Université des sciences et des technologies Houari Boumediene
Faculté d'Electronique et d'Informatique
Département Informatique

Master spécialité : Bio-informatique
Module : Data-Mining
Intitulé du projet :

Etude de statut de survie du Cancer de Sein Haberman Breast Cancer (H)

Réalisé par :

- BOUZIDI Abdeldjalil 201504000052
- ZERROUKHI Djillali 201500007450

Année universitaire : 2019/2020

Table des matières

1. Introduction :

L'intelligence artificielle est une branche de l'informatique qui met en œuvre un ensemble de techniques a des machines pour simuler l'intelligence humaine. L'intelligence artificielle a le grand rôle d'évolution qu'on vu aujourd'hui à cause de sa vaste couverture dans différents domaines.

Actuellement, l'intelligence artificielle se déploie fortement dans tous les domaines et précisément dans le domaine médical et en médecine. Ses techniques permettent de lutter contre les cancers et d'accélérer le diagnostic de certaines maladies. Une technique de l'IA permet de passer des modèles existants pour prédire le statut de survie du cancer du sein (Haberman Breast Cancer), cette technique est l'apprentissage profond (Deep Learning) qui s'appuie sur les réseaux de neurones artificiels.

2. Problématique :

La problématique qui se pose à ce moment est comment générer un model capable à apprendre les relations dans le jeu de données du cancer de sein Haberman et de se généraliser pour la prédiction ?

3. Objectif :

L'objectif de ce projet est de réaliser une application capable de prédire le statut de survie d'un patient atteint d'un cancer de sain.

4. Solution proposée :

Afin de d'atteindre notre objectif, nous proposons un réseau de neurones profond dédié à l'apprentissage de la classification binaire de statut de survie des patients.

Nous allons concevoir une IHM qui permettra de faciliter l'utilisation de l'application par un médecin.

5. Démarches de la construction de la solution à base de réseaux de neurones :

- ***Présentation des données :***

Le jeu de données constitue de quatre attributs y compris l'attribut classe « Age », « Year », « Number », « Status ». Touts les attributs sont de même type 'numérique'.

- **INPUT :** 3 Attributs
  - Age : age de patient au moment de l'operation ([0 - 100] ans)
  - Year : année de l'operation ([58 – 70] === year-1900)
  - Number : nombre de nœuds positives détectées
- **TARGET :**
  - Status : statut de survie du patient (1 : le patient survécu 5ans ou plus / 2 : le patient est décidé dans les 5ans).

Nous remarquons que les attributs ont des différents domaines de définition. Donc nous avons besoin d'un encodage qui permet aux trois attributs de se varier dans un domaine de définition commun.

- ***Codification des données :***

Dans cette partie, nous allons encoder le jeu de données :

- **TARGET :** nous choisissons l'encodage « 1-of-2 » et donc on aura deux vecteurs

Si la sortie désirée = 1	**1**	**0**
Si la sortie désirée = 2	**0**	**1**

- **INPUT :** afin de mettre les attributs dans la même échèle, nous utilisons la standardisation « z-score » :

$$\mathbf{X'} = \frac{\boldsymbol{X - Mean(Ai)}}{\boldsymbol{Std(Ai)}}$$

- ***Génération des données de train-test (Inputs & Targets) :***

Une fois codifier les données sont enregistrées dans un fichier « habermanEncoded.txt », ensuite nous allons diviser le jeu de données en partie de train et de test.

Taches de cette partie :

- Récupération de Dataset.
- Division de jeu de données en deux paties (70% train, 30% test).
- Affecter les inputs (train/test) et les targets (train/test).

- ***Outils utilisés pour le développement de la solution :***

Pour le développement de la solution, nous avons utilisé le langage Python3.6 et la bibliothèque d'apprentissage automatique « Tensorflow » « Keras ».

Pour l'interface graphique, nous créons des pages html/css puis les connectons avec python à l'aide de la bibliothèque « Flask ».

6. Apprentissage des réseaux de neurones (configuration) :

Les données générées précédemment seront utilisées pour l'apprentissage de notre réseau de neurones. Un vecteur de taille trois en entré (age, year, node) après standardisation, en sortie nous aurons un vecteur de taille deux contient la probabilité [0...1] que la sortie $_{i=0}$ ou $_{i=1}$ soit la sortie désirée.

Afin d'obtenir une architecture en adéquation avec les besoins de notre apprentissage, nous avons adopter une représentation génétique (Bit-String) d'un réseau de neurones quelconque (individu) puis nous avons appliqué l'algorithme génétique pour trouver une population des individus les mieux adaptés(meilleures topologies). Par la suite, nous avons commencé notre apprentissage avec les meilleures paramètres (architecture, fonctions d'activation, etc…).

Couche	H1	H2	H3	H4	H5	H6	Output	MSE	Accuracy
Nombre de neurones Exemple :1	256	128	64	32	16	4	2	0.01	0.98
Nombre de neurones Exemple :2	100	100	100	0	0	0	2	0.19	0.73
Nombre de neurones Exemple :3	300	200	100	200	300	0	2	0.192	0.764

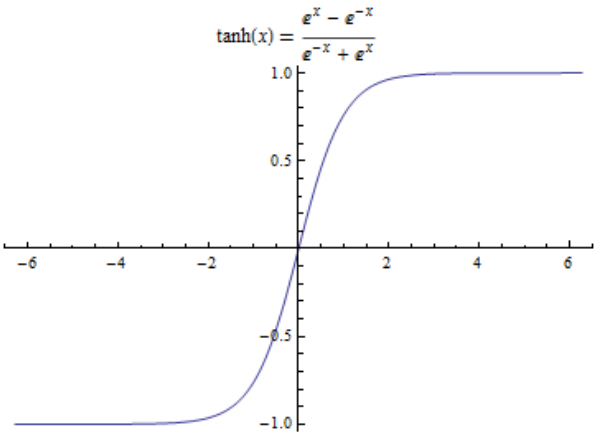
Notre réseau de neurones choisis (Best) est constitué de 7 couches, 6 couches cachées et la couche de sortie. (Exemple : 1).

Pour notre problème, toutes les architectures de la forme ($H_1 > H_2 > \ldots > H_n$) donnent des bons résultats (erreur et régression).

- ***Fonction d'activation :***

Toutes les couches ont la même fonction d'activation : « Tanh » définie par :

$$\mathbf{O(net)} = \frac{e^{-net} - e^{net}}{e^{-net} + e^{net}}$$



La fonction « Tanh » produit un resultat compris entre -1 et 1, elle est préférable à la fonction « Sigmoide » car elle produit des valeurs positives et négative (c.à.d. elle converti les valeurs de très grand nombre négatif en entré en une valeur égale à -1 en sortie).

- ***Fonction d'erreur :***

Nous avons utilisé comme fonction de mesure de l'erreur, la fonction « Mean Squared Erreur » est toujours positive et plus qu'elle converge vers zéro l'algorithme d'apprentissage est plus performant. Elle est définie par :

$$\frac{1}{2}\sum(\boldsymbol{Target - Output})^2$$

- ***Fonction d'apprentissage :***

Pour que notre model apprend, nous avons utilisé la fonction d'apprentissage « Adam ». C'est un algorithme populaire dans le domaine de Deep Learning, car il permet d'obtenir rapidement de bons résultats. Adam est approprié pour les problèmes avec gradient clairsemé très bruyant et c'est le cas pour notre problème.

7. Résultats de l'apprentissage :

Les courbes ci-dessous représente les performances de notre apprentissage (Erreur == Loss) et (Regression == Accuracy) exprimées en fonction de nombre d'itérations.

Ces résultats correspondent à nos désirs et confirment l'efficacité de l'approche utilisée. En effet, l'erreur de « train » est minimale : 0.01 ainsi que celle du « test » qui est égale à 0.0094. L'apprentissage a pu atteindre un très haut niveau de précision à savoir 98.7% sur l'ensemble de données de « train » et 98.4% sur les données de « test ».

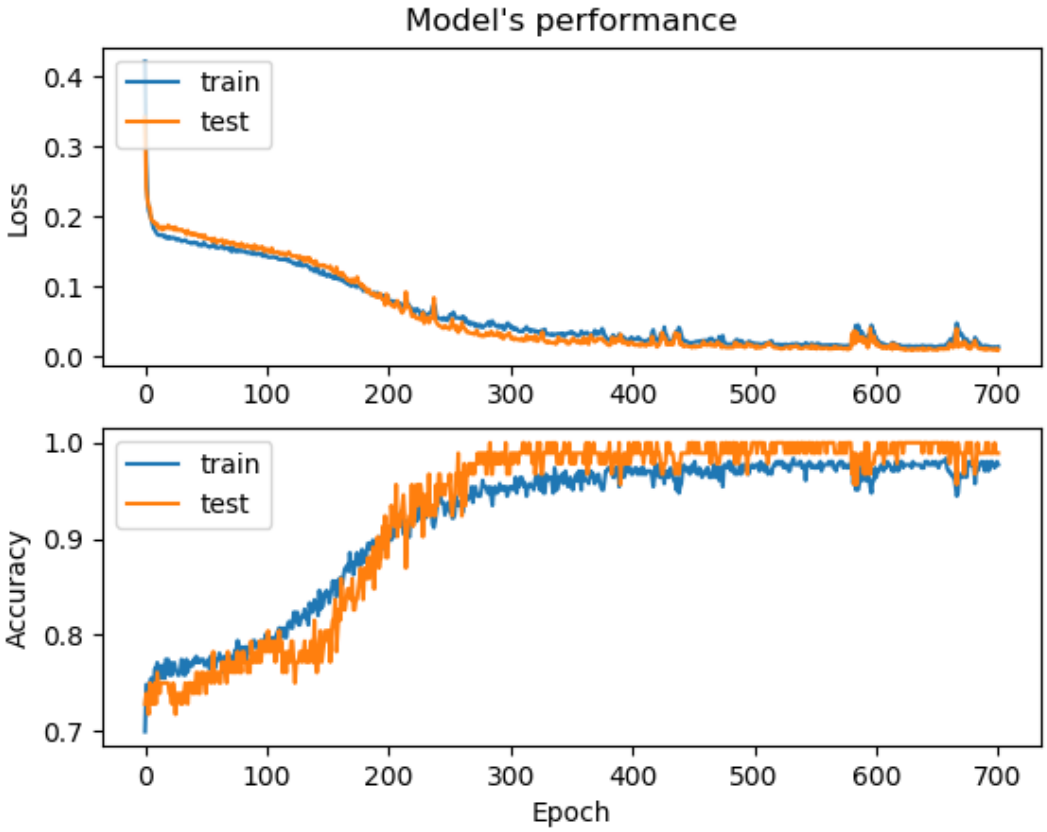


Fig.1.Graphes de performance.

8. Conclusion :

Ce travail nous a conduit à acquérir et approfondir nos connaissances dans l'apprentissage automatique et les réseaux de neurones profonds. La partie critique dans ce projet c'est de bien choisir l'architecture de réseaux de neurones adéquate. Ainsi la variété des paramètres à ajuster nous a posé un problème d'optimisation combinatoire qui a été résolu avec l'utilisation d'une des approches des algorithme évolutifs : « Algorithme Génétique ».

9. Bibliographie et références :

DataMining2 : Cour de réseaux de neurones artificiels – Chapitre3 / Mr. Guessoum

Annexes :

I. *Algorithmes génétiques & réseaux de neurones :*

La performance d'un réseau de neurones dépend de plusieurs facteurs y compris l'architecture, la fonction d'activation, le taux d'apprentissage, etc... L'ajustement de ces paramètres est un problème d'optimisation combinatoire.

L'algorithme génétique est un algorithme de recherche approchée basé sur les mécanismes de la sélection naturelle et de la génétique. L'optimisation de notre problème nécessite une bonne représentation génétique d'individu (réseau de neurones) et une fonction d'évaluation.

Il existe trois opérateurs d'évaluation dans les algorithmes génétiques :

- **La sélection :** chois des individus les mieux adaptés.
- **Le croissement :** mélange par la reproduction de particularités des individus.
- **La mutation :** altération aléatoire des particularités des individus.

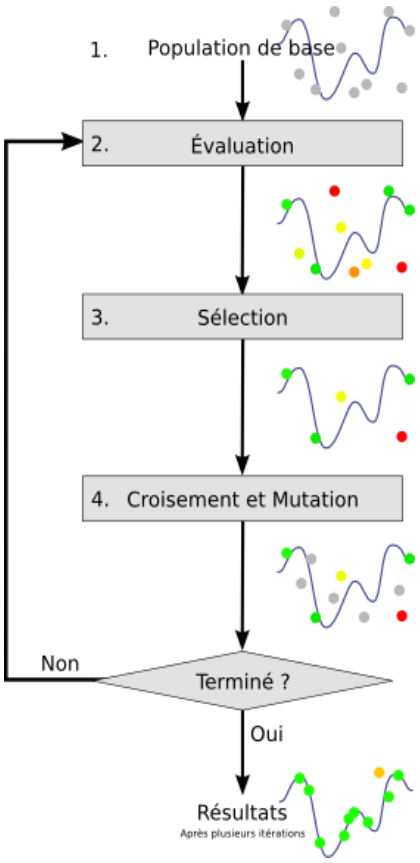


Fig.2.organigrame de l'algorithme Génétique.

- ***Codage d'individus :***

Nous avons proposé une représentation génétique du réseau de neurones où chaque réseau est représenté par un Bit-String (chromosomes) de taille « 55 ».

1 - 8	9 - 16	17 - 24	25 - 32	33 - 40	41 - 48	49 - 51	52 - 53	54 - 55
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXX	XX	XX

- Cette représentation contient six couches cachées au maximum ([1 – 48]), chaque 8bits est un codage binaire de nombre de neurones par couche (ex : 00000000 == 0 unité, cette couche est vide et donc elle ne sera pas empilée).
- Les trois bits [49 - 51] : représentent le taux d'apprentissage (un dictionnaire de 8 valeurs différentes de taux d'apprentissage).
- Les deux bits [52 - 53] : représentent la fonction d'activation (un dictionnaire de 4 valeurs : 'tanh', 'sigmoid', 'elu', 'prelu').
- Les deux derniers bits [54 - 55] : représentent le « slope » de la fonction d'activation (concerne la fonction 'elu' et 'prelu').

- ***Fonction d'évaluation (Fitness) :***

Chaque individu (réseau) doit être évalué avant la sélection pour la reproduction. La fonction objectif à minimiser est « RMSE : Root Mean Equared Error ».

$$\textbf{RMSE} = \sqrt{\frac{\mathbf{1}}{\mathbf{2}}\sum(\boldsymbol{Target} - \boldsymbol{Output})^2}$$

- ***Implémentation et outils :***

Nous avons utilisé le package prédéfini des algorithmes génétiques sous Python « DEAP ».