INSTITUT **S**UPERIEUR **P**OLYTECHNIQUE DE **M**ADAGASCAR

Compte rendu sur le mini projet de
Réseaux de Neurones Artificiels

Parcours : **I**nformatique de **G**estion, **G**énie **L**ogiciel et **I**ntelligence **A**rtificielle **(IGGLIA)**

PREDICTION ET MODELISATION DE SERIES TEMPORELLES AVEC LES RESEAUX DE NEURONES ARTIFICIELS MULTICOUCHES

Présenté par : Monsieur RAMINOMIANDRISOA Harivony (N° 40)

Année Universitaire : 2022-2023

AVANT-PROPOS

L'intelligence Artificielle et les réseaux de neurones artificiels sont des domaines interconnectés au cœur de la révolution technologique. L'IA englobe toutes les technologies visant à créer des systèmes capables de simuler l'intelligence humaine, tandis que les RNA sont une technique spécifique d'IA inspirée du fonctionnement du cerveau humain, utilisée pour résoudre des problèmes complexes.

Dans le cadre du cursus de formation à l'Institut Supérieur Polytechnique de Madagascar (ISPM), et surtout concernant la matière « Réseaux de neurones artificiels », dirigé par le Professeur RABOANARY Rolland, chaque étudiant est tenu de réaliser un mini projet sur la « Prédiction et modélisation de séries temporelles par réseaux de neurones artificiels multicouches ».

REMERCIEMENTS

Je tiens à remercier le bon Dieu, qui nous a permis de terminer ce mini-projet ainsi que les nombreuses personnes qui m'ont aidé tout au long de ce travail.

Je aussi à exprimer mes remerciements aux personnes ci-dessous :

- Le Professeur **RABOANARY Julien Amédée**, Recteur et fondateur de l'Institut Supérieur Polytechnique de Madagascar.
- Le Professeur **RABOANARY Rolland**, Professeur de Réseaux de Neurones Artificiels
- Tout le corps enseignant et administratif de l'I.S.P.M
- Nos familles qui nous ont soutenu par leurs aides que ce soit moralement ou financièrement
- Toutes les personnes ayant contribué de près ou de loin à la réalisation de ce projet.

Liste des figures :

1 -INTRODUCTION

De nos jours, aucun domaine ne se dissocie de l'informatique. Il est devenu indispensable surtout dans le cadre de l'automatisation. En effet, l'Intelligence Artificielle permet de résoudre rapidement et facilement de nombreux problèmes difficiles.

Les réseaux de neurones artificiels sont des algorithmes pour tâches cognitives et qui simulent le traitement de l'information biophysique.

Un réseau quant à lui composé par un ensemble de nœuds (ou unité) connecté par des liens orientés ou connexion. Les unités des réseaux multicouches sont organisées en couches. Elles sont connectées aux unités du niveau suivant et seulement à celle-là.

Un seul neurone ne peut effectuer que des tâches simples comme les fonctions OR ou AND par exemple d'où l'utilisation de plusieurs en couches pour des problèmes plus complexes.

L'application des réseaux multicouches à propagation de l'information vers l'avant à la prédiction de séries temporelles consiste à prédire l'évolution future d'un système à partir des mesures passées faite sur celle-ci.

La série de Hénon est utilisée pour analyser des données temporelles. Elle est surtout utilisée dans la finance, la météorologie, …

Elle est obtenue en générant les relations de récurrence suivantes :

$$x_{n+1} = y_n + 1 - ax_n{}^2$$

$$y_{n+1} = bx_n$$

Il dépend de deux paramètres, a et b, qui ont ici pour valeurs :

a = 1,4

b = 0,3.

Pour ces valeurs, l'attracteur de Hénon est chaotique. Pour d'autres valeurs de a et b, il peut être chaotique, intermittent ou converger vers une orbite périodique. Un aperçu du comportement de l'attracteur peut être donné par son diagramme orbital.

L'attracteur fut nommé par Michel Hénon, un astronome français en 1976. Il est caractérisé par sa structure fractale et sa sensibilité aux conditions initiales, ce qui signifie que de petites variations initiales peuvent conduire à des trajectoires très différentes au fil du temps. C'est aussi un exemple important dans l'étude des systèmes dynamiques non linéaires et du chaos.

2 - METHODOLOGIE et ALGORITHME

2.1 - Takens :

Algorithme de TAKENS : Il s'agit d'un algorithme qui fait appel au système dynamique en Physique pour déterminer la dimension de plongement de l'espace de phase reconstruit. Cependant, il est aussi capable de donner le nombre d'unités dans la couche d'entrées d'un réseau de neurone.

2.2 -Présentation de l'algorithme de TAKENS :

• Soit une suite de données (série temporelle) obtenus à des intervalles de temps identiques :

U1, U2, …, ui, …, Un, …

• Construire une séquence de vecteur à partir de cette série.

$X_i = [U_i, U_{i+\tau}, U_{i+2\tau}, \ldots, U_{i+n\tau}]$
τ : paramètre de délais $Prendre\ \boldsymbol{n}\ assez\ grand$

• Définir la matrice de covariance :

$\Theta < (i)\ x\ (i)\ t >$: matrice carrée (n, n)

• Déterminer les vecteurs propres ainsi que les valeurs propres $\boldsymbol{\lambda}$ de $\boldsymbol{\theta}$ que l'on range par ordre décroissant. (Algorithme de Jacobi).

• L'erreur d'approximation moyenne est :

$$\varepsilon I = \sqrt{\lambda i + 1}$$

• Tracer εI en fonction de *I*.

• La 1ère valeur de *I* correspondant au 1er plateau de la courbe donne la dimension de plongement de l'espace de phase reconstruit et qui est aussi égale au nombre d'unités d'entrées du réseau de neurone.

2.3 - Algorithme d'apprentissage par descente de gradient pour réseau multicouche

La phase d'apprentissage permet d'obtenir les valeurs optimales des poids qui relient les divers nœuds entre la couche d'entrée et la couche cachée, puis, la couche cachée et la couche de sortie. Dans notre cas, nous allons utiliser l'algorithme d'apprentissage par descente de gradient pour réseau multicouche.

Algorithme

-On fait entrer un prototype à la fois.

-On considère un réseau à M couches.

-Notation :

V_i^M : Sortie de la i^{ème} unité dans la mi^{ème} couche.

m =1,2,3,…,M avec 1 : couche d'entrée et M : couche de sortie.

W_{ij}^m : Poids de la connexion de V_1^m à V_i^i

Le procédé de propagation est :

Etape1 : Initialiser les poids à de petites valeurs aléatoires.

Etape 2 : Choisir un prototype ζ_k^M et l'appliquer à la couche d'entrée (m=1). C'est-à-dire V_k^{1m}={ζ_k^M} pour tout k.

Etape 3 : Propager le signal vers l'avant à travers le réseau en utilisant : V_i^m=g(h_i^m)=g($\sum_j W_{ij}^m V_j^{m-1}$) pour chaque i et m jusqu'à ce que les sorties finales V_i^M aient été toutes calculées.

Etape 4 : Calculer les deltas pour la couche de sortie : δ_i^m=g'(h_i^M)[ζ_i^M –V$_i^M$–] en comparant les sorties actuelles V_i^M avec les sorties désirées ζ_i^M pour le prototype M désiré.

Etape 5 : Calculer les deltas pour les couches précédentes en propageant les erreurs vers l'arrière : δ_i^{m-1}=g'(h$_i^{m-1}$) $\sum_j W_{ij}^m \delta_j^m$ pour m=M,M-1,…,3 jusqu'à ce qu'un delta ait été calculé pour chaque unité.

Etape 6 : Utiliser ΔW_{ij}^m=$\eta\delta_i^m V_j^{m-1}$ pour mettre à jour toutes les connexions suivantes :
W_{ij}^{new}=W_{ij}^{old}+ΔW

Etape 7 : Retourner à l'étape 2 et répéter le processus pour les autres prototyp

2.4 - Méthode de Jacobi

Définition :

Précédemment, nous avons vu qu'il est essentiel de trouver une méthode pratique et efficace pour trouver les valeurs propres et les vecteurs propres.

En effet, dès que la dimension de la matrice est supérieure ou égale à 3, il n'y a plus de méthode efficace pour chercher les valeurs propres à partir des polynômes caractéristiques
qui est **P (λ)=det (A-λI),** car le degré de ce polynôme est égal à la dimension de la matrice.

La méthode de Jacobi permet de calculer efficacement les valeurs propres et les vecteurs propres sous l'hypothèse que la matrice soit symétrique et définie positive.

2.4.1 - L'algorithme :

Matrice de rotation

On appelle matrice de rotation en dimension 2 toutes matrices de la forme :

$$R = \begin{pmatrix} a & -b \\ b & a \end{pmatrix}$$

Avec $a^2+b^2 = 1$.

Un exemple particulier qui vérifie cette matrice :

$$R^{-1}(\alpha) = \begin{pmatrix} cos\,\alpha & sin\,\alpha \\ -sin\,\alpha & -cos\,\alpha \end{pmatrix}$$

L'intérêt de la matrice de rotation est que le calcul de l'inverse se fait de manière très rapide selon le calcul ci-dessous :

$$R^{-1}(\alpha) = \begin{pmatrix} cos\,\alpha & -sin\,\alpha \\ sin\,\alpha & -cos\,\alpha \end{pmatrix}$$

En dimension n, la forme générale d'une matrice de rotation est la suivante :

$$\begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} \overset{\begin{array}{ccccccc} & & i & & j & & \end{array}}{\begin{pmatrix} 1 & & 0 & & & 0 \\ & cos\alpha & \dots & -sin\alpha & & \\ & \vdots & \ddots & \vdots & & \\ & sin\alpha & \dots & cos\alpha & & \\ 0 & \dots & 0 & \dots & 1 & \end{pmatrix}}$$

Or une matrice de rotation est une matrice orthogonale et on sait que si P est une matrice orthogonale, et si A est une matrice symétrique donnée, alors la matrice A1=PAP-1 et A sont des matrices similaires ou semblables, c'est-à-dire A1 et A ont exactement les mêmes valeurs propres.

Pour trouver les valeurs propres, nous allons générer une suite de matrices de la manière suivante :

A : Matrice originale

$A1=P0\ A\ P0^{-1}$
$A2=P1\ A1\ P0^{-1}$
…
$An=Pn\text{-}1\ An\text{-}1\ Pn\text{-}1^{-1}$

Or, nous allons construire les matrices Pi comme étant des matrices de rotation, donc une matrice orthogonale ; par conséquent, Les matrices A, A1, …, Ai+1 sont des matrices similaires. Donc, ils ont les mêmes valeurs propres.

La construction des matrices Pi doit se faire tel qu'à chaque itération, on génère des 0 en dehors de la diagonale. Théoriquement, lorsque i tends vers l'infini, on obtient une matrice diagonale et les éléments de la diagonale sont exactement les valeurs propres de A.

Pratiquement, on arrête l'itération lorsque les éléments en dehors de la diagonale sont nuls ou presque nuls.

Construction de la matrice P Supposons qu'à la i-ème itération, on a : Ai+1=*Pi Ai* P_i^t. Pour pouvoir éliminer les éléments en dehors de la diagonale :

Si $a_{jj}^{(i)} \neq a_{kk}^{(i)}$

$$(P_i)_{jj}=(P_i)_{kk}=\frac{1}{2}\left(1+\frac{b}{\sqrt{c^2+b^2}}\right)$$

Si $a_{jj}^{(i)} = a_{kk}^{(i)}$

$P_i)_{jk}$

$(P_i)_{jj}=(P_i)_{kk}=\frac{\sqrt{2}}{2}$

$_{jj}\text{-}\ a^{(i)}_{kk}))$ et $b=|a_{jj}-a_{kk}|$

$(P_i)_{kj}=-(P_i)_{jk}=\frac{\sqrt{2}}{2}$

Le choix des indices j et k doit correspondre à l'élément de $a^{(i)}{}_{jk}$ ayant la plus grande valeur absolue. À chaque itération, un algorithme de recherche du maximum en valeur absolue parmi les éléments de Ai est nécessaire. Le programme est interrompu lorsque les éléments hors de la diagonale deviennent nuls ou sont inférieurs à une tolérance définie, par exemple, inférieurs à 10^{-3}

Architecture du réseau neuronal optimale :

Concernant l'architecture optimale du réseau neuronal, pour prédire les valeurs du système basé sur les solutions du système de Lorenz, qui représente la série temporelle utilisée pour déterminer cette architecture. En principe, pour résoudre le problème de

prédiction, il est recommandé d'utiliser un réseau à trois couches, structuré comme suit :

- Le nombre d'unités dans la couche d'entrée est déterminé par l'algorithme de Takens.
- Conformément au théorème de Cybenko, une couche cachée est suffisante pour approximer toutes les fonctions continues.
- Pour la prédiction des séries temporelles, une seule unité de sortie est généralement suffisante.
- Le nombre d'unités dans la couche cachée est ajusté en explorant différentes structures de réseau jusqu'à ce que l'on obtienne celle qui minimise l'erreur d'apprentissage.

Détermination du nombre d'unités pour la couche d'entrée

En se basant sur l'algorithme de Takens mentionné précédemment, le nombre d'unités pour la couche d'entrée est défini à 2.

Détermination du nombre d'unités pour la couche cachée

Il est crucial de choisir une structure de réseau capable de prédire avec précision les valeurs de la série temporelle du système de Lorenz. Pour ce faire, un nombre limité d'unités pour la couche cachée est initialement choisi.

Pour chaque valeur de ce nombre, un processus d'apprentissage est effectué sur les valeurs de la série sur un nombre spécifié de périodes, en calculant et en enregistrant l'erreur quadratique normalisée (NMSE) pour chacune de ces périodes. À chaque itération de période, une comparaison de la NMSE est effectuée entre la période actuelle et la période précédente pour décider de l'arrêt de l'apprentissage sur l'architecture respective. Enfin, les valeurs de NMSE de chaque architecture proposée sont comparées pour sélectionner l'architecture de réseau optimale, caractérisée par la plus faible erreur d'apprentissage

3 - ENONCE DU PROBLEME

3.1 - PREDICTION ET MODELISATION DE SERIES TEMPORELLES PAR RESEAUX DE NEURONES ARTIFICIELS MULTICOUCHES

Le série de Héron obtenue en générant les relations de récurrence suivantes :

$$x_{n+1} = y_n + 1 - a x_n^2$$

$$y_{n+1} = b x_n$$

Travail à effectuer :

Prendre : a=1,4 et b=0,3

1. Générer les 500 premières valeurs de x_n et de y_n en prenant $x_0=0$ et $y_0=0$.
2. Tracer y_n en fonction de x_n
3. **Prédictions** :
   Faire les prédictions sur la série formée par les valeurs des X_n.
   a. Déterminer l'architecture optimale du réseau permettant de faire les meilleures prédictions.
   b. Faire l'apprentissage du réseau.
   c. Effectuer des prédictions à un pas. (10 valeurs prédites pour 10 valeurs existantes). Faire figurer sur un même graphe les valeurs prédites et les valeurs attendues.
4. Quelles remarques faites-vous sur l'évolution des résultats dans les deux cas de prédictions ?

Remarques :

1. Ecrire les résultats numériques avec 8 chiffres significatifs et faire figurer les erreurs de prédictions correspondantes.
2. Le travail doit contenir les programmes détaillés utilisés.

4 - OUTILS DE REALISATION

4.1 - C# :

J'ai opté pour le langage C# comme langage de programmation.

Figure 1 : Logo deC# ;
Sources : wikipedia.org

C# étant de base le langage C qui devient C++ en ajoutant les techniques de programmation orientée objet où tout doit être incorporé dans des classes. Puis C++ devient C# en ajoutant les techniques de construction de programmes sur la base de composants avec des propriétés et évènements, ce qui rend le développement de programmes notamment plus facile.

Ces avantages sont que ses types sont précisément conformes à l'architecture .NET, les vérifications de types sont plus élaborées, la fonctionnalité de libération automatique des objets appelée « garbage collector » et le remplacement des pointeurs par des références qui offrent des possibilités identiques, qui offre plus de sûreté tout en ne perdant nullement de performance.

4.2 - Microsoft Visual Studio 2022 :

Figure 2 : Logo de Visual Studio 2022
Sources : wikipedia.org

Microsoft Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications Web ASP.NET, des Services Web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# et Visual J# utilisent tous le même environnement de développement intégré (IDE, Integrated Development Environment), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages.

Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du Framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications Web ASP et de Services Web XML grâce à Visual Web Developer.

5 - INTERFACE ET EXTRAIT CODE

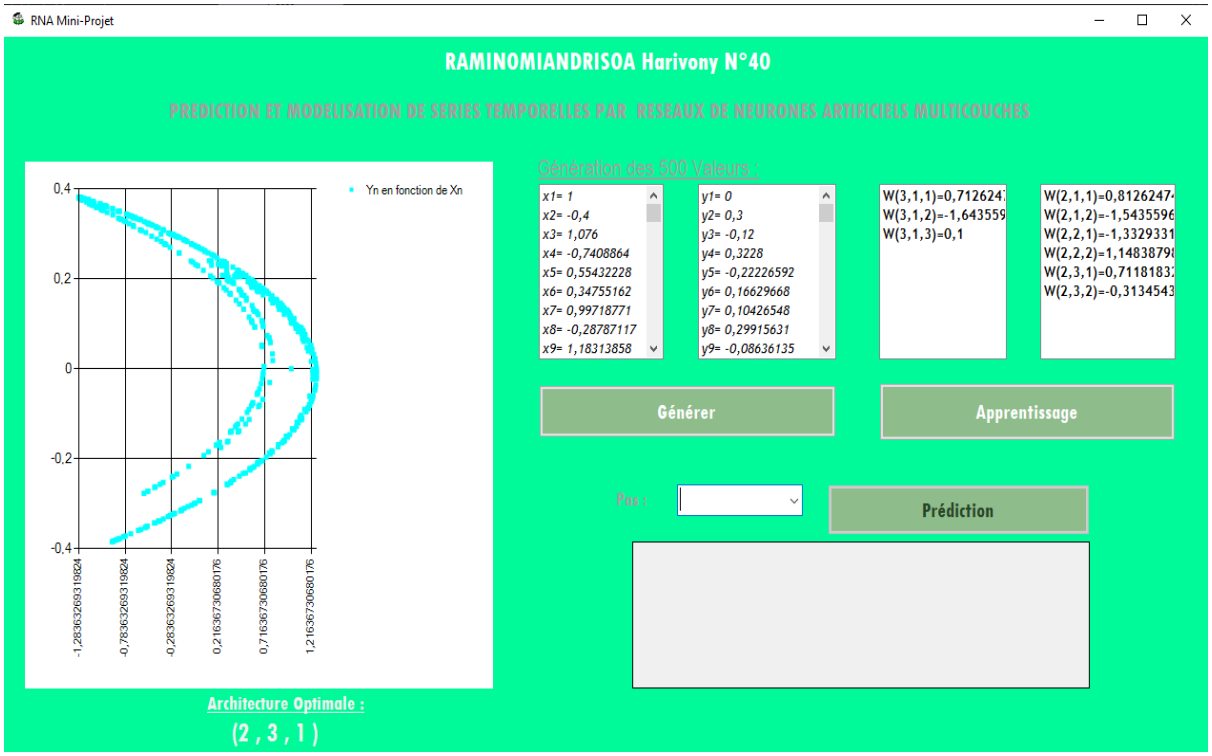

Figure 3 : Interface de l'apprentissage
Source : Auteur

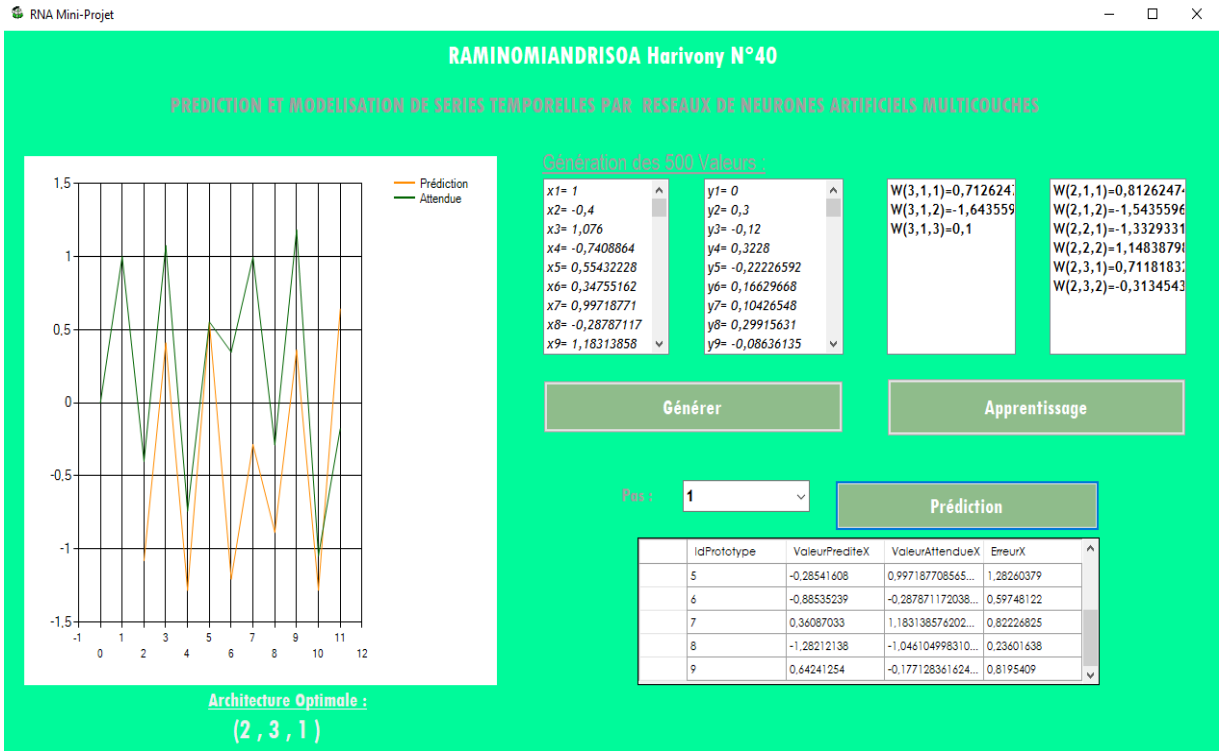

Figure 4 : Interface de la prédiction
Source : Auteur

```
public static double NMSE()
{
    double res, somme = 0;
    double[] xCh = new double[100];
    xCh = Xchapeau();
    for (int i = 0; i < NbPrototype; i++)
    {
        somme += (serieTemporelle[i + NombreUniteEntree] - xCh[i]) * (serieTemporelle[i + NombreUniteEntree] - xCh[i]);
    }
    res = somme / (NbPrototype * Variance(serieTemporelle) * Variance(serieTemporelle));
    return res;
}
private static double Variance(double[] serie)
{
    double variance, moyenne, somme1 = 0, somme2 = 0;
    int n = NbPrototype;
    for (int i = 0; i < n; i++)
    {
        somme1 += serie[i];
    }
    moyenne = somme1 / n;
    for (int i = 0; i < n; i++)
    {
        somme2 += Math.Pow(serie[i], 2);
    }
    variance = (somme2 / n) - (moyenne * moyenne);
    return Math.Round(variance, 8);
}
```

Figure 5 : Extait de code (NMSE)
Source : Auteur

```
//Apprentissage poids caché
for (int k = 1; k <= NombreUniteCachee; k++)
{
    for (int l = 1; l <= NombreUniteEntree; l++)
    {
        DeltaW[2, k, l] = pas * Delta[2, k] * V[1, l];
    }
}
for (int l = 1; l <= NombreUniteCachee; l++)
{
    DeltaW[3, 1, l] = pas * Delta[3, 1] * V[2, l];
}
//Apprentissage poids d'entree
for (int k = 1; k <= NombreUniteCachee; k++)
{
    for (int l = 1; l <= NombreUniteEntree; l++)
    {
        w[2, k, l] += Math.Round(DeltaW[2, k, l], 8);

    }

}
for (int k = 1; k <= NombreUniteCachee; k++)
{
    w[3, 1, k] += Math.Round(DeltaW[2, 1, k], 8);

}
```

Figure 6 : Extait de code (Apprentissage)
Source : Auteur

6- CONCLUSION

On a pu voir que les réseaux de neurones permettent la Prédiction et la modélisation de séries temporelles, et malgré les marges d'erreurs qui sont minimes, on a pu en tirer des prédictions assez précises. Certes, la recherche de l'architecture optimale du réseau est une tâche ardue mais elle en vaut la chandelle.

On a donc pu observer en pratique l'utilisation des réseaux de neurones artificiels ; cependant, avec une meilleure architecture pourrait-on encore améliorer la précision ?

BIBLIOGRAPHIE

Cours données à l'ISPM :

I. **Monsieur RABOANARY Andry Heriniaina :**
   1- Programmation Orientée Objet (2è année)
   2- Algorithme Avancée (4è année)

II. **Monsieur RAKOTOMALALA Manjaka :**
   3- Lanagage C# (3è année)

III. **Professeur RABOANARY Julien Amédée :**
   4- Intelligence Artificielle (3è année et 4è année)

IV. **Professeur RABOANARY Rolland :**
   5- Réseaux de neurones artificiels (4è année)

Webographie :

5- https://thenextweb.com/microsoft/2015/03/19/microsoft-is-open-sourcing-visual-studios-build-tool-msbuild/

6- https://www.brandeps.com/logo/C/C-Sharp-01

7- https://fr.wikipedia.org/wiki/Attracteur_de_H%C3%A9non

Table des matières