes d'entrée et d'oubli fusionne pour devenir une (update gate). Et enfin la porte de sortie est remplacée par une porte de réinitialisation (reset gate) [62].

LSTM, c'est que GRU a l'avantage d'être moins lourde en calculs car possédant moins de paramètres et d'équations, mais elle a des performances équivalentes au modèle LSTM de base. [62].

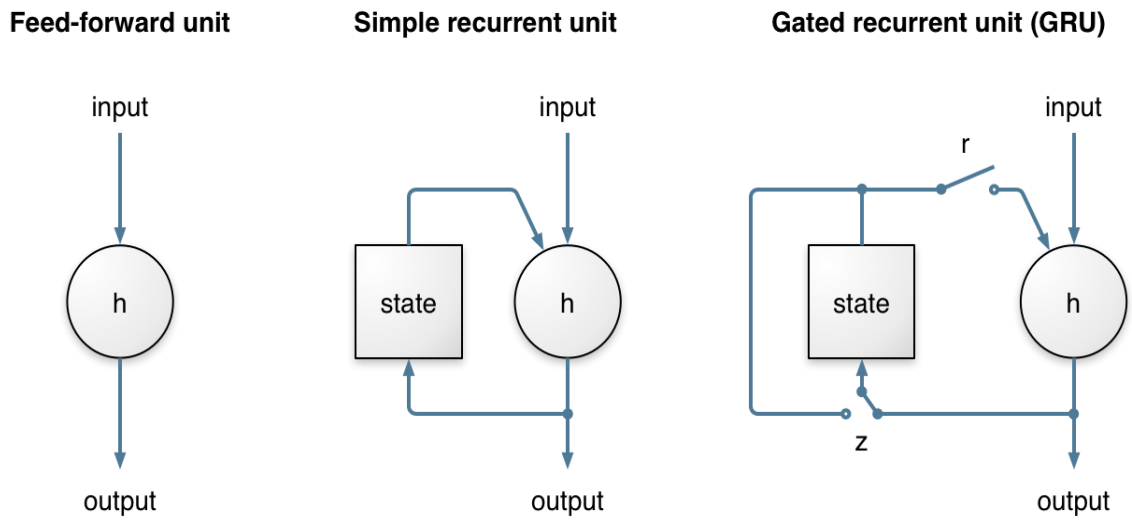



**Figure 2.19 :** Réseau récurrent à portes [4].

## 3. Les réseaux accusatoires génératifs (GAN)

Les réseaux accusatoires génératifs (Generative Adversarial Networks où GAN) sont des architectures de réseaux neuronaux profonds composées (contradictoire) de deux réseaux, opposant les uns aux autres : le générateur et le discriminateur [63].

Le *générateur*, génère de nouvelles instances de données, tandis que l'autre, le *discriminateur*, les évalue pour en vérifier l'authenticité. c'est-à-dire que le discriminateur décide si chaque instance de données qu'il examine appartient ou non à l'ensemble de données d'apprentissage réel[63].

Voici les étapes qu'un GAN prend [63]:

- Le générateur prend des nombres aléatoires et renvoie une image.
- Cette image générée est introduite dans le discriminateur à côté d'un flux d'images provenant du jeu de données réel de terrain.
- Le discriminateur prend à la fois des images réelles et fausses et renvoie des probabilités, un nombre compris entre 0 et 1, 1 représentant une prédiction d'authenticité et 0 représentant fictif.

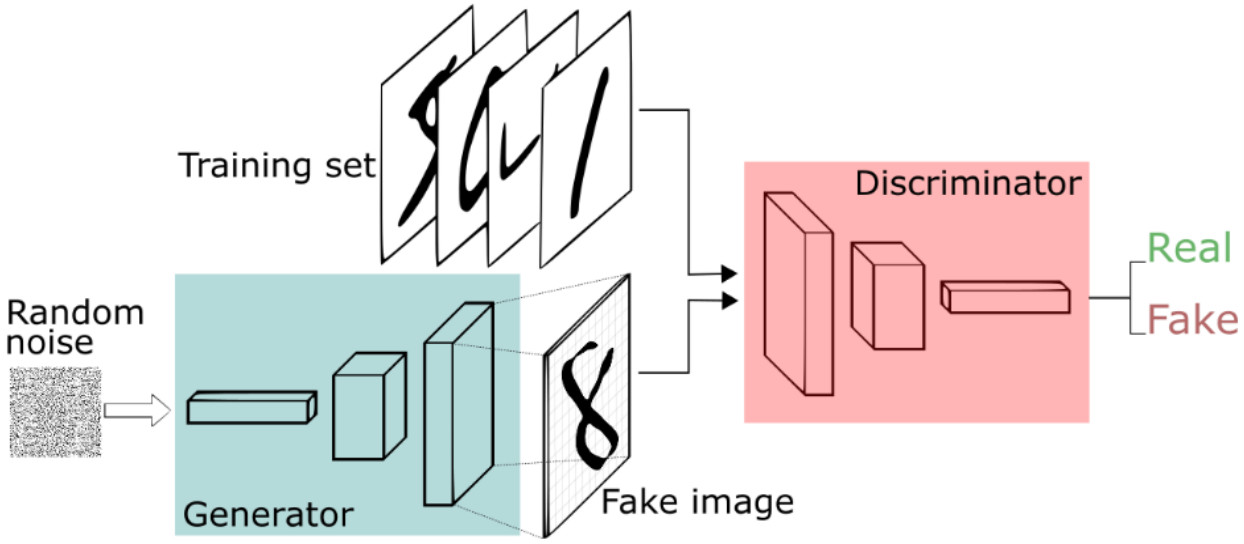



***Figure 2.19 :*** *Les réseaux accusatoires génératifs (GAN)[63].*

## VII. Conclusion

Dans ce chapitre nous avons étudié les réseaux de neurones artificiels, leurs fonctionnements ainsi que les différentes variantes qui existent, dans ce qui suit nous allons étudié en détail l'une des plus importantes variantes des RNAs , les auto-encodeurs.

CHAPITRE 3 : **les auto-encodeurs**

# I. Introduction

Nous avons vu dans La première partie de ce mémoire les concepts d'extractions des motifs fréquents, Dans la deuxième partie, concernée les réseaux de neurones. Nous allons maintenant commencer à plonger dans des architectures d'apprentissage approfondi spécifiques: les auto-encodeurs.

# II. Définition

Les auto-encodeurs sont un type spécifique de réseaux neuronaux à anticipation où l'entrée est identique à la sortie. Ils compressent l'entrée dans un code de dimension inférieure, puis reconstruisent la sortie à partir de cette représentation. Le code est un "résumé" ou "compression" compact de l'entrée, également appelé représentation en espace latent.

Un autoencoder est composé de 3 composants: encodeur, code et décodeur. Le codeur compresse l'entrée et produit le code, le décodeur reconstruit ensuite l'entrée uniquement à l'aide de ce code [64].

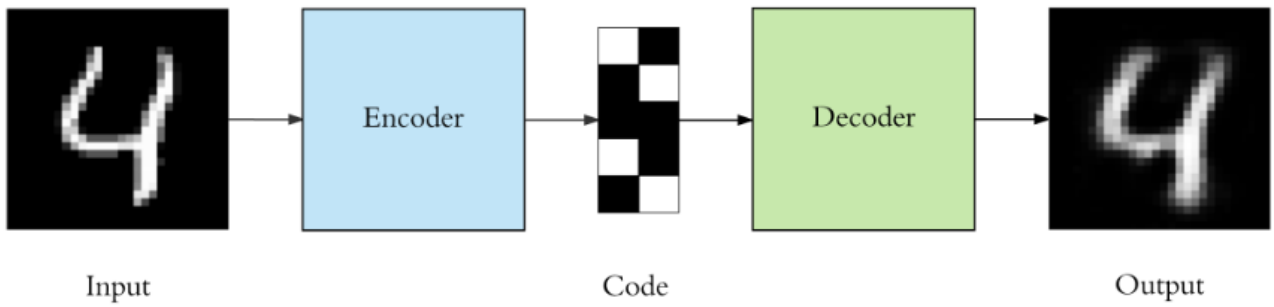



**Figure 3. 1 :** Architecture d'un auto-encodeur [64].

Pour construire un auto-codeur, nous avons besoin de 3 éléments: une méthode de codage, une méthode de décodage et une fonction de perte pour comparer la sortie à la cible. [64][27]

# III. Fonctionnement général

D’une manière général Les auto-encodeurs encodent les valeurs d'entrée *x, à l'* aide d'une fonction *f*. Il décode ensuite les valeurs codées *f (x)*, à l'aide d'une fonction *g*, pour créer des valeurs de sortie identiques aux valeurs d'entrée.

L'objectif d'Auto-encodeur est de minimiser les erreurs de reconstruction entre l'entrée et la sortie. Cela aide les auto-encodeurs à apprendre les fonctionnalités importantes présentes dans les données. Lorsqu'une représentation permet une bonne reconstruction de son entrée, elle conserve une grande partie de l'information présente dans l'entrée[28]

Décomposons cela étape par étape, en prenant par exemple les produits achetés par les clients comme ensemble de données [28] :

**Étape 1: Prenez la première ligne des données client pour tous les produits achetés dans un tableau en tant qu'entrée.**

1 signifie que le client a acheté le produit.
0 signifie que le client n'a pas acheté le produit.

**Étape 2: Encodez l'entrée dans un autre vecteur h. h est un vecteur de dimension inférieure à celle de l'entrée.**
Nous pouvons utiliser la fonction d'activation sigmoïde pour h car elle est comprise entre 0 et 1.
W est le poids appliqué à l'entrée et b est le terme de biais.

$$h = f(Wx + b) \quad (3,1)$$

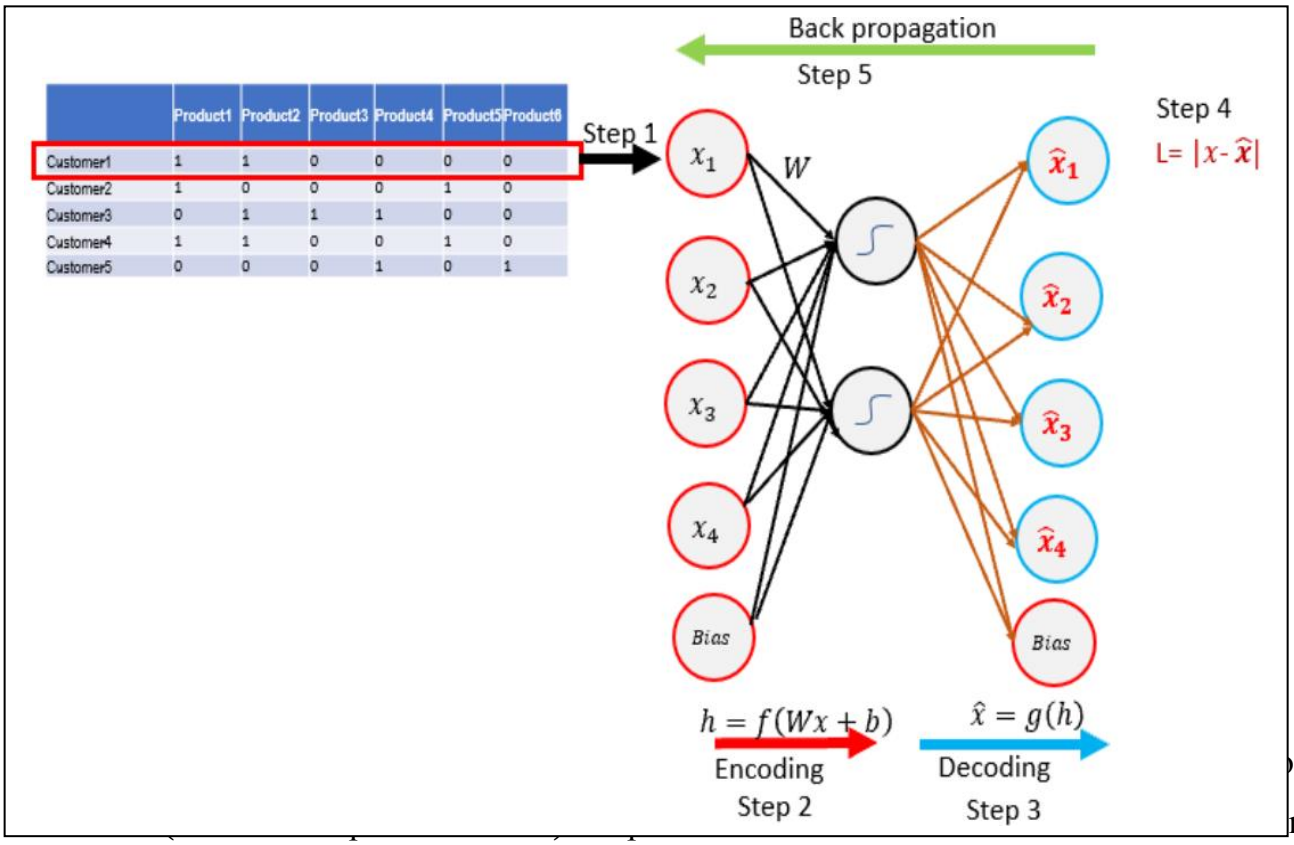
**Étape 3: Décodez le vecteur h pour recréer l’entrée.** La sortie aura la même dimension que l'entrée

**Étape 4: Calculer l'erreur de reconstruction L.** L'erreur de reconstruction est la différence entre le vecteur d'entrée et de sortie. Notre objectif est de minimiser l’erreur de reconstruction afin que la sortie soit similaire au vecteur d’entrée.

**Étape 5: Retournez l'erreur de la couche en sortie vers la couche en entrée pour mettre à jour les poids.** Les poids sont mis à jour en fonction du degré de responsabilité de l’erreur.
Le taux d’apprentissage détermine la quantité de mises à jour que nous mettons à jour.

**Étape 6: Répétez les étapes 1 à 5 pour chacune des observations du jeu de données.** Les poids sont mis à jour après chaque observation

**Étape 7: Répétez plusieurs époques.** L'époque est lorsque toutes les lignes de l'ensemble de données ont transité par le réseau de neurones.




l'encodeur, la couche cachée ainsi que la dernière couche formant le décodeur [29] comme décrit dans la figure ci-dessous.

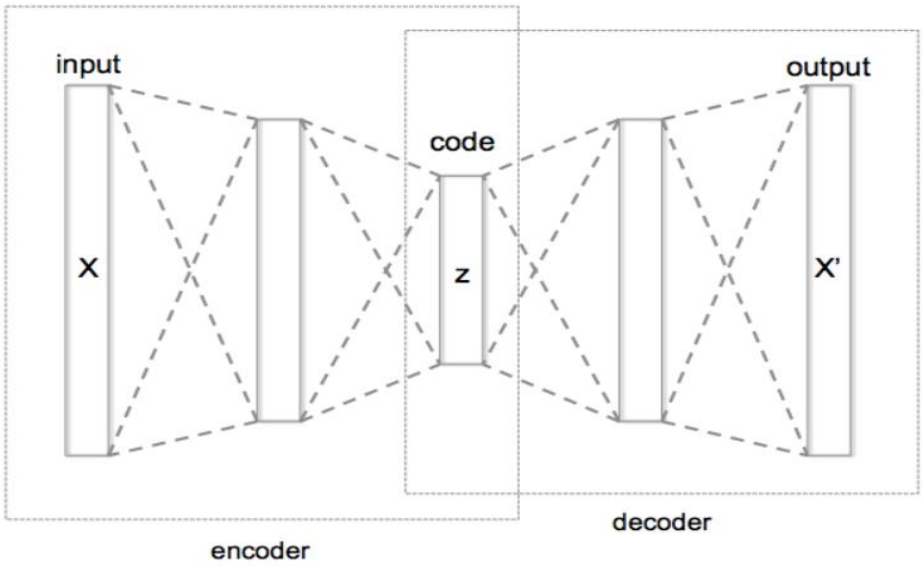



Figure 3.2 Structure schématique d'un auto-encodeur avec 3 couches cachées entièrement connectées |30]

L'encodeur calcule, à partir de x, le vecteur h de taille m (nombre de neurones cachés) ainsi[29]:

$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \quad (3,3)$$

où $W^{(1)}$ est une matrice de taille m × n, et $b^{(1)}$ un vecteur de biais de taille m.

σ(.) est une fonction d'activation de type tangente hyperbolique définie par [29]:

$$\sigma(\mathbf{y}) = \frac{e^{\mathbf{y}} - e^{-\mathbf{y}}}{e^{\mathbf{y}} + e^{-\mathbf{y}}} \quad (3,4)$$

Le décodeur cherche à reconstruire le vecteur x à partir de la couche cachée h. Le résultat de cette reconstruction est le vecteur $\tilde{\mathbf{x}}$ tel que :

$$\tilde{\mathbf{x}} = \sigma(\mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}) \quad (3,5)$$

où $\tilde{\mathbf{x}}$ est un vecteur de taille m, W(2) est une matrice de poids de taille n × m et b(2) est un vecteur de biais de taille n.

Durant l'apprentissage, l'auto-encodeur tente de réduire une erreur de reconstruction l entre x et $\tilde{\mathbf{x}}$. Il utilise l'erreur quadratique moyenne (MSE) (Mean Squared Error)

$$(l_{\text{MSE}}(\mathbf{x}, \tilde{\mathbf{x}}) = ||\mathbf{x} - \tilde{\mathbf{x}}||^2) \quad (3,6)$$

de manière à minimiser l'erreur de reconstruction totale LMSE avec l'ensemble de paramètres θ = {W(1), b(1),W(2), b(2)} :

$$L_{\text{MSE}}(\theta) = \frac{1}{d}\sum_{\mathbf{x}\in D} l_{\text{MSE}}(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{1}{d}\sum_{\mathbf{x}\in D} ||\mathbf{x} - \tilde{\mathbf{x}}||^2 \quad (3,7)$$

Prenant à titre d'exemple, Nous avons une entrée en 3 dimensions correspondant à un vecteur dans un espace 3D. Nous le passons ensuite à travers deux couches cachées de 4 nœuds chacune. Et le résultat final est un vecteur 1D ou un scalaire. Donc, le vecteur d'entrée *x* a 1 ligne et 3 colonnes. Pour le transformer en un espace 4D, nous devons le multiplier avec une matrice *3x4* . Puis dans un autre espace 4D, on multiplie par une matrice *4x4* . Et enfin, pour le réduire à un espace 1D, nous utilisons une matrice *4x1*[65] .

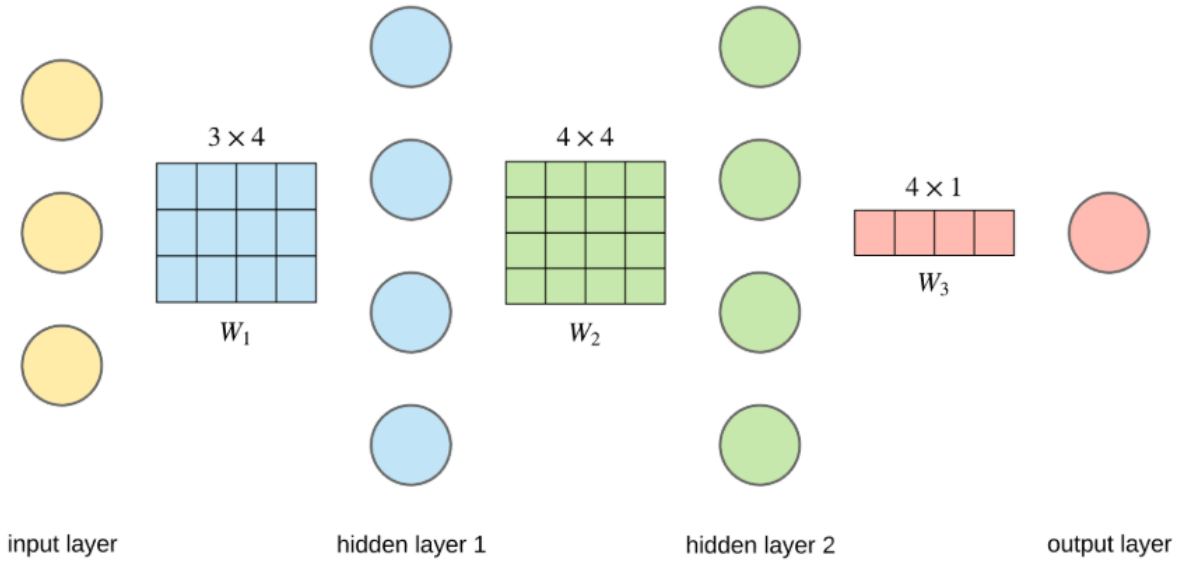



Figure 3.3 exemple d'un auto-encodeur avec trois couche cachées |30]

## V. Propriétés des auto-encodeurs

Les auto-encodeurs sont principalement un algorithme de réduction de dimensionnalité (ou compression) avec quelques propriétés importantes:

- **Données spécifiques:**

- Les auto-encodeurs ne peuvent comprimer que des données similaires à celles sur lesquelles ils ont été entraînés [4]
- Les auto-encodeurs sont différent des algorithmes de compression car ils apprennent des fonctionnalités spécifiques aux données[4] Nous ne pouvons donc pas nous attendre à ce qu'un codeur automatique formé à l'aide de chiffres manuscrits compresse les photos de paysages[64],
- Font des suppositions générales sur les données, mais pas sur les types spécifiques exemple : .mp3 .png .mp4 [4].

- **Les auto-encodeurs subissent des pertes :**

- Ce qui signifie que les sorties décompressées seront dégradées par rapport aux entrées originales [4].

## VI. Domaines d'utilisation

Les auto-encodeurs peuvent être utilisé dans plusieurs domaines nous citons à titre d'exemple [28] :

- **La réduction de la dimensionnalité linéaire et non linéaire :** Encode l'entrée de la couche masquée dans une dimension inférieure à celle de l'entrée. La couche cachée est ensuite décodée en sortie. La couche de sortie a la même dimension que l'entrée. L'uto-encodeur réduit la dimensionnalité des données linéaires et non linéaires. Il est donc plus puissant que PCA.

- **Les moteurs de recommandation :** Ceci utilise des encodeurs profonds pour comprendre les préférences de l'utilisateur et recommander des films, des livres ou des éléments.

- **L'extraction de caractéristiques :** les auto-encodeurs essaient de minimiser l'erreur de reconstruction. Pour réduire l'erreur, il apprend certaines des caractéristiques importantes présentes dans l'entrée. Il reconstruit l'entrée à partir de l'état codé présent dans la couche cachée. Le codage génère un nouvel ensemble de fonctionnalités combinant les fonctionnalités d'origine. Le codage dans les auto-codeurs aide à identifier les caractéristiques latentes présentes dans les données d'entrée**.**

- **Reconnaissance des images :** Les auto-encodeurs empilés sont utilisés pour la reconnaissance des images et pour apprendre différentes caractéristiques d'une image.

# VII. Variantes des auto-encodeurs

## 1. Auto-encodeur de débruitage

Lorsqu'il y a plus de nœuds dans la couche cachée, le réseau risque d'apprendre la «fonction d'identité», également appelée «fonction nulle», ce qui signifie que la sortie est égale à l'entrée, ce qui marque le code Auto-encodeur comme inutile…Les auto-encodeurs de débruitage (Denoising) résolvent ce problème en corrompant volontairement les données en tournant de manière aléatoire certaines des valeurs d'entrée à zéro. En général, le pourcentage de nœuds d'entrée mis à zéro est d'environ 50%.[66][30][32]

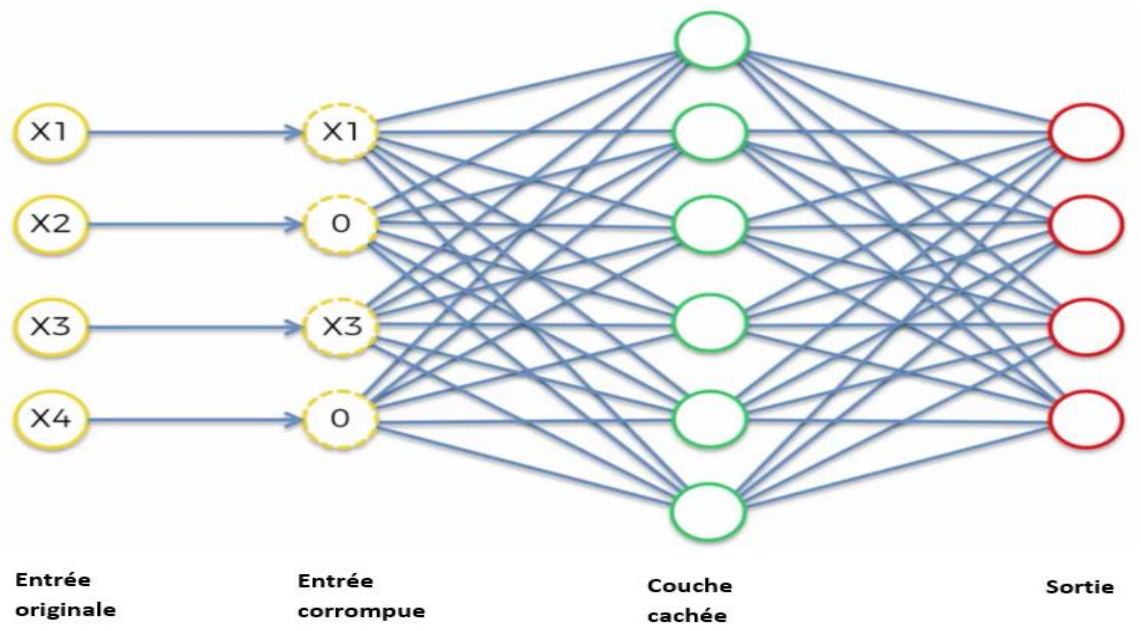



Figure 3.3 Architecture d'un DAE (*Denoising Autoencoder*) |66]

Lors du calcul de la fonction Perte, il est important de comparer les valeurs de sortie avec l'entrée d'origine et non avec l'entrée corrompue. Ainsi, le risque d'apprendre la fonction d'identité au lieu d'extraire des fonctionnalités est éliminé.[66]

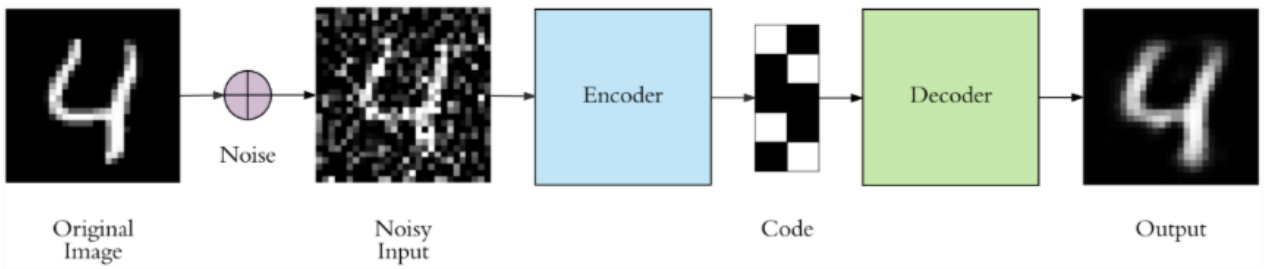



Figure 3.4 exemple de fonctionnement d'un DAE*D*[64]

## 2. Auto-encodeurs variationnels

Les auto-encodeurs variationnels (VAE) possèdent une propriété fondamentalement unique qui les sépare des auto-encodeurs standards, le codeur ne produise pas un vecteur de codage de taille *n*, mais deux signaux de taille n: un vecteur de moyennes**, μ** , et un autre vecteur d'écarts-types, **σ** [67].

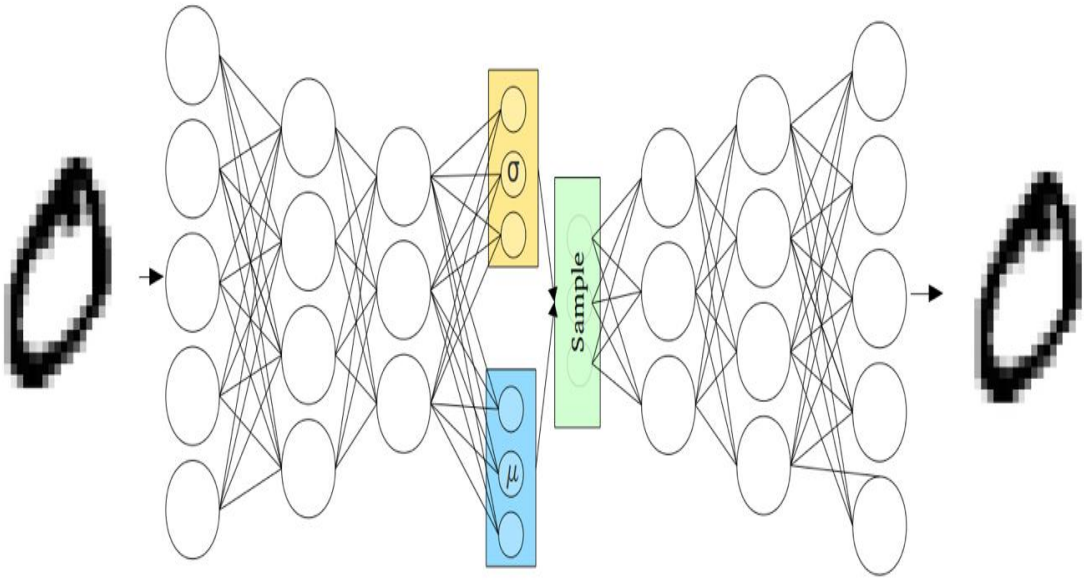



Figure 3.5 Architecture d'un auto-encodeur variationnel [67]

Cette génération stochastique signifie que, même pour la même entrée, alors que la moyenne et les écarts-types restent les mêmes, le codage réel varie légèrement à chaque passage, simplement en raison de l'échantillonnage.[67]

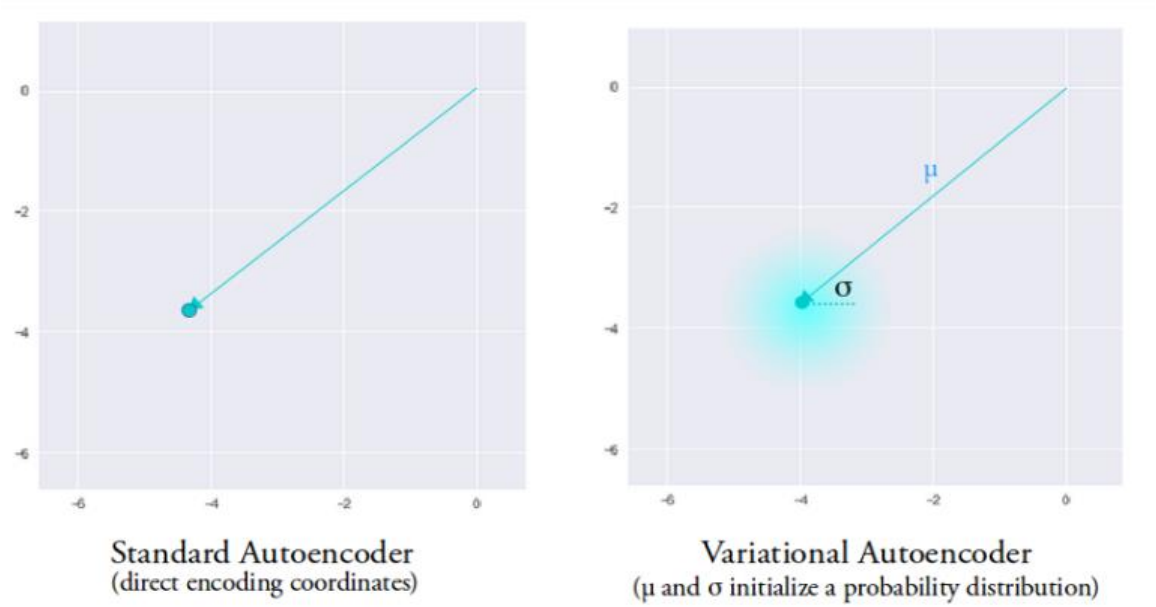



Figure 3.6 Comparaison entre un auto-encodeur standard et variationnel [67]

## VIII. Conclusion

Dans ce chapitre nous avons vu un type particulier de réseau de neurones, les auto-encodeurs pour pouvoir expliquer dans le chapitre suivant le choix de la structure utilisé.

CHAPITRE 4 : **Approche proposée**

# I. Introduction

L'extraction des itemsets fréquents à partir de données incertaines a attiré beaucoup l'intention des chercheurs de la communauté de Data Mining, et par conséquent un nombre important des algorithmes a été proposé dans la littérature.

Dans ce qui suit, on veut traiter ce problème en utilisant un des modèles d'apprentissage profond, pour tester l'utilité de ce modèle dans l'extraction des motifs, et voir s'il y possibilité d'avoir de nouveaux ou meilleurs résultats par rapport aux approches existantes.

# II . Motivations attribuées à l'usage d'une architecture profonde

Dans un apprentissage supervisé classique, on extrait d'abord des caractéristiques des données brutes (comme par exemple un histogramme sur une image)[77], le programmeur doit être très, très explicite lorsqu'il indique à l'ordinateur le type d'éléments qu'il doit rechercher pour déterminer l'erreur. C'est un processus laborieux appelé *extraction de caractéristiques*, et le taux de réussite de l'ordinateur dépend entièrement de la capacité du programmeur à définir de manière précise le jeu de caractéristiques. [78]

Puis on passe ces caractéristiques à un classifieur, seule cette phase est apprise de façon supervisée ; l'extraction de caractéristiques est statique. [77]


**Figure 4.1.** schéma général de l'apprentissage classique [77]

L'implication de la précision humaine pour définir les caractéristiques engendre des erreurs, ce qui nous a pousser à illiminer l'apprentissage classique dans notre approche.

De plus, certaines familles de fonctions pouvant être représentées par un réseau de neurones de profondeur k, nécessitent un nombre exponentiel d'unités afin d'être représentées par une architecture de profondeur moindre, soit de k −1. Ainsi, afin qu'un réseau peu profond puisse bien représenter n'importe quelle fonction, il devra posséder un nombre exponentiel de paramètres. On sait par ailleurs qu'une bonne généralisation passe

par un nombre d'exemples d'entraînement qui croît au moins linéairement avec le nombre de paramètres dans le réseau [79].

Il s'ensuit qu'une architecture peu profonde est inefficace en terme de paramètres nécessaires et donc de nombre nécessaire d'exemples, ce qui nous ont permis d'écarter cette possibilité aussi.

De plus, des études ont démontré que les architectures profondes étaient indispensables à une modélisation efficace de distributions complexes pour l'obtention d'un meilleur niveau de généralisation.[79]

Dans un apprentissage profond, on va directement travailler sur les données brutes. La phase d'extraction des caractéristiques est elle aussi apprise, et n'est plus statique.[77]


**Figure 4.2.** schéma général de l'apprentissage profond [77]

Donc, nous optons pour le deep learning pour la réalisation de notre travail, pour plus de détail sur le deep learning veuillez-vous référer au chapitre2.

## III. Motivations attribuées aux choix des techniques utilisées

Les types d'analyses du data mining sont six, ils se répartissent en deux techniques descriptive et prédictive, comme le montre le tableau suivant :

| Techniques descriptives | | Techniques prédictives | | |
|---|---|---|---|---|
| Corrélation simple | Corrélation complexe | Présent | | Futur |
| | | Variable cible numérique | Variable cible catégorielle | |
| 1 : Description | 2 : Classification | 4 : Estimation | 5 : Segmentation | 6 : Prévision |
| | 3 : Association | | | |

**Tableau 1.4** Comparaison entre techniques descriptives et prédictives [83]

1. Les techniques descriptives [83]

(archétype : la classification) On les appelle aussi :technique non supervisées, ils permettent de :

• Décrire.

• Résumer, synthétiser, réduire, classer.

• Mettre en évidence des informations présentes mais cachées par le volume des données.

• Pas de variable cible à prédire.

• Elles produisent des modèles de classement : typologie, méta-typologie.

2. Les techniques prédictives [83]

(archétype : le scoring) : On les appelle aussi : techniques supervisées.

• Prédire.

• Extrapoler de nouvelles informations à partir des informations présentes.

• Les techniques prédictives présentent une variable cible à prédire.

• L'objectif est de prévoir la variable cible mais aussi de classer à partir de la variable cible.

• Elles sont plus délicates à mettre en œuvre que les techniques descriptives.

• Elles demandent plus d'historique que les techniques descriptives.

• Elles produisent des modèles de prédiction.

Les modèles d'architectures du deep-learning utilisent des méthodes prédictives autrement dit de classification, par contre les méthodes d'extraction des motifs fréquents, sont des méthodes descriptives, notre travail consiste à appliquer une méthodes prédictive :deep learning (utilisations des auto-encodeurs que nous allons voir leurs utilité

par la suite) pour traiter les données qui seront par la suite utilisé par la technique descriptive pour extraire les itemsets fréquents incertains. Ce qui est considéré comme travail de recherche original, car l'alliance de ces deux techniques n'a pas fait l'objet de traitement par les chercheurs auparavant.

## IV. Prétraitement des données

Les ensembles données (datasets) que nous avons utilisés proviennent du site : http://fimi.ua.ac.be/data/, préparés par Roberto Bayardo à partir des jeux de données UCI et de la PUMSB.

Le référentiel d'apprentissage automatique UCI (UCI Machine Learning Repository ) est une collection de bases de données, de théories de domaine et de générateurs de données qui sont utilisés par la communauté d'apprentissage automatique pour l'analyse d'algorithmes d'apprentissage automatique.[80]

Ses datasets conviennent particulièrement à l'extraction de règles d'association, parmi les datasets téléchargeable dans ce site notre choix c'est porté sur 'Mashroom datasets'.

Notre travail consiste à manipuler des données incertaines, donc nous avons ajouté du bruit (noise) à ses datasets pour avoir des données probabilistes.

Le bruit que nous avons ajouté correspond à l'hypothèse qu'une probabilité existentielle d'un élément suit une distribution normale t ~ N (μ, σ2), où μ est tiré au hasard de la plage [0,7, 0,9] et σ au hasard tiré de la plage [1/21, 1/12]. Ensuite, nous tirons une valeur de t et l'affectons à un élément dans les jeux de données d'origine comme sa probabilité existentielle.

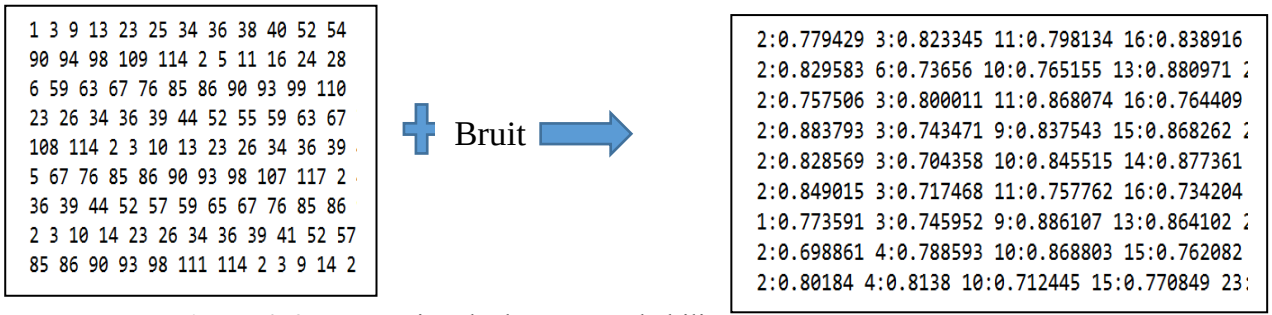



**Figure 4. 3 :** génération de données probabilistes à p

## V. Présentation de l'approche proposée

Dans la construction de notre système nous passons par deux phases, la première phase est l`entraînement du modèle et la deuxième phase est le déploiement et l'utilisation du modèle pour extraire les itemsets fréquents.

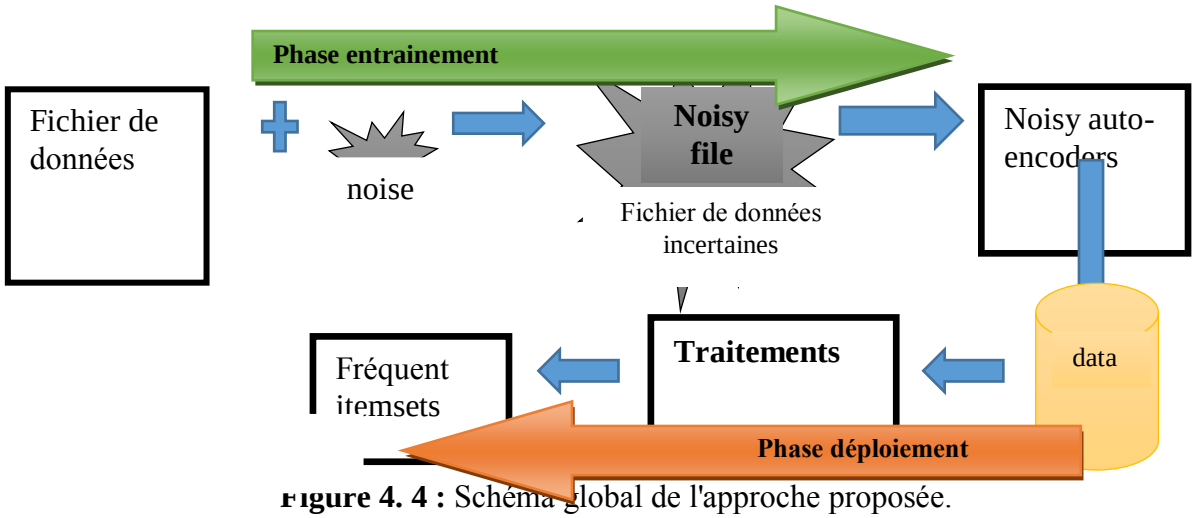



**Figure 4. 4 :** Schéma global de l'approche proposée.

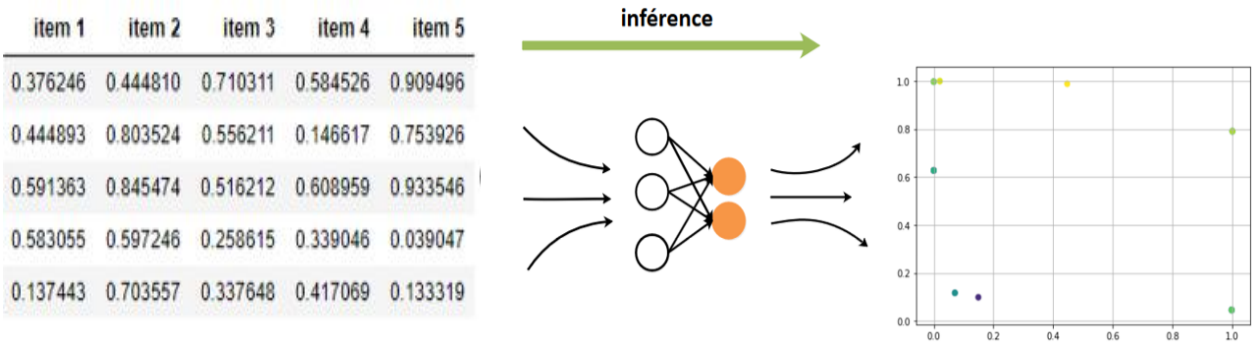
### 1. Phase d'entrainement

Dans cette phase, nous créons un modèle et nous l'entraînons sur un dataset de listes d'itemsets, jusqu'à ce que nous obtenions un modèle entrainé qui est prêt à être utilisé pour extraire les itemsets fréquents.

Ce fichier (modèle entrainé) sera sauvegardé puis utilisé comme entrée pour l'auto-encodeurs de débruitage (DEAD).

### 2. Phase de déploiement

Dans cette phase, le modèle est déjà entraîné. Il suffit d'utiliser une partie ou la globalité du model pour trouver les itemsets fréquents.



| item 1 | item 2 | item 3 | item 4 | item 5 |
|---|---|---|---|---|
| 0.376246 | 0.444810 | 0.710311 | 0.584526 | 0.909496 |
| 0.444893 | 0.803524 | 0.556211 | 0.146617 | 0.753926 |
| 0.591363 | 0.845474 | 0.516212 | 0.608959 | 0.933546 |
| 0.583055 | 0.597246 | 0.258615 | 0.339046 | 0.039047 |
| 0.137443 | 0.703557 | 0.337648 | 0.417069 | 0.133319 |



**Figure 4. 5 :** La phase de déploiement du modèle.

# VI. Modèle de réseau profond adopté

Comme nous l'avons déjà dit au préalable, nous allons utiliser en premier lieu une méthode prédictive pour entrainer notre dataset, et le modèle de neurones le plus approprié c'est les auto-encodeurs.

## 1. Les motivations pour utiliser l'auto-encodeur

- **Les étiquettes des transactions**

L'étape la plus essentielle dans l'entrainement des réseaux de neurones est le calcul de la fonction de coût, qui est calculé de la façon suivante (étiquette de transaction – sortie produite par le réseau), donc on ne peut pas entrainer un réseau si on n'a pas ses étiquettes.

- **Les propriétés de l'espace latent**

L'espace latent des auto-encodeurs possède la propriété de regrouper les transactions identiques donc on peut utiliser cette propriété pour grouper les transactions et calculer l'expected supports de ces transactions.

## 2. Dimensions de l'auto-encodeur utilisé

Nous avons choisi les nombres de neurones et de couches suivants :

- **Nombre de neurones**
  - Couche d'Entrée : nombre de neurones = nombre des items.
  - Couche cachée 1 : nombre de neurones =nombre des items/ 2 (division entiére)
  - Couche cachée 2 : un
  - Couche cachée 3 : nombre de neurones =nombre des items/ 2 (division entiére)
  - Couche de sortie : nombre de neurones = nombre des items.

- **Nombre de couches**

  Pour ne pas tomber dans le problème d'overfitting nous avons opté pour 5 couches :
  - Couche d'Entrée
  - Couche de sortie
  - 3 Couches intermédiaires

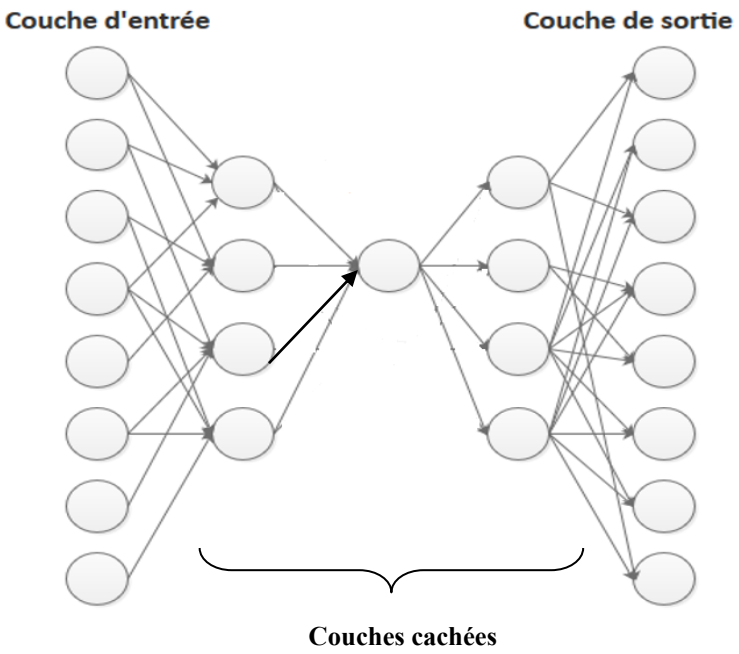



**Figure 4. 6 :** Illustration du nombre de couche et de neurones dans le modèle.

### 3. Fonction d'activation

Les fonctions d'activation sont une composante extrêmement importante des réseaux de neurones profonds. Ils décident si un neurone doit être activé ou non. Si l'information que le neurone reçoit est pertinente pour l'information donnée ou si elle doit être ignorée [77].

$$Y = \text{Activation}\left(\sum (w \times x) + b\right)$$

**W**: poids, **X** : entrée, **B** : biais

Pour notre travaille nous avons opté pour la fonction d'activation Sigmoid. La raison principale pour laquelle nous utilisons la fonction Sigmoid est qu'elle existe entre (0 à 1). Cette propriété de la fonction sigmoïde est utile lorsque nous travaillons avec des listes d'itemsets, de plus c'est l'intervalle qui nous intéresses car nous manipulons des probabilités.

Équation de la fonction Sigmoid : $f(x) = \frac{1}{1+e^{-x}}$

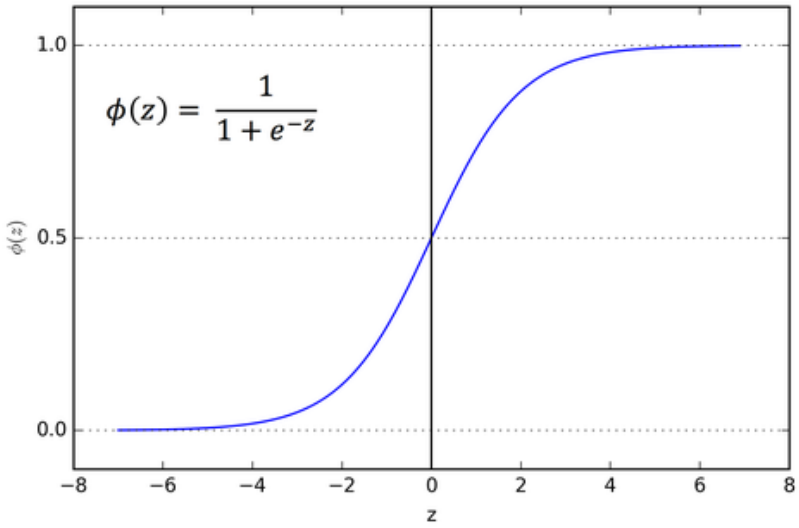



**Figure 4. 6 :** Représentation graphique de la fonction d'activation Sigmoid [78].

- **Initialisation des poids**

Dans notre modèle nous avons initialisé les poids avec l'initialisation de Xavier [88] :

$$var(W) = \frac{2}{Nin+Nout}$$

**N*in***, **N*out* :** nombre des entrées et de sortie dans la couche.

- **Initialisation des biais**

Nous initialisons les biais à zéro, puisque la rupture de l'asymétrie est fournie par les petits nombres dans les poids [87].

## V. Pseudo algorithme

L'algorithme proposé est nommé UN-Autoencoder, il se présente comme suit :

**Algorithme : UN-Autoencoder**

- **Phase d'entrainement :**

Entrainer le noisy file pour n itérations

- **Phase de déploiement :**

**Entrées :** *données entrainées*
**Pour** chaque (transactions) *Tj* **faire**
Obtenir sa représentation latente
**Si la** représentation latent existe déjà **alors** multiplier les exp-supports.
**Sinon**
Ajouter la représentation latente
**Fin Si.**
**Fin Pour.**
**Sortie :** itemsets globaux avec leurs exp-supports.

- **Phase d'extraction des itemsets fréquents :**

**Entrées :** itemsets globaux, minSup.
**Pour** chaque **item** calculer exp-support.

**Si** (exp-Supp-item < minSup*)* **alors** éliminer l'item.
**Fin Si**
**Fin Pour.**
Sorties : les itemsets fréquents incertain et leurs exp-support réspectifs

## VI. Conclusion

Dans ce chapitre nous avons proposé un modèle de réseau neuronal profond (auto-encodeur) qui peut être utilisé pour trouver des itemsets fréquents. Nous avons montré les étapes à prendre et les paramètres à considérer pour entraîner un auto-encodeur.

Dans le chapitre suivant on définit les outils et logiciels de programmation utilisés pour implémenter et tester notre algorithme.

CHAPITRE 5 : ***tests et expérimentations***

## *I. Introduction*

Après avoir décrit notre solution, nous aborderons dans ce chapitre la partie des expérimentations de notre approche. En première partie, nous présenterons l'environnement de travail et les outils de développement utilisés. Ensuite, nous présentons notre jeu d'essai. Suivie d'une discussion pour mieux comprendre notre approche.

Enfin, nous présenterons notre application et les résultats obtenus

## II. Environnement de développement

### 1. Environnement matériel

Un ordinateur portable ASUS avec les caractéristiques suivantes :

- 4 GO RAM
- Intel® Core(TM) i7-4500U CPU 1.8 GHz, 2401 MHz, 2 coeur(s),

4 processurs logiques

- Système d'exploitation : Windows 8.1 de 64-bit

### 2. Environnement logiciel

#### A. Python :

C'est un langage de programmation dynamique de haut niveau, interprété et polyvalent, qui met l'accent sur la lisibilité du code. La syntaxe dans Python aide les programmeurs à coder en moins d'étapes que Java ou C ++. Le langage fondé en 1991 par le développeur Guido Van Rossum présente une programmation facile et amusante. Le Python est largement utilisé dans les grandes organisations en raison de ses multiples paradigmes de programmation. Ils impliquent généralement une programmation fonctionnelle impérative et orientée objet. Il possède une vaste bibliothèque standard complète dotée d'une gestion automatique de la mémoire et de fonctionnalités dynamiques.[68]

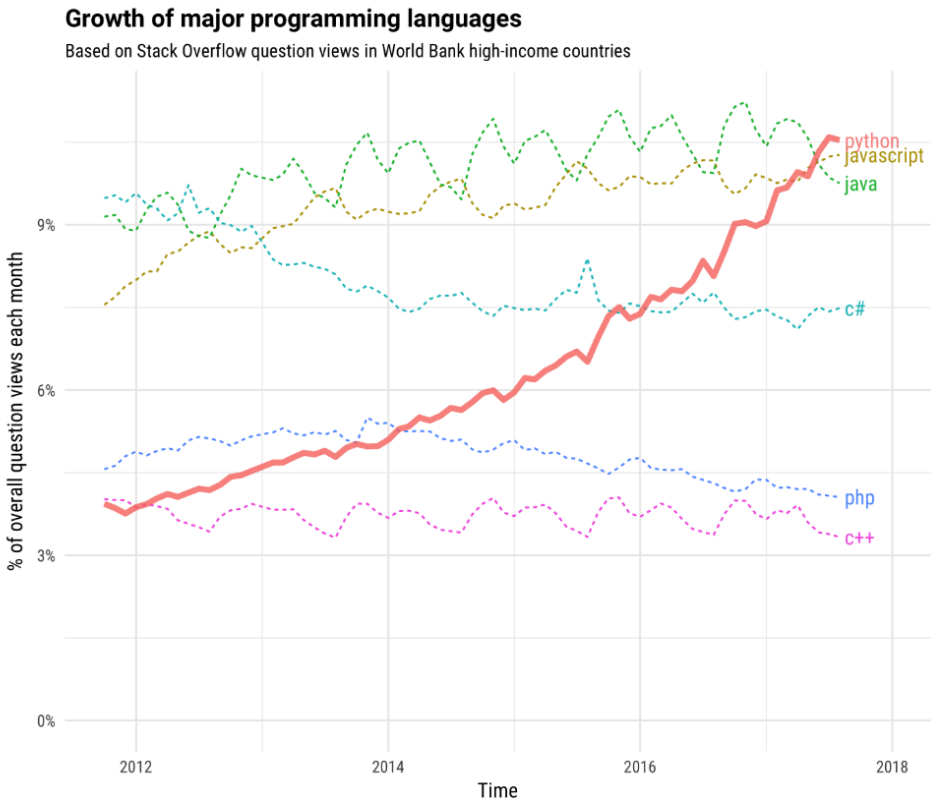



**Figure 5. 1 :** La croissance des langages de programmation [69].

## B. Jupyter Notebook

Jupyter Notebook est une application client-serveur créée par l'organisation à but non lucratif Project Jupyter. Elle a été publiée en 2015. Elle permet la création et le partage de documents Webauformat JSON constitués d'une liste ordonnée de cellules d'entrées et de sorties et organisés en fonction des versions successives du document. Les cellules peuvent contenir, entre autres, du code, du texte au format Markdown, des formules mathématiques ou des contenus médias (Rich Media). Le traitement se fait avec une application client fonctionnant par Internet, à laquelle on accède par les navigateurs habituels.[72]

Un Jupyter Notebook, c'est deux choses à la fois [73]:

- Une application Web interactive dans laquelle on peut développer, documenter, exécuter et partager du code. C'est un excellent outil notamment dans le domaine scientifique : vous importez des données, vous les affichez, vous les étudiez, vous les exploitez avec des algorithmes traduits en programmes Python, il est utilisé aussi pour l'apprentissage automatique.

- Un document qui permet d'intégrer en plus du code, des équations écrites en LATEX, du texte formaté en Markdown (convertit du format txt en html), différents médias (audio, vidéo) et qui s'exporte dans un certain nombre de formats (HTML, PDF, LATEX…)

## C. Bibliothèques utilisées

### • Pandas

*Pandas* est une bibliothèque open source sous licence BSD fournissant des structures de données hautes performances et faciles à utiliser, ainsi que des outils d'analyse des données pour le langage de programmation Python .[70]

Pandas est une librairie python qui permet de manipuler facilement des données à analyser à travers [71]:

- Manipuler des tableaux de données avec des étiquettes de variables (colonnes) et d'individus (lignes),ces tableaux sont appelés DataFrames.
- On peut facilement lire et écrire ces dataframes à partir ou vers un fichier tabulé.
- On peut facilement tracer des graphes à partir de ces DataFrames grâce à matplotlib.

### • Tensorflow

TensorFlow est un outil open source d'apprentissage automatique développé par Google. Le code source a été ouvert le 9 novembre 2015 par Google et publié sous licence Apache.

Il est fondé sur l'infrastructure DistBelief, initiée par Google en 2011, et est doté d'une interface pour Python et Julia. Il est l'un des outils les plus utilisés en IA dans le domaine de l'apprentissage machine. Il dispose d'un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires qui permet aux chercheurs de faire évoluer l'état de l'art en ML (méta langage) et aux développeurs de créer et de déployer facilement des applications propulsées par ML.[75]

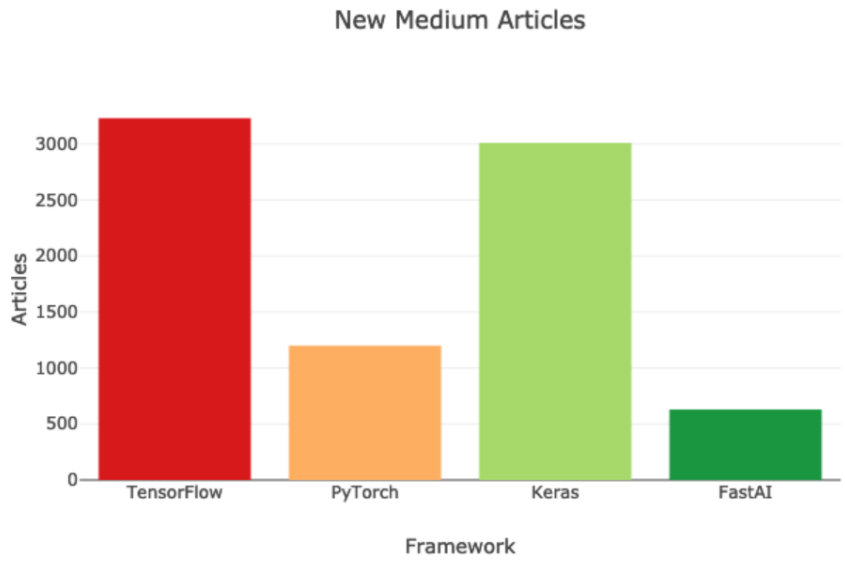



**Figure 5.2** comparaison entre les principales framework(juin 2019) d'après des articles [75]

- Numpy

Son nom complet: Numeric Python. Cela signifie que le module gère les nombres. NumPy est un logiciel à code source ouvert pour la création et la gestion de tableaux multidimensionnels et matrices. Cette bibliothèque comprend diverses fonctions permettant de gérer de tels tableaux complexes.[75]

## III. Création du modèle

Pour la création de notre modèle d'extraction des itemsets, on passe par deux étapes, l'étape importation des données et l'étape d'entrainement, et afin d'extraire les itemsets fréquents incertains, nous allons exploiter l'espace latent généré par les auto-encodeurs, et cela se fait en utilisant juste l'encodeurs sans le décodeur (la première partie des auto-encodeurs) car les auto-encodeurs rassemblent les itemsets ressemblant (identiques) ensembles (cote à cote).

### 1. Importation des données

Importons notre noisy dataset (après traitement, voir chapitre 4), avec deux chiffres après la virgule pour avoir un nombre important d'itemsets fréquents incertains, autant que fichier csv avec pandas.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.98 | 0.03 | 0.11 | 0.68 | 0.03 | 0.71 | 0.55 | 0.36 | 0.88 | 0.12 | 0.25 | 0.86 | 0.38 |
| 1 | 0.91 | 0.16 | 0.57 | 0.71 | 0.35 | 0.18 | 0.08 | 0.45 | 0.09 | 0.79 | 0.08 | 0.24 | 0.16 |
| 2 | 0.73 | 0.51 | 0.29 | 0.58 | 0.97 | 0.81 | 0.92 | 0.11 | 0.37 | 0.90 | 0.87 | 0.55 | 0.73 |
| 3 | 0.87 | 0.28 | 0.86 | 0.54 | 0.17 | 0.11 | 0.39 | 0.40 | 0.34 | 0.25 | 0.76 | 0.39 | 0.35 |
| 4 | 0.89 | 0.67 | 0.10 | 0.42 | 0.29 | 0.92 | 0.69 | 0.40 | 0.93 | 0.52 | 0.05 | 0.87 | 0.16 |
| 5 | 0.39 | 0.18 | 0.33 | 0.20 | 0.29 | 0.77 | 0.67 | 0.46 | 0.36 | 0.85 | 0.45 | 0.68 | 0.16 |
| 6 | 0.06 | 0.56 | 0.41 | 0.07 | 0.68 | 0.02 | 0.55 | 0.02 | 0.67 | 0.85 | 0.64 | 0.19 | 0.57 |
| 7 | 0.36 | 0.31 | 0.50 | 0.67 | 0.78 | 0.31 | 0.49 | 0.65 | 0.96 | 0.16 | 0.16 | 0.88 | 0.75 |
| 8 | 0.35 | 0.17 | 0.79 | 0.31 | 0.78 | 0.29 | 0.12 | 0.28 | 0.91 | 0.01 | 0.27 | 0.24 | 0.58 |
| 9 | 0.41 | 0.94 | 0.49 | 0.20 | 0.25 | 0.09 | 0.77 | 0.99 | 0.01 | 0.98 | 0.49 | 0.29 | 0.95 |
| 10 | 0.43 | 0.49 | 0.86 | 0.07 | 0.30 | 0.47 | 0.91 | 0.62 | 0.41 | 0.78 | 0.85 | 0.02 | 0.33 |
| 11 | 0.87 | 0.77 | 0.47 | 0.74 | 0.30 | 0.05 | 0.22 | 0.56 | 0.71 | 0.50 | 0.82 | 0.71 | 0.45 |
| 12 | 0.64 | 0.14 | 0.39 | 0.89 | 0.96 | 0.92 | 0.95 | 0.51 | 0.19 | 0.79 | 0.83 | 0.18 | 0.29 |
| 13 | 0.28 | 0.37 | 0.51 | 0.61 | 0.87 | 0.08 | 0.22 | 0.01 | 0.49 | 0.99 | 0.69 | 0.25 | 0.36 |

**Figure 5.3.** Partie d'un data sets csv utilisé

## 2. Optimisation et entrainement

### A. Cout (loss) et optimisation

Souvent, lorsque nous construisons un modèle d'apprentissage automatique, nous développons une fonction de coût capable de mesurer l'efficacité de notre modèle. Cette fonction pénalisera toute erreur de notre modèle (en assignant un coût) par rapport aux valeurs actuelles des paramètres. Ainsi, en minimisant la fonction de coût, nous pouvons trouver les paramètres optimaux qui donnent les meilleures performances du modèle

Pour notre model on a utilisé Mean squared error (Erreur quadratique moyenne) comme fonction de coût. Supposons que $\hat{X}_i$ soit le vecteur dénotant les valeurs de n nombre de prédictions. Aussi, $X_i$ est un vecteur représentant n nombre de valeurs vraies, La formule de l'erreur quadratique moyenne est donnée ci-dessous :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Xi} - Xi)^2$$

Lorsque la valeur de MSE est nulle, cela signifie qu'il y a une précision parfaite trouvée par l'estimateur. Cette condition est idéale et généralement impossible à réaliser dans la pratique [81].

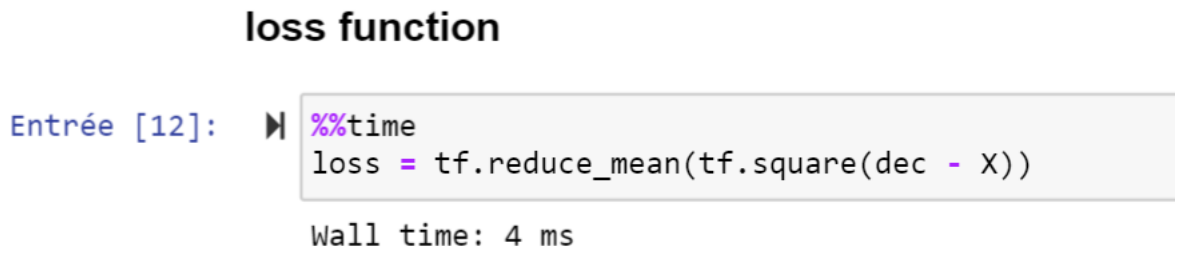



**Figure 5.4.** loss function de notre application utilisé

### B. Optimisation de la fonction cout

Le choix de l'algorithme d'optimisation pour votre modèle d'apprentissage en profondeur peut faire la différence entre de bons résultats en minutes, heures et jours.

C'est dernière années plusieurs algorithmes ont étaient proposés pour optimiser la fonction cout et réduire l'erreur, pour notre cas nous avons optimiser le fonction cout en utilisant Adam (voir annexe)

- Adam : Adaptive Moment Estimation

Adam est un algorithme d'optimisation qui peut être utilisé à la place de la procédure classique de descente de gradient stochastique pour mettre à jour les poids de réseau itératifs basés sur les données d'entraînement.[82], il a était considéré en 2019, comme dernière tendance en matière d'optimisation de l'apprentissage en profondeur. (latest trend in deep learning optimization)

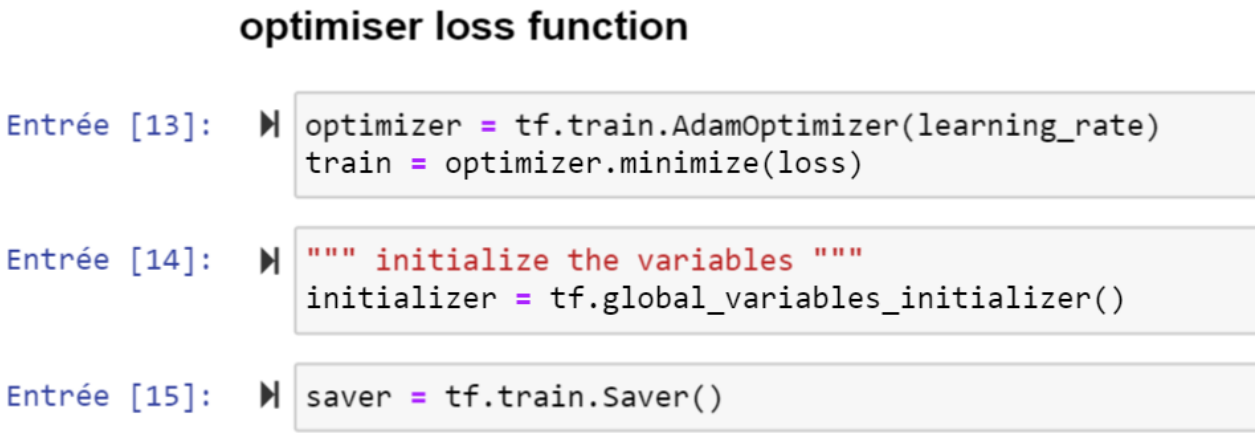
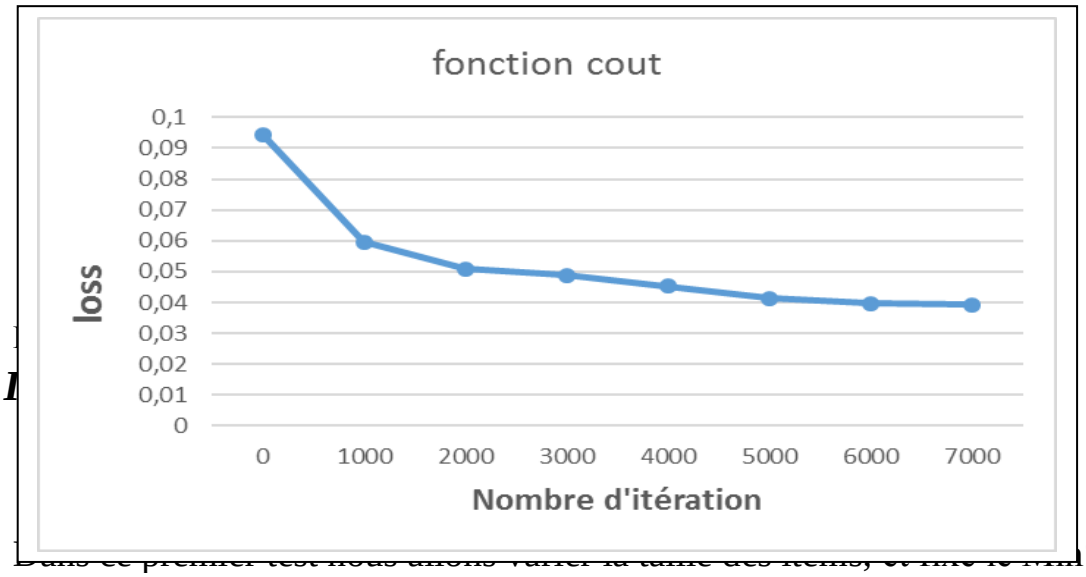



**Figure 5.5.** Utilisation de la fonction Adam pour l'optimisation de la fonction cout

## 3. L'entrainement

Dans cette phase, nous entraînerons le modèle pour 7000 itérations jusqu'à ce que la fonction de coût converge. La figure 5.4 montre la minimisation de la fonction de coût par rapport au nombre d'itérations nous remarquons que la fonction de cout n'a pas à partir de la

5000 itération, donc on peut entrainer notre modèle pour 5000 itérations la valeur de la fonction de cout sera toujours la même.




Sup = 1, et le nombre d'itération à 100, et nous allons constater le temps de réponse

| nb items | 6 | 13 | 20 | 50 | 80 |
|---|---|---|---|---|---|
| Temps de réponse (ms) | 76 | 60 | 69,1 | 458 | 476 |

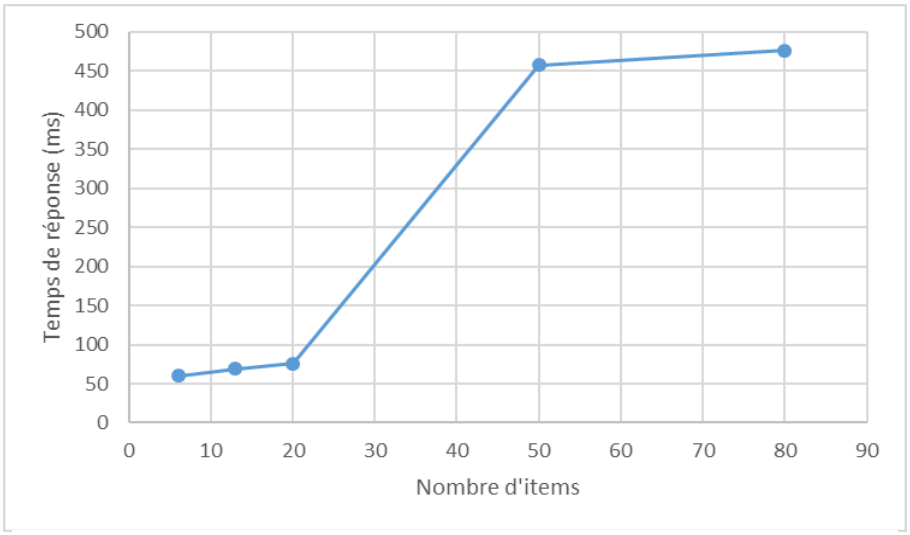



Figure 5.7 temps de réponse par rapport au nombre d'items

Interprétation :

Le temps de réponse croit avec le nombre d'items, mais ça reste toujours < 1seconde

## 2. Deuxième expérience

Dans cette expérience nous allons fixer le MinSup =1 et le nombre de transition (itération) à 100 , et nous allons voir l'espace mémoire du dataset

| nb items | 6 | 13 | 20 | 50 | 80 |
|---|---|---|---|---|---|
| espace mémoire (KB) | 4,8 | 10,2 | 15,6 | 38,8 | 62 |

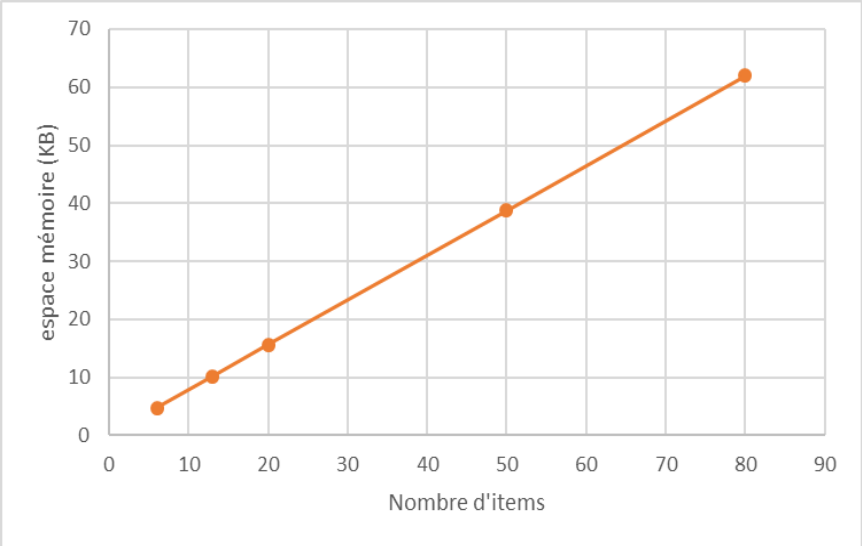



**Figure 5.8** espace mémoire par rapport au nombre d'items

Espace mémoire après compression du dataset, est obtenu grâce à la fonction info(), par exemple pour un data set de taille 600, 6 items et 100 transactions, après la compression résultant de notre modèle, on obtient 99 transaction avec leurs supports général, ces transactions vont être utilisée pour obtenir l'exp-support des itemsets fréquent final

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 6 columns):
0.24    99 non-null float64
0.85    99 non-null float64
0.57    99 non-null float64
0.59    99 non-null float64
0.63    99 non-null float64
0.38    99 non-null float64
dtypes: float64(6)
memory usage: 4.8 KB
```

*Figure 5.9* **:** Information sur la base de donnes après la compression.

### 3. Troisième expérience

Nous allons varier le MinSup, le nombre de transactions est fixé à 100, et le nombre d'items à 80, et nous allons voir le nombre d'itemsets généré par notre application

| Min-sup | 3 | 1 | 0 |
|---|---|---|---|
| nb itemsets | 26 | 163 | 819 |

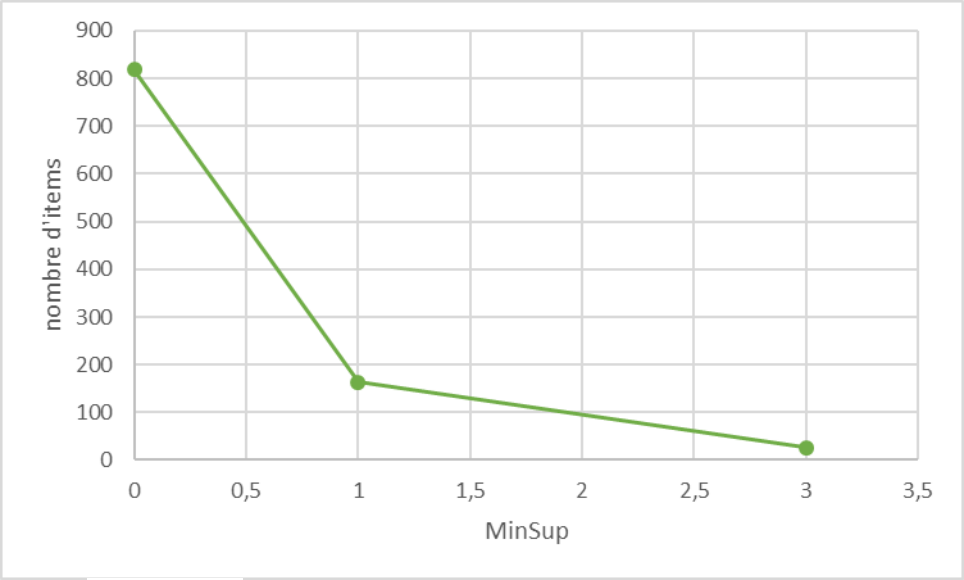



**Figure 5.10** Nombre d'itemsets généré par rapport à MinSup

Interprétation

Le nombre d'itemset diminue à chaque fois que min-sup augmente (voir chapitre 1)

### 4. Quatrième expérience

Dans cette expérience nous allons varier la taille des datasets, et nous calculons le temps de la fonction cout

| taille du dataset | 600 | 1300 | 2000 | 5000 | 8000 |
|---|---|---|---|---|---|
| loss (ms) | 4 | 4 | 4 | 4 | 4 |

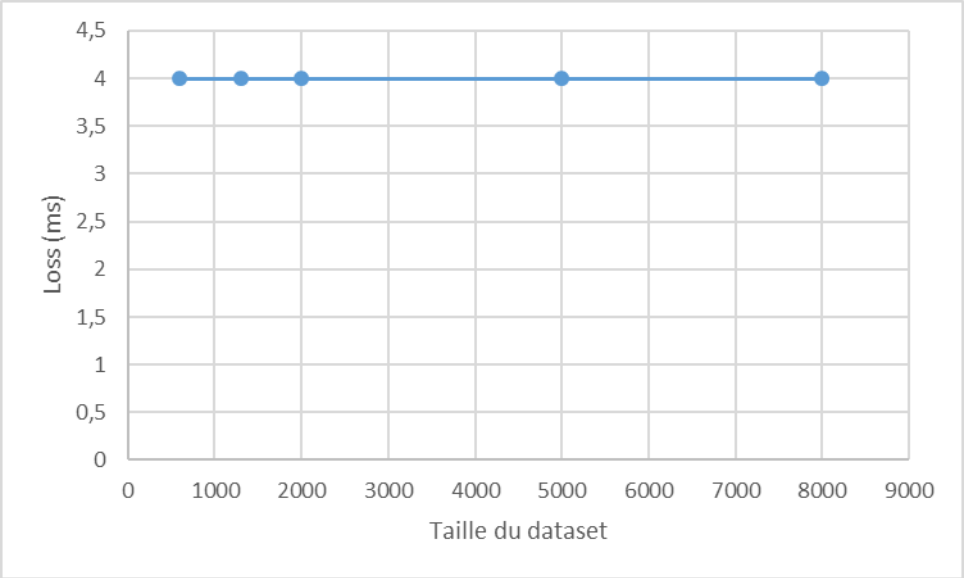



**Figure 5.11.** fonction cout par rapport à la taille des datasets

On remarque que le temps de la fonction cout est fixe quelque soit la taille du dataset

## *V. Avantage d'utilisation de notre approche*

### 1. Base de données qui contient beaucoup de répétitions

Les bases de données qui contiennent beaucoup de répétitions seront plus compressées, ce qui se traduira par un meilleur temps d'exécution.

2. **Si on veut une meilleure compréhension des résultats**
   Cette avantage est expliqué dans les apportes de notre approche.
3. Utilisation du modèle entrainé pour manipuler les fichiers quelque soit leur taille, le temps est toujours le même

- Les inconvénients de notre approche

- **Les réseaux de neurones profonds sont des boites noires**
  Nous sommes confrontés à ce problème lorsque nous essayons de donner un sens à l'espace latent dans le but de l'exploiter un peu plus, parce que le nombre de paramètres et de variables est important.
  **Solution :**
  Visualiser l'espace latent pour une meilleure compréhension des résultats.

## *VI. Conclusion*

Dans ce chapitre nous avons présenté l'environnement matériel et l'environnement logiciel, les bibliothèques utilisées pour implémenter notre approche. On a effectué des tests pour vérifier la qualité des résultats de notre approche et son l'efficacité.

# *Conclusion générale et Perspectives*

## 1. Conclusion générale

L'extraction des motifs fréquents à partir des données incertaines reste toujours l'un des sujets d'actualités pour les chercheurs car les domaines et applications qui lui font appels ne cessent de s'accroitre.

A travers ce document, on a abordé le domaine d'extractions des itemsets fréquents à partir des données incertaines, à travers l'étude des principaux algorithmes qui traitent ce sujet ainsi que les lois les utilisant dans le premier chapitre, par la suite nous avons étudié le deep learning et les réseaux de neurones où nous avons appris leur fonctionnement, nous avons consacré le troisième chapitre à l'étude des auto-encodeurs, vu que notre application se basée dessus, dans le quatrième chapitre nous avons proposé notre approche qui consiste à la création et l'entrainement d'un auto-encodeur (technique prédictives) en utilisant sa partie encodage pour faire une compression des données, et puis l'extraction des motifs fréquents (technique descriptives) en utilisant l'expected support. Enfin nous avons terminé notre travail par le chapitre cinq, test et validation qui nous a permis d'implémenter et tester notre solution.

Par la même occasion, ce travail nous a permis d'approfondir nos connaissances sur la notion d'extraction d'itemsets fréquents, et d'apprendre de nouvelles techniques concernent le Deep Learning.

## 2. Perspectives

Les perspectives de ce travail préliminaire sont nombreuses. Tout d'abord, nous devons valider la démarche proposée avec des algorithmes réels, U-Apriori par exemple, en terme de performance et exactitude.

Il est probablement intéressant de pouvoir adopter notre approche pour traiter le problème des valeurs imparfaites (manquantes, imprécises...) et voir comment ces méthodes pourraient traiter ces données.

Utiliser d'autre supports de calcul, pour voir les résultats obtenus et faire une comparaison.

En fin, ce travail étant une œuvre humaine, ce n'est pas un modèle parfait, c'est pourquoi nous restons ouverts à toutes les critiques et sommes prêts à recevoir toutes les suggestions et remarques tendant à améliorer davantage cette étude, étant donné que tout travail informatique a été toujours l'œuvre d'une équipe.

# Annexes

## 1. Importation des données avec pandas

**IMPORTS THE NOISY FILE**

Entrée [3]:
```
df = pd.read_csv ('C:\\Users\\s.abadi\\Desktop\\test1\\test.csv',sep=';',header=None)
df.head(20)
```

Out[3]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.98 | 0.03 | 0.11 | 0.68 | 0.03 | 0.71 | 0.55 | 0.36 | 0.88 | 0.12 | 0.25 | 0.86 | 0.38 |
| 1 | 0.91 | 0.16 | 0.57 | 0.71 | 0.35 | 0.18 | 0.08 | 0.45 | 0.09 | 0.79 | 0.08 | 0.24 | 0.16 |
| 2 | 0.73 | 0.51 | 0.29 | 0.58 | 0.97 | 0.81 | 0.92 | 0.11 | 0.37 | 0.90 | 0.87 | 0.55 | 0.73 |
| 3 | 0.87 | 0.28 | 0.86 | 0.54 | 0.17 | 0.11 | 0.39 | 0.40 | 0.34 | 0.25 | 0.76 | 0.39 | 0.35 |
| 4 | 0.89 | 0.67 | 0.10 | 0.42 | 0.29 | 0.92 | 0.69 | 0.40 | 0.93 | 0.52 | 0.05 | 0.87 | 0.16 |
| 5 | 0.39 | 0.18 | 0.33 | 0.20 | 0.29 | 0.77 | 0.67 | 0.46 | 0.36 | 0.85 | 0.45 | 0.68 | 0.16 |
| 6 | 0.06 | 0.56 | 0.41 | 0.07 | 0.68 | 0.02 | 0.55 | 0.02 | 0.67 | 0.85 | 0.64 | 0.19 | 0.57 |
| 7 | 0.36 | 0.31 | 0.50 | 0.67 | 0.78 | 0.31 | 0.49 | 0.65 | 0.96 | 0.16 | 0.16 | 0.88 | 0.75 |
| 8 | 0.35 | 0.17 | 0.79 | 0.31 | 0.78 | 0.29 | 0.12 | 0.28 | 0.91 | 0.01 | 0.27 | 0.24 | 0.58 |
| 9 | 0.41 | 0.94 | 0.49 | 0.20 | 0.25 | 0.09 | 0.77 | 0.99 | 0.01 | 0.98 | 0.49 | 0.29 | 0.95 |
| 10 | 0.43 | 0.49 | 0.86 | 0.07 | 0.30 | 0.47 | 0.91 | 0.62 | 0.41 | 0.78 | 0.85 | 0.02 | 0.33 |
| 11 | 0.87 | 0.77 | 0.47 | 0.74 | 0.30 | 0.05 | 0.22 | 0.56 | 0.71 | 0.50 | 0.82 | 0.71 | 0.45 |

## 2. Création du model avec TensorFlow

# AUTOENCODER

## number of inputs and the hidden layer and the number of outputs

Entrée [6]:
```
num_inputs = nb_item
hidden_neu_1 = nb_item // 2
hidden_neu_2 = 1
hidden_neu_3 = hidden_neu_1
num_outputs = num_inputs
learning_rate = 0.01
```

Entrée [7]:
```
X = tf.placeholder(tf.float32, shape=[None,num_inputs])
```

Entrée [8]:
```
activ_func = tf.nn.sigmoid
```

Entrée [9]:
```
def encoder(x):
    with tf.variable_scope("encoder", reuse=tf.AUTO_REUSE):
        hidden_layer_1 = tf.layers.dense(inputs=X, units=hidden_neu_1, activation=activ_func)
        z  = tf.layers.dense(inputs=hidden_layer_1, units=hidden_neu_2, activation=activ_func)

    return z
```

Entrée [10]:
```
def decoder(Z):
    with tf.variable_scope("decoder", reuse=tf.AUTO_REUSE):
        hidden_layer_3 = tf.layers.dense(inputs=Z, units=hidden_neu_3, activation=activ_func)
        output_layer = tf.layers.dense(inputs=hidden_layer_3, units=num_outputs, activation=activ_func)
    return output_layer
```

Entrée [11]:
```


sampled = encoder(X)
dec = decoder(sampled)
```

## RUNNING SESSION

Entrée [16]:

```
num_steps = 7000

with tf.Session() as sess:
    sess.run(initializer)

    for iteration in range(num_steps+1):
        outputs = sess.run([train, loss],feed_dict={X:df})
        if iteration % 1000 == 0:
            print('Iter: {}'.format(iteration))
            print('Loss: {:.4}'. format(outputs[1]))
            print()
    saver.save(sess, './save/UNCERTAINautoencoder.ckpt')
```

```
Iter: 0
Loss: 0.1022

Iter: 1000
Loss: 0.06254

Iter: 2000
Loss: 0.0511

Iter: 3000
Loss: 0.04453

Iter: 4000
Loss: 0.0406

Iter: 5000
Loss: 0.03877

Iter: 6000
Loss: 0.03721

Iter: 7000
Loss: 0.03688
```

## Exemple de résultats avec min sup=1

Entrée [24]:

```
%%time
result = []
for itemset, support in find_frequent_itemsets(compressed_tran, compressed_tran_sup, 1, True):

    result.append((itemset,support))
```

```
Wall time: 362 ms
```

Entrée [25]:

```
res_df = pd.DataFrame(result, columns=['itemset', 'supp'])

res_df.sort_values("supp", inplace=True, ascending=False)
res_df
```

Out[25]:

| | itemset | supp |
|---|---|---|
| **4095** | [i13] | 5 |
| **7167** | [i11, i12, i13] | 5 |
| **3071** | [i11, i12] | 5 |
| **6143** | [i12, i13] | 5 |
| **5119** | [i11, i13] | 5 |
| ... | ... | ... |
| **3634** | [i1, i2, i5, i6, i10, i11, i12] | 1 |
| **3632** | [i1, i5, i6, i10, i11, i12] | 1 |
| **3630** | [i1, i2, i3, i4, i6, i10, i11, i12] | 1 |
| **3628** | [i1, i3, i4, i6, i10, i11, i12] | 1 |
| **8190** | [i1, i2, i3, i4, i5, i6, i7, i8, i9, i10, i11,... | 1 |

## Bibliographie :


1. Nombré CI, Brou K, Kimou K. ALOA2i: OPTIMISATION D'EXTRACTION DES K-ITEMSETS FREQUENTS (POUR K≤ 2) ALOA2i: OPTIMIZATION OF EXTRACTION K-itemsets FREQUENT (FOR K≤ 2). 2016;
2. Takfarinas KENOUCHE. Présentation sur le Data Mining [Internet]. Données & analyses présenté à; 06:47:45 UTC [cité 22 janv 2020]. Disponible sur: https://fr.slideshare.net/takfa92i/prsentation-data-mining
3. fr_Tanagra_Itemset_Mining.pdf [Internet]. [cité 22 janv 2020]. Disponible sur: http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/fr_Tanagra_Itemset_Mining.pdf
4. Bouzourine A, Taha Aidi fecel. Deep Learning pour l'extraction des itemsets Fréquents [MASTER 2, ingénieurie des logiciels]. saad dahleb BLIDA; 2018.
5. BENATTOU I, LARBI K. Extraction des itemsets fréquents à partir des données imparfaites. [BLIDA]: saad dahleb BLIDA; 2016.
6. Bernecker et al. - Probabilistic Frequent Pattern Growth for Itemset .pdf. Disponible sur: https://www.sciencedirect.com/science/article/pii/S1877050915023224
7. Chui C-K, Kao B, Hung E. Mining Frequent Itemsets from Uncertain Data. In: Zhou Z-H, Li H, Yang Q, éditeurs. Advances in Knowledge Discovery and Data Mining [Internet]. Berlin, Heidelberg: Springer Berlin Heidelberg; 2007 [cité 23 janv 2020]. p. 47-58. Disponible sur: http://link.springer.com/10.1007/978-3-540-71701-0_8
8. Cuzzocrea A, Leung CK, MacKinnon RK. Approximation to Expected Support of Frequent Itemsets in Mining Probabilistic Sets of Uncertain Data. Procedia Comput Sci. 2015;60:613-22.
9. Bernecker T, Kriegel H-P, Renz M, Verhein F, Zuefle A. Probabilistic frequent itemset mining in uncertain databases. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09 [Internet]. Paris, France: ACM Press; 2009 [cité 23 janv 2020]. p. 119. Disponible sur: http://portal.acm.org/citation.cfm?doid=1557019.1557039
10. Bernecker T, Kriegel H-P, Renz M, Verhein F, Zü A. Probabilistic Frequent Pattern Growth for Itemset Mining in Uncertain Databases (Technical Report). :21.
11. Yue - 2015 - Review of Algorithm for Mining Frequent Patterns f.
12. Calders et al. - 2010 - Efficient Pattern Mining of Uncertain Data with Sa.
13. Cours_de_data_mining_3-Modelisation-EPF.
14. A Brief Survey on Frequent Patterns Mining of Uncertain Data.
15. Tobji MAB, Yaghlane BB. Extraction des itemsets fréquents à partir de données évidentielles : application à une base de données éducationnelles. :22.
16. Mining Frequent Itemsets from Uncertain Data Presented by Chun-Kit Chui, Ben Kao, Edward Hung Department of Computer Science, The University of Hong Kong. - ppt download [Internet]. [cité 23 janv 2020]. Disponible sur: https://slideplayer.com/slide/4777765/
17. Machine Learning : Introduction à l'apprentissage automatique | SUPINFO, École Supérieure d'Informatique [Internet]. [cité 23 janv 2020]. Disponible sur: https://www.supinfo.com/articles/single/6041-machine-learning-introduction
18. DDOC_T_2018_0008_ZIMMER. :341–356, 1982
19. Apprentissage Supervisé Vs. Non Supervisé [Internet]. Le DataScientist. 2019 [cité 15 sept 2019]. Disponible sur: https://le-datascientist.fr/apprentissage-supervise-vs-non-supervise
20. Snapshot [Internet]. [cité 23 janv 2020]. Disponible sur: https://le-datascientist.fr/apprentissage-supervise-vs-non-supervise

21. Salsabil et Amaria - présenté et soutenu publiquement par.pdf [Internet]. [cité 23 janv 2020]. Disponible sur: http://dspace.univ-tlemcen.dz/bitstream/112/10630/1/Ms.EBM.Lakhdari%2BSaidi.pdf
22. Auto-encodeur [Internet]. Data Analytics Post. Disponible sur: https://dataanalyticspost.com/Lexique/auto-encodeur/
23. Khandelwal R. Deep Learning Autoencoders [Internet]. Medium. 2019 [cité 23 janv 2020]. Disponible sur: https://medium.com/datadriveninvestor/deep-learning-autoencoders-db265359943e
24. Auto-encodeur — Wikipédia [Internet]. [cité 23 janv 2020]. Disponible sur: https://fr.wikipedia.org/wiki/Auto-encodeur
25. Biais [Internet]. Data Analytics Post. [cité 23 janv 2020]. Disponible sur: https://dataanalyticspost.com/Lexique/biais/
26. Filtrage avec auto-encodeur [Internet]. [cité 23 janv 2020]. Disponible sur: http://vision.gel.ulaval.ca/~jflalonde/cours/4105/h15/tps/results/projet/OLGAG28/index.html
27. Giguère - Auto-encodeurs et Word2vec.pdf [Internet]. [cité 23 janv 2020]. Disponible sur: http://www2.ift.ulaval.ca/~pgiguere/cours/DeepLearning/08-Autoencoder.pdf
28. Initiez-vous au Machine Learning [Internet]. OpenClassrooms. [cité 23 janv 2020]. Disponible sur: https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning
29. Snapshot [Internet]. [cité 23 janv 2020]. Disponible sur: https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning
30. Deep Learning Explained in 7 Steps - Updated [Internet]. Data Driven Investor. [cité 23 janv 2020]. Disponible sur: https://www.datadriveninvestor.com/deep-learning-explained/
31. Larochelle H. Étude de techniques d'apprentissage non-supervisé pour l'amélioration de l'entraînement supervisé de modèles connexionnistes [Internet]. Université de Montréal; 2009 [cité 23 janv 2020]. Disponible sur: http://oatd.org/oatd/record?record=handle%5C%3A1866%5C%2F6435
32. Snapshot [Internet]. [cité 23 janv 2020]. Disponible sur: https://medium.com/@LZigaDigital/episode1-le-vendredi-cest-ai-9eada2c4a6dc
33. R L. Focus : Le Réseau de Neurones Artificiels ou Perceptron Multicouche [Internet]. Pensée Artificielle. 2018 [cité 23 janv 2020]. Disponible sur: http://penseeartificielle.fr/focus-reseau-neurones-artificiels-perceptron-multicouche/
34. Entraînez un réseau de neurones simple [Internet]. OpenClassrooms. [cité 23 janv 2020]. Disponible sur: https://openclassrooms.com/fr/courses/4470406-utilisez-des-modeles-supervises-non-lineaires/4730716-entrainez-un-reseau-de-neurones-simple
35. L +Bastien. Réseau de neurones artificiels : qu'est-ce que c'est et à quoi ça sert ? [Internet]. LeBigData.fr. 2019 [cité 23 janv 2020]. Disponible sur: https://www.lebigdata.fr/reseau-de-neurones-artificiels-definition
36. les fonctions d'activations [Internet]. Racine Ly. [cité 23 janv 2020]. Disponible sur: https://www.racinely.com/post/les-fonctions-d-activation-part-i
37. Deep Learning, les fonctions d'activation | [Internet]. [cité 23 janv 2020]. Disponible sur: https://www.supinfo.com/articles/single/7923-deep-learning-fonctions-activation
38. chapitre3.pdf [Internet]. [cité 23 janv 2020]. Disponible sur: http://thesis.univ-biskra.dz/1558/7/chapitre3.pdf
39. Touzet - INTRODUCTION AU CONNEXIONNISME.pdf [Internet]. [cité 23 janv 2020]. Disponible sur: http://www.touzet.org/Claude/Web-Fac-Claude/Les_reseaux_de_neurones_artificiels.pdf
40. Doncescu - Les réseaux de neurones artificiels.pdf [Internet]. [cité 23 janv 2020]. Disponible sur: http://conf.laas.fr/ignotus/archives/Doncescu_reseaux_neuronnes.pdf
41. Khattabi - Introduction à la classification.pdf [Internet]. [cité 27 janv 2020]. Disponible sur: http://www.ieee.ma/uaesb/pdf/algo-classification.pdf
42. ZIGH EHLEM. Calibration d'une Camèra, Utilisation des Rèseaux de Neurones Artificiels. 2003. 129 p.

43. de Runz - Imperfection, temps et espace modélisation, analy.pdf [Internet]. [cité 27 janv 2020]. Disponible sur: https://tel.archives-ouvertes.fr/file/index/docid/560668/filename/CReSTIC-1615.pdf
44. Leung C, Mateo M, Brajczuk D. A Tree-Based Approach for Frequent Pattern Mining from Uncertain Data. 2008. 653 p.
45. Du K-L, Swamy MNS. Neural Networks and Statistical Learning [Internet]. London: Springer London; 2014 [cité 27 janv 2020]. Disponible sur: http://link.springer.com/10.1007/978-1-4471-5571-3
46. J14.pdf [Internet]. [cité 27 janv 2020]. Disponible sur: https://jep-taln2016.limsi.fr/actes/Actes%20JTR-2016/Papers/J14.pdf
47. Snapshot [Internet]. [cité 27 janv 2020]. Disponible sur: https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/
48. McClelland C. The Difference Between Artificial Intelligence, Machine Learning, and Deep Learning [Internet]. Medium. 2019 [cité 27 janv 2020]. Disponible sur: https://medium.com/iotforall/the-difference-between-artificial-intelligence-machine-learning-and-deep-learning-3aa67bff5991
49. RÉSEAUX DE NEURONES, L'apprentissage des réseaux de neurones formels - Encyclopædia Universalis [Internet]. [cité 27 janv 2020]. Disponible sur: https://www.universalis.fr/encyclopedie/reseaux-de-neurones-formels/5-l-apprentissage-des-reseaux-de-neurones-formels/
50. SHARMA S. What the Hell is Perceptron? [Internet]. Medium. 2019 [cité 27 janv 2020]. Disponible sur: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53
51. SHARMA S. Activation Functions in Neural Networks [Internet]. Medium. 2019 [cité 27 janv 2020]. Disponible sur: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
52. Le réseau de neurones artificiel - ppt video online télécharger [Internet]. [cité 27 janv 2020]. Disponible sur: https://slideplayer.fr/slide/5489483/
53. Amazon.fr - Python Machine Learning - Second Edition: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow - Sebastian Raschka, Vahid Mirjalili - Livres [Internet]. [cité 27 janv 2020]. Disponible sur: https://www.amazon.fr/Python-Machine-Learning-scikit-learn-TensorFlow/dp/1787125939
54. What's the Difference Between Machine Learning Techniques? | Electronic Design [Internet]. [cité 27 janv 2020]. Disponible sur: https://www.electronicdesign.com/markets/automotive/article/21804976/whats-the-difference-between-machine-learning-techniques
55. Qu'est ce qu'un réseau de neurones convolutif (ou CNN) ? [Internet]. OpenClassrooms. [cité 27 janv 2020]. Disponible sur: https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn
56. CS 230 - Pense-bête de réseaux de neurones convolutionnels [Internet]. [cité 27 janv 2020]. Disponible sur: https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-convolutionnels
57. Crouspeyre C. Comment les Réseaux de neurones à convolution fonctionnent [Internet]. Medium. 2017 [cité 27 janv 2020]. Disponible sur: https://medium.com/@CharlesCrouspeyre/comment-les-r%C3%A9seaux-de-neurones-%C3%A0-convolution-fonctionnent-b288519dbcf8
58. Karn U. An Intuitive Explanation of Convolutional Neural Networks [Internet]. Medium. 2016 [cité 27 janv 2020]. Disponible sur: https://medium.com/@ujjwalkarn/https-ujjwalkarn-me-2016-08-11-intuitive-explanation-convnets-7bf60a14746b
59. CS 230 - Pense-bête de réseaux de neurones récurrents [Internet]. [cité 27 janv 2020]. Disponible sur: https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-recurrents

60. Réseaux de neurones récurrents · Intelligence artificielle [Internet]. [cité 27 janv 2020]. Disponible sur: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/recurrent_neural_networks.html
61. Yan S. Understanding LSTM and its diagrams [Internet]. Medium. 2017 [cité 27 janv 2020]. Disponible sur: https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714
62. Dey R, Salem FM. Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. :5.
63. A Beginner's Guide to Generative Adversarial Networks (GANs) [Internet]. Pathmind. [cité 27 janv 2020]. Disponible sur: http://pathmind.com/wiki/generative-adversarial-network-gan
64. Dertat A. Applied Deep Learning - Part 3: Autoencoders [Internet]. Medium. 2017 [cité 27 janv 2020]. Disponible sur: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798
65. Dertat A. Applied Deep Learning - Part 1: Artificial Neural Networks [Internet]. Medium. 2017 [cité 27 janv 2020]. Disponible sur: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6
66. Monn D. Denoising Autoencoders explained [Internet]. Medium. 2017 [cité 27 janv 2020]. Disponible sur: https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2
67. Shafkat I. Intuitively Understanding Variational Autoencoders [Internet]. Medium. 2018 [cité 27 janv 2020]. Disponible sur: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf
68. Solutions M. Advantages and Disadvantages of Python Programming Language [Internet]. Medium. 2017 [cité 27 janv 2020]. Disponible sur: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121
69. The Incredible Growth of Python | Stack Overflow [Internet]. Stack Overflow Blog. 2017 [cité 27 janv 2020]. Disponible sur: https://stackoverflow.blog/2017/09/06/incredible-growth-python/
70. Bibliothèque d'analyse de données Python - pandas: Bibliothèque d'analyse de données Python [Internet]. [cité 27 janv 2020]. Disponible sur: https://pandas.pydata.org/
71. Introduction à Pandas [Internet]. [cité 27 janv 2020]. Disponible sur: http://www.python-simple.com/python-pandas/panda-intro.php
72. Jupyter Notebook [Internet]. IONOS Digitalguide. [cité 27 janv 2020]. Disponible sur: https://www.ionos.fr/digitalguide/sites-internet/developpement-web/jupyter-notebook/
73. Kraëber J-M. Tutoriel : Utiliser un Jupyter Notebook [Internet]. Lycée Antoine de Saint-Exupery. 2019 [cité 27 janv 2020]. Disponible sur: http://lycee-saint-exupery.fr/tutoriel-utiliser-un-jupyter-notebook/
74. TensorFlow Tutorial For Beginners [Internet]. DataCamp Community. 2019 [cité 27 janv 2020]. Disponible sur: https://www.datacamp.com/community/tutorials/tensorflow-tutorial
75. TensorFlow [Internet]. TensorFlow. [cité 27 janv 2020]. Disponible sur: https://www.tensorflow.org/?hl=fr
76. Cours d'initiation au machine learning [Internet]. Google Developers. [cité 27 janv 2020]. Disponible sur: https://developers.google.com/machine-learning/crash-course/glossary?hl=fr
77. Initiez-vous aux autoencodeurs [Internet]. OpenClassrooms. [cité 27 janv 2020]. Disponible sur: https://openclassrooms.com/fr/courses/5801891-initiez-vous-au-deep-learning/5814621-initiez-vous-aux-autoencodeurs
78. Que signifie Deep learning (apprentissage par réseau neuronal profond)? - Definition IT de Whatis.fr [Internet]. Whatis.com/fr. [cité 27 janv 2020]. Disponible sur: https://whatis.techtarget.com/fr/definition/deep-learning-reseau-neuronal-profond
79. UCI's datasets | page 1 [Internet]. data.world. [cité 27 janv 2020]. Disponible sur: https://data.world/uci
80. Das K, Jiang J, Rao JNK. Mean squared error of empirical predictor. Ann Stat. avr 2004;32(2):818-40.

81. Brownlee J. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning [Internet]. Machine Learning Mastery. 2017 [cité 27 janv 2020]. Disponible sur: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
82. Brownlee J. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning [Internet]. Machine Learning Mastery. 2017 [cité 27 janv 2020]. Disponible sur: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
83. Cours_de_data_mining_3-Modelisation-EPF (1).pdf. http://bliaudet.free.fr/IMG/pdf/Cours_de_data_mining_3-Modelisation-EPF.pdf