CLASSIFICATION ET RÉSEAUX DE NEURONES

2.1 Introduction

La classification de données est un problème délicat qui a apparu dans de nombreux domaines tels que l'analyse d'image, le diagnostic médical, l'apprentissage automatique...etc, et cela à cause de l'explosion de la quantité de données traitée par les systèmes informatiques lors de ces dernières années.

Dans le domaine de la sécurité informatique, la classification sert à classifier le trafic réseau ou les données du système surveillé en deux classes principale (normale/attaque,légal/illégal,...etc).En effet, les méthodes de classification sont implémentées dans les systèmes de détection d'intrusions pour les aider à prendre des décisions et générer des alertes en cas d'attaques avec le moins de fausses alarmes possibles. Pour ce faire, plusieurs méthodes de l'intelligence artificielle peuvent être utilisées, telle que les réseaux de neurones.

Dans ce chapitre, nous allons présenter la classification des données vu que c'est une phase principale dans le processus de génération de notre modèle de détection d'intrusions. Nous commençons tout d'abord par sa définition, ses catégories et les méthodes de classement les plus utilisées dans chacune. Ensuite, on focalise sur les réseaux de neurones en exposant leur historique, leur définition, leurs techniques d'apprentissage et leur architecture avant de passer aux détails de perceptron multicouches, vu que c'est l'outil de notre travail, et nous terminons par les avantages et les inconvénients des réseaux de neurones et leurs domaines d'applications.

2.2 Classification

2.2.1 Définition

La classification est l'opération statistique qui consiste à regrouper des objets(individus ou variables) en un nombre limité de classes de sort qu'on doit satisfaire deux conditions principales :[21]

- Ces classes ne sont pas prédéfinies par l'analyste mais découvertes au cours de l'opération.
- Une homogénéité interne et hétérogénéité externe, c'est-à-dire les classes de la classification regroupent les objets ayant des caractéristiques similaires et séparent les objets ayant des caractéristiques différentes.

2.2.2 Architecture typique d'une application basée sur la classification

La classification est une tâche très importante dans le data mining [1], mais le développement d'un outil de classification dans n'importe quel domaine doit être réalisé en appliquant d'autre phases d'extraction et de l'analyse d'informations qui sont montrées dans la figure suivante :

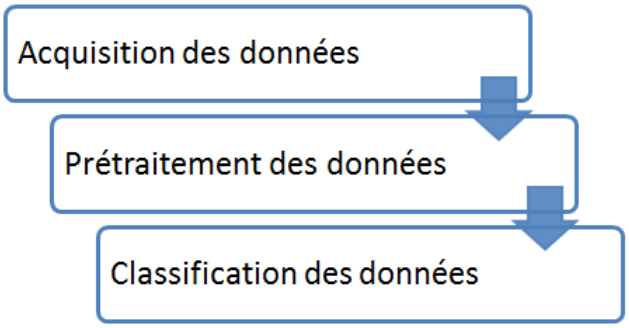


Figure 2.1 – Processus du data mining

- **Acquisition des données :** D'une manière générale, il s'agit de mettre en place l'ensemble d'instrumentation (capteurs, matériel d'acquisition, etc.) de façon à reproduire le phénomène observé le plus fidèlement possible. [22]. Dans notre cas, il s'agit de placer les sondes (IDS) pour écouter le trafic réseau.
- **Pré-traitement de données :** Cette phase correspond au filtrage des informations en ne conservant que ce qui est pertinent dans le contexte d'étude puisque ces données peuvent contenir plusieurs types d'anomalies (elles peuvent être omises à cause des erreurs de frappe ou à cause des erreurs dues au système lui-même, elles peuvent être incohérentes donc on doit les écarter ou les normaliser...etc). Parfois on est obligé à faire des transformations sur les données pour unifier leur poids.
- **Classification des données :** Dans cette étape, on doit choisir la bonne technique pour extraire les connaissances des données (les réseaux de neurones, les arbres de décision, les réseaux bayésiens...etc). Dans notre cas, la classification des connexions TCP/IP, on se base sur les réseaux de neurones.

2.2.3 Catégories de classification

Il existe un grand nombre des méthodes pour la résolution des problèmes de classification. Cependant, il est possible de les regrouper sous forme d'une hiérarchie de méthodes puisque certaines de ces approches partagent des caractéristiques communes, soit dans le type d'apprentissage utilisé (apprentissage supervisé ou non), ou bien dans la sortie réalisée (groupes disjoints ou classification floue).

La figure suivante montre une taxonomie des méthodes de classification dérivée de celle de Jain et Dubes (1988) :

1. Le **data mining** est un procédé d'exploration et d'analyse de grands volumes de données en vue d'une part de les rendre plus compréhensibles et d'autre part de découvrir des corrélations significatives

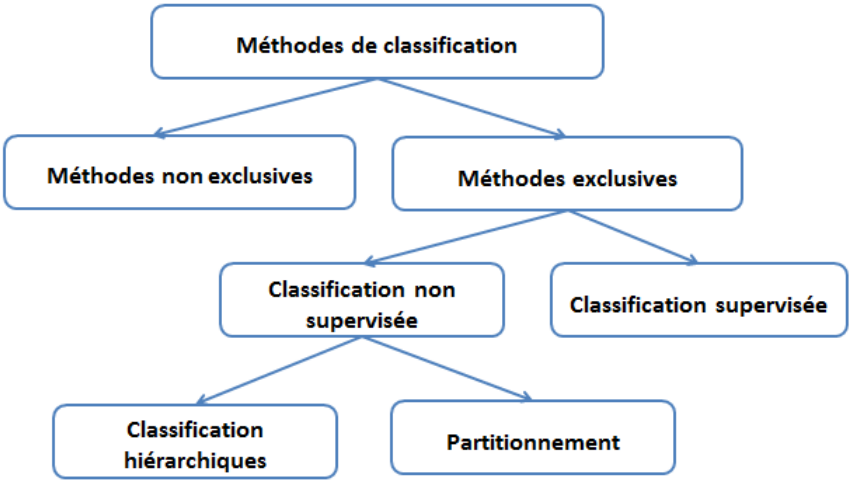


Figure 2.2 – Les méthodes de classification
[23]

a) Classification non exclusive

Une méthode est dite non exclusive si chaque objet est associé à une densité de probabilité qui indique pour chacune des classes la probabilité que l'objet considéré y appartienne (une classification floue).

b) Classification exclusive

Une méthode est dite exclusive si un objet ne peut être affecté qu'à une classe et une seule. Elle peut être divisée en deux autres classes :

I. **Classification non supervisée :** il s'agit d'extraire à partir d'une population des classes ou groupes d'individus présentant des caractéristiques communes. Le nombre et la définition des classes n'étant pas donnés a priori[24]. Le clustering regroupe un ensemble de techniques qui visent à regrouper les enregistrements d'une base de données en des groupes selon leur rapprochement les uns des autres en ne se basant sur aucune information antérieure.

il existe deux approches pour le clustering : le partitionnement et les méthodes hiérarchique.

- **Partitionnement :** consiste à construire plusieurs partitions puis les évaluer selon certains critères. Parmi les méthodes de partitionnement les plus connues on trouve **la méthode des k-moyennes(K-Means)**.
  Le principe de l'algorithme *k-means* est le suivant :
  - Choisir k objets formant ainsi k clusters.
  - (Ré)affecter chaque objet O au cluster Ci de centre Mi tel que distance(O,Mi) est minimale.
  - Recalculer Mi de chaque cluster.
  - Aller à la deuxième étape si on vient de faire une affectation.

Figure 2.3 – Exemple de partition obtenue en utilisant le K-means

- **Classification ascendante hiérarchique :** consiste à créer une décomposition hiérarchique des objets selon certains critères. Elle a pour objectif de construire une suite de partitions emboitées des données en n classes, n-1 classes,...,1 classe. Ces méthodes peuvent être vues comme la traduction algorithmique de l'adage « *qui se ressemble s'assemble* ».
  Le principe de l'algorithme CAH peut être exprimé comme suit :
  — A l'étape initiale, les n individus constituent des classes à eux seuls.
  — On calcule les distances deux à deux entre les individus, et les deux individus les plus proches sont réunis en une classe.
  — La distance entre cette nouvelle classe et les n-2 individus restants est ensuite calculée, et à nouveau les deux éléments les plus proches sont réunis.
  — Ce processus est réitère jusqu'à ce qu'il ne reste plus qu'une unique classe constituée de tous les individus.

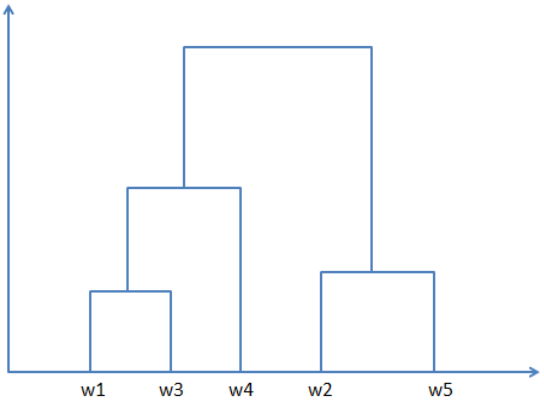


Figure 2.4 – Exemple de dendogramme (arbre hiérarchique)

II. **Classification supervisée :** elle consiste à inférer à partir d'un échantillon d'exemples classés une procédure de classification. Le problème est alors d'être capable d'associer à tout nouvel objet sa classe la plus adaptée, en se servant des exemples déjà étiquetés. [25]
Il existe plusieurs méthodes de classification supervisée, les plus connues sont : la méthode de k plus proche voisins et la méthode d'arbre de décision.

- **K plus proche voisins :** l'algorithme des k-plus proches voisins est un des algorithmes de classification les plus simples. Le seul outil dont on a besoin est une distance entre les éléments que l'on veut classifier.
  On considère que l'on dispose d'une base d'éléments dont on connait la classe, on parle de la base d'apprentissage, bien que cela soit de l'apprentissage simplifié. Dès que l'on reçoit un nouvel élément que l'on souhaite classifier, on calcule sa distance à tous les éléments de la base. Si cette base comporte 50 éléments, alors on calcule 50 distances et on obtient donc 50 nombres réels. Si k=5 par exemple, on cherche alors les 5 plus petits nombres parmi ces 50 nombres. Ces 5 nombres correspondent donc aux 5 éléments de la base qui sont les plus proches de l'élément que l'on souhaite classifier. On décide d'attribuer à l'élément à classifier la classe majoritaire parmi ces 5 éléments.
- **Arbres de décision :** les arbres de décision représentent l'une des techniques les plus connues et les plus utilisées en classification. Leur succès est notamment dû à leur aptitude à traiter des problèmes complexes de classification. En effet, ils offrent une représentation facile à comprendre et à interpréter, ainsi qu'une capacité à produire des règles logiques de classification.[26]
  Un arbre de décision est caractérisé par :
  — chaque nœud correspond à un test sur la valeur d'un ou plusieurs attributs.
  — chaque branche partant d'un nœud correspond à une ou plusieurs valeurs de ce test.
  — à chaque feuille, une valeur de l'attribut cible est associée.

  L'utilisation des arbres de décision dans les problèmes de classification se fait en deux étapes principales :
  — la construction d'un arbre de décision à partir d'une base d'apprentissage.
  — la classification ou l'inférence consistant à classer une nouvelle instance à partir de l'arbre de décision construit dans la première étape.

  Voici un exemple d'arbre de décision et la partition qu'il implique :

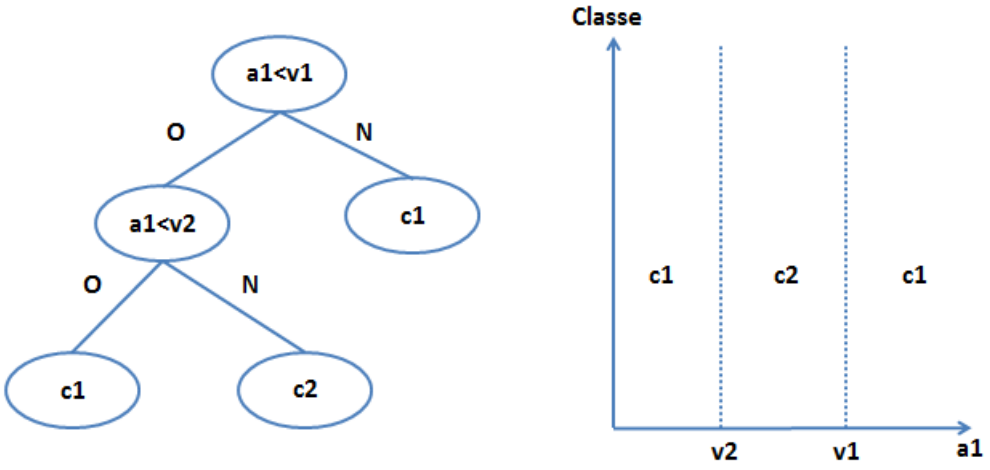


Figure 2.5 – Exemple d'une classification par arbre de décision
[27]

Il existe d'autres méthodes pour la classification supervisée, telle que la méthode de réseaux de neurones qui sert à résoudre les problèmes de classification avec les modèles non-linéaires. Cette approche sera abordée en détail dans le reste de ce chapitre.

2.3 Réseaux de neurones

Les réseaux de neurones artificiels sont inspirés de la méthode de travail du cerveau humain qui est totalement différente de celle d'un ordinateur. Le cerveau humain se base sur un système de traitement d'information parallèle et non linéaire, très compliqué, ce qui lui permet d'organiser ses composants pour traiter, d'une façon très performante et très rapide, des problèmes très compliqués tel que la reconnaissance des formes.[28]

2.3.1 Historique

Les premières tentatives de modélisation du cerveau sont anciennes et précèdent même l'informatique, voici quelques dates qui ont marqué l'histoire du domaine connexionniste :[29] [30] [31]

- En 1890, W. Jones a introduit le concept de mémoires associatives et proposa une loi de fonctionnement pour l'apprentissage dans les réseaux de neurones, connue plus tard sous le nom de «Loi de Hebb »
- En 1943, le neurologue Warren Sturgis McCulloch et le logicien Walter Pitts ont proposé un modèle simplifié de neurone biologique appelé neurone formel capable de représenter des fonctions booléennes simples.
- En 1949, D. Hebb a présenté dans son ouvrage « *The Organization of Behavior* »une règle d'apprentissage. De nombreux modèles de réseaux aujourd'hui s'inspirent encore de la règle de Hebb.
- En 1958, le premier succès est apparu quand F.Rosenblatt a présenté le premier modèle opérationnel nommé « Perceptron ».C'était le premier système artificiel qui pouvait apprendre par expérience, y compris lorsque son instructeur commettait des erreurs.
- En 1969, Les recherches sur les réseaux de neurones ont été pratiquement abandonnées lorsque M. Minsky et S. Papert ont publié leur livre « *Perceptrons* »(1969) et démontré les limites théoriques du perceptron mono-couche du point de vue performance.
- En 1982, Hopfield développe un modèle qui utilise des réseaux totalement connectés basés sur la règle de Hebb pour définir les notions d'attracteurs et de mémoire associative. En 1984 c'est la découverte des cartes de Kohonen avec un algorithme non supervisé basé sur l'auto-organisation et suivi une année plus tard par la machine de Boltzman (1985).
- En 1986, Rumelhart, Hinton et Williamsrreur ont introduit le perceptron multi-couches qui repose sur la rétro-propagation du gradient de l'erreur dans les systèmes à plusieurs couches.

A nos jours, l'utilisation des réseaux de neurones dans divers domaines ne cesse de croître. Les applications en sont multiples et variées.

2.3.2 Neurone artificiel et neurone biologique

a) Neurone biologique

Un neurone biologique comprend :[32]

— le corps cellulaire, qui fait la somme des influx qui lui parviennent. Si cette somme dépasse un certain seuil, il envoie lui-même un influx par l'intermédiaire de l'axone.

— l'axone, qui permet de transporter les influx en provenance du corps cellulaire vers d'autres cellules.

— les dendrites, qui sont les récepteurs principaux du neurone, captant les signaux qui lui parviennent.

— les synapses, qui permettent aux neurones de communiquer avec les autres via les axones et les dendrites.

La figure suivante représente un neurone biologique :

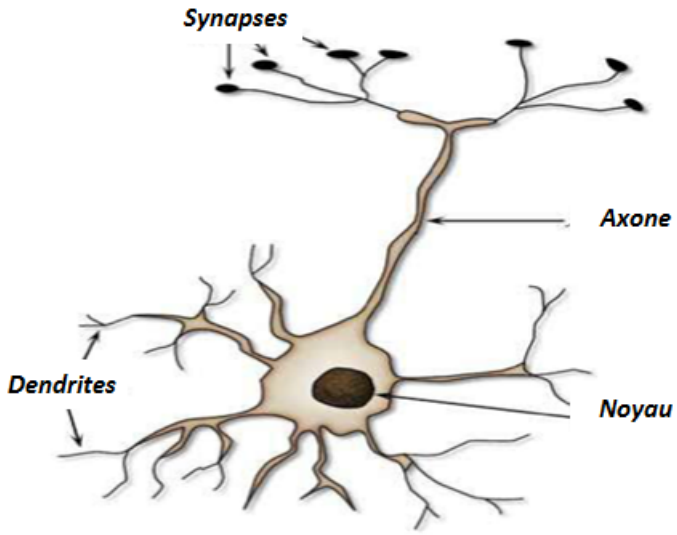


Figure 2.6 – Un neurone biologique
[33]

b) Neurone artificiel

Un neurone formel (ou simplement « 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 sa « sortie ». Un neurone est donc avant tout un opérateur mathématique, dont on peut calculer la valeur numérique par quelques lignes de programme informatique.[34]

Un réseau de neurones formel peut être représenté comme suit :

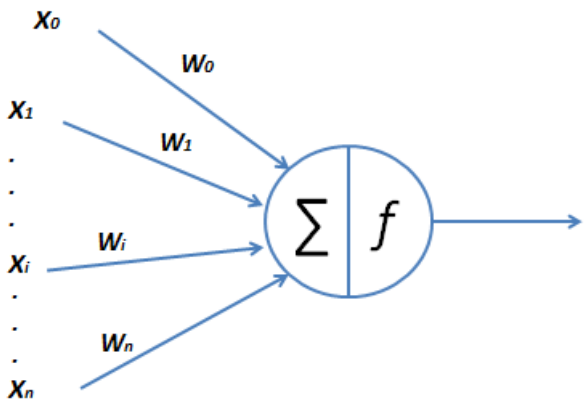


Figure 2.7 – Un neurone formel

La structure d'un neurone artificiel est en fait inspirée de la structure des neurones biologiques et la figure suivante montre la mise en correspondance entre ces deux types de neurones :

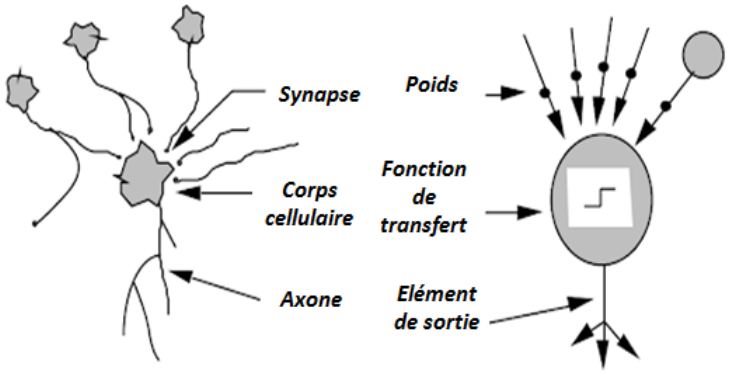


Figure 2.8 – Une correspondance entre neurone artificiel et neurone biologique [33]

2.3.3 Fonctionnement des réseaux de neurones

Un neurone est une unité de calcul élémentaire qui combine des entrées réelles $X_1, X_2, ...X_n$ en une sortie réelle S. Ces entrées proviennent soit d'autres éléments « processeurs »(neurones), soit de l'environnement et elles n'ont pas la même importance, donc à chaque entrée X_i est associé un poids (ou coefficient synaptique) W_i. En règle générale, l'activité en entrée est mesurée par la somme pondérée des entrées $\sum W_i X_i$. Puis on applique une fonction de transfert f (appelée fonction d'activation) au résultat afin d'obtenir la sortie.

Il existe de nombreuses formes possibles pour la fonction de transfert. Les plus courantes sont montrées ci-dessous :

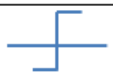
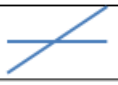
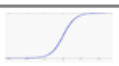
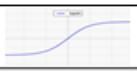
Nom de la fonction	*Relation d'entrée/sortie*	*Icône*
Seuil	a=0 si n<0 a=1 si n≥0	
Seuil symétrique	a=-1 si n<0 a=1 si n≥0	
Linéaire	a=n	
Linéaire saturée	a=0 si n<0 a=n si 0≤n≤1 a=1 si n>1	
Linéaire saturée symétrique	a=-1 si n<-1 a=n si -1≤n≤1 a=1 si n>1	
Linéaire positive	a=0 si n<0 a=n si n≥0	
Sigmoïde	$a=\frac{1}{1+e^{-n}}$	
Tangente hyperbolique	$a=\frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$	
compétitive	a=0 si n maximum a=1 autrement	C

Tableau 2.1 – Les fonctions de transfert les plus utilisées
[35]

2.3.4 Apprentissage des réseaux de neurones

Les réseaux de neurones ont la capacité d'apprendre à partir d'exemples de manière assez analogue à celle des humains basée sur leur expérience, cet apprentissage peut être de différents modes :[36]

- **L'apprentissage non supervisé :** le réseau doit détecter des points communs aux exemples présentés, par la modification des poids, afin de fournir la même sortie pour des entrées aux caractéristiques proches.
  L'apprentissage non supervisé est bien adapté à la modélisation des données complexes (images, sons, etc.), généralement des données symboliques, où l'on possède des règles moins précises qui gouvernent le comportement du système à modéliser par les réseaux de neurones.
- **L'apprentissage supervisé :** dans le cas de l'apprentissage supervisé, l'adaptation intervient directement lorsque le système compare la réponse qu'il a calculé avec la réponse attendue en fonction des entrées fournies. La performance de l'apprentissage est déterminée par l'intermédiaire d'un critère à optimiser.
- **Le mode hybride :** ce mode prend en fait les deux autres approches, puisqu'une partie des poids doit être déterminée par apprentissage supervisé et l'autre partie par apprentissage non-supervisé.

2.3.5 Architecture des réseaux de neurones

On distingue deux grands types d'architectures de réseaux de neurones : les réseaux de neurones non bouclés et les réseaux de neurones bouclés.

- **Les réseaux de neurones bouclés :**
  Dans ce type de réseau, les neurones sont connectés dans n'importe quel sens d'une façon cyclique, c'est-à-dire, lorsqu'on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ. Ce type de réseau peut être schématisé comme suit :

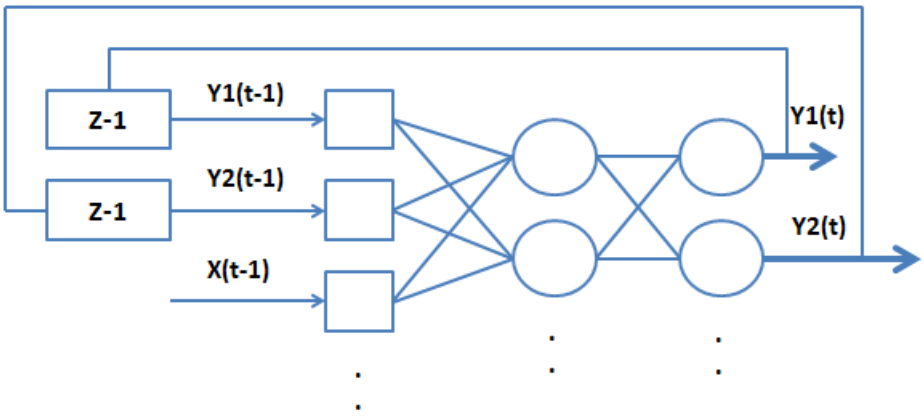


Figure 2.9 – Un réseau de neurone bouclé

- **Les réseaux de neurones non-bouclés :**
  Dans ce type de réseau, les connexions entre les neurones des différentes couches partent dans un seul sens. On a d'abord une couche d'entrée avec seulement des entrées du réseau, ensuite les couches cachées contenant des neurones cachés et en dernier la couche de sortie englobant les neurones visibles. Un tel réseau peut être représenté comme suit :

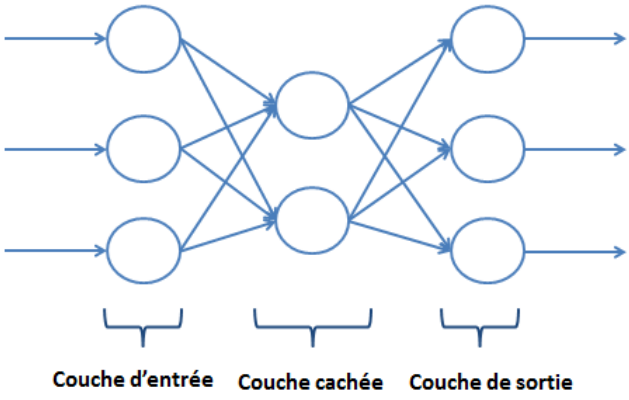


Figure 2.10 – Un réseau de neurone non-bouclé

2.3.6 Quelques modèles de réseaux de neurones

À nos jours, où les chercheurs n'ont de cesse que d'inventer de nouveaux modèles de réseaux de neurones plus adaptés à la résolution des problèmes particuliers, on trouve plusieurs modèles qui sont disponibles et largement utilisés, parmi eux on peut citer les suivants : [37]

- **Le perceptron simple :**
  C'est le modèle le plus simple qui comporte uniquement deux couches, une pour les entrées constituée de n neurones élémentaires dont la fonction d'activation est linéaire est une couche de sortie constituée d'un ou plusieurs neurones.

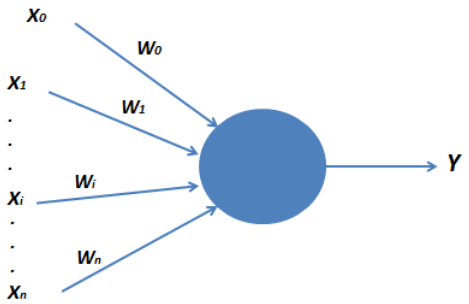


Figure 2.11 – Un perceptron simple(mono-couche)

- **Le perceptron multi-couches :**
  C'est le type le plus connu des réseaux de neurones. C'est une extension du perceptron mono-couche avec la présence d'une couche intermédiaire constituée d'une ou plusieurs couches cachées. Les connexions n'existent qu'entre les cellules d'une couche avec les cellules de la couche suivante. Et vu que ce modèle de réseaux de neurones est celui que nous allons utiliser pour mettre en œuvre notre modèle de détection d'intrusions, il sera détaillé dans la suite de ce chapitre.

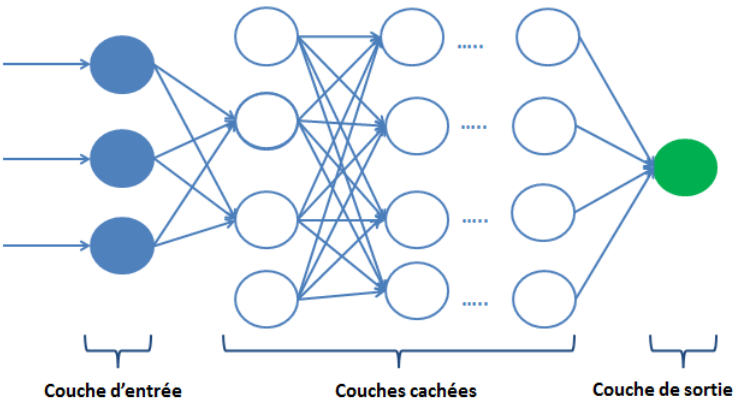


Figure 2.12 – Un perceptron multi-couches

- **Modèle de Kohonen :**
  Les cartes topologiques ou cartes auto organisatrices ont été introduites pour la première fois par T. Kohonen en 1981. Elles sont réalisées à partir d'un réseau à deux couches(une couche d'entrée et une couche de sortie).Notons que les neurones de la couche d'entrée sont entièrement connectés à la couche de sortie.Les neurones de la couche de sortie sont composés d'un certain nombre de neurones régulièrement répartis sur une carte.

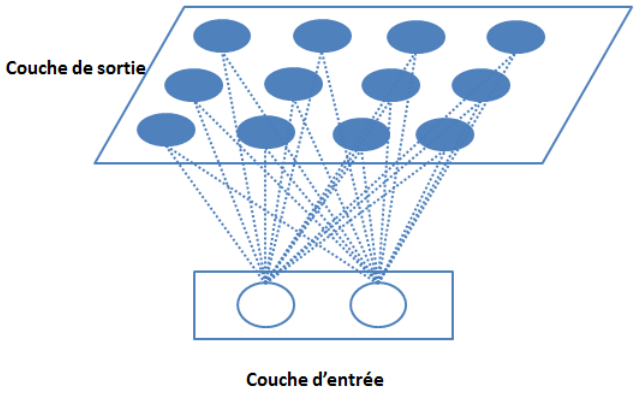


Figure 2.13 – Le modèle de Kohonen

- **Modèle de Hopfield :**
  Le modèle de Hopfield fut présenté en 1982. Ce modèle très simple est basé sur le principe des mémoires associatives. C'est d'ailleurs la raison pour laquelle ce type de réseau est dit associatif (par analogie avec le pointeur qui permet de récupérer le contenu d'une case mémoire). Le modèle de Hopfield utilise l'architecture des réseaux complètement connectés et récurrents. Les sorties sont en fonction des entrées et du dernier état pris par le réseau.

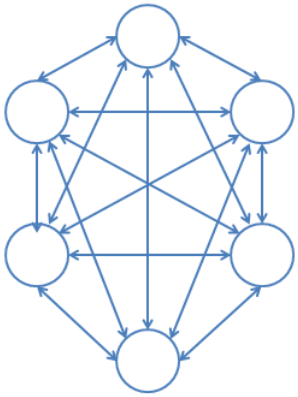


Figure 2.14 – Le modèle de Hopfield

2.3.7 Perceptron multi-couche

Le perceptron multicouche (MLP [2]) est un des réseaux de neurones les plus utilisés pour des problèmes d'approximation, de classification et de prédiction. Ce type de réseau est dans la famille générale des réseaux à propagation vers l'avant, c'est-à-dire qu'en mode normal d'utilisation, l'information se propage dans un sens unique, des entrées vers les sorties sans aucune rétroaction. Son apprentissage est de type supervisé par correction des erreurs. Dans ce cas uniquement, le signal d'erreur est rétro-propagé vers les entrées pour mettre à jour les poids des neurones. Il est habituellement constitué de deux ou trois couches de neurones totalement connectés.

a) Mise en oeuvre du réseau de neurones MLP

La mise en œuvre des réseaux de neurones comporte à la fois une partie conception, qui permet de choisir la meilleure architecture possible, et une partie de calcul numérique, pour réaliser l'apprentissage d'un réseau de neurones. En effet, un MLP peut avoir un nombre de couches et un nombre de neurones par couche quelconques, mais en vue de perfectionner le fonctionnement du MLP d'un part et réduire au maximum le temps de calcul d'autre part, on doit chercher une architecture optimale au point de vue nombre de couches , nombre de neurones par couche et nombre de sorties possibles. À partir d'une architecture de réseau de neurones donnée et des exemples disponibles (la base d'apprentissage), on détermine les poids optimaux, par l'algorithme de la rétro-propagation des erreurs, pour que la sortie du modèle s'approche le plus possible du fonctionnement désiré.[36]

b) Apprentissage des réseau MLP

L'apprentissage d'un réseau de neurones multicouches se fait généralement par l'algorithme de la rétro-propagation « back-propagation »qui est l'exemple d'apprentissage supervisé le plus utilisé. Cette technique de rétro-propagation du gradient permet de calculer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première. Le principe de cet algorithme peut être décrit en trois étapes principales : l'acheminement de l'information à travers le réseau, la rétro-propagation des sensibilités et calcul du gradient et enfin l'ajustement des paramètres par la règle du gradient approximé.[38]

- **Propagation avant :** dans cette étape, on calcule la valeur d'entrée pour chaque neurone *j* de la couche cachée ou bien de la couche de sortie. Cette valeur est égale à la somme pondérée des valeurs de sortie des neurones de la couche antérieure. Elle est calculée à partir de la formule suivante [39] :

$$\mathrm{VE_j} = \sum_{i \in pred(j)} W_{ij} VA_i$$

On calcule ensuite la valeur d'activation de chaque neurone *j* en utilisant une fonction de transfert f :

$$VA_\mathrm{j} = f(VE_\mathrm{j})$$

Il existe plusieurs fonctions d'activation, la plus utilisée dans le cas des réseaux de neurones multi-couches est la fonction sigmoïde, qui est utilisée dans ce travail. Donc la valeur d'activation est calculée comme suit :

2. Multi Layer Perceptron

$$VA_{\mathrm{j}} = f(VE_{\mathrm{j}}) = \frac{1}{1 + e^{-VE_{\mathrm{j}}}}$$

On répète cette opération jusqu'à le calcul des valeurs d'activation des neurones de la couche de sortie. La différence entre ces valeurs et les valeurs désirées représente l'erreur d'apprentissage appelée delta (Δ).

- **Rétro-propagation :** après le calcul de l'erreur d'apprentissage dans la phase de propagation avant, ce dernier va être distribué à travers les couches du réseau en allant des sorties vers les entrées (vers l'arrière) afin de pouvoir ajuster dans la phase suivante les poids du réseau.

  On calcule l'erreur (Δ) de chaque neurone s de la couche de sortie par la formule suivante [39] :

$$\Delta_{\mathrm{s}} = VA_{\mathrm{s}}(1 - VA_{\mathrm{s}})(\textit{val désirée de S} - VA_{\mathrm{s}})$$

  On calcule ensuite l'erreur de chaque neurone caché *j* en utilisant la formule suivante [39] :

$$\Delta_{\mathrm{j}} = VA_{\mathrm{j}}(1 - VA_{\mathrm{j}}) \sum_{k \in succ(j)} w_{jk}\Delta_k$$

- **Mise à jours des poids :** lorsque on distribue l'erreur d'apprentissage sur tous les neurones de la couche cachée, on passe à la mise à jours des poids et des biais à l'aide des formules suivantes [39] :

$$w_{ij} = w_{ij} + \alpha * VA_i * \Delta_j$$
$$b_j = b_j + \alpha * \Delta_j$$

  Où :α est le taux d'apprentissage choisi par l'utilisateur(entre 0 et 1)

On répète ce processus (propagation avant,rétro-propagation et mise à jours des poids) autant de fois que le nombre d'exemples de la base d'apprentissage, lorsque on termine on calcule la somme des erreurs au carré (SEC) tel que [39] :

$$SEC = \frac{1}{n} \sum_{i=1}^{n} (y_k^{des} - y_k)^2$$

Où : y_k^{des} est la valeur désirée dans l'exemple d'apprentissage
y_k est la valeur de sortie calculée par le MLP

Le problème consiste donc à minimiser la valeur de SEC en fonction de l'ensemble des valeurs de pondération des nœuds et des liaisons. Si la valeur du SEC n'est pas optimal, on répète la procédure avec d'autres valeurs initiales des poids (w_{ij}) et d'autres valeurs des biais (b_j).

La phase d'apprentissage d'un réseau de neurone peut donc être résumée par l'algorithme suivant :

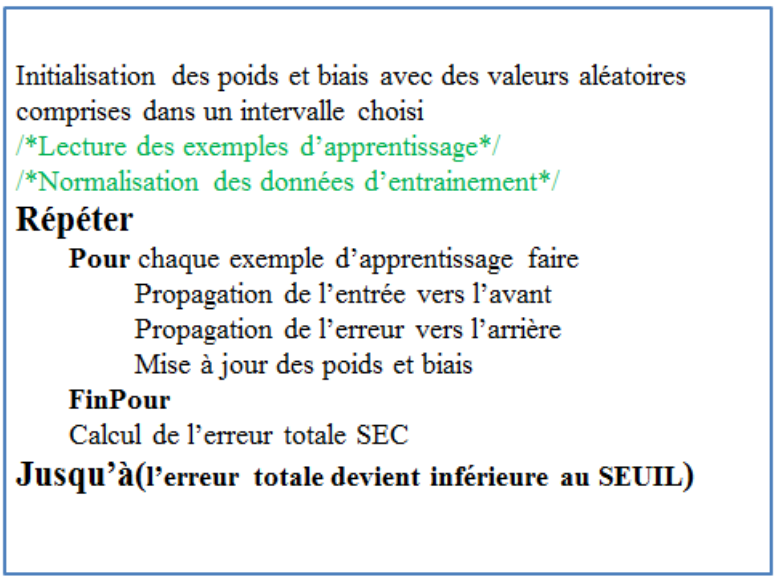


Figure 2.15 – L'algorithme de rétro-propagation de gradient

c) Validation :

La phase de validation vient après la phase d'apprentissage, elle utilise la matrice des poids optimaux et les vecteurs d'entrées qui correspondent aux exemples de la base de test. La validation dans un réseau de neurone correspond à l'opération de propagation d'états.

2.3.8 Avantages et limites des réseaux de neurones

a) Avantages

Les réseaux de neurones artificiels offrent plusieurs avantages, parmi eux on peut citer :[40] [41]

- **Réutilisabilité :** Un réseau de neurones n'est pas programmé pour une application mais pour une classe de problèmes : après une phase d'apprentissage adéquate, il peut traiter de nombreuses tâches.
- **Robustesse :** Les couches cachées du réseau de neurone forment une représentation abstraite des données (concepts), qui permettent de savoir catégoriser des données non traitées lors de l'apprentissage (non prévues).
- **Temps de réponse :** C'est l'un des avantages principaux du réseau de neurones : en effet une fois que le réseau a appris, il peut sortir quasi-instantanément la réponse. En fait, les opérations que fait un réseau de neurones sont très simples du point de vue informatique et peu gourmandes en CPU.
- **Parallélisme :** l'architecture d'un réseau de neurone permet le traitement parallèle et rapide des informations et aide à l'implantation des applications ayant notamment des contraintes temps-réel.

b) Limites

Bien que les réseaux de neurones soient capables d'effectuer beaucoup de tâches, ils souffrent néanmoins de certaines limites dont on peut citer : [40] [41]

- **Choix des attributs :** Avant de passer des exemples à un réseau de neurones, il faut trouver une structure permettant au réseau de bien apprendre les exemples(choix des valeurs initiales des poids du réseau, le nombre de neurones cachés nécessaires,réglage du pas d'apprentissage...etc).
- **Temps d'apprentissage :** Un réseau doit parfois apprendre les exemples plusieurs dizaines de milliers de fois. Si la base d'exemples est énorme, le temps d'apprentissage risque d'être démesuré et le réseau perd son pouvoir de généralisation(il reconnait les données de l'échantillon d'apprentissage, mais plus de nouvelles données similaires).
- **Exploitabilité :** Il existe une grande difficulté pour expliquer les résultats obtenus par le réseau de neurones, car ce dernier fonctionne comme une boite noire (on lui passe des entrées et il ressort un résultat) où les connaissances sont intelligibles pour l'utilisateur.
- **Sur apprentissage :** Un réseau de neurones peut donner une très grande précision face aux exemples d'entrainement, mais se comporte très mal avec les nouveaux exemples. cela représente un phénomène très connu en apprentissage qui est le sur-apprentissage ou l'apprentissage par cœur. Le sur apprentissage donne, généralement, des modèles à faible capacité de généralisation, et par conséquent la mesure de précision n'est pas suffisante pour qualifier les performances de ces modèles.

2.3.9 Domaines d'applications des réseaux de neurones

Depuis leur apparition, les réseaux de neurones ont été largement utilisés dans plusieurs domaines. On peut citer :[42]

- **Traitement d'image :** compression d'images, reconnaissance automatique de caractères et de signatures, reconnaissance de formes dans la lecture automatique des codes postaux, classification...etc.
- **Traitement du signal :** traitement de la parole, l'élimination du bruit et de l'écho, classification...etc.
- **La fouille de données :** extraction des connaissances et les problèmes d'optimisation et de classification.
- **La simulation :** simulation boîte noire, prévisions météorologiques et les estimations de probabilité de succès.
- **L'automatique :** l' identification/modélisation des systèmes non linéaires, la modélisation des procédés industriels et la Robotique.

Il existe d'autres domaines où on utilise différents types de réseaux de neurones selon le problème à résoudre. Voici un tableau qui représente la correspondance entre chaque domaine d'application et le type de réseau de neurones le plus approprié :

Domaine d'application	*Type de RNA*
Reconnaissance de formes	Mlp, Hopefied, Kohonen, PNN
Mémoires associatives	Hopefied, MLP récurrent, Kohonen
Optimisation	Hopefied, ART, CNN
Approximation de fonctions	MLP, RBF
Modélisation et control	MLP, MLP récurrent, FLN
Traitement d'images	CNN, Hopefied
Classification et clustering	MLP, Kohonen, RBF, ART, PNN

Tableau 2.2 – Correspondance type de RNA / Domaine d'application
[43]

2.4 Conclusion

Les réseaux de neurones artificiels sont considérés comme des approches très intéressantes dans le domaine de l'intelligence artificielle. Ils sont connus par leur puissance d'apprentissage et généralisation.

Dans ce deuxième chapitre, nous avons présenté dans la première partie une introduction au domaine de classification des données en détaillant l'architecture des applications basées sur cette notion ainsi que leur catégories avec les techniques utilisées dans chacune.

Dans la deuxième partie, nous avons introduit les définitions de base pour les réseaux de neurones vu que c'est la technique que nous allons utiliser pour réaliser ce travail, et cela en spécifiant leur principe de fonctionnement, leurs types d'apprentissage et leur architecture avec une brève présentation de quelques modèles de ces derniers, où nous avons concentré sur les réseaux de neurones multi-couches(MLP) qui sont les plus adaptés à la classification du trafic réseau, et enfin nous avons terminé par les avantages et les inconvénients de cette technique de classification supervisée ainsi que ses domaines d'application.

CHAPITRE 3

OPTIMISATION COMBINATOIRE

3.1 Introduction

Les problèmes d'optimisation occupent actuellement une place importante dans la communauté scientifique où il est nécessaire de trouver des solutions optimales pour des problèmes très compliqués et difficiles à résoudre. Parmi les méthodes les plus utilisées pour la résolution de tels problèmes, on trouve les métaheuristiques qui englobent un ensemble d'algorithmes capables de résoudre des problèmes assez compliqués en s'inspirant d'analogies avec la physique (recuit simulé), avec la biologie (algorithmes évolutionnaires)ou encore l'éthologie (colonies de fourmis, essaims particulaires).

Dans ce chapitre, nous avons présenter le concept d'optimisation combinatoire et ses différentes techniques, où nous avons concentrer sur les métaheuristiques vu que c'est la méthode d'optimisation que nous allons utiliser pour générer notre modèle de détection d'intrusions. Plus particulièrement, la méthode de recuit simulé et celle de recherche tabou qui sont utilisées dans ce travail conjointement avec le réseau de neurones multicouches pour optimiser les poids de ce dernier afin de perfectionner notre modèle de détection d'intrusions.

3.2 Optimisation Combinatoire

3.2.1 Définition

L'optimisation combinatoire recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction avec ou sans contraintes. En théorie, un problème d'optimisation combinatoire est défini par un ensemble d'instances. A chaque instance du problème est associé un ensemble discret de solutions S, un sous-ensemble X de S représentant les solutions admissibles (réalisables) et une fonction de coût f (ou fonction objectif) qui assigne à chaque solution $s \in$X le nombre réel (ou entier) $f(s)$. Résoudre un tel problème (plus précisément une telle instance du problème) consiste à trouver une solution $s^* \in$ X optimisant la valeur de la fonction de coût f. Une telle solution s^* s'appelle une solution optimale ou un optimum global.[44]

3.2.2 Classification des méthodes d'optimisation combinatoire

La résolution d'un problème d'optimisation combinatoire est réalisée à l'aide des méthodes illustrées dans la figure suivante :

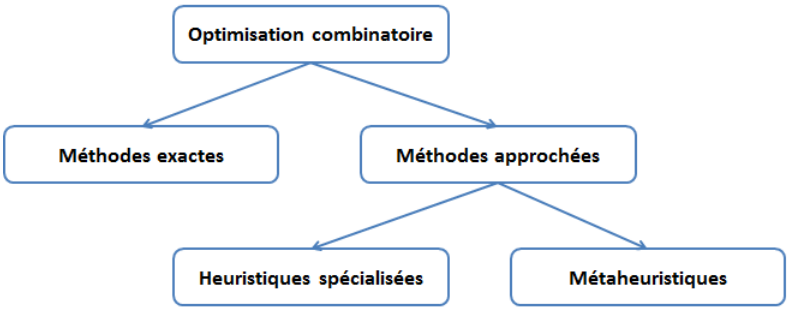


Figure 3.1 – Classification des méthodes d'optimisation
[45]

a) Méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles (développées depuis une trentaine d'années) telles les techniques de séparation et évaluation (Branch and Bound) ou la programmation dynamique. Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable.[21]

b) Méthodes approchées

Lorsque l'on dispose d'un temps de calcul limité ou lorsqu'on est confronté à des problèmes difficiles ou de taille importante, on peut avoir recours aux méthodes approchées, en se contentant de rechercher une solution de bonne qualité. Dans ce cas le choix est parfois possible entre une heuristique spécialisée et une métaheuristique :[46]

- **Heuristique :** Le mot *heuristique* vient du grec « eurisko »qui signifie « je trouve »d'où la célèbre Eureka d'Archimède. Une heuristique, ou méthode approximative, est un algorithme qui fournit rapidement(en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation difficile. Une méthode heuristique est généralement conçue pour un problème particulier, en s'appuyant sur sa structure propre.
- **Métaheuristique :** Le mot *métaheuristique* est dérivé de la composition de deux mots grecs : méta signifiant « au-delà »et heuristique. En effet, ces algorithmes se veulent des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changement profond dans l'algorithme employé.

3.3 Métaheuristiques

3.3.1 Définition

Contrairement aux méthodes exactes, les métaheuristiques permettent de s'attaquer aux instances problématiques de grande taille en fournissant des solutions satisfaisantes dans un délai raisonnable. Il n'y a aucune garantie de trouver des solutions globales optimales ou même des solutions limitées. Les métaheuristiques ont reçu de plus en plus de popularité au cours des 20 dernières années. Leur utilisation dans de nombreuses applications montre leur efficacité pour résoudre des problèmes vastes et complexes. Parmi ces domaines, on peut citer les suivants : [45]

- Conception technique, optimisation de la topologie et optimisation structurelle en électronique et VLSI, aérodynamique, dynamique des fluides, télécommunications, automobile, et la robotique.
- Apprentissage automatique et fouille de données en bioinformatique et biologie computationnelle, et les finances.
- Simulation et identification en chimie, physique et biologie, traitement du signal et de l'image...etc.
- Planification des problèmes de routage, planification des robots, problèmes d'ordonnancement et de production, logistique et transport, gestion de la chaîne d'approvisionnement...etc.

3.3.2 Concepts fondamentaux des métaheuristiques

Les métaheuristiques ne nécessitent pas une connaissance particulière sur les problèmes d'optimisation à résoudre. Il suffit d'associer une ou plusieurs variables à une ou plusieurs solutions (optimum).

Il existe deux points critiques pour toute métaheuristique :[47]

- **Diversification :** Le principe de diversification d'une méthode d'optimisation donnée correspond à sa capacité de parcourir aisément l'espace de recherche pour obtenir des solutions très différentes les unes des autres. Autrement dit, c'est une mécanisme pour une exploration assez large de l'espace de recherche.

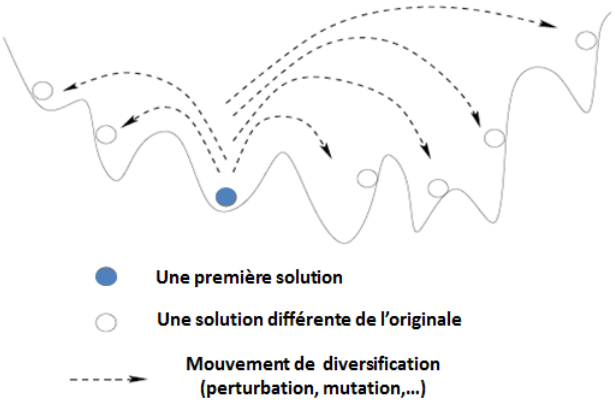


Figure 3.2 – Processus de diversification d'une solution

- **Intensification :** Autant le principe de diversification essaye de déplacer les solutions dans d'autres zones de l'espace de recherche, autant le processus d'intensification vise à forcer une solution donnée à tendre vers l'optimum local de la zone à laquelle elle est attachée. En effet, Elle permet une exploitation de l'information accumulée durant la recherche.

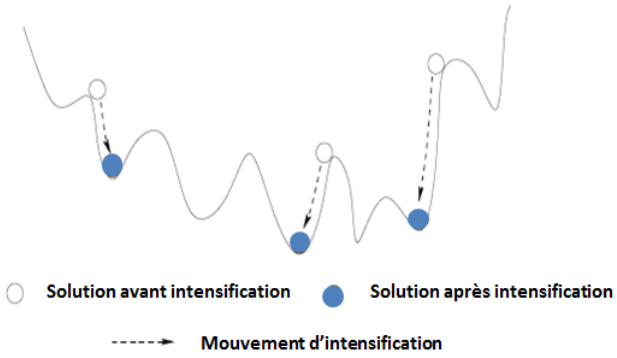


Figure 3.3 – Processus d'intensification de plusieurs solutions

3.3.3 Classification des métaheuristiques

Une manière de classifier les métaheuristiques est de distinguer celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois(méthodes de trajectoire).

Voici une figure qui montre cette classification avec des exemples typiques pour chaque catégorie :

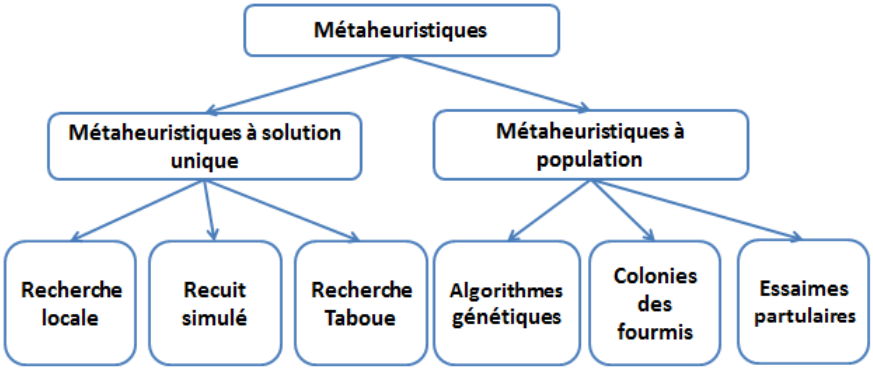


Figure 3.4 – Classification des métaheuristiques

a) Métaheuristiques à solution unique

Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage[48].