"Celui qui deviendra leader en ce domaine sera le maître du monde"

Vladimir Poutine

Table des matières

Table des figures

1 Introduction

Le spam, soit la junk mail ou le pourriel, est une tactique de marketing efficace. Le spam contient généralement de la publicité (Le crédit financier, les casinos en ligne, les montres de contrefaçon, les diplômes falsifiés, les logiciels « craqués » et nombre de superstitions (notamment l'astrologie)). Une autre forme de spam qui est l'hameçonnage consiste à tromper le destinataire en faisant passer un courriel pour un message de sa banque ou d'un quelconque service protégé par mot de passe. Le but est de récupérer les données personnelles des destinataires (notamment des mots de passe, un numéro de carte bancaire) en les attirant sur un site factice enregistrant toutes leurs actions.

Un premier système anti-spam en temps réel est le Real-time Blackhole List (RBL) géré par Paul Vixie et qui consiste à vérifier si l'adresse IP de l'émetteur fait partie de la liste des adresses IP susceptibles d'être un spam. Il est ensuite converti par Eric Ziegast en DNS Black Listing (DNSBL) qui est une méthode permettant de consulter une liste noire d'émetteurs de courrier électronique en utilisant le protocole DNS.

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 (en couches, complets. . .), 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...

Le succès croissant des réseaux de neurones sur la plupart des autres techniques statistiques peut s'attribuer à leurs remarquable faculté à donner un sens, extraire des règles et des tendances à partir de données compliquées, bruitées et imprécises. Ils peuvent s'utiliser pour extraire des modèles et détecter des tendances reposant sur des fonctions mathématiques compliquées qui sont trop difficiles, voire impossible, à modéliser à l'aide de techniques analytiques ou paramétriques traditionnelles. L'une des propriétés intéressantes des réseaux de neurones est qu'ils savent prévoir avec précision des données qui ne faisaient pas partie des données d'apprentissage, un processus

connu sous le nom de généralisation.

La sélection des primitives est une étape importante dans tout système de reconnaissance de formes. Cette sélection des primitives est considérée comme un problème d'optimisation combinatoire et a fait l'objet de recherche dans de nombreuses disciplines. L'objectif principal de la sélection des primitives est de réduire le nombre de celles-ci en éliminant les primitives redondantes et non pertinentes du système de reconnaissance. Le second objectif de cette sélection de primitives est aussi de maintenir et/ou améliorer la performance du classificateur utilisé par le système de reconnaissance. Les algorithmes génétiques sont utilisés pour résoudre ce type de problème de sélection des primitives dans la reconnaissance de chiffres isolés. Les résultats obtenus lors de la sélection des primitives ont permis de réduire la complexité du réseau de neurones utilisé.

Ce Livrable se décompense en 5 grandes parties :

Partie I : Introduction des concepts à Anti-Spam ,RBL (Realtime Blackhole List) , DNSBL (DNS Black Listing) et ANN (Artificial neural networks
Partie II : Représentation des connaissances , où nous expliquerons les métriques choisies
Partie III : Construction et Apprentissage d'un réseaux de neuronnes
Partie IV : Jeu de Test
Partie V : Conclusion

2 Représentation des connaissances

Dans cette partie, nous aurons à formater la donnée brute en entrée [MAILS : Spam et ham confondus] sous un format exploitable et orienté objectif.

2.1 Liste d'imports

2.1.1 import os

Le module OS dans Python fournit un moyen d'utiliser le système d'exploitation dépendant fonctionnalité. Les fonctions que le module OS fournit vous permet d'interfacer avec le système d'exploitation sous-jacent que Python est en cours d'exécution - que Windows, Mac ou Linux. Vous pouvez trouver des informations importantes sur votre emplacement ou sur le processus.

2.1.2 import dns.resolver

dnspython est une boîte à outils DNS pour Python. Il supporte presque tous les types d'enregistrements. Il peut être utilisé pour les requêtes, les transferts de zone et les mises à jour dynamiques. dnspython fournit un accès haut et bas au DNS.

2.1.3 import sys

Ce module donne accès à certaines variables utilisées ou maintenues par l'interpréteur et à des fonctions qui interagissent fortement avec l'interpréteur. C'est toujours disponible.

2.1.4 import socket

Ce module donne accès à l'interface de la prise BSD. Il est disponible sur tous les systèmes Unix, Windows, MacOS et probablement sur des plateformes supplémentaires.

2.2 Ramification du texte brut

```
In [401]: import os
list_train=os.listdir(r'C:\Users\asus\Desktop\Etudes\S4\ProjetAI\train')
list_train
file=r'C:\Users\asus\Desktop\Etudes\S4\ProjetAI\train\jargon.txt'
def tokenize_text(list_train):
    dir=r'C:\Users\asus\Desktop\Etudes\S4\ProjetAI\train'
    Gtoken=[]
    for file in list_train:
        token=[]
        chaine="  "
        f = open(dir+'\\'+file, 'r')
        token.append(file)
        for line in f.readlines():
            if len(token)<=4:
                token.append(line)
            else:
                chaine=chaine+line
        token.append(chaine)
        f.close()
        Gtoken.append(token)
    return Gtoken
p=tokenize_text(list_train)
```

FIGURE 1 – Fonction de Ramification de la donnée brute en Liste de donnée exploitable

La fonction balaie en profondeur le contenu d'un dossier de mails de l'utilsateur. elle stock les noms des fichiers dans une liste. En bouclons sur l'ensemble des fichiers elle crée un token par fichier sous le format : [DATE , FROM , TO , OBJECT , MESSAGE]. La fonction retourne un tableau de liste contenant l'ensemble des tokens résultants.

2.3 RBL et DNSBL : Real-time Blackhole et DNS Black Listing

La liste noire en temps réel SMTP (RBL) est un mécanisme permettant de publier les adresses IP des serveurs SMTP à partir desquels ou via lesquels les spammeurs fonctionnent. Il existe un certain nombre d'organisations qui compilent ces informations à la fois gratuitement http ://www.spamhaus.org, et à but lucratif http ://www.mail-abuse.com

2.4 Liste des Domaines suspects

```
def Listing_RBL(x):
    open("data.txt", "w").close()
    bls = ["zen.spamhaus.org", "spam.abuse.ch", "cbl.abuseat.org", "virbl.dnsbl.bit.nl", "dnsbl.inps.de
    "ix.dnsbl.manitu.net", "dnsbl.sorbs.net", "bl.spamcannibal.org", "bl.spamcop.net",
    "xbl.spamhaus.org", "pbl.spamhaus.org", "dnsbl-1.uceprotect.net", "dnsbl-2.uceprotect.net",
    "dnsbl-3.uceprotect.net", "db.wpbl.info","all.s5h.net","b.barracudacentral.org","bl.emailbasura.org
"bl.spamcannibal.org","bl.spamcop.net","blacklist.woody.ch",
"bogons.cymru.com","cbl.abuseat.org","cdl.anti-spam.org.cn",
"combined.abuse.ch","db.wpbl.info","dnsbl-1.uceprotect.net","wormrbl.imp.ch","xbl.spamhaus.org","z.mail
"zen.spamhaus.org"]
    data = socket.gethostbyname(x)
    myIP =data
    length=len(bls)
    print (myIP)
    for bl in bls:
        try:
            my_resolver = dns.resolver.Resolver()
            query = '.'.join(reversed(str(myIP).split("."))) + "." + bl
            answers = my_resolver.query(query, "A")
            answer_txt = my_resolver.query(query, "TXT" )
            print ('IP: %s IS listed in %s (%s: %s)' %(myIP, bl, answers[0], answer_txt[0]))
        except dns.resolver.NXDOMAIN:
            fichier = open("data.txt", "a+")
            fichier.write('NOT_LISTED \n')
```

FIGURE 2 – Fonction de detection des domaines suspects en se basant sur RBL et DNSBL

```
            fichier.write('NOT_LISTED \n')
        except socket.gaierror:
            print ("")
        except dns.resolver.NoNameservers:
            print ("")
        except dns.resolver.Timeout:
            print ("")
```

FIGURE 3 – Fonction de detection des domaines suspects en se basant sur RBL et DNSB (suite)

2.4.1 Fonctions de Contributions : Métrique dégagée du "FROM :"

```
In [405]: def extract_domain(chaine):
              pos=chaine.find('@')
              pos2=chaine.find('>')
              domaine=chaine[pos+1:pos2]
              return domaine

In [350]: extract_domain("<marie.dupuis@noos.fr>")
Out[350]: 'noos.fr'
```

FIGURE 4 – Fonction d'extraction d'un domaine à partir d'une adresse mail

```
In [403]: def count_file_line():
              f = open("data.txt","r")
              l=[]
              for line in f.readlines():
                  l.append(line)
              f.close()
              return len(l)
```

FIGURE 5 – Compteur de liens NON-LISTANTS le domaine comme étant suspect

2.5 Métrique dégagée du "FROM :"

```
In [404]: def metric_FROM():
              #la creation du fichier data.txt est obligatoire à ce niveau avant de commencer
              j=count_file_line()# on retourne de degree d'inquiétude 1-degre de safety (NON LISTER )
              return 1-j/31
```

FIGURE 6 – Métrique dégagant le degré d'inquiétude de la source d'envoi

Cette fonction vient définir la première primitive de notre réseaux de neuronne, une entrée qui montrera le degré d'inquiétude de la source d'envoie du mail sur la base des RBL.

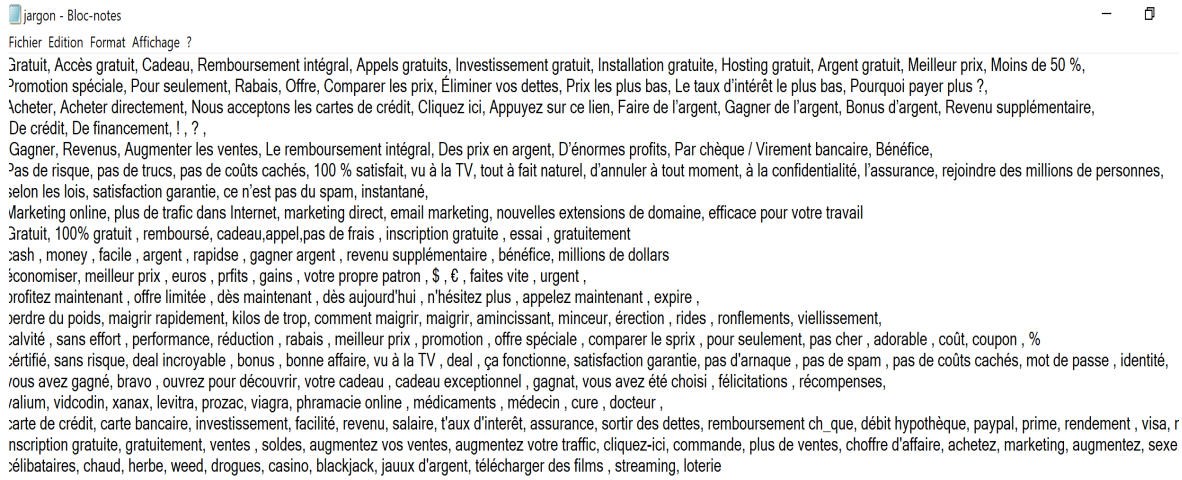
2.6 Métrique dégagée de "OBJECT :" et "SUBJECT :"

```
In [407]: def metric_SUBJECT_MESSAGE(subject,message,file):
              #traitement subject evalué à 30%
              f=open(file,"r")
              text=f.read()
              #text=text.split(',')
              sub=0
              mes=0
              subject=subject.split(' ')
              message=message.split(' ')

              for x in subject:
                  if x in text :
                      sub=sub+1
              for x in message:
                  if x in text :
                      mes=mes+1
              sub=sub/len(subject)
              mes=mes/len(message)
              res=sub*0.300 + mes*0.700
              #traitement subject evalué à 70%

              f.close()
              return res
```

FIGURE 7 – Métrique dégagant les fréquences des mots suspects propres au SPAM

Cette Métrique présente la deuxième primitive de notre réseaux de neuronne, elle admet par subjectivité que l'OBJECT peut être révélateur mais le SUBJECT vient affirmer et appuiyer la présente hypothèse d'être devant un SPAM. Nous établissons ici une mesure de fréquance classique basée sur la recherche dans un dictionnaire , ici appelé " jargon.txt" qui présente les mots les plus récurrents dans la langue française utilisés par les spameurs.

2.7 Jargon Spameurs en langue française


jargon - Bloc-notes

Fichier Edition Format Affichage ?

Gratuit, Accès gratuit, Cadeau, Remboursement intégral, Appels gratuits, Investissement gratuit, Installation gratuite, Hosting gratuit, Argent gratuit, Meilleur prix, Moins de 50 %,
Promotion spéciale, Pour seulement, Rabais, Offre, Comparer les prix, Éliminer vos dettes, Prix les plus bas, Le taux d'intérêt le plus bas, Pourquoi payer plus ?,
Acheter, Acheter directement, Nous acceptons les cartes de crédit, Cliquez ici, Appuyez sur ce lien, Faire de l'argent, Gagner de l'argent, Bonus d'argent, Revenu supplémentaire,
De crédit, De financement, ! , ? ,
Gagner, Revenus, Augmenter les ventes, Le remboursement intégral, Des prix en argent, D'énormes profits, Par chèque / Virement bancaire, Bénéfice,
Pas de risque, pas de trucs, pas de coûts cachés, 100 % satisfait, vu à la TV, tout à fait naturel, d'annuler à tout moment, à la confidentialité, l'assurance, rejoindre des millions de personnes,
selon les lois, satisfaction garantie, ce n'est pas du spam, instantané,
Marketing online, plus de trafic dans Internet, marketing direct, email marketing, nouvelles extensions de domaine, efficace pour votre travail
Gratuit, 100% gratuit , remboursé, cadeau,appel,pas de frais , inscription gratuite , essai , gratuitement
cash , money , facile , argent , rapidse , gagner argent , revenu supplémentaire , bénéfice, millions de dollars
économiser, meilleur prix , euros , prfits , gains , votre propre patron , $, € , faites vite , urgent ,
profitez maintenant , offre limitée , dès maintenant , dès aujourd'hui , n'hésitez plus , appelez maintenant , expire ,
perdre du poids, maigrir rapidement, kilos de trop, comment maigrir, maigrir, amincissant, minceur, érection , rides , ronflements, viellissement,
calvité , sans effort , performance, réduction , rabais , meilleur prix , promotion , offre spéciale , comparer le sprix , pour seulement, pas cher , adorable , coût, coupon , %
certifié, sans risque, deal incroyable , bonus , bonne affaire, vu à la TV , deal , ça fonctionne, satisfaction garantie, pas d'arnaque , pas de spam , pas de coûts cachés, mot de passe , identité,
vous avez gagné, bravo , ouvrez pour découvrir, votre cadeau , cadeau exceptionnel , gagnat, vous avez été choisi , félicitations , récompenses,
valium, vidcodin, xanax, levitra, prozac, viagra, phramacie online , médicaments , médecin , cure , docteur ,
carte de crédit, carte bancaire, investissement, facilité, revenu, salaire, t'aux d'interêt, assurance, sortir des dettes, remboursement ch_que, débit hypothèque, paypal, prime, rendement , visa, r
nscription gratuite, gratuitement, ventes , soldes, augmentez vos ventes, augmentez votre traffic, cliquez-ici, commande, plus de ventes, choffre d'affaire, achetez, marketing, augmentez, sexe
célibataires, chaud, herbe, weed, drogues, casino, blackjack, jauux d'argent, télécharger des films , streaming, loterie

FIGURE 8 – Jargon Spameurs en langue française tout domaines confondus Les différents sujets d'un spam : Accès à des services gratuits(Appels internationales, Remboursement sur des achats), des promotions sur des achats Offre de services qui permettent d'augmenter son revenue Conseils et services de marketing Recommandation et offres de services médicaux

2.8 Quantification de la donnée

```
In [415]: file=r'C:\Users\asus\Desktop\Etudes\S4\ProjetAI\jargon.txt'
          def Quantify(list_train):
              p=tokenize_text(list_train)
              list=[]
              for x in p :
                  item=[]
                  domaine=extract_domain(x[2])
                  Listing_RBL(domaine)
                  item.append(x[0])
                  item.append(metric_FROM())
                  item.append(metric_SUBJECT_MESSAGE(x[4],x[5],file))
                  list.append(item)
              return list

In [416]: print (Quantify(list_train))
```

FIGURE 9 – Fonction de quantification de la donnée

Cette fonction permet de rassembler tout les blocs précédemment réalisé pour arriver à un dataset acceptable du point de vue format à celui demandé en entrée d'un réseaux de neuronne.

```
[['spam1.txt', 0.25806451612903225, 0.17843137254901958], ['spam2.txt', 0.16129032258064513, 0.1506944
4444444444], ['spam3.txt', 0.25806451612903225, 0.28385826771653544], ['spam4.txt', 0.2258064516129032
5, 0.175], ['spam5.txt', 0.25806451612903225, 0.19209302325581395], ['spam6.txt', 0.16129032258064513,
0.12727272727272726]]
```

FIGURE 10 – Résultat de la quantification

Ceci présente le token final qui quantifie notre donnée en entrée du réseaux de neuronnes selon deux axes majeurs : le taux d'inquiétude de la provenance et le taux de mots suspects en Titre et en Contenu.

3 Réseaux de Neuronnes

3.1 Justification du Modèle : Bonne Pratique

Notre Réseaux de Neuronnes (ANN) comporte 2 entrées et une sortie. Le choix du nombre des layers et des nodes cachés est justifés par l'approche de "Jeff Heaton" 'the optimal size of the hidden layer is usually between the size of the input and size of the output layers'.

3.2 Construction du Réseaux de Neuronnes : Approche Orienté objet

In [127]:
```
import numpy as np
from scipy import optimize
class Anti_SPAM_ANN_Approach(object):
    def __init__(self, Lambda=0):
        #Define Hyperparameters
        self.inputLayerSize = 2
        self.outputLayerSize = 1
        self.hiddenLayerSize = 3

        #Weights (parameters)
        self.W1 = np.random.randn(self.inputLayerSize,self.hiddenLayerSize)
        self.W2 = np.random.randn(self.hiddenLayerSize,self.outputLayerSize)

        #Regularization Parameter:
        self.Lambda = Lambda

    def forward(self, X):
        self.z2 = np.dot(X, self.W1)
        self.a2 = self.sigmoid(self.z2)
        self.z3 = np.dot(self.a2, self.W2)
        yHat = self.sigmoid(self.z3)
        return yHat
```

FIGURE 11 – Approche Orienté objet : initialisation de l'objet ANN et Forward pour la propagation des scalaires à travers les layers

```
    def sigmoid(self, z):
        #Apply sigmoid activation function to scalar, vector, or matrix
        return 1/(1+np.exp(-z))

    def sigmoidPrime(self,z):
        #Gradient of sigmoid
        return np.exp(-z)/((1+np.exp(-z))**2)

    def costFunction(self, X, y):
        #Compute cost for given X,y, use weights already stored in class.
        self.yHat = self.forward(X)
        J = 0.5*sum((y-self.yHat)**2)/X.shape[0] + (self.Lambda/2)*(np.sum(self.W1**2)+np.sum(self.W2**
        return J

    def costFunctionPrime(self, X, y):
        #Compute derivative with respect to W and W2 for a given X and y:
        self.yHat = self.forward(X)

        delta3 = np.multiply(-(y-self.yHat), self.sigmoidPrime(self.z3))
        #Add gradient of regularization term:
        dJdW2 = np.dot(self.a2.T, delta3)/X.shape[0] + self.Lambda*self.W2

        delta2 = np.dot(delta3, self.W2.T)*self.sigmoidPrime(self.z2)
        #Add gradient of regularization term:
```

FIGURE 12 – Fonction d'activation et de coût et dérivées mathématiques

3.2.1 fonction sigmoid et dérivée

Elle représente la fonction de répartition de la loi logistique et est souvent utilisée dans les réseaux de neurones parce qu'elle est dérivable ce qui est une contrainte pour l'algorithme de rétropropagation de l'erreur de Werbos qui a permis à la fin des années 80 de créer des réseaux multicouches et de relancer toute la recherche sur les modèles connexionnistes à l'arrêt depuis 1969 ; et parce que la forme de la dérivée de sa fonction inverse est extrêmement simple, et facile à calculer - ce qui améliore les performances des algorithmes.

3.2.2 fonction coût et dérivée

Fonction coût et sa dérivée apporte une amélioration suite à un choix de demystification en gradient

```
        dJdW1 = np.dot(X.T, delta2)/X.shape[0] + self.Lambda*self.W1

        return dJdW1, dJdW2

    #Helper functions for interacting with other methods/classes
    def getParams(self):
        #Get W1 and W2 Rolled into vector:
        params = np.concatenate((self.W1.ravel(), self.W2.ravel()))
        return params

    def setParams(self, params):
        #Set W1 and W2 using single parameter vector:
        W1_start = 0
        W1_end = self.hiddenLayerSize*self.inputLayerSize
        self.W1 = np.reshape(params[W1_start:W1_end], \
                             (self.inputLayerSize, self.hiddenLayerSize))
        W2_end = W1_end + self.hiddenLayerSize*self.outputLayerSize
        self.W2 = np.reshape(params[W1_end:W2_end], \
                             (self.hiddenLayerSize, self.outputLayerSize))

    def computeGradients(self, X, y):
        dJdW1, dJdW2 = self.costFunctionPrime(X, y)
        return np.concatenate((dJdW1.ravel(), dJdW2.ravel()))
```

FIGURE 13 – Calcul du gradient

3.3 Apprentissage du Réseaux de Neuronnes

```
class trainer(object):
    def __init__(self, N):
        #Make Local reference to network:
        self.N = N

    def callbackF(self, params):
        self.N.setParams(params)
        self.J.append(self.N.costFunction(self.X, self.y))

    def costFunctionWrapper(self, params, X, y):
        self.N.setParams(params)
        cost = self.N.costFunction(X, y)
        grad = self.N.computeGradients(X,y)

        return cost, grad
```

FIGURE 14 – Enveloppe sur la fonction coût

```
    def train(self, X, y):
        #Make an internal variable for the callback function:
        self.X = X
        self.y = y

        #Make empty list to store costs:
        self.J = []

        params0 = self.N.getParams()

        options = {'maxiter': 200, 'disp' : True}
        _res = optimize.minimize(self.costFunctionWrapper, params0, jac=True, method='BFGS', \
                                 args=(X, y), options=options, callback=self.callbackF)

        self.N.setParams(_res.x)
        self.optimizationResults = _res
#Train_DATA
X=np.array(([0.16129032258064513, 0.31875], [0.19354838709677424, 0.3322222222222222], [0.1290322580645
Y=np.array(([0],[0],[0],[1],[1],[1],[1],[1],[1]),dtype=float)
```

FIGURE 15 – Optimisation du coût et données d'apprentissage

```
#Test_DATA
XT=np.array(([0.16129032258064513, 0.175],
 [0.12903225806451613, 0.182],
 [0.12903225806451613, 0.23890243902439023]),dtype=float)
YT=np.array(([1],[1],[1]),dtype=float)

#Scaling of Data
X=X/np.amax(X,axis=0)
Y=Y/1
```

FIGURE 16 – Standarisation des données et données de test

3.4 Période d'apprentissage

Ici nous utilisons les fichiers du dossier 'train'et qui représente 2/3 de nos ressources en données.

```
In [128]: NN=Anti_SPAM_ANN_Approach()
          NN.forward(X)

Out[128]: array([[ 0.49715626],
                 [ 0.49645157],
                 [ 0.49635454],
                 [ 0.49717301],
                 [ 0.49866999],
                 [ 0.49914825],
                 [ 0.49763902],
                 [ 0.49991476],
                 [ 0.49961696],
                 [ 0.50078049]])
```

FIGURE 17 – Propagation des valeurs dans ANN

```
In [130]: T=trainer(NN)

In [131]: T.train(X,Y)

          Optimization terminated successfully.
                   Current function value: 0.000000
                   Iterations: 63
                   Function evaluations: 77
                   Gradient evaluations: 77

In [132]: Y

Out[132]: array([[ 0.],
                 [ 0.],
                 [ 0.],
                 [ 0.],
                 [ 1.],
                 [ 1.],
                 [ 1.],
                 [ 1.],
                 [ 1.],
```

FIGURE 18 – Apprentissage et Optimisation des performances du ANN

```
In [94]: NN.forward(X)

Out[94]: array([[  2.76617641e-17],
                [  9.44790120e-04],
                [  1.01495250e-22],
                [  5.61276423e-17],
                [  1.00000000e+00],
                [  1.00000000e+00],
                [  9.98305320e-01],
                [  1.00000000e+00],
                [  1.00000000e+00],
                [  1.00000000e+00]])
```

FIGURE 19 – Valeurs estimées par ANN de Y

4 Tests

```
In [135]: NN.forward(XT)

Out[135]: array([[ 1.],
                 [ 1.],
                 [ 1.]])

In [136]: YT

Out[136]: array([[ 1.],
                 [ 1.],
                 [ 1.]])
```

FIGURE 20 – Estimation de YT à partir d'un jeu de données XT

5 Conclusion

les réseaux de neurones reposent à Présent sur des bases mathématiques solides qui permettent d'envisager des applications dans presque tout les domaines y compris industriel et à grande échelle, notamment dans le domaine de la classification. A travers cette application

6 Références

References Anaconda.org. (2018). : : Anaconda Cloud. [online] Available at : https ://anaconda.org/ [Accessed 20 Jan. 2018].

Apprendre-python.com. (2018). Apprendre à utiliser des tuples en python - documentation python français french. [online] Available at : http ://apprendre-python.com/page-apprendre-tuples-tuple-python [Accessed 20 Jan. 2018].

Csmining.org. (2018). Spam email datasets * - Csmining Group. [online] Available at : http ://csmining.org/index.php/spam-email-datasets-.html [Accessed 20 Jan. 2018].

Fr.wikipedia.org. (2018). Fonction de Heaviside. [online] Available at : https ://fr.wikipedia.org/wiki/Fonction$_de_Heaviside$[$Accessed 20 Jan. 2018$].

Fr.wikipedia.org. (2018). Sigmoïde (mathématiques). [online] Available at : https ://fr.wikipedia.org/wiki/Sigmo%C3%AFde$_(math\%C3\%A9matiques)$[$Accessed 20 Jan. 2018$].

lazzaroni, t. (2018). Spam words : la liste des mots français interdits dans vos emailing. [online] Blog Codeur.com. Available at : https ://www.codeur.com/blog/spam-words-liste-francais-mots-interdits-emailing/ [Accessed 20 Jan. 2018].

network ?, H. (2018). How to choose the number of hidden layers and nodes in a feedforward neural network ?. [online] Stats.stackexchange.com.

Available at : https ://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw [Accessed 20 Jan. 2018].

Pypi.python.org. (2018). PyPI - the Python Package Index : Python Package Index. [online] Available at : https ://pypi.python.org/ [Accessed 20 Jan. 2018].