REPUBLIQUEALGERIENNEDEMOCRATIQUEETPOPULAIRE

Ministère de l'ENSIGNEMENET Supérieur et de la recherche scientifique

Université 8 Mai 1945-GUELMA

Faculté des Sciences et de la Technologie

Département de Génie Electrotechnique et Automatique

Réf : ……/2022

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

Domaine Académique : Sciences et Technologie

Filière: Automatique

Spécialité : Automatique et Informatique Industrielle

Par : Bouachria Salah Eddin et Belkharchiche Foued

THÈME

COMPARAISON ENTRE LES DEUX MÉTHODES DE CLASSIFICATION LE PERCEPTRON MULTICOUCHES ET L'AUTO ENCODEUR

Soutenu publiquement le :19 /06/2022 devant le jury composé de :

BOUDJEHEME BADRE EDDINE	Professeur	Univ.Guelma	Président
BOUCEREJE	MCA	Univ.Guelma	Examinateur
DEBECHE MEHDI	MAA	Univ.Guelma	Encadreur

Résume :

L'objectif générale de ce mémoire est d'explorer les différentes techniques de classification et les algorithmes fondamentaux largement utilisés par celle-ci, en se concentrons beaucoup plus sur les techniques à base de réseaux de neurones artificiels. Notre approche pratique sera l'implémentation de deux modèles de calssifieur à bas de réseaux de neurone ,un MLP en premier lieu et un autoencodeur en second lieu pour la classification de la célèbre base de données IRIS afin de comparer leur performances en terme de précision globale

Mots clés :

Intelligence artificielle, apprentissage automatique, classification, supervisée, non supervisée réseaux de neurones artifiiels, MLP auto encodeur, base de donnée IRIS

Remerciement :

Au début, nous remercions Allah qui nous a aidés à accomplir ce travail, et qui a été avec nous à tous les moments de notre chemin d'étude. A nos chers parents, ,. Nos sincères remerciements à tous les membres du jury qui nous ont fait l'honneur d'examiner ce travail. Un remerciement spécial à notre encadreur Mr.Debeche Mehdi pour sa patience, et ses conseils judicieux pour élaborer ce travail

Un remerciement à notre professeur Moussawi Abdelkarim, docteur Ghadjeti Mohammed, L'équipe du laboratoire de génies électrique , et génie mécanique , sans oublier nos frères Fnides Hocine,et Slimani Mouloud.

TABLES DES MATIÈRES

Chapitre 1 : la classification

Chapitre 2 : Méthodes neuronales :

Chapitre 3 : Classification profonde

Chapitre 4 : Application

LES FIGURES

Chapitre 1 : la classification

Chapitre 2 : Méthodes neuronales :

Chapitre 3 : Classification profonde

Chapitre 4 : Application

LES TABLEAUX

CHAPITER 1:

Classification

1- INTRODUCTION :

La classification est une méthode très importante en analyse de données et peut être appliquée dans plusieurs domaines, elle inclut des méthodes classiques dont le but est de former des classes selon des critères de distance ou de similarité, cependant, elles sont limitées compte tenu de leur temps d'exécution et de la capacité de fonctionnement des machines. Ainsi, avec l'arrivée de l'intelligence artificielle, différentes méthodes de classification ont été développées et parviennent à contourner certaines limitations des méthodes classiques.

En 1943, McCulloch et Pits ont conçu le premier modèle de réseau de neurones inspiré du fonctionnement des cellules nerveuses biologiques.

C'est dans ce cadre qu'est né le perceptron multicouche pour traiter des situations plus complexes en termes de taille de données.

Par la suite, d'autres algorithmes ont été proposés suivant la nature des donnés. On note par exemple les réseaux de neurones à convolution pour le traitement de donnés d'images. Des travaux de recherches récents ont proposé un cadre général de classification profonde (DeepCluster) pour intégrer certaines méthodes de classification classiques, notamment la méthode K-means et les modèles de mélange gaussien dans les modèles d'apprentissage profond.

Plusieurs études et algorithmes ont été menés dans ce cadre parmi elles les auto encoders qui sont des méthodes neuronales apprenant à reconstruire les entrées originales en sortie le plus fidèlement possible.

Selon la nature des données et la méthode de classification intégrée, l'auto encoder est généralement utilisé comme branche principale pour l'extraction des caractéristiques avant la formation des classes.

D'autres modèles utilisent le perceptron multicouche pour la tache de classification.[1]

2 Définition :

La classification dans son énoncé le plus simple consiste à assembler des choses Similaires, et si elle est décrite plus complètement, c'est l'arrangement des choses Selon la similitude ou la différence, en d'autres termes Il s'agit de trier ou de

regrouper des choses.

La classification est incluse dans tous les domaines, l'apprentissage automatique on Est un il repose souvent sur la catégorisation des choses. Dans ce cas, la machine est Comme un enfant qui apprend à trier des jouets : voici la poupée, voici la voiture, voici Le camion, etc. Mais le problème qui se pose alors est-ce que l'enfant connaît la vraie Signification de la poupée ou de la voiture, Pouvoir distinguer n'importe quelle Poupée, peu importe sa forme, sa couleur et son mode de fabrication, En d'autres Termes, pourra-t-il généraliser ce qu'il a appris ?

En classification, nous aurons toujours besoin d'un professeur. Les données doivent Être classées avec des caractéristiques afin que la machine puisse attribuer des classes appropriées en fonction de ces caractéristiques.

En fait, nous pouvons catégoriser à Peu près tout, des utilisateurs en fonction de leurs intérêts (comme le fait l'algorithme de Facebook), des articles en fonction de la langue ou du sujet (ce qui est important pour les moteurs de recherche), de la musique en fonction du style, que ce soit du jazz, du hip-hop, etc. (ce que nous voyons dans les listes de lecture Spotify), et même catégoriser vos e-mails. C'est le type d'apprentissage automatique le plus utilisé .

L'objectif du processus d'apprentissage est de produire un modèle capable de classer toute nouvelle entrée en fonction des connaissances antérieures[2]

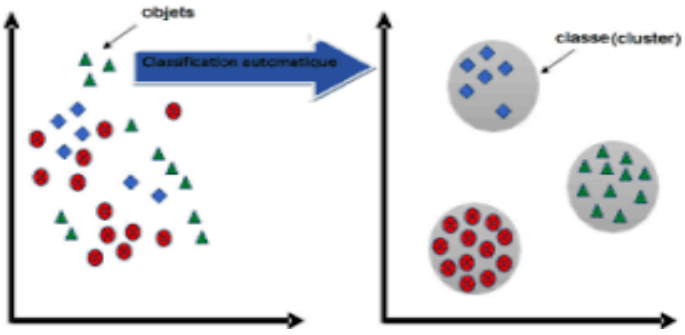


Figure 01 : principe de la classification

3 DEFINITION FORMELLE

Soit $x \in R^d$ un ensemble représentant un espace à d dimensions, appelé *Espace des instances.* La donnée $x \in X$ est appelée une instance et représente un point dans l'espace X. L'instance x est présentée sous forme d'un vecteur de taille d, $x = (x(1), \ldots, x(d))$ où chaque composante $x(i) \in R$ est une valeur discrète ou continue. Soit Y un ensemble fini de classes où chaque classe $y \in Y$ est présentée sous forme d'une valeur discrète appelée étiquette de classe (un nom ou identifiant unique pour la classe).

Le classifieur se présente alors sous forme d'une fonction de classification h (appelée aussi modèle de classification) permettant d'associer une donnée $x \in X$ à une étiquette de classe $y \in Y$, $h : X \to y = h(x)$

Notez que pour le problème de classification, l'espace des réponses $Y \in N$ est discret et fini, vu que chaque $y \in Y$ représente une classe. Lorsque Y est continu (c.à.d. $Y \in R$), on parle alors de problème de régression [3]

4 - Problème de la classification

Le problème de classification est également appelé problème de régression logistique (régression logistique).

En fait, on parle de régression mathématiquement, étant donné un ensemble de points qui peuvent être ajustés avec une courbe. Si la courbe est une droite, on parle de régression linéaire, et si la courbe est une courbe quadratique, alors on l'appelle régressions non linéaire ca de la régression logistique .La régression linéaire est généralement utilisé pour prédire les résultats de valeurs continues, exemple prévision des prix des maisons en fonction de la taille de la maison ,donc étant donné la taille d'une maison, il y a toujours un prix de maison correspondant.

Dans la vraie vie, il existe de nombreux problèmes qui ne sont pas conformes au modèle de régression linéaire, ce qui signifie que leurs valeurs ne sont pas continuées, comme les problèmes de classification binaire où il n'y a que deux résultats : soit 1(vrais/oui/oui/peut) ou 0(Faux/non/non/ne peut pas). Les problèmes de classification les plus courants sont les suivants : Outlook détermine si un e-mail nouvellement reçu est un spam ou non , et en

médecine jugé sur la base de certains phénomènes si un patient a une maladie , en entrant dans l'école, décidez si de l'admission ou pas à l'école et ainsi de suite. Il n'est clairement pas très raisonnable d'utiliser l'algorithme de régression linéaire pour résoudre le problème de classification.

Parmi les problèmes de classification on cite:

- Problème de double classification
- Problème de classification multiple

4.1 Méthode d'évaluation d'un modèle d'apprentissage automatique (problème de classification)

1. Taux d'erreur et précision
2. Précision et taux de récupération
3. Courbe P-R.
4. Courbe ROC et valeur AUC
5. Courbe KS et valeur KS

4.2 Résolvez les problèmes de classification en trois étapes :

1. Sélectionnez un modèle
2. Pour déterminer la fonction de perte, l'une des idées les plus simples consiste à
3. calculer le nombre d'erreurs de classification
4. Trouvez la fonction optimale qui minimise la fonction de perte.

4.3 EXEMPLES DE PROBLEMES DE CLASSIFICATION

Prenons par exemple le jugement de savoir si un patient a une tumeur maligne. Dans cet exemple, la seule valeur distinctive qui détermine si une tumeur est maligne ou bénigne est la taille de la tumeur. Supposons que nous ayons une série d'ensembles de données, la taille de la tumeur pour de nombreux patients et si la tumeur est bénigne ou maligne. Nous prenons 1 comme malin et 0 comme bénin, et mettons la taille de la tumeur dans l'ensemble de données aux coordonnées du système de coordonnées, et utilisons si la tumeur dans l'ensemble de données est maligne ou bénigne comme ordre, et les résultats obtenus sont affichés dans la figure suivante :

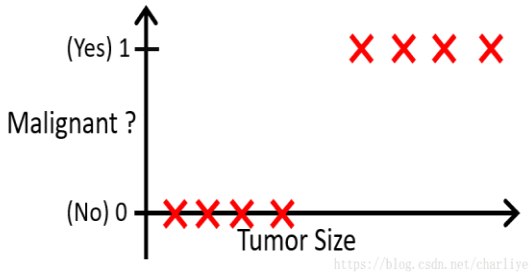


Figure 02 : taille de tumeur(maligne ou bénigne)

Nous avons pu voir que les quatre croix rouges à l'extrême gauche indiquent que les tumeurs de ces patients sont bénignes, et les quatre croix rouges à droite indiquent que les tumeurs de ces patients sont malignes. Si nous utilisons toujours l'algorithme de régression linéaire pour ce faire, nous pouvons utiliser une ligne droite pour ajuster les huit points du graphique. Selon l'algorithme de régression linéaire mentionné précédemment, nous pouvons utiliser la méthode des moindres carrés pour faire disparaître cette ligne droite des huit points. Le contraste croisé est le plus proche. Comme indiqué ci-dessous :

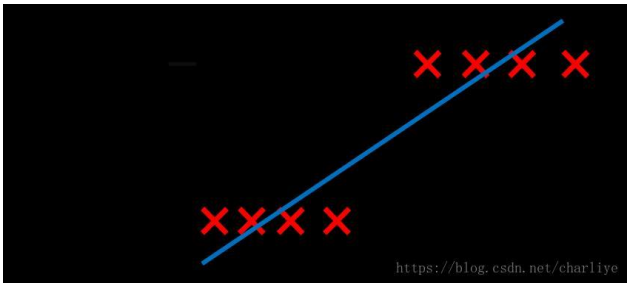


Figure 03 : première étape

Nous supposons que cette ligne bleue peut répondre à nos besoins, alors nous pouvons prendre le point médian de l'axe des ordonnées, qui est de 0,5 comme critère pour juger d'une tumeur maligne ou bénigne. Autrement dit, si la valeur y est supérieure à 0,5, la tumeur est considérée comme maligne, et si la valeur est inférieure à 0,5, la tumeur est considérée comme bénigne. Cela signifie que toutes les tumeurs avec une valeur x qui atteignent un y supérieur à 0,5, c'est-à-dire des tumeurs plus grandes qu'une certaine taille de tumeur, sont considérées comme malignes. Comme le montre la figure ci-dessous, les tumeurs plus petites que x sont considérées comme bénignes et les tumeurs plus grandes que x sont considérées comme malignes.

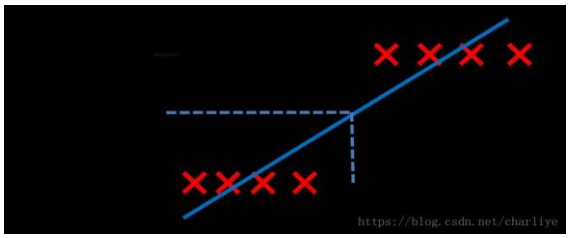


Figure 04 : deuxième étape

Mais ce résultat est-il raisonnable ? Supposons que nous ajoutions la taille de la tumeur et l'état d'un autre patient, comme le montre la figure suivante :

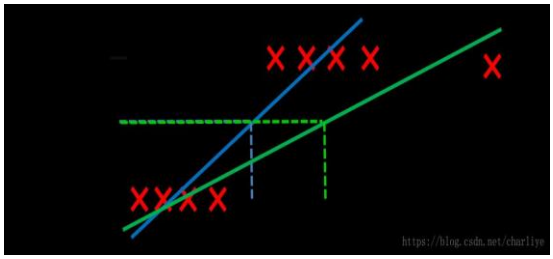


Figure 05 : troisième étape

C'est-à-dire que les données nouvellement ajoutées se trouvent à l'extrême droite. À l'heure actuelle, l'archétype n'est plus pertinent pour ce point nouvellement ajouté. Que devrions-nous faire ? Continuons à déplacer la ligne droite dans le graphique pour qu'elle corresponde aux données nouvellement ajoutées ? Si nous faisons cela, nous constaterons que la valeur x d'origine a changé (se déplace vers la droite) et que tout le modèle a changé, et nous pouvons constater que le modèle obtenu par régression linéaire ne résout pas bien notre problème.

Problème y vaut probablement 1 ou 0, utiliser la régression linéaire, qui peut être bien supérieure à 1 ou bien inférieure à 0 n'est pas très raisonnable, par conséquent, l'algorithme de régression linéaire n'est pas très adapté à ce type de problème, mais par contre l'algorithme de régression peut garantir la valeur de 0 et 1 avec des valeurs entre les deux.

Nous avons donc introduit un algorithme de régression logistique pour traiter ce problème de classification.

Nous voulons que y prenne les valeur 0 et 1 avec des valeurs entre les deux. La fonction de type S (ou fonction de représentation de la fonction logistique) peut répondre à nos besoins. La valeur de la fonction sigmoïde prend les valeurs 0 et 1 entre les deux. Supposons que la

fonction ressemble à la courbe en rouge sur la figure 06, et pour qu'elle ressemble à cela, elle est estimée par la méthode de probabilité maximale.

Nous avons juste besoin de savoir à quoi ressemble la fonction d'hypothèse et comment utiliser la régression logistique pour résoudre le problème de classification.

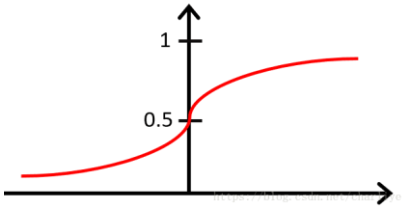


Figure 06 : la fonction d'hypothèse

Il existe un concept de régression logistique appelé limite de décision , la limite de décision est une caractéristique de la fonction d'hypothèse. Étant donné la valeur thêta du paramètre du compensateur en $y(x)$ vous pouvez calculer les conditions $y = 1$. Trouvez le sommet de la ligne X1 + X2 = 3 est $y = 1$, et le bas est $y = 0$. X1 + X2 = 3 sont les limites de la décision. Tant qu'il y a des paramètres, les limites de décision peuvent être entièrement définies. Au lieu d'utiliser l'ensemble d'apprentissage pour définir les limites de décision, nous utilisons l'ensemble d'apprentissage pour ajuster les paramètres et définir les limites de décision par des paramètres [4]

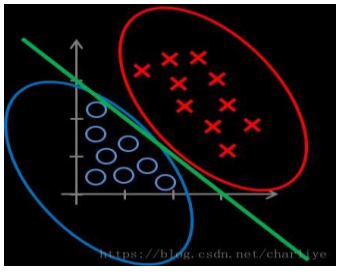


Figure 07 : Limite de décision

5 DOMAINES D'APPLICATION

La classification a connu une croissance importante ces dernières années et désormais appliqué dans plusieurs domaines, de la reconnaissance de forme à la médecine en passant même par le domaine militaire.

Elle est largement utilisée en reconnaissance de formes : reconnaissance des caractères manuscrite comme la reconnaissance de code postale, les chèques et les cartes, reconnaissance de parole…. etc. On retrouve des applications en :

- L'ingénierie : génie mécanique, génie électrique, l'industrie.
- L'informatique : Classification des documents, classification des donnés, segmentation d'image, moteur de recherche... etc.
- La médecine : Système d'aide au diagnostic, c'est l'une des plus importantes applications de la classification dans le domaine médical, un système qui apportera des informations au médecin afin de l'aider à sa décision.
- Marketing : Par exemple, une entreprise peut collecter des informations sur ces clients afin de savoir leur satisfaction.
- Multimédia : la classification des documents web afin d'offrir une organisation thématique de ces derniers, l'accessibilité ou non d'une page web pour enfant à partir de son contenu. autres domaines : comme la géographie, l'astronomie ou en biologie. Cette diversité reflète le grand intérêt de la classification dans la recherche scientifique.[5]

6- les algorithmes de La classification :

- **ARBRE DE DÉCISION**: classe l'utilisation d'une structure d'arbre avec des règles if-then, exécutant l'entrée via une série de décisions jusqu'à ce qu'elle atteigne une condition de terminaison. Capable de modéliser des processus de décision complexes et très intuitif, mais peut facilement surdimensionné les données.
- **FORÊT ALÉATOIRE** : un ensemble d'arbres de décision, avec sélection automatique de l'arbre le plus performant. Fournit la force de l'algorithme d'arbre de décision sans problème de surapprentissage.
- **NAÏVE BAYES CLASSIFIER** :un classificateur basé sur les probabilités. Calcule la probabilité que chaque point de données existe dans chacune des catégories cibles. Simple à mettre en œuvre et précis pour un large éventail de problèmes, mais sensible à l'ensemble des catégories sélectionnées.
- **K-VOISIN LES PLUS PROCHE** : classe chaque point de données en analysant ses voisins les plus proches parmi les exemples d'apprentissage. Simple à mettre en œuvre et à comprendre, efficace pour de nombreux problèmes, en particulier ceux à faible dimensionnalité. Fournit une précision inférieure par rapport aux algorithmes supervisés, et nécessite beaucoup de calculs.[6]

7 - Méthodes de classification classiques :

L'objectif des méthodes de classification est de regrouper en classes assez homogènes un ensemble d'individus décrits par p variables. De ce fait, afin d'affecter chaque individu dans la classe qui le représente le mieux, il faudra commencer par se donner une mesure de distance

- Soit une mesure de distance : On appelle d'une mesure de distance entre les individus, si pour tout$x, yetz \in \{1, ..., \eta\}$, on a :

1. $d(x, y) > 0$
2. $d(x, y) = d(y, x)$.
3. $d(x, z) \leq d(x, y) + d(y, z)$

Lorsque les deux premières conditions sont vérifiées, on parle de dissimilarité.

- Soit une mesure de similarité : On appelle s un indice de similarité entre les individus,

Si pour tout x et $y \epsilon \sum (1 ..., n)$, on a :

1. $s(x, y) \geq 0$
2. $s(x, y) = s(y, x)$
3. $s(x, y)$ Augmente lorsque $x \; et \; y$ se ressemblent davantage

Généralement, $s(x, y) \leq 1$ pour tout $1 \leq (x, y) \leq n$

Pour former les classes, nous devons choisir un critère d'agrégation qui peut être basé sur une distance, une dissimilarité ou une similarité. Ce critère nous permettra de déterminer si des classes sont suffisamment similaires pour n'en former qu'une seule.

Il existe plusieurs méthodes de montage que nous décrirons ici en termes de distance

- **Le saut minimum** : d(A, B) = min $d(x, y)$avec $x \in$ A et $y \in$ B C'est la plus petite distance
- entre deux membres des groupes A et B.
- **Le saut m maximum** : d(A, B) = max $d(x, y)$ avec $x \in$ A et $y \in$ B C'est la plus grande distance entre deux membres des groupes A et B.

Ces deux méthodes sont sensibles aux données aberrantes

- **Le saut moyen** : $d(A + B) = \frac{1}{N_A + N_B} \Sigma_i \in A \Sigma_j \in B\left(x_i, x_j\right)$

Avec N_A et $N_B$le nombre d'individus respectivement de deux classes A et B. C'est la moyenne des distances deux à deux entre les individus de A et B.

- **Méthode du centroïde** : $d(\mathrm{A}, \mathrm{B}) = d(\tilde{x}_A, \tilde{x}_B)$

Avec $\tilde{x}_A = \frac{l}{\mathbb{N}_A}\Sigma_{i\epsilon A}x_i$, $\tilde{x}_B = \frac{1}{N_B}\Sigma_{j\in B}x_j$ et $\tilde{x}_{AB} = \frac{\tilde{x}_A N_A + \tilde{x}_B N_B}{N_B + N_B}$

Les centres de gravité respectivement de A, de B et de la réunion de A et B.

- **Méthode de la médiane** : $d(A,B) = d(\tilde{x}_A, \tilde{x}_B)$ avec $\tilde{x}_{AB} = \frac{\tilde{x}_A + \tilde{x}_B}{2}$ la moyenne des centres de gravité de A et B. La moyenne des centres de gravité de A et B.
- **Méthode de Ward** : $d(A,B) = \frac{N_A N_B}{N_A + N_B} d^2(\tilde{x}_A, \tilde{x}_B)$ La méthode de Ward correspond à la perte d'inertie résultant de l'agrégation de la classe A et de celle de B.

Elle tient également compte de la taille des classes contrairement à celle de la médiane. Ces trois dernières méthodes s'appliquent uniquement sur des données quantitatives.

7.1 Méthodes de la classification hiérarchique

Il existe deux familles de classification hiérarchique :

La classification ascendante hiérarchique (CAR) et la classification descendante hiérarchique. On s'intéresse à la première famille dont les méthodes sont constituées par une suite de partitions emboitées. Les distances définies précédemment permettent itérativement de construire un arbre de classification dit dendrogramme qui va aider sur le choix du nombre de classes à retenir. Ainsi, en découpant cet arbre à une certaine hauteur choisie, on produira la partition désirée. La procédure de la CAR consiste à rassembler successivement les individus suivant le plus grand nombre de degré de leurs ressemblances, c'est-à-dire en des partitions de moins en moins fines. L'algorithme de la CAR le plus simple est le suivant :

Etape 1 : Chaque individu forme sa propre classe

Etape 2 : On regroupe les deux individus (ou groupes d'individus) les plus proches, c'est-à- dire ceux dont la ressemblance est maximale

Etape k : Ainsi de suite, on regroupe les classes les plus proches jusqu'à ce que tous les individus soient dans la même classe

De façon générale, le problème majeur de ces méthodes de classification demeure la détermination du nombre de classes.

7.2 Méthodes de classification non hiérarchique :

Les méthodes de classification non hiérarchique dites de partitionnement ont pour objectif de fixer dès le début le nombre de classes à former. Le problème auquel elles se proposent de

répondre peut s'énoncer de la façon suivante :

Etant donné un ensemble de n individus, on cherche à les classer en K classes ($2 \leq K \leq \eta - 1$)

Etant donné un critère W permettant de mesurer la qualité d'une partition sur l'ensemble de toutes les partitions possibles en K classes sur les n individus, on détermine la partition P' qui optimise W.

Cependant, à cause de l'explosion combinatoire (lorsque n dépasse quelques dizaines), il est impossible d'énumérer toutes les partitions possibles.

Il existe plusieurs méthodes de partitionnement. Mais la plus utilisée et la méthode des centres mobiles dont les K-means et une généralisation de cette dernière appelée méthode des nuées dynamiques.

7.2.1 Méthode des centres mobiles :

L'idée générale de la méthode des centres mobiles consiste à fixer K classes et de déplacer les individus d'une classe à l'autre jusqu'à l'obtention d'une meilleure partition. La procédure générale de la méthode des centres mobiles est la suivante :

- Choisir une configuration initiale
- Affecter les individus aux classes les plus proches
- Calculer les nouveaux centres de ces classes
- Répéter l'algorithme jusqu'à ce qu'il y ait une stabilité de toutes les classes ou un nombre d'itérations maximal atteint sinon on recommence en réaffectant

S'agissant de la méthode des K-means, développée par McQueen en 1967, on choisit dans un premier temps K individus au hasard, considérés comme centres de classes. Par la suite, on calcule la distance entre chaque individu et ces K centres de classes et on affecte les individus aux centres les plus proches. Puis, on calcule les centres de gravité de ces classes et on réaffecte les individus aux nouveaux centres les plus proches. Ainsi de suite, on reprend le même procédé jusqu'à ce que l'algorithme soit convergent.

L'avantage de la méthode des centres mobiles est qu'elle n'utilise que les distances des individus autour des centres, et par conséquent elle peut classer de grands ensembles de données. L'inconvénient majeur est le fait de fixer a priori le nombre de classes à former.

En plus, la partition finale à retenir dépend fortement du nombre de classes initialement fixé

et donc de la partition initiale.

7.2.2 Méthode des nuées dynamiques :

C'est une extension de la méthode des K-means . Ici, au lieu de choisir un seul individu comme centre de gravité, on considère un ensemble d'individus représentatifs de la classe. Cet ensemble est appelé noyau de la classe.

L'algorithme de la méthode des nuées dynamiques est le suivant :

- On sélectionne au hasard k sous-ensembles notés N_j^0 de q objets parmi les n objets de l'ensemble E. Les N_j^0 sont les noyaux.
- On affecte ensuite les objets i de E aux classes j dont la distance D(i, N_j^0) est minimale. On obtient alors une partition P^0 = ($P_{1,\dots}^0 \cdot, P_k^0$) de E.
- On détermine dans chaque classe P_j^0 les q objets, notés N_j^0 qui minimisent D(N_j^1, P_j^0).
- On réitère les deux dernières étapes jusqu'à l'obtention d'une partition stable. Puisque le choix initial des noyaux se fait au hasard, il faudra alors répéter la procédure plusieurs fois de suite, en partant à chaque fois d'une famille différente de noyaux. On appelle « forme forte », un ensemble d'individus qui auront été classés ensemble dans toutes les partitions.

Il est intéressant de déterminer ces individus car il faut une certaine homogénéité pour être dans la même classe malgré le choix de différentes familles de noyaux. Enfin, la méthode des nuées dynamiques peut s'appliquer même sur des données dont la notion de centre de gravité n'a pas de sens.

7.3 Modèle de mélange :

7.3.1 Définition :

La ressemblance entre les éléments d'un même groupe peut se traduire par le fait qu'ils proviennent tous d'une même loi de probabilité. La combinaison linéaire des lois associées à chaque classe conduit à un mélange de lois donné par :

$$P(X) = \sum_{k=1}^{k} \pi_k \, p_k(X)$$

Avec $X = (X_1, \cdots, X_n)$l'ensemble des individus ;

$\pi_k = p(z_{ik} = 1)$, la probabilité a priori que l'individu X_i provienne du groupe G_k ,

$z = (z_1, \cdots, z_i, \ldots, z_n)$ Leur partition en *K* classes $G_1, \cdots G_k$ où $z_i = (z_{i1}, \cdots, z_{ik})$ Tel que $z_{ik} = 1$ si $i \in G_k$ et $Z_{ik} = 0$ sinon .

$P_k(x) = P(X = X_i / z_{ik} = 1)$, la probabilité conditionnelle qu'un individu $X = X_i$appartienne au groupe G_k

Ainsi, on appelle modèle de mélange de lois m, le couple constitué par la loi paramétrique $p(X, \theta)$ et un espace θ contenant l'ensemble des paramètres du modèle, dans lequel évolue $\theta = (\pi_k, \alpha_k)$ Avec α_k le paramètre de la loi qui peut éventuellement être vectoriel.

7.3.2 Cas Gaussien :

Dans le cadre des modèles de mélange gaussien, les données $X_i (i = 1, \ldots, n)$ sont des variables continues de $\mathbb{R}^d$ et la densité conditionnelle des composantes s'écrit :

$$P_k(x_i) = F(x_i, \alpha_k) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_k|^{\frac{1}{2}}} exp\left\{-\frac{1}{2}(x_i - \mu_\kappa)^T \Sigma_k^1 (x_i - \mu_K)\right\}$$

Avec $\alpha_k = (\mu_k, \Sigma_k), \mu_k \in \mathbb{R}^d$,la moyenne de la composante et $\Sigma_k \in \mathbb{R}^{d \times d}$ sa matrice de variance-covariance.[7]

7.3.3- CATEGORIES DE la CLASSIFICATION

Les méthodes de classification analysent des entrées (des données : images,signal, document, état dePatient...) pour en déduire des sorties (classes, catégories, diagnostic...), autrement ditÉtiqueter ces différents donnés en leur associant une classe.

L'apprentissage automatique se propose de construire automatiquement une telle procédure.En effet, deux types de classement peuvent être distingués :[8]

Supervisé (apprentissage avec superviseur) et **Non Supervisé** (apprentissage sans superviseur)

8.Classification supervisée :

8.1 Définition :

La classification supervisée est l'une des techniques qui consiste à inférer ou à apprendre à partir d'un sous ensemble d'échantillon d'exemples$X = \{X_i \in D / i \in \{1, \ldots, n\}, n < N\}$, d'apprentissage ,de l'ensemble de toutes les échantillons possibles D , caractérisés par P variables qualitatives ou quantitatives $(x_{i1}, x_{i2}, \cdots, x_{in})$ appelées attributs et dont la

classe $Y_i \in Y$ est connu d'avance , des modèles, des fonctions f(X_i) ou des règles de décision tel que la règle Si Alors qui permettent de prédire le comportement ou la classe de tout nouvelle échantillon (de test) $X' \in D$ dont la classe est inconnu . On parle alors de classification binaire si le nombre de classes $|Y|$est 2 , et de classification multi classe dans le cas où le nombre de classes $|Y|$est supérieur à 2.[9]

La fiabilité et la pertinence du model appris est évalué et validé à partir des échantillons indépendant de test cela permet de savoir si le model a sous ou sur appris afin de pouvoir effectuer les changement adéquat pour remédier à ces deux problèmes

Les systèmes d'apprentissage peuvent être basés sur des hypothèses probabilistes (Classifieur naïf de Bayes, méthodes paramétriques), sur des notions de proximité (plus proches voisins) , sur des recherches dans des espaces d'hypothèses (arbres de décision, réseaux de neurones)[10]

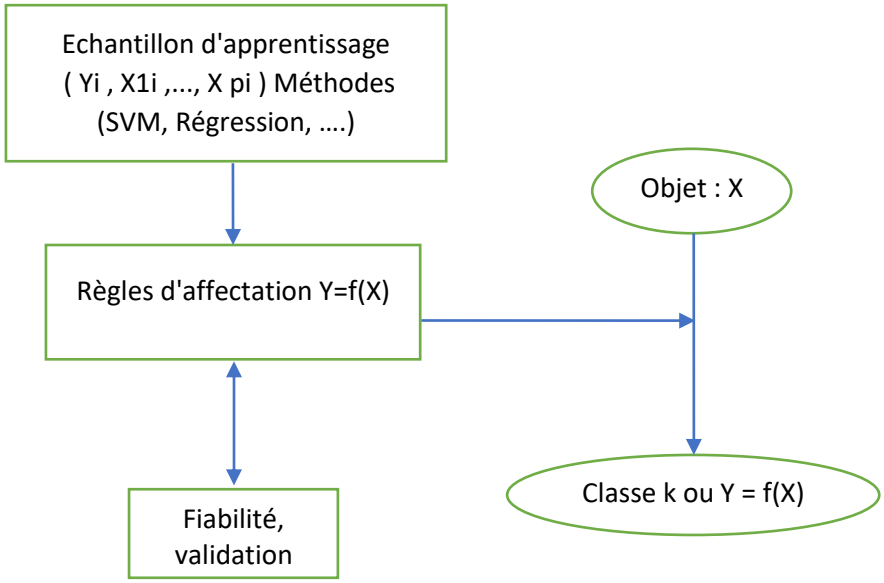


Schéma 01 : La fiabilité et la pertinence du mode

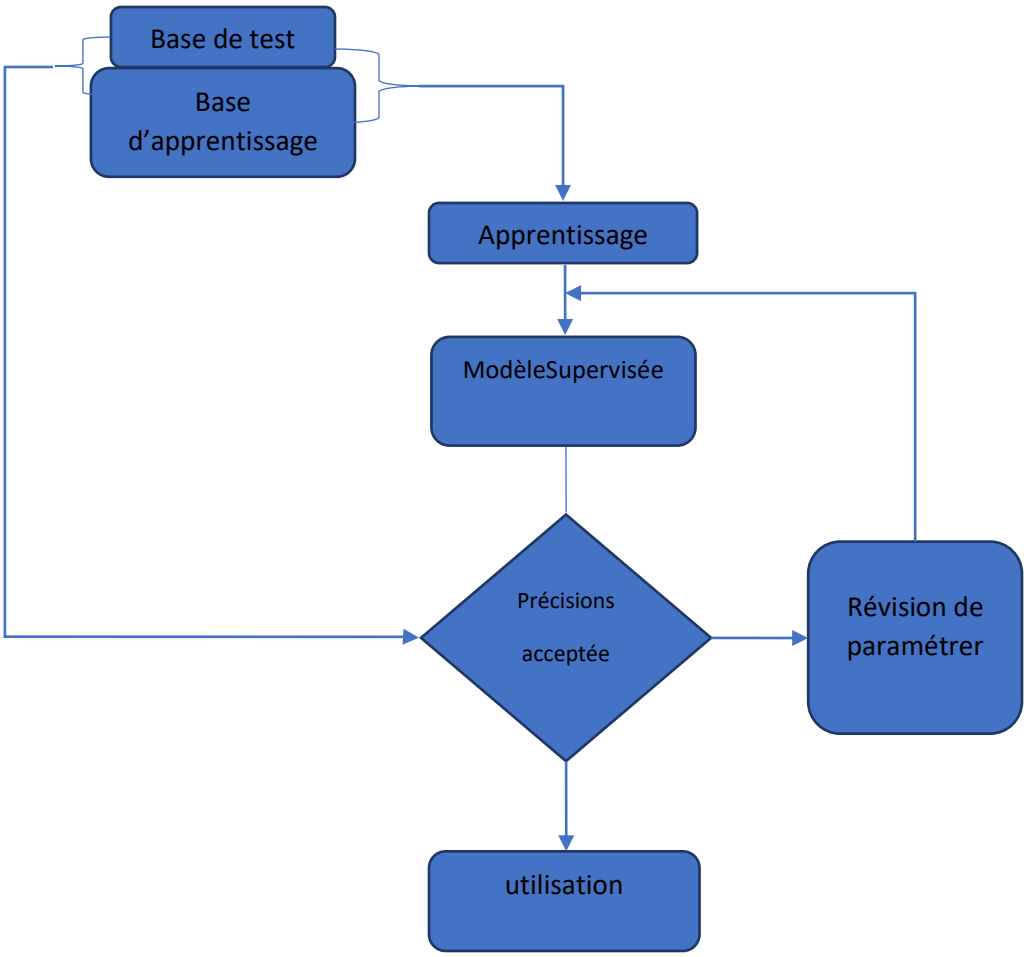


Schéma 02 : modèle supervisée

8.2 Evaluation du modèle :

Indicateurs de performance d'un modèle de classification

- Il est très important d'avoir des mesures permettant de quantifier l'efficacité du modèle appris face à de nouvelle données n'appartenant à la base d'apprentissage pour cela on utilise les différentes métriques dont parmi les suivantes :[11]

La matrice de confusion :

La matrice de confusion est un outil puissant qui sert à évaluer la qualité d'un modèle de classification, l'avantage de cette matrice est qu'elle est simple à lire et à comprendre.

Elle offre une visualisation rapide afin d'analyser les performances d'un modèle et d'identifier les tendances qui peuvent aider à modifier les paramètres du modèle utilisée.

Dans une matrice de confusion les lignes correspondent aux classes réelles et les colonnes correspondent aux classes prédites.

Penons comme exemple un classificateur binaire, qui classe chaque personne selon son état de santé après un diagnostic c'est à dire. S'il est malade ou pas malade, s'il est malade donc le teste est positive sinon le teste est négative.

Vrais positifs (VP) :Est le nombre de personne malade c'est à dire, la personne est malade.

Vrais négatifs (VN) : Est le nombre de personne pas malade.

Faux négatifs (FN) : Est le nombre de personne qui ne sont pas malade mais ils sont classés comme malade

Faux positifs (FP) : est le nombre de personne qui sont malade mais ils sont classés comme pas malade.

		Classe prédite	
		Pas malade	Malade
Classe réelle	Pas malade	VN	FP
	Malade	FN	VP

Table 01 : matrice de confusion

- L'obtention d'une matrice diagonale indique que le classifieur est parfait.
- plusieurs paramètres résument la matrice de confusion et sont décrit comme suit :
  - Précision : Proportion d'éléments bien classés pour une classe donnée elle est représentée par l'équation suivante

$$Précision_{de\ la\ classe} = \frac{VP}{VP + FP}$$

  - Précision globale : (over all accuracy) :probabilité qu'un individu soit correctement classé par un test, c'est-à-dire la somme des vrais positifs et des vrais négatifs divisée par le nombre total d'individus testés.

$$Précisionglobale_{de\ la\ classe} = \frac{VP + VN}{VP + VN + FP + FN}$$

- **Rappel :** Proportion d'éléments bien classés par rapport au nombre d'élément de la classe à prédire elle est représentée par l'équation suivante

$$Rappel_{de\ la\ classe} = \frac{VP}{VP + FN}$$

- **Précision moyenne :**Sommes des proportionsd'éléments bienclassés par rapport au nombre d'élément de la classe à prédire divisé par le nombre de classe *NC,* elle est représentée par l'équation suivante :

$$Précisionmoyenne_{de\ la\ classe} = \left(\frac{VP}{VP + FN} + \frac{VN}{VN + FP}\right)/NC$$

- **F-mesure :** Mesure de compromis entre précision et rappel

$$\text{Moyenne harmonique} = F - mesure_{de\ la\ classe} = \frac{2 * (Précision * Rappel)}{Précision + Rappel}$$

NB : La matrice de confusion peut être utiliser même pour la multi classification c a d pour un nombre de classe supérieur à deux.

La courbe ROC (receiver operating characteristic):

La courbe ROC est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification. Cette courbe trace le taux de vrais positifs en fonction du taux de faux positifs

La représentation de la courbe ROC offre aussi un indice de classification appelé AUC (Area under the curve) plus cette aire rapproche a un plus le modèle de classification est performant est performant.

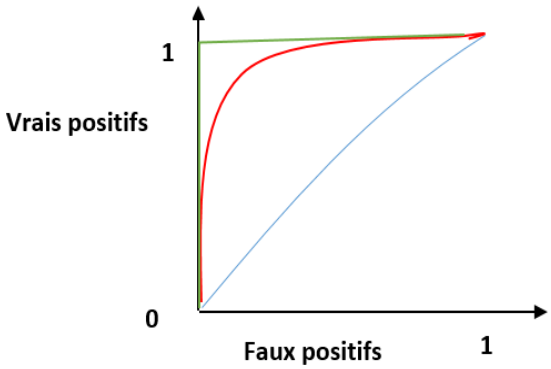


Figure 09 - La courbe ROC

- La courbe bleue correspond à un modèle de classification aléatoire.
- La courbe rouge correspond à un bon modèle de classification.
- La courbe verte correspond à modèle de classification parfait.[12]

9 - Classification non supervisée :

9.1 Définition :

La classification non supervisée est un apprentissage sans superviseur ou sont étiquètes en englais clustering figure (09)il consiste à séparer ou à diviser un ensemble d'échantillon en un certain nombre de groupes dont le nombre est inconnu d'avance , de sorte que l'ensemble d'échantillons présentant des traits de caractéristiques d'avantage similaires soit regroupé ensemble.

Ceci est fait en optimisant un critère visant à regrouper les individus dans des classes, chacune le plus homogène possible et, entre elles, les plus distinctes possible.

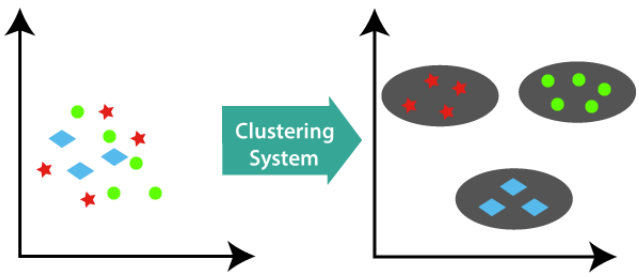


Figure10 - Classification non supervisé ou clustering

Parmi les principales méthodes d'apprentissage non supervisées, on cite les méthodes par partitionnement telles que les algorithmes des k-moyennes ou et les méthodes de regroupement hiérarchique.[13]

CHAPITRE 2 :

Méthodes neuronales

1.INTRODUCTION :

Un réseau de neurones se définit comme un ensemble d'algorithmes dont la conception est à l'origine très schématiquement inspirée du fonctionnement des cellules nerveuses biologiques. En effet, l'expérimentation menée sur les réseaux de neurones formels datent de 1943, suite aux travaux de McCulloch et Pitts.

Ensuite, différents algorithmes ont été proposés selon le domaine d'étude et les résultats attendus. Les méthodes neuronales sont susceptibles de contourner des problèmes complexes que les ordinateurs classiques ne peuvent pas en assurer la gestion.

En effet, si les bases à traiter sont volumineuses, les tâches peuvent se révéler fastidieuses et coûteuses en ressources et, dans ce contexte, l'intelligence artificielle devient incontournable.

De nos jours, les réseaux de neurones peuvent être appliqués dans de nombreux secteurs dont notamment :

- Le traitement d'images reconnaissance de caractères, de signatures, ou de formes, compression d'images, cryptage, classification, etc.
- Le traitement du signal : filtrage, classification, identification de sources, traitement de la parole, etc.
- Contrôle : commande de processus, diagnostic de pannes, contrôlé de qualité, etc.
- Optimisation : planification, gestion, finance, etc. Simulation : prévisions météorologiques, recopies de modèles, etc.

Nous abordons dans cette partie, les différentes règles et types d'apprentissage d'un réseau de neurones, puis, une application avec la méthode du perceptron multicouches.[1]

2 - Définition :

Un réseau neuronal est l'association, en un graphe plus ou moins complexe, d'objets élémentaires, les neurones formels. Les principaux réseaux se distinguent par l'organisation du graphe, c'est-à-dire leur architecture, son niveau de complexité (le nombre de neurones, présence ou non de boucles de rétroaction dans le réseau), par le type des neurones (leurs fonctions de transition ou d'activation) et enfin par l'objectif visé : apprentissage supervisé ou non, optimisation, systèmes dynamiques, etc.[2]

3-Descriptions des réseaux de neurones :

Il existe plusieurs types de réseaux suivant la façon dont sont connectés les neurones entre eux. En informatique, les neurones sont calculés via un programme informatique, mais ils peuvent être parfois réalisés sur un circuit électronique. Un réseau de neurones possède deux états, un état d'apprentissage et un état de fonctionnement optimal. Afin de former le réseau, il doit y avoir une période d'enseignement et de répétition, tout comme un enfant qui développe son cerveau grâce à l'enseignement de ses parents.

Les réseaux de neurones apprennent par un processus de rétro propagation. Cela implique la comparaison de la sortie du réseau de neurones avec la sortie pour laquelle il a été conçu. Cette différence sert à modifier les poids des connexions entre les neurones du réseau et à le rendre plus précis. Une fois le réseau suffisamment calibré, il atteint un niveau d'autonomie exceptionnel où il n'est plus nécessaire de le superviser.

Prenons un exemple, où nous supposons que nous avons enseigné à un réseau de neurones à distinguer une voiture automobile d'un engin de travaux publics. Pour cela nous lui avons montré 15 voitures et 15 engins différents, en lui expliquant bien à quoi correspondait chaque image.

Il atteint alors un état d'autonomie et est capable de distinguer par lui-même si les nouvelles entrées correspondent plutôt à une voiture ou à un engin. Si on lui montre une variante, par exemple, une voiture plus grande il va être capable de catégoriser cet objet, en voiture et non en engin de travaux publics. Simplement sur la seule base de son expérience et sans aucune aide, tout comme un être humain. Au final, nous avons appris à un ordinateur à apprendre. .

Les réseaux de neurones sont utilisés dans la vie de tous les jours pour plusieurs problématiques complexes comme :

- L'estimation des trafics maritime, ferroviaire, routier ;
- Les prévisions météorologiques ;
- La classification d'espèces animales et végétales, par analyse de l'ADN ;
- Les banques, afin de vérifier les montants des transactions financières ;
- La poste pour trier le courrier en fonction du code postal ;
- Les opérations boursières et bien d'autres utilisations ;

La recherche sur les réseaux de neurones a gagné en popularité et a atteint le sommet au début des années 1990. Jusqu'en 2006, les scientifiques n'ont pas su former les réseaux de neurones à dépasser les approches plus traditionnelles, sauf pour quelques problèmes spécialisés. Ce qui a changé à partir de cette date, c'est la découverte de techniques d'apprentissage dans les réseaux de neurones profonds. Ces techniques sont maintenant connues comme apprentissage profond.

Elles ont été développées d'avantage et aujourd'hui les réseaux de neurones et l'apprentissage profond ont obtenus des performances exceptionnelles sur de nombreux problèmes importants. Ils sont déployés à grande échelle par des entreprises telles que Microsoft, Google, IBM et Facebook.

Dans l'approche conventionnelle de la programmation, nous disons à l'ordinateur ce qu'il faut faire, diviser un grand problème en de nombreuses petites tâches définies avec précision, que l'ordinateur peut facilement effectuer. En revanche, dans un réseau de neurones, nous ne disons pas à l'ordinateur comment résoudre 6notre problème. Au lieu, il apprend des données d'observation, trouvant sa propre solution au problème en question.

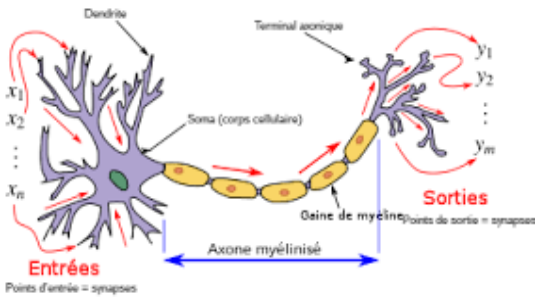


Figure 01 : Les neurones biologiques

4- Les neurones biologiques et artificiels :

Le cerveau est l'organe le plus complexe du corps humain. L'unité fondamentale du cerveau humain est le neurone. Un petit morceau de cerveau de la taille du grain de riz, contient plus de 10. 000 neurone. A sa base, le neurone est optimisé pour recevoir des informations d'autres neurones, traiter ces informations de manière unique et envoyer son résultat à d'autres cellules.

Le neurone reçoit ses entrées le long d'une structure en forme d'antenne appelée dendrite. Après avoir été pondéré par la force de leurs connexions, les entrées sont additionnées dans

le corps de la cellule. Cette somme est ensuite transformée en un nouveau signal qui se propage le long de l'axone de la cellule et envoyée à d'autres cellules.

Nous pouvons traduire cette compréhension fonctionnelle des neurones de notre cerveau en un modèle artificiel que nous pouvons représenter sur notre ordinateur. Dans le domaine des technologies de l'information, un réseau de neurones est un système logiciel et / ou matériel qui imite le fonctionnement des neurones biologiques. Les réseaux neuronaux, aussi appelés réseaux de neurones artificiels (RNA ou ANN, en anglais).

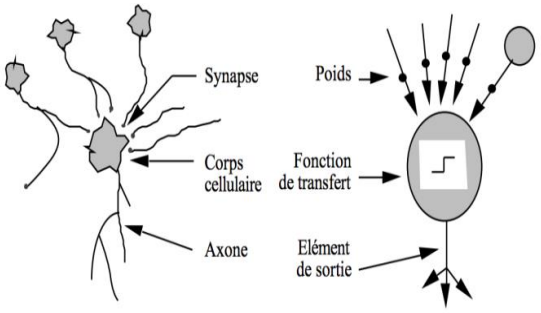


Figure 02 : Les neurones biologiques et artificiels

5-Le neurone formel :

Le neurone formel est un dispositif à plusieurs entrées et une sortie, qui prend pour modèle schématiquement le neurone biologique : il correspond respectivement aux dendrites et au point de départ de l'axone (cône d'émergence du neurone biologique).

Le neurone formel est une modélisation mathématique sous forme d'une fonction algébrique non linéaire, qui reprend les principes du fonctionnement du neurone biologique et dont la 7valeur dépend des paramètres appelés coefficients ou poids. Il constitue l'unité minimale d'un réseau de neurones artificiels.

6 - Le perceptron simple :

Un réseau de neurones est une structure de réseau constituée d'un nombre de nœud interconnectés par des liaisons directionnelles. Chaque nœud représente une unité de traitement et les liaisons représentent les relations causales entre les nœuds. Le réseau le plus simple est celui de neurones monocouches appelé perceptron terme forgé par Franck Rosenblatt en 1958 et l'assimilant à un modèle de rétine artificiel. Ce réseau est représenté par une couche de neurones d'entrée et de plusieurs neurones de sortie. Il constitue l'un des

tous premiers algorithmes d'apprentissage supervisé. Il s'agit d'un neurone artificiel inspiré par la théorie cognitive de Friedrich Hayek et celle de Donald Hebb. Dans sa version la plus simple, le perceptron n'a qu'une seule sortie.

La structure d'un perceptron est représentée dans la figure 03. Les valeurs d'entrée $(x_1, x_2, \dots, x_n)$ et les poids associés aux entrées $(w_{1j}, w_{2j}, \cdots, w_{n_j})$ sont les variables de la fonction de combinaison qui détermine la somme pondérée du neurone net_j . Cette somme est ensuite passée à la fonction d'activation qui détermine la valeur de sortie du neuroneo_j . Enfin, le neurone corrige ses poids afin de s'approcher de la valeur attendue (S).

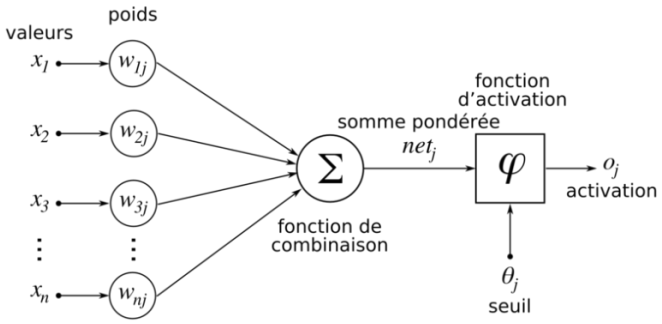


Figure03 : Structure d'un perceptron simple

La somme pondérée des entrées par les poids (weights) w_{ij},associés aux entrées est appeléepotentiel, notéS.

$$S = \sum_{i=1}^{n} w_{ij} + b$$

S : est la somme pondérée du neurone net_j

w_{ij} : est le poids de l'entrée i.

x_i : est la valeur de l'entrée i

b : entrée spéciale (biais).

- **Biais(biais)** : c'est une valeur d'entrée qui est toujours égale à 1, ce qui permet une normalisation grâce à la valeur du poids correspondant ;
- **La fonction d'activation (φ)** : transforme la somme pondérée pour obtenir la valeur de sortie. Le choix de la fonction de sortie dépend de l'application et du comportement souhaité ;

7- Le perceptron multicouche (PMC) :

Le perceptron multicouche est un classifieur linéaire de type réseau neuronal formel organisé

en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. Les connexions sont toujours dirigées des entrées à travers le réseau vers les sorties, sans interconnections entre les neurones de la même couche.

Les trois couches d'un MLP et le rôle de chacune :

1. Couche d'entrée : La première couche elle est complètement connectée vers l'avant et reçoit toutes les valeurs d'entrées du réseau à des fins d'apprentissage. Donc le nombre de neurones dans la couche d'entrées est égal au nombre de données en entrées (ou égale au nombre de dimensions du problème) ;
2. Couches cachées : couches qui succèdent à la couche d'entrée, constituées d'une ou de plusieurs couches intermédiaires. Elles relient la couche d'entrées à la couche de sorties. Le choix du nombre de couches et la taille (nombre de neurones) de chaque couche est empirique ;
3. Couche de sortie : La troisième couche ou la couche de résultat. Elle donne le
4. résultat obtenu par le réseau. Donc, le nombre de neurones dans la couche de sorties
5. égale au nombre de classes ;

La figure 1 illustre la structure d'un MLP présentant trois neurones en entrée, deux couches cachées et deux neurones en sortie.

Lorsque tous les neurones d'une couche sont connectés aux neurones de la couche suivante, on parle alors de couches complètement connectées.[3]

7.1 Principe de fonctionnement de multicouche :

Fonctionnement Le fonctionnement du perceptron multicouches est organisé en trois parties figure 04

La couche d 'entrée : Elle est constituée d'un ensemble de neurones qui portent le signal d'entrée et reçoivent les données sources que l'on veut utiliser pour l'analyse.

Les couch es cachées: Elles constituent le véritable moteur de calcul du perceptron. C'est la phase de l'apprentissage. Ici, le réseau étudiera les relations entre les différentes variables et effectue ensuite les pondérations. Le choix du nombre de couches est arbitraire mais il se fait généralement par essai erreur, et les sorties d'une couche sont les entrées de la suivante.

La couch e de sortie : Elle donne enfin le résultat final obtenu par notre réseau.[4]

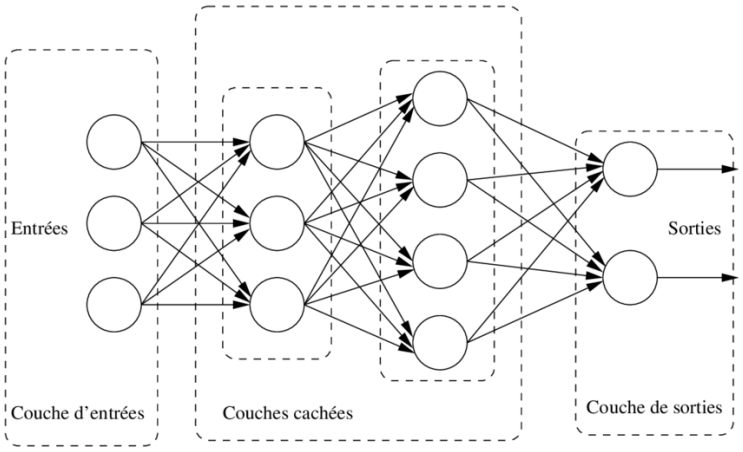


Figure 04 :Organisation du perceptron multicouches

7.2 Fonctions d'activation :

Dans le domaine des réseaux de neurones artificiels, la fonction d'activation est une fonction mathématique non linéaire appliquée à u ! n signal en sortie d'un neurone artificiel. Le terme de fonction d'activation est synonyme de potentiel d'activation en biologie. C'est un seuil de stimulation qui une fois atteint entraîne une réponse du neurone. Les fonctions d'activation sont utilisées selon leurs caractéristiques d'étendue de non linéarité, dedifférentiabilité de convergence vers l'identité en 0 de continuité ou de monotonie.[5]

Plusieurs fonctions d'activation ont été mises au point. Nous retenons quelques-unes

A. La fonction seuil :

Également appelée modèle tout ou rien, elle applique une valeur seuil sur son entrée et permet de prendre des décisions binaires

$$f(x) = \begin{cases} 0 & si \quad x < 0 \\ 1 & si \quad x > 0 \end{cases}$$

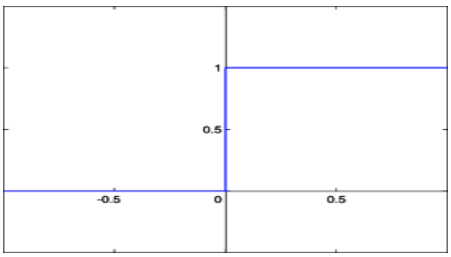


Figure 05 : fonction Seuil

A'. La fonction linéaire $f(x) = ax, a \in \mathbb{R}^*, x \in \mathbb{R}$[6]

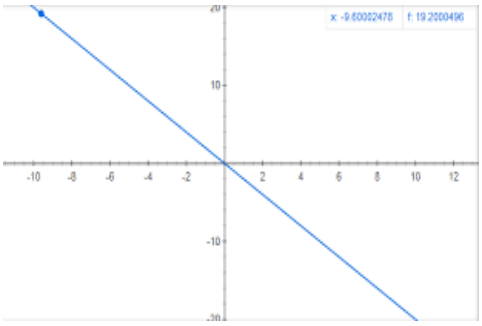


Figure 06 : fonction linéaire

B. Fonction Sigmoïde (Marche Douce ou Logistique) :

Elle est souvent utilisée comme fonction d'activation pour les réseaux de neurones, dans lesquels les valeurs de sortie désirées sont soit binaires soit dans un intervalle compris entre 0 et 1. Le but de cette fonction est de réduire la valeur d'entrée à un intervalle entre 0 et 1.
Si la valeur en entrée est un très grand nombre positif, la fonction convertira cette valeur en une probabilité de 1.A l'inverse, si la valeur en entrée est un très grand nombre négatif, la fonction convertira cette valeur en une probabilité de 0. L'équation ci-dessous représente la fonction Sigmoïde

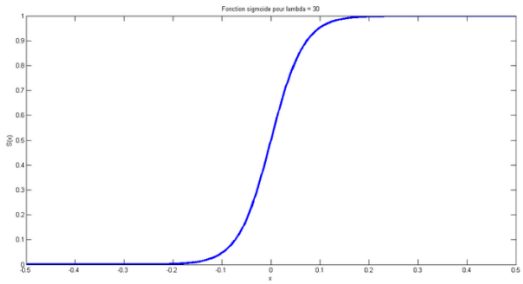


Figure 07 :la function Sigmoïde

$$\sigma = \frac{1}{1 + e^{x}}$$

C. Fonction Tanh (Tangente Hyperbolique) :

Cette fonction ressemble à la fonction Sigmoïde. La différence avec la fonction Sigmoïde est que la fonction Tanh produit un résultat compris entre -1 et 1. La fonction Tanh est en terme général préférable à la fonction Sigmoïde car elle est centrée sur zéro. Les grandes entrées négatives tendent vers -1 et les grandes entrées positives tendent vers 1.

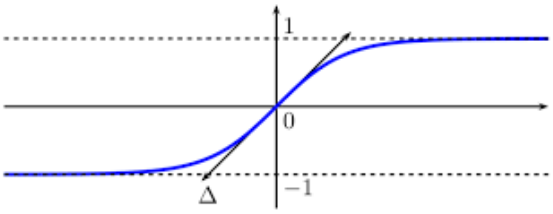


FIGURE 08 : Fonction tangente hyperbolique

D. FonctionReLU (Unité de Rectification Linéaire) :

: Elle est inspirée par des motivations biologiques. Elle a été introduite pour la première fois en 2011 par Glorot et al, Les avantages de ReLU incluent le calcul efficace, 10l'activation clairsemée, etc. Elle est favorisée pour les réseaux neuronaux profonds et plus vite, L'équation ci-dessous représente la fonction ReLU : Si l'entrée est négative la sortie est 0. Et si elle est positive alors la sortie est x. Cette fonction est la plus utilisée.

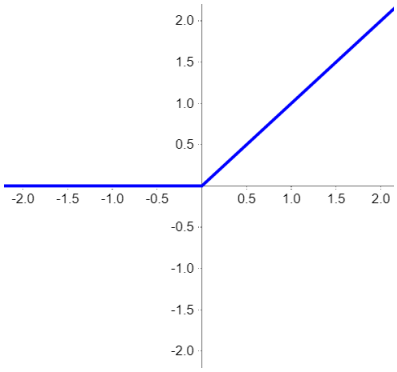


Figure 09 :la function RELU

$$F(x) = max(0, x)$$

E. FonctionSoftmax :

Est également appelée fonction exponentielle normalisée. Elle est couramment utilisée pour représenter une distribution de probabilité pour les K sorties possibles. Ainsi, la fonction softmax a été fortement utilisée dans les tâches de classification. Elle est définie comme :[7]

$$F(x)_i = \frac{e_i^x}{\sum_{k=1}^{k} e^{kx}}$$

8 La rétropropagation :

La technique de rétropropagation (Backpropagation en anglais) est une méthode qui permet de calculer le gradient de l'erreur pour chacun des neurones du réseau de la dernière couche vers la première. On appelle souvent technique de rétropropagation du gradient, l'algorithme

classique de correction des erreurs basé sur le calcul du gradient grâce à la rétropropagation, mais cela n'est pas toujours le cas. La correction des erreurs peut se faire selon d'autres méthodes. Le plus souvent, dans le cas des réseaux de neurones, la méthode de correction d'erreurs agit en corrigeant de manière significative les coefficients synaptiques qui contribuent à engendrer une erreur importante tout en pondérant également les neurones générant une erreur moins conséquente.

La procédure de rétropropagation peut se résumer dans les étapes suivantes :

— Initialiser tous les poids à de petites valeurs aléatoires dans l'intervalle [-0.5, 0.5].

— Normaliser les données d'entraînement.

— Permuter aléatoirement les données d'entraînement.

— Pour chaque donnée d'entraînement n :

1. Calculer les sorties observées en propageant les entrées vers l'avant.

2. Ajuster les poids en rétro propageant l'erreur observée. Répéter les étapes 1 et 2 jusqu'à un nombre maximum d'itérations ou jusqu'à ce que l'erreur soit en dessous d'un certain seuil.[8]

9Algorithme de rétropropagation :

Soit le couple $(\vec{x}(n), \vec{d}(n))$désignant la n^edonnée d'entrainement du réseau où :

$$\vec{x}(n) =< x_1(n), \dots\dots, x_p(n) > et\vec{d}(n) =< d_1(n), \dots, d_q(n) > \qquad (1)$$

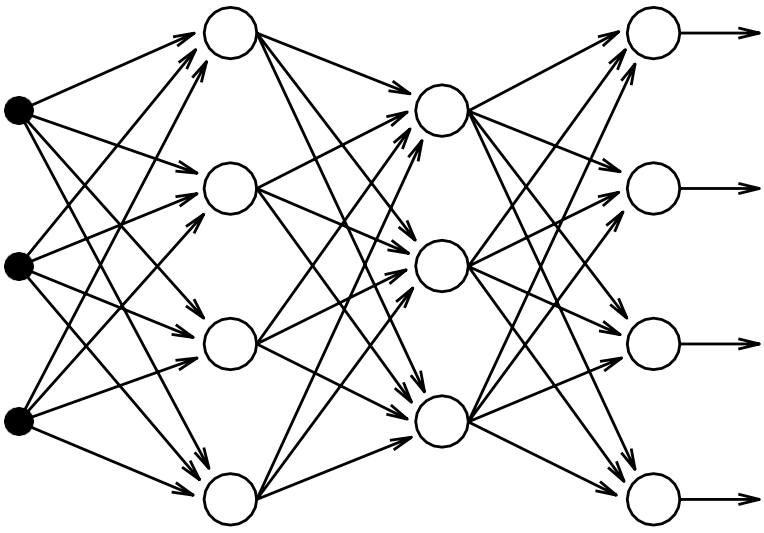


Figure 10 : Exemple d'un réseau de type perceptron multicouche

Correspondent respectivement aux p entrées et aux q sorties désirées du système.

L'algorithme de rétropropagation consiste alors à mesurer l'erreur entre les sorties désirées$\vec{d}(n)$et les sorties observées $\vec{y}(n)$

$$\vec{y}(n) =< y_1(n), \cdots, y_q(n)\rangle \qquad (2)$$

Résultant de la propagation vers l'avant des entrées $\vec{x}(n)$ et a rétroportages cette erreur à travers les couches du réseau en allant des sorties vers les entrées cas de la couche de la sortie. L'algorithme de rétropropagation procédé à l'adaptation des poids neurone par neurone en commençant par la couche de sortie.

Soit l'erreur observée $e_j(n)$ pour le neurone de sortie$j$ et la donnée d'entraînement$n$:

$$e_j(n) = d_j(n) - y_j(n) \qquad (3)$$

Ou $d_j(n)$ correspondà la sortie Désirée du neurone$j$ et $y_j(n)$ à sa sortie observée.

1 Cas de la couche de sortie

$$y_0 = -1$$

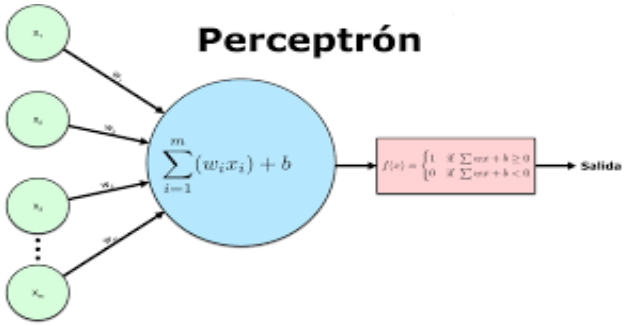


Figure 11 : Modèle du neurone

– La variable n représentera toujours la donnée d'entrainement c'est-à-dire le couple contenant un vecteur d'entrées et un vecteur de sorties désirées.

– L'objectif de l'algorithme est d'adapter les poids des connexions du réseau de manière à minimiser la somme des erreurs sur tous les neurones de sortie

– L'indice j représentera toujours le neurone pour lequel on veut adapter les poids.

Soit $E(n)$ la somme des erreurs quadratiques observées sur l'ensemble C des neurones de sortie :

$$E(n) = \frac{1}{2}\sum_{j\epsilon^c} e_j^2(n) \qquad (4)$$

La sortie $y_j(n)$ du neurone j est définie par : ´

$$y_j(n) = \varphi\left[v_j(n)\right] = \varphi\left[\sum_{i=0}^{r} w_{ji}(n)\, y_i(n)\right] \qquad (5)$$

Ou` φ[.] est la fonction d'activation du neurone, $v_j(n)$ est la somme pondérée des entrées du ´ neurone$j$, $w_{ji}(n)$ est le poids de la connexion entre le neurone i de la couche précédente et ´ le neurone j de la couche courante, et $y_i(n)$ est la sortie du neurone i . On suppose ici que la couche

Couche précédente contient $r$neurones numérotés de 1 a r , que le poids $w_{j_0}(n)$correspond au biais du neurone j et qu'entrée $y_0(n)$ = - 1. La figure 11 illustre l'équation 5

L'indice i représentera toujours un neurone sur la couche précédente par rapport au neurone$j$; on suppose par ailleurs que cette couche contient r neurones

Pour corriger l'erreur observée, il s'agit de modifier le poids $w_{ji}(n)$ dans le sens oppose´ au gradient $\frac{\partial E(n)}{\partial w_{ji}(n)}$ de l'erreur

– Cette dérivée partielle représente un facteur de sensibilité : si je change un peu $w_{ji}(n)$, est-ce que c.a. changé beaucoup $E(n)$? Si oui, alors je vais changer beaucoup $w_{ji}(n)$ dans le sens inverse de cette dérivée car cela devrait me rapprocher beaucoup du minimum local. Sinon, je dois changer seulement un peu $w_{ji}(n)$ pour corriger l'erreur car je suis tout près de ce minimum !

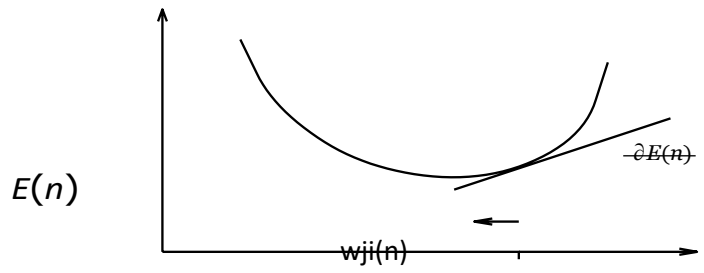


Figure 12 : Gradient de l'erreur totale

– Puisqu'il y a r neurones sur la couche précédant la couche de sortie, il y a aussi r poids à adapter, et il importe donc de remarquer que la courbe de la figure 12 correspond en ` fait à une hyper-surface de ` r + 1 dimensions ! Par la règle de chainage des dérivées partielles, qui nous dit que $\frac{\partial f(y)}{\partial x} = \frac{\partial f(y)}{\partial y} \cdot \frac{\partial y}{\partial x}$, on obtient :

$$\frac{\partial E(n)}{\partial w_{ji(n)}} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial_{vj}(n)}{\partial w_{ji}} \quad (6)$$

Et on exprime la variation de poids $\Delta w_{ji}(n)$ sous la forme suivante :

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji(n)}}$$

Avec $0 \leq \eta \leq 1$ représentant un taux d'apprentissage ou gain de l'algorithme.

Evaluons maintenant chacun des termes du gradient. ´

$$\frac{\partial E(n)}{\partial e_j(n)} = \frac{\partial \left[\frac{1}{2}\sum_{k\epsilon C} e_k^2(n)\right]}{\partial e_j(n)} \quad (7)$$

$$= \frac{1}{2} \cdot \frac{\partial e_j^2(n)}{\partial e_j(n)}$$

$$= e_j(n)$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial \left[d_j(n) - y_j(n)\right]}{\partial y_j(n)}$$

$$= -1$$

$$= \frac{\partial y_j(n)}{\partial_{vj}(n)} = \frac{\partial \left[\frac{1}{1+e^{-v_j(n)}}\right]}{\partial v_j(n)}$$

$$= \frac{e^{-v_j(n)}}{\left[1+e^{-v_j(n)}\right]^2}$$

$$= y_j(n)\left[\frac{e^{-v_{j(n)}}}{1+e^{-v_j(n)}}\right]$$

$$= y_j(n)\left[\frac{e^{-v_j(n)}+1}{1+e^{-v_j(n)}} - \frac{1}{1+e^{-v_{j(n)}}}\right]$$

$$= y_j(n)\left[1 - y_j(n)\right]$$

Et finalement :

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = \frac{\partial \left[\sum_{l=0}^{r} w_{j_l}(n) y_l(n)\right]}{\partial w_{j_i}(n)}$$

$$= \frac{\partial \left[W_{ji}(n) y_i(n)\right]}{\partial W j_i(n)}$$

$$= y_i(n)$$

Nous obtenons donc :

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)y_j(n)\big[1 - y_j(n)\big]y_j(n) \qquad (8)$$

Et la règle dite du “delta” pour la couche de sortie s’exprime par : `

$$\Delta_{wj_i}(n) = -\eta \frac{\partial E(h)}{\partial w_{ji}(n)} = \eta\delta_j(n)y_j(n) \qquad (9)$$

Avec

$$\delta_j(n) = e_j(n)y_j(n)[1 \cdot yi(n)] \qquad (10)$$

Qui correspond à ce qu’on appelle le “gradient local”.

– Jusqu’ici, nous avons traite seulement le cas de la couche de sortie ! Il reste maintenant à faire l’adaptation des poids sur les couches cachées.

– Mais le problème est qu’on ne dispose plus de l’erreur observée ! ?

2 - Cas d’une couche cachée

Considérons maintenant le cas des neurones sur la dernière couche cachée (le cas des ´ autres couches cachées est semblable).

– La variable n désignera toujours la donnée d’entrainement c’est-à-dire un couple de ` vecteurs d’entrées et de sorties désirées.

– L’objectif sera toujours d’adapter les poids de la couche courante en minimisant la somme des erreurs sur les neurones de la couche de sortie.

– Les indices i et $j$désigneront respectivement (comme précédemment) un neurone sur ´ la couche précédente et un neurone sur la couche courante. ´

– L’indice k servira maintenant à designer un neurone sur la couche suivante. ´ Reprenons l’expression de la dérivée partielle de l’erreur totale ´$E(n)$ par rapport à`w_{j_i}mais en ne dérivant plus par rapport ´ à l’erreur $e_j(n)$ car celle-ci est maintenant inconnue :

$$\frac{\partial E(n)}{\partial^w ji(n)} = \frac{\partial E(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{j_i}(n)} \qquad (11)$$

Par rapport aux résultats obtenus pour la couche de sortie, les deux derniers termes de cette ´ équation restent inchangées, seul le premier terme requiert d’être évalue :

$$\frac{\partial E(n)}{\partial y_j(n)} = \frac{\partial\left[\frac{1}{2}\sum_{k\in c} e_j^2(n)\right]}{\partial y_j(n)} \qquad (12)$$

Notre problème ici, contrairement au cas des neurones de la couche de sortie, est que tous les $e_k(n)$ dans la somme ci-dessus dépendent de $y_j(n)$. On ne peut donc pas se débarrasser de ´ cette somme ! Néanmoins, nous pouvons écrire :

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_{k\in C}\left[e_k(n)\cdot\frac{\partial e_k(n)}{\partial y_i(n)}\right]$$

$$\sum_{k\in c}\left[e_k(n)\cdot\frac{\partial e_k(n)}{\partial V_k(n)}\cdot\frac{\partial v_k(n)}{\partial y_j(n)}\right]$$

$$\sum_{k\in c}\left[e_k(n)\cdot\frac{\partial\left[d_k(n)-\varphi\left(v_k(n)\right)\right]}{\partial v_k(n)}\cdot\frac{\partial\left[\Sigma_L w_{k_l}(n)y_L(n)\right]}{\partial y_j(n)}\right]$$

$$\sum_{k\in c}\left[e_k(n)\cdot(-y_k(n)[1-y_k(n)]).w_{k_j}\right]$$

Et en substituant l'équation 10 on obtient :

$$\frac{\partial E(n)}{\partial y_j(n)}' = \sum_{k\in C}\delta_k(n)w_{kj}(n) \qquad (13)$$

A. Sommaire de l'algorithme (règle du "delta")

En substituant l'équation 13 dans l'équation 11, on obtient : ´

$$\frac{\partial E(n)}{\partial w_{j_i}(n)} = -y_i(n)\left[1-y_j(n)\right]\left[\sum\nolimits_{k\in\Sigma}\delta_k(n)W_{j_i}(n)\right]y_i(n) \quad (14)$$

$$\Delta W_{j_i}(n) = -\eta\frac{\partial E(n)}{\partial W_{ji}(n)} = \eta\delta_k(n)w_{k_j}(n) \qquad (15)$$

$$\delta_j(n) = y_i(n)\left[1-y_j(n)\right]\sum\nolimits_{k\epsilon C}\delta_k(n)W_{kj}(n) \qquad (16)$$

On peut démontrer que les équations 15 et 16 sont valides pour toutes les couches cachées. ´

Notez bien, cependant, que dans le cas de la première couche cachée du réseau, puisqu'il n'y a pas de couche précédente de neurones, il faut substituer la variable $y_i(n)$ par l'entrée $x_i(n)$.

Sommaire de l'algorithme (règle du "delta")

L'algorithme de rétropropagation standard se résume donc à la série d'étapes suivantes :

´ 1. Initialiser tous les poids à de petites valeurs aléatoires dans l'intervalle ´ [−0.5, 0.5] ;

2. Normaliser les données d'entraînement ;

3. Permuter aléatoirement les données d'entraînement ;

4. Pour chaque donnée d'entraînement$n$:

(a) Calculer les sorties observées en propageant les entrées vers l'avant ;

´ (b) Ajuster les poids en retro propageant l'erreur observée :

$$W_{j_i}(n) = w_{ji}(h-1) + \Delta W_{j_i}(n) = W_{j_i}(n-1) + \eta\delta_j(n)y_i(n) \qquad (17)$$

B. Ou le "gradient local" est défini par : ´

$$\delta_j(n) = \begin{cases} e_j(n)y_j(n)\left[1-y_j(n)\right] & \text{Si j } \in \text{ couche de sortie} \\ y_j(n)\left[1-y_j(n)\right]\sum_k \delta_k(n)w_{k_j}(n) & \text{Si } j \in \text{ couche cachée} \end{cases} \qquad (18)$$

Avec 0 ≤ η ≤ 1 représentant le taux d'apprentissage et $y_i(n)$ représentant soit la sortie du neurone i sur la couche précédente, si celui-ci existe, soit l'entrée´ i autrement.

Répéter les étape 3 et 4 jusqu'à un nombre maximum d'itérations ou jusqu' à ce que la ` racine de l'erreur quadratique moyenne (EQM) soit inferieure a un certain seuil.

C. Règle du "delta généralise"

L'équation 17 d écrite qu'on appelle la règle du "delta" pour l'algorithme de rétropropagation des erreurs. L'équation suivante, nomme règle du "delta généralisé", d écrit une autre variante de l'algorithme :

$$W_{ji}(n) = w_{j_i}(n-1) + \delta_j(n) + \alpha\Delta w_{j_i}(n-1) \qquad (19)$$

Ou` 0 ≤ α ≤ 1 est un paramètre nomme´ Momentum qui représente une espèce d'inertie dans ` le changement de poids.[9]

CHAPITRE 3 :

Classification par Autoencodeur

1. Introduction :

Les réseaux de neurones ne se limitent pas forcément à des problématiques d'apprentissage supervisé, même si cela reste leur utilisation principale.

En effet, il existe aussi des réseaux de neurones qui permettent de faire de l'apprentissage non supervisé. Les plus utilisés dans ce domaine sont probablement les Auto-Encodeurs.

Ces réseaux n'ont pas pour objectif de prédire ce qui se trouve dans une donnée mais seulement de chercher à trouver une représentation pertinente et compressée de la donnée en tirant justement parti du pouvoir de représentation des réseaux de neurones.

Auto Encodeurs : sont des algorithmes d'apprentissage non supervisé à base des réseaux de neurones artificiels, qui permettent de construire une nouvelle représentation une d'une donnée. Généralement, celle-ci est plus compacte, et présente moins de descripteurs, ce qui permet de réduire la dimensionnalité de la donnée.

L'architecture d'un auto-encodeur est constituée de deux parties : l'encodeur et le décodeur.[1]

2Architecture et formulation mathématique d'un autoencodeur:

On peut décomposer un auto encodeur en deux parties, à savoir un encodeur, f_θ, suivi d'un décodeur, g_ψ. L'encodeur permet de calculer le code $Z_j = f_\theta(x_i)$ pour chaque échantillon d'apprentissage en entrée (x_{ij}), avec i allant de 1 jusqu'à n, n étant le nombre de lignes et j allant de 1 jusqu'à p, p étant le nombre de colonnes. Le décodeur vise à reconstituer l'entrée à partir du code $z_i: \hat{x}_i = g_\psi(Z_i)$. Les paramètres de l'encodeur et du décodeur sont appris simultanément pendant la tâche de reconstruction, tout en minimisant la fonction objective :

$$\mathcal{J}_{AE}(\theta, \Psi) = \sum_{i=1}^{n} \mathcal{L}\left(x_i, g_\Psi\left(f_\theta(x_i)\right)\right) = \sum_{i=1}^{n} \mathcal{L}\left(x_i, g_\Psi(z_i)\right) = \sum_{i=1}^{n} \mathcal{L}\left(x_i, \hat{x}_i\right) \qquad (1)$$

$\mathcal{L}$Étant une fonction de coût permettant de mesurer la divergence entre l'échantillon d'apprentissage en entrée et les données reconstruites. f_θ et$g_\Psi$sont des fonctions de transition :

$f_\theta: X \to F \;\; et \; g_\Psi: F \to X$ OùX et F sont respectivement les ensembles d'entrée et de sortie de L'encodeur ;

$$f_\theta, g_\Psi = \underset{f_\theta, g_\Psi}{\arg\min} \left|\left|x - \left(f_\theta o g_\psi\right)(x)\right|\right|^2 \qquad (2)$$

où $\left(f_\theta o g_\psi\right)(x) = f_\theta[g_\Psi(x)]$ pour tout $x \in X$,

Dans le cas où il n'y a qu'une seule couche cachée, l'étape d'encodage prend l'entrée $x \in \mathbb{R}^p = X$ et l'associe à $z \in \mathbb{R}^k = \mathcal{F}, p \geq k$:

$$z = f_\theta(wx + b) \qquad (3)$$

Où z est généralement appelé code, variable latente ou représentation latente, θ est une fonction d'activation (e.g., sigmoïde, ReLU...), W est une matrice de poids du réseau de neurones et b un vecteur de biais.

Ensuite, l'étape de décodage associe z à la reconstruction $\hat{x}$ de forme identique à x :

$$\hat{x} = g_\Psi(w'z + b') \qquad (4)$$

Oùw'etb' pouvant être différents ou non de w et b de l'encodeur, selon la conception de l'autoencodeur. La figure 01illustre l'architecture d'un autoencodeur.

L'encodeur représente la partie du réseau qui compresse l'entrée dans un espace latent représentant la couche code. Le code est la couche cachée qui est généralement représentée sous forme compressée dans une dimension réduite. Il constitue aussi l'entrée alimentée du décodeur. Le décodeur est la partie du réseau qui tente de reconstruire l'entrée à partir de l'espace latent.[2]

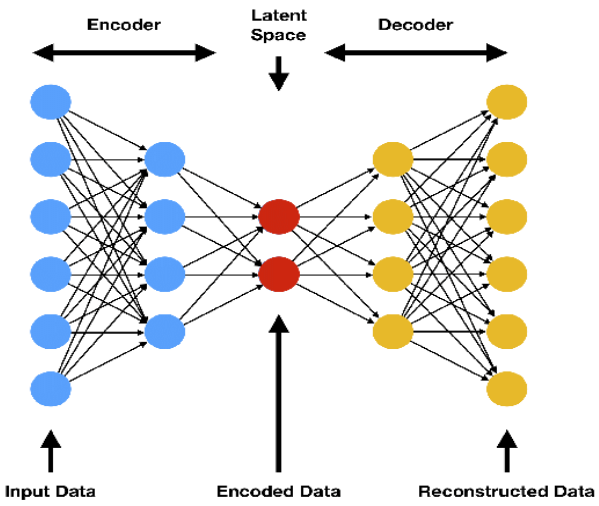


Figure01 : principe de l'autoencodeur

3 - les différents types d'auto-encodeurs

Il existe différents types d'auto-encodeurs, issus de deux principes générale d'autoencodage les auto-encodeurs over-complete et les auto-encodeurs under-complete selon la contrainte supplémentaire imposée pour limiter la capacité de représentation de l'autoencodeur lors de l'optimisation de la fonction objectif. Dans le cas d'un auto-encodeur over-complete, l'espace latent a une taille plus grande que les données d'entrée et de sortie ce type d'autoencodage ajoutent un terme de régulation dans la fonction objective. Un auto-encodeur under-complete désigne une architecture dont l'espace latent présente une réduction de dimension.

A- Autoencodeurs épars

Les autoencodeurs épars utilisent une méthode alternative pour introduire le goulot d'étranglement sans nécessiter une réduction du nombre de nœuds au niveau des couches cachées. Plus précisément, il s'agit de construire une fonction objective en pénalisant les activations dans une couche compte tenu de l'observation en question. Donc le réseau de neurones effectue l'encodage et le décodage en se fondant sur l'activation d'un certain nombre de neurones. Une des stratégies consiste à ajouter un terme supplémentaire dans la fonction objectif pendant l'entraînement afin de pénaliser la divergence de Kullback-Leibler entre les marginaux des unités cachées $\widehat{p}_j$ et un taux de parcimonie souhaité ρ :

$$J_{AE}(\theta_i\psi) = \sum_{i=1}^{n} \mathcal{L}(x_i, \hat{x}_i) + \lambda \sum_{j=1}^{m} kL(\hat{p}_j, p) \qquad (5)$$

Où $\mathcal{L}$ étant une fonction de coût permettant de mesurer la divergence entre l'échantillon d'apprentissage en entrée et les données reconstruites, $\hat{p}_j = \sum_{i=0}^{n} z_j(x_i)$ représente également l'activation moyenne de l'unité cachée j pour l'entrée xi sur la distribution de données et z_j est l'activation de l'unité cachée j.

$\mathrm{k.L}(\hat{\mathrm{p}}_\mathrm{j}, \mathrm{p}) = \mathrm{p}\left[\log \frac{\mathrm{p}}{\mathrm{p}_\mathrm{j}}\right] + (1-\mathrm{p})\left[\log \frac{1-\mathrm{p}}{1-\mathrm{p}_\mathrm{j}}\right]$ la divergence de Kullback-Leibler entre un paramètre de rareté p et son approximation $\hat{p}_j$. Une autre approche consiste à ajouter dans la fonction objective unterme de régularisation qui pénalise la norme L_1 du code latent pour toute entrée x donnée. Dans ce cas, la fonction objective sera :

$$J_{AL} = (\theta, \psi) = \sum_{i=1}^{n} \mathcal{L}\left(x_i, \hat{x}_i\right) + \lambda \|z\|_1 \qquad (6)$$

$\mathcal{L}$Étant une fonction de coût permettant de mesurer la divergence entre l'échantillon d'apprentissage en entrée et les données reconstruites, L_1 est la norme des fonctions à valeurs dans $\mathbb{R}$ dont la valeur absolue est intégrable au sens de Lebesgue

B - Autoencodeurs débruteurs

Une autre façon de limiter la capacité de représentation d'un autoencodeur consiste à ajouter du bruit aux données d'entrée pendant l'entraînement afin d'avoir le plus possible des sorties proches des entrées. L'autoencodeur est entraîné pour reconstruire son entrée x à partir de sa version corrompue $\tilde{x}$ $\mathcal{L}$ étant une fonction de coût permettant de mesurer la divergence entre l'échantillon d'apprentissage en entrée et les données reconstruites.

C - Auto encodeurs contractifs :

L'auto-encodeur contractif est un auto encodeur dans lequel on pénalise la norme de Frobenius de la matrice Jacobienne des activations de l'encodeur par rapport à l'entrée en ajoutant un terme supplémentaire dans la fonction objectif afin d'apprendre des représentations utiles :

$$J_{AE}(\theta, \psi) = \sum_{i=1}^{n} \mathcal{L}\left(x_i, \hat{x}_i\right) + \lambda \sum_{i=1}^{m} \|\nabla_x z_i\|_F^2$$

$\mathcal{L}$Étant une fonction de coût permettant de mesurer la divergence entre l'échantillon d'apprentissage en entrée et les données reconstruites, z_i l'activation de l'unité cachée i et m est le nombre d'unités cachées.

La norme de Frobenius est définie comme suit :

$\|A\|_F = tr(A * A)$ avec A $\in M_{min}$ A^* la matrice adjointe de A.

D - Auto-encodeur variationnel

Le modèle d'auto-encodeur variationnel hérite de l'architecture de l'auto-encodeur, mais fait des hypothèses fortes concernant la distribution des variables latentes. Il utilise l'approche variationnelle pour l'apprentissage de la représentation latente, ce qui se traduit par une composante de perte additionnelle et un algorithme d'apprentissage spécifique fondé sur un estimateur bayésien variationnel du gradient stochastique [3]

4 Algorithme de L’auto encodeur :

L'algorithme d'apprentissage d'un auto-encodeur peut être résumé comme suit : Pour chaque entrée x',Effectuer un passage vers l'avant afin de calculer les activations sur toutes les couches cachées, puis sur la couche de sortie pour obtenir une sortie x',Mesurer l'écart entre x' et l'entrée x, généralement en utilisant l'erreur quadratique, Rétropropager l'erreur vers 'arrière et effectuer une mise à jour des poids.

Un auto-encodeur est bien souvent entrainé en utilisant l'une des nombreuses variantes de la rétropropagation, e.g., méthode du gradient conjugué, algorithme du gradient.

5Utilisation d'un auto encodeur pour la classification :

Les réseaux neuronaux à couches cachées multiples peuvent être utilisés pour résoudre des problèmes complexes de classification de données, tels que la classification d'images de lettres, ou même de petites images d'objets d'une catégorie spécifique ou d'autres bases de données décrite directement avec des attributs tel que la base iris . Chaque couche peut apprendre des caractéristiques à différents niveaux d'abstraction. Cependant, l'entraînement d'un réseau neuronal à couches cachées multiples peut s'avérer difficile dans la pratique. Un moyen efficace d'entraîner des réseaux neuronaux multicouches consiste à entraîner une couche à la fois. Pour ce faire, il suffit d'entraîner un réseau spécial appelé autoencodeur pour chaque couche cachée .[4]

CHAPITER4 :

Application

1- travail effectuer :

On a appliquer deux modèles de calssifieur à bas de réseaux de neurone un MLP en premier lieu et un auto encodeur(Auto-codeurs empilés ou Stacked Autoencoders en englais) en second lieu pour la classification de la célèbre base de données IRIS sous Matlab, et on a comparer le tau de classification des deux modèles en terme de precision globale de classification et cela en faisant varier le pourcentage d'échantillon de la base de d'apprentissage et de test selon le partage suivant:

Train / test %10/90 20/80 30/70 40/60 50/50 60/40 70/30 80/20 90/10

présentation de la base de donnée utilisé :

2 - La base de données iris de Fisher appelée en anglais iris Flower data set ou fisher's iris data est :

Présenté en 1936 par Ronald Fisher dans son article « The use of multiple measurements in taxonomic problems » est l'une des bases de données la plus populaire et la plus connue dans le domaine de la classification par les réseaux de neurones et autoencodeur IRIS data set comprend trois classes, chaque classe contient 50 objects, chaqu 'un de ces classes se référé à un type de fleur iris La liste des attributs présentes dans la base IRIS peut être décrite comme catégorielle, nominale et continue.

Les experts ont mentionné qu'il n'y avait aucune valeur manquante dans aucun attribut de cet ensemble de données. L'ensemble de données est complet. La première des classes se distingue linéairement des deux autres, les deux secondes n'étant pas linéairement séparables l'une de l'autre, chaque fleur parmi les 150 est caractérisées par quatre attributs numériques figure comme suit :

- Longueur sépale
- Largeur sépale
- Longueur des pétales
- Largeur des pétales

On se rend compte que la largeur des pétales est toujours inférieure à la longueur des pétales et la largeur des sépales également inférieure à la longueur des sépales

Caractéristiques	La taille minimale de la caractéristiques	La taille maximale de la caractéristique
Longue de sépale	4,3	7,9
Largeur de sépale	2,0	4,4
Longueur de sépale	1,0	6,9
Longueur de pétale	0,1	2,5

Table 01 - Information générales sur les caractéristiques d'entrée

Figure01 - Les trois types de fleur iris

Le cinquième attribut est l'attribut prédictif qui est l'attribut de classe qui signifie le nom de classe d'identification, dont chacun est l'un des suivants : IRIS Setosa, IRIS Versicolour ou IRIS Virginica,

3logiciels utlisé

Nous avons utilisé le logiciel Matlab qui est un langage de programmation et de simulation de haut niveau crée par CleveMoler à la fin des années 1970, et développé par la société the mathworks, sa première version étaient en 1984, il fournit des constructions qui permettent une programmation claire à petite et à grande échelle.

Ce logiciel, prend en charge de multiples paradigmes de programmation, y compris orientés objet, impératifs, fonctionnels et procéduraux,Il permet l'utilisation d'un ensemble de fonctions prédéfinies appelées boîtes à outils et des fonctions et programmes définis par l'utilisateur. MATLAB Neural Network Toolbox fournit des outils pour la conception, la mise en œuvre, la visualisation et la simulation de réseaux de neurones et le autoencodeur pour diverses applications.

Figure 02 - logo du Matlab

4 Configuration des données :

La base de données iris est sauvegardée dans le fichier Matlab sous irisdata.mat, le chargement du fichier se faire par l'utilisation de la commande loadfisheriris, qui permet de charger toute les entrées et les sorties de la base de données comme suit :

4.1 Meas 150×4 :150 lignes par 4 colonnes chaque ligne contient 4 attributs qui représentent les dimensions d'une fleur iris.

4.2 Species$\mathbf{150} \times \mathbf{1}$:150 lignes par une seul colonne divisé par trois classes de fleur identique, contient le nom de fleur iris setosa, iris versicolor et iris virginica.

5 Les ensembles de création d'un modèle de classification supervisé :

La création d'un modèle de classification superviseur provient généralement de Plusieurs ensembles de données. En particulier, deux ensembles sont couramment Recommandés, training set et test set, chaque un de ces ensembles joue un rôle important pour l'obtention d'un modèle de classification finale, apte à prédire des résultats les plus juste possible, c'est deux ensembles sont décrits comme suit :

5.1 Training set :

Celui-ci va absorbe la grande quantité de données. En effet, c'est sur cette ensemble que le réseau va itérer pour adapter les paramètres des réseaux (poids et biais) et les ajuster au mieux.

5.2 Test set :

Ce dernier a un rôle diffèrent des autres phases, puisqu'il ne servira pas à ajuster les données, En effet, il va avoir pour rôle d'évaluer le réseau sous sa forme finale, et de voir comment il arrive à prédire[1]

6 - Description de l'architecture des deux classifieurs :

6.1 Le classifieur MLP :

- Pour notre MLP on a choisi d'utiliser une couche d'entrée deux couches cachées de taille 5 et 25 ayant pour fonction la même fonction d'activation logsig et une couche de sortie linéaire pour la classification produisant un vecteur 3∗ 1 correspondant aux trois classes définies comme suit :

- Le vecteur de sortie de la classe Setosa représenté par : [1, 0,0]'
- Le vecteur de sortie de la classe Versicolor représenté par : [0, 1,0]'
- Le vecteur de sortie de la classe Virginica représenté par : [0, 0,1]'.

Nous utilisons pour crée notre MLP figure (88) la fonction de Matlab newff en utilisant la fonction d'entraînement trainlm ,cette fonction met à jour les états de poids et de biais selon l'optimisation de l'algorithme de rétropropagation de Levenberg-Marquardt qui est souvent l'algorithme de rétropropagation le plus rapide.

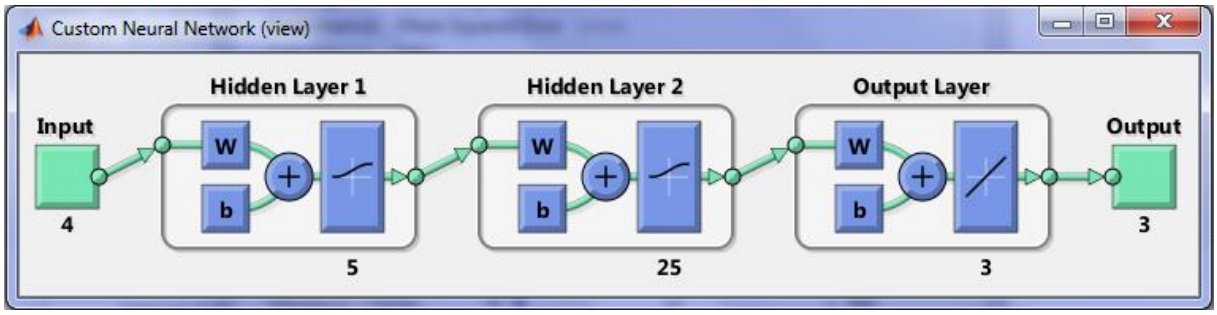


Figure03 : fonctionnement de PML dans Matlab

6.2 Le classifieur Autoencodeur (stacked autoencodor) :

pour notre application nous entraînons un réseau neuronal à deux couches cachées de taille 20 sur la base de données IRIS. Nous utilisons tout d'abord l'autoencodeur pour entraîner les deux couches cachées (couches d'encodages) séparément, de manière non supervisée.

ensuite, nous entraînons la dernière couche de classification softmax de taille 3 correspondant

Aux trois classes définies comme suit :

- Le vecteur de sortie de la classe Setosa représenté par : [1, 0,0]'
- Le vecteur de sortie de la classe Versicolor représenté par : [0, 1,0]'
- Le vecteur de sortie de la classe Virginica représenté par : [0, 0,1,]'.

Avec la crossentropy (mesure de différence entre deux probabilités de distribution pour une variable prise aléatoirement) comme critère ou cout de minimisation et connectons ces couches ensemble pour former un réseau empiléd'où le nom Autoencodeurs empilés ou stacked autoencodors, qui est entraîné pour ladernière fois sous

supervision, notre classifieur est représenter sur la figure(fff) suivante :

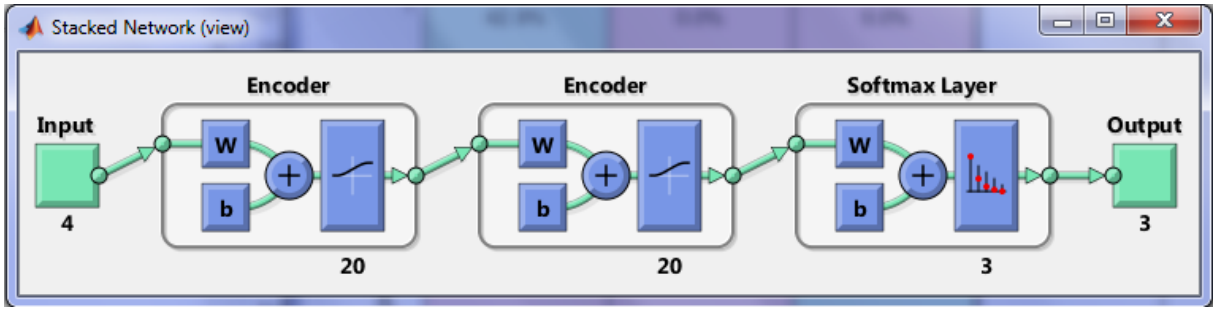


figure 04 : fonctionnement de l'aotoencodeur dans Matlab

7 Résultats obtenus :

Tout d'abord soit la précision globale pour les deux échantillons d'apprentissage et de test suivant :

PGA : précision globale d'apprentissage

PGT : précision globale de teste

7.1 Résultats obtenus pour le MLP :

Le tableau et la courbe figure 05 si dessous montre l'influence du nombre d'échantillon d'apprentissage et de test sur la précision d'apprentissage et de test pour la méthode de Classification par un MLP :

Train /test %	10/90	20/80	30/70	40/60	50/50	60/40	70/30	80/20	90/10
PGA	100	100	95.56	98.33	96	98.89	96.19	99.17	99.26
PGT	90.37	87.5	96.19	96.67	98.67	93.33	95.56	93.1	100

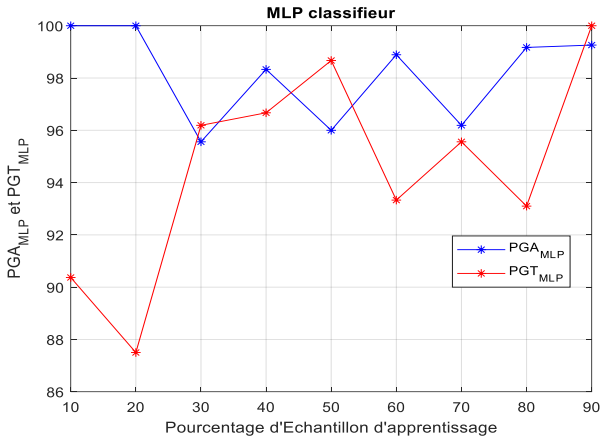


Figure 05 : PGA ET PGT D'UN MLP

7.2 Résultat obtenus pour L'Autoencodeur :

Le tableau et la courbe figure 05 si dessous montre l'influence du nombre d'échantillon d'apprentissage et de teste sur la précision d'apprentissage et de teste pour la méthode de classification par un Autoencodeur :

Train /test %	10/90	20/80	30/70	40/60	50/50	60/40	70/30	80/20	90/10
PGA	100	100	100	100	100	100	100	100	100
PGT	96.3	96.7	92.4	92.2	93.3	95	95.6	86.2	100

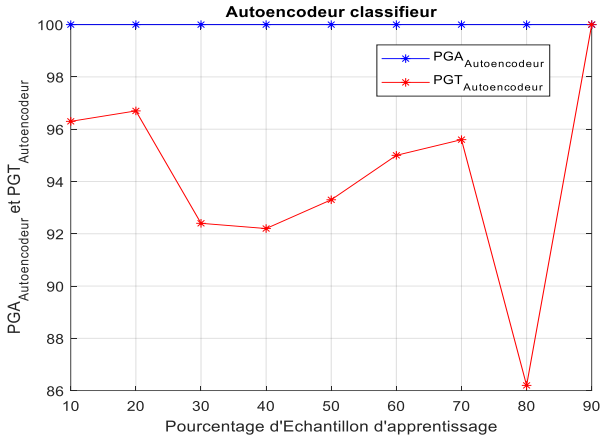


Figure 06 : PGA ET PGT D'UN AUTOENCODEUR

7.3 Discussion des résultats et conclusion :

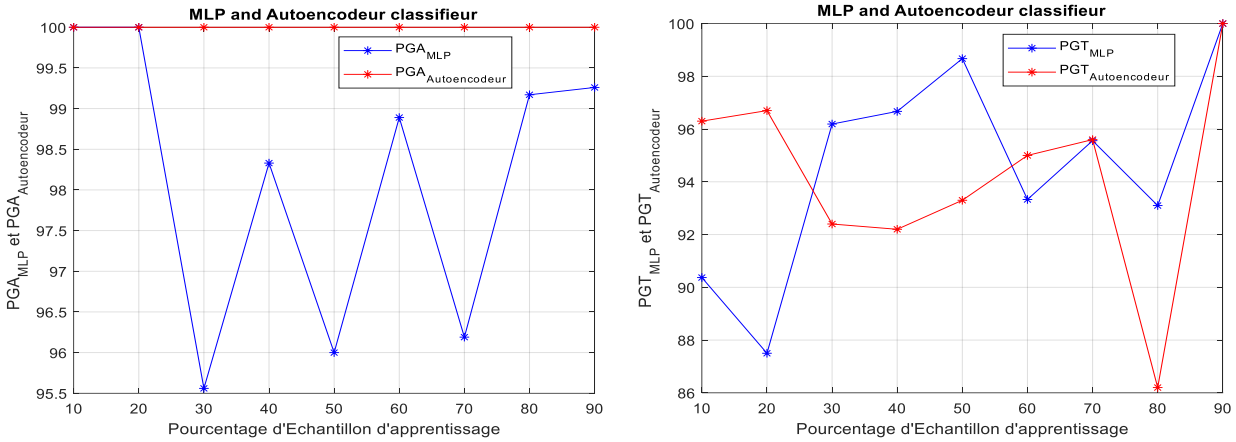
Discussion des résultats :

Vue les résultats obtenus pour les deux classifieurs il n'a pas vraiment de corrélation entre le nombre d'échantillons (apprentissage et test) utiliser par les deux classifieur et la précision globale de classification sois pour l'échantillon d'apprentissage ou l 'échantillon de test par exemple pour le classifieur MLP le PGA est maximale pour un choix Train /test de 10/90 et 20/80 puis redescend pour 30/70 remonte pour 40/60 puis redescend pour 50/50 la même chose pour le PGT .pour le classifieur Autoencodeur en remarque que quelque soit le choix Train /test le PGA est maximal 100% et constant quant au PGT il crois et décrois en permanence cela est dus peut être au problème d'initialisation aléatoire des poids et des biais qui sont différent à chaque fois ,cela peut être aussi dus à un mauvais choix des tailles des réseaux pour les deux classifieur ce qui à causé un sur apprentissage des deux classifieurs ,aussi un seul essais est insuffisant pour certifier qu'il y'a ou qu'il n'ya vraiment pas de corrélation entre le nombre d'échantillon la précision de classification on aurais dus refaire l'expérience pour chaque échantillons plusieurs fois (exemple une dizaine de fois) et prendre la moyenne

Conclusion :

On conclut que d'après les deux courbes a et b de la figure 07,qu'on ne peut pas dire qu'un classifieur et meilleur que l'autre en terme de précision globale puisque pour certaine pourcentage d'échantillon de train/test les MLP donne de meilleur résultat pour le PGT que

l'Autoncodeur exemple sur 30/70, 40/60 et 50/50 et pour d'autre l'Autoncodeur l'emporte sur MLP exemple 10/90 20/80 et 60/40 et pour d'autres il sont presque équivalent exemple 70/30 et 90/10 donc en général d'après les résultats obtenus ils sont équivalent oùils sont complémentaires, par contre pour le PGA l'Autoncodeur l'emporte sur le MLP



Courbeb :PGA MLP ET AUTOcourbe (a) : PGT MLP ET AUTO

Figure 07 : comparaison entre les deux méthodes

Conclusion générale :

On conclut que d'après les deux courbes a et b de la figure,qu'on ne peut pas dire qu'un classifieur et meilleur que l'autre en terme de précision globale puisque pour ertaine pourcentage d'échantillon de train/test les MLP donne de meilleur résultat pour le PGT que l'Autoencodeur exemple sur 30/70, 40/60 et 50/50 et pour d'autre l'Autoencodeur l'emporte sur MLP exemple 10/90 20/80 et 60/40 et pour d'autres il sont presque équivalent exemple 70/30 et 90/10 donc en général d'après les résultats obtenus ils sont équivalent outils sont complémentaires, par contre pour le PGA l'Autoencodeur l'emporte sur le MLP

Enfin, nous souhaitons que notre travail soit un apport pour les futurs étudiants qui cherchent à travailler dans ce domaine.

Bibliographie

CHAPITER 1 :

[1]INIVERSITE DE Québec « classification apprentissage profond et réseaux de neurones : application en science des donnes » par JEHN NOEL DIBOCOR DIOUR années 2020

[2]https://fr.wikipedia.org/wiki/Apprentissage_automatique

[3]These, Classification et apprentissage actif à partir d'un flux de données évolutif en présence d'étiquetage incertain. Mohamed-RafikBouguelia.Université de lorraine, janvier 2016

[4]https://arabicprogrammer.com/article/7771120334/

[5]Université 8 Mai 1945 – Guelma « Application des réseaux de neurone pour la classification des Données » par Bourougaa Nour El Islam et Seridi Merouane années 2021

[6]Université Mouloud MAMMERI de Tizi-Ouzou « Implémentation et évaluation des modèles de recommandation basés sur la machine et deep learning » par :BIRI Lydia et GUEMAT Yasmine Promotion : 2019 / 2020

[7] INIVERSITE DE Québec « classification apprentissage profond et réseaux de neurones : application en science des donnes » par JEHN NOEL DIBOCOR DIOUR années 2020

[8]Université 8 Mai 1945 – Guelma « Application des réseaux de neurone pour la classification des Données » par Bourougaa Nour El Islam et Seridi Merouane années 2021

[9]https://www.Fouille-de-donnees-JV-PART2.pdf

[10] https://www./classificationsupervisee.//

[11]https://www.Fouille-de-donnees-JV-PART2.pdf

[12]Université 8 Mai 1945 – Guelma « Application des réseaux de neurone pour la classification des Données » par Bourougaa Nour El Islam et Seridi Merouane années 2021

[13]https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-explo-classif.pdf

CHAPITER 2 :

[1]INIVERSITE DE Québec « classification apprentissage profond et réseaux de neurones : application en science des donnes » par JEHN NOEL DIBOCOR DIOUR années 2020

[2]Université Mouloud MAMMERI de Tizi-Ouzou « Implémentation et évaluation des modèles de recommandation basés sur la machine et deep learning » par :BIRI Lydia et GUEMAT Yasmine Promotion : 2019 / 2020

[3] INIVERSITE Mohamed Sadik ben Yahia de Jijel " Utilisation des réseaux de neurones convolutifs pour la suppression automatique des filigranes » par : Krine Zohra. Abdi Ahlem Années 2020

[4]INIVERSITE DE Québec « classification apprentissage profond et réseaux de neurones : application en science des donnes » par JEHN NOEL DIBOCOR DIOUR années 2020

[5] INIVERSITE Mohamed Sadik ben Yahia de Jijel " Utilisation des réseaux de neurones convolutifs pour la suppression automatique des filigranes » par : Krine Zohra. Abdi Ahlem Années 2020

[6]INIVERSITE DE Québec « classification apprentissage profond et réseaux de neurones : application en science des donnes » par JEHN NOEL DIBOCOR DIOUR années 2020

[7] INIVERSITE Mohamed Sadik ben Yahia de Jijel " Utilisation des réseaux de neurones convolutifs pour la suppression automatique des filigranes » par : Krine Zohra. Abdi Ahlem Années 2020

[8]INIVERSITE Mohamed Sadik ben Yahia de Jijel " Utilisation des réseaux de neurones convolutifs pour la suppression automatique des filigranes » par : Krine Zohra. Abdi Ahlem Années 2020

[9]https://reussirlem1info.files.wordpress.com/2012/05/mlp.pdf

CHAPITER 3:

[1]INIVERSITE DE Québec « classification apprentissage profond et réseaux de neurones : application en science des donnes » par JEHN NOEL DIBOCOR DIOUR années 2020

[2]UNIVERSITÉ DU QUÉBEC « CLASSIFICATION, RÉDUCTION DE DIMENSIONNALITÉ ET RÉSEAUX DE NEURONES : DONNÉES MASSIVES ET SCIENCE DES DONNÉES » PAR Aboubakry Moussa SOW Novembre 2020

[3]https://atcold.github.io/pytorch-Deep-Learning/fr/week07/07-3/

[4]https://fr.wikipedia.org/wiki/Auto-encodeur

CHAPITER 4 :

[1]Université 8 Mai 1945 – Guelma « Application des réseaux de neurone pour la classification des Données » par Bourougaa Nour El Islam et Seridi Merouane années 2021