Notes de cours sur l’intelligence artificielle :

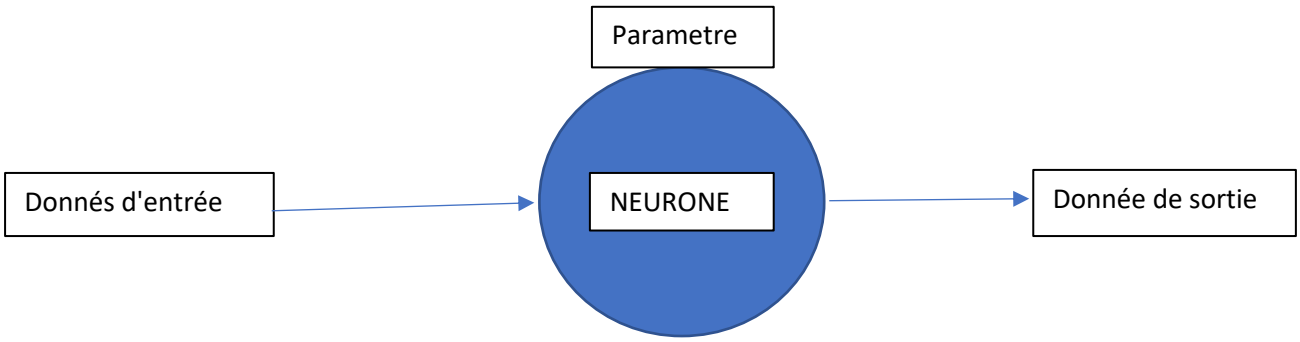
I : c’est quoi une IA

L'intelligence artificielle (IA) vise donc à reproduire au mieux, à l'aide de machines, des activités mentales, qu'elles soient de l'ordre de la compréhension, de la perception, ou de la décision. Par là même, l'IA est distincte de l'informatique, qui traite, trie et stocke les données et leurs algorithmes. Le terme « intelligence » recouvre ici une signification adaptative, comme en psychologie animale. Il s'agira souvent de modéliser la résolution d'un problème, qui peut être inédit, par un organisme.

En gros

un IA est un programme informatique qui fonctionne non pas avec des fonctions mais des neurones. Ces neurones fonctionne sur le même principe des données en entrée est des données en sortie souvent normalisé (on y reviendra plus tard) a la seule différence c’est que contrairement a des fonction les neurones peuvent être paramétré par des valeur

.



Les données de sortie sont appelées prédiction plus notre réseau est fiable plus les données produite par les neurones seront fiable jusque-là logique

II : L'ENTRAINEMENTS D’UNE IA

En Machine Learning on parle souvent d’entrainer une IA comme on l'a vue au-dessus il faut des paramètres pour faire fonctionner notre neurone

III : LA CONSTRUCTION D'UN NEURONE

Pour construire un Neurone il nous faut un problème a résoudre

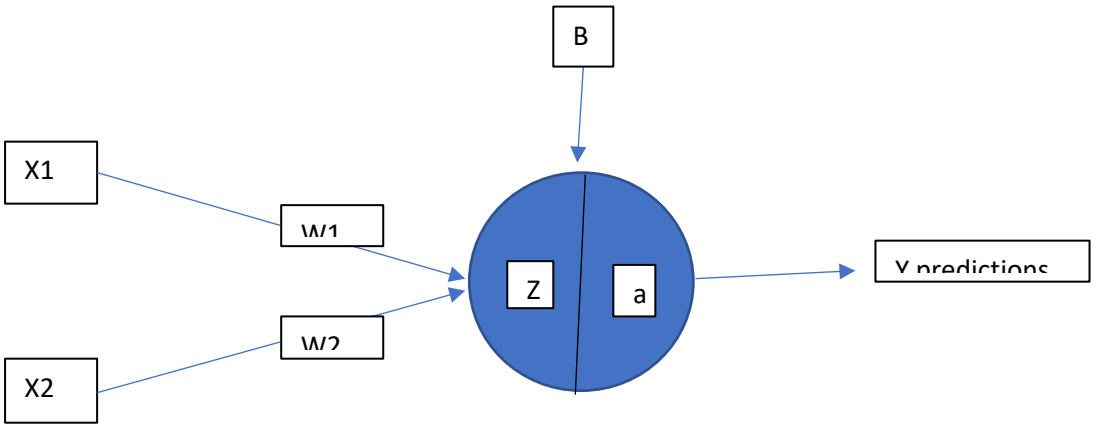
ICI nous devrons déterminer si une plante est toxique ou non en fonction de 2 paramètres la longueur d'une feuille et sa largeur
nous avons donc 2 données a rentrée dans notre neurone soit :

Longueur = X1
largeur = X2

Mtn on va parler math :

Le perceptron : c'est une formule mathématique qui sert a séparé deux classes de point dans notre exemple toxique Y = 1, non toxique Y = 0

ATTENTION SCHEMA TRES TOXIQUE



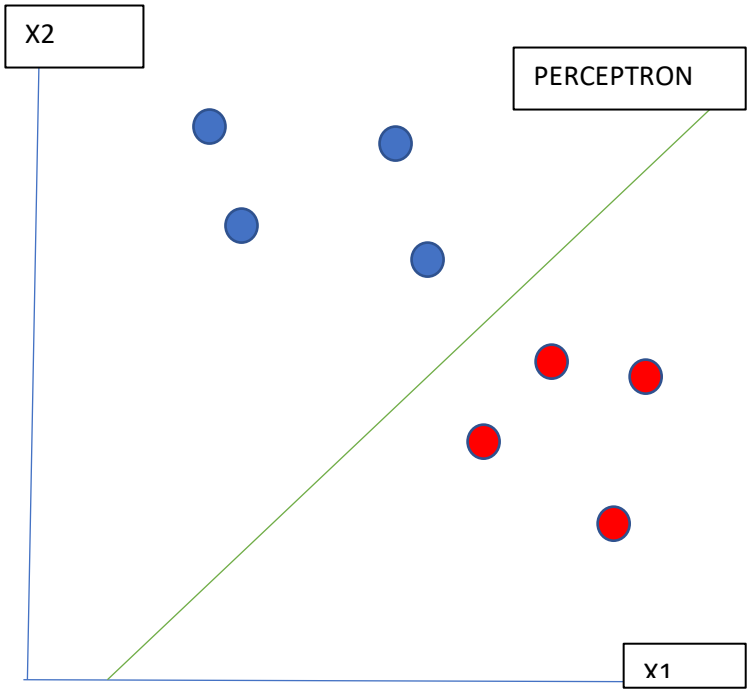
PAS DE PANIQUE on explique

ICI le but est de trouver les paramètres soit W1 W2 et b les plus précis pour déterminer avec n'importe quelle x1, x2 la sortie la plus probable

A : LE PERCEPTRON

Attention voici la formule qui a révolutionné le monde du machine Learning :

Z(x1 , x2) = (W1X1 + W2X2) +b



Concrètement ici c'est un exemple de séparation de classe de point grâce au perceptron et on peut modifier les paramètre W et b pour trouver la meilleure position de la droite

B : La fonction Sigmoïde ou Logistique

Concrètement cette fonction permet de normaliser tous nos sorties entre 0 et 1 cela va être très utile pour déterminer la fiabilité de notre résultat au cas par cas.

A(Z) = 1 / 1 + e^-z

Grace a ça aux lieux d'avoir des valeurs relou comme –2.1 puis 1.7 on aura 10% 81% beaucoup plus lisible pour nous

C : Fonction Gout

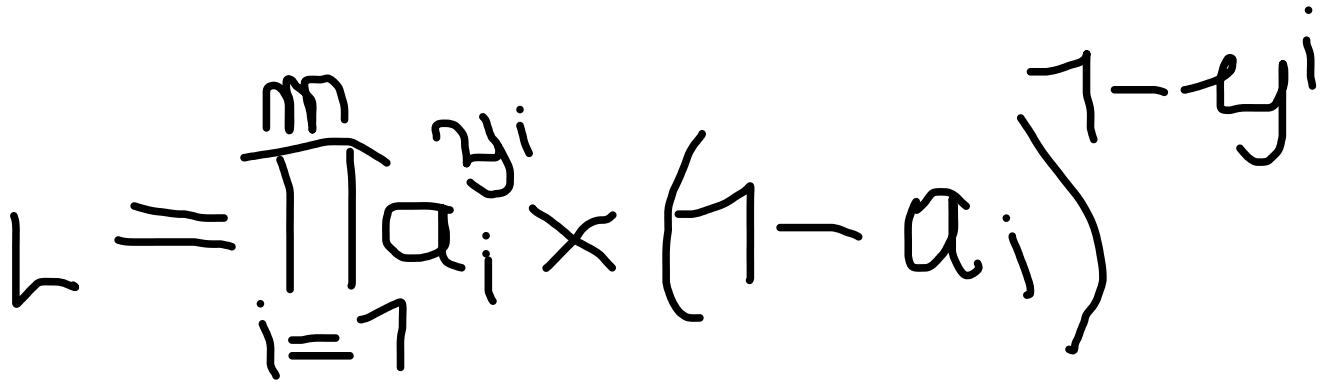
Imaginons que notre modelé a fonctionné

Avec 6 plante

	T1	T2	T3	T4	T5	T6

Toxique ou non	Toxique	Toxique	Toxique	Non Toxique	Non Toxique	Non Toxique
Donné normalisé	Y =1	Y =1	Y = 1	Y=0	Y= 0	Y= 0
Donné produite par notre neurone	P1 = 0.93	P1 = 0.35	P1 = 0.74	P1 = 0.95	P1 = 0.58	P1 =0.74

pour calculer la vraisemblance ou va faire le produit de toute la probabilité



C'"est horrible mais pas de panique c'est juste le produit de toutes nos données plus le RES est proche de 100% alors cool plus porche de 0% alors pas cool 😱

exercice

On essaye avec notre modelé ? Le produit de toutes nos données

RES L = 0.93 * 0.35 * 0.74 * 0.95 * 0.58 * 0.74 = 0.10 si ont arrondie pas ouf quoi

Pb quand on multiplie des proba on tend vers 0 donc plus on a de donnée plus le nombre et petit et ca l'ordinateur ne va plus arriver a calculer a un moment il faut donc ruser any idea ?

Vous vous rappeler du log (log de produit = somme des produit)

Notre formule de vraisemblance on va simplement rajouter on ne change l'ordre de nos termes donc c'est tous good

log(0.92 * …. * 0.74) = log(0.93) + log(….) + log(0.74)

= - 2.320623472730579

J'avoue la démonstration de la simplification j'ai la flemme

Donc abracadabra

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m} y_i log(a_i) + (1-y_i) log(1-a_i)$$

D : La descente de gradient

OUTIL INCROYABLE qui sert à ajuster nos paramètres pour minimiser la fonction LogLoss En gros sert à améliorer la fiabilité de notre Neurone

Court de math tous ca ca la dérivé d'une fonction montre comment elle varie

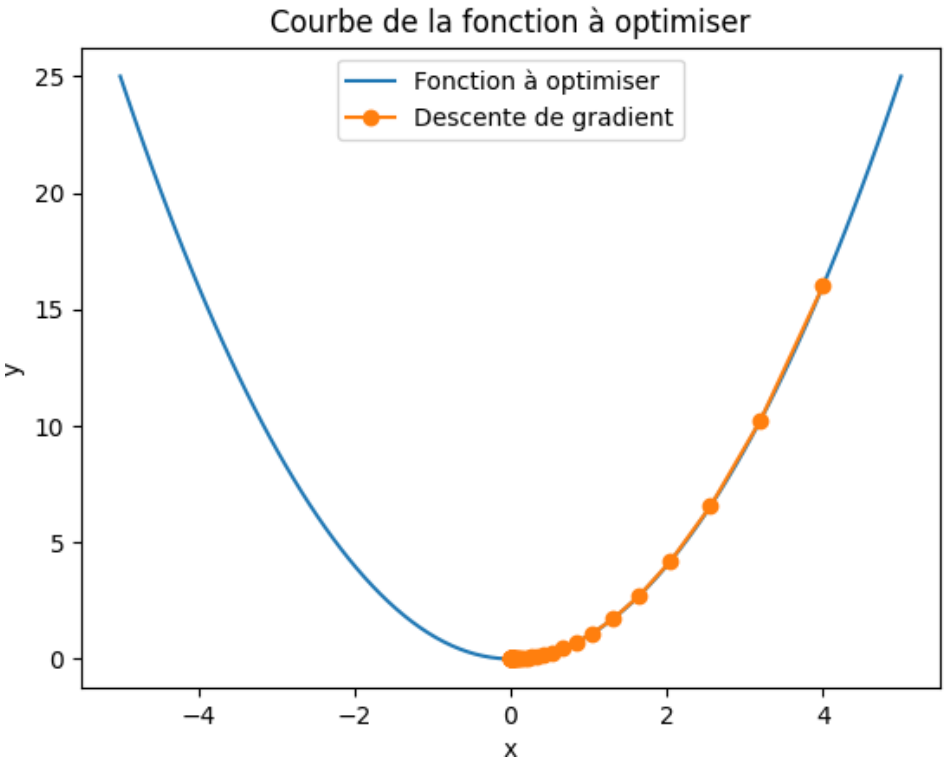
$$^{w}t + 1 = {}^{w}t - \alpha \frac{\partial L}{\partial W_t}$$

Ou W t +1 c'est le paramètre a l'instant d'après

Wt c'est les paramètres à l'instant actuel

Alpha c'est la taille du déplacement sur la courbe

Et le truc bizarre à la fin c'est le gradient a l'instant T constitué des dérivés partiel de la fonction gout et des paramètres a l'instant T



Voila un exemple de la courbe de la descente de gradient on voit bien que au fur est a mesure on se rapproche de la valeur la plus basse possible le minimum

IV) Vectorisation

C’est quoi ??????

En gros pour l’informatique c’est mettre nos données dans des tableaux pour pouvoir appliquer nos programmes a toutes les données en une seule fois
imaginons un tableau de chiffre pour multiplier tous les chiffres par deux on faire une boucle for pour parcourir le tableau grave a la vectorisation on va pouvoir multiplier toutes les données en une fois

```
1  import numpy as np#imaginons un petit exemple
2
3  Liste_Exemple = [1, 2, 3, 4]
4  print([i * 10 for i in Liste_Exemple])
5
6  Tropfort = np.array([1, 2, 3, 4])
7  print(Tropfort * 10)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

farkas@farkas-CoinCoin:~$ /bin/python3 /home/farkas/Docu
test.py
[10, 20, 30, 40]
[10 20 30 40]
```

Plus simple plus rapide bref trop cool est c’est ultra important en machine Learning

Optimise tous comme ça au lieux de parcourir nos données on fait tous d’un coup

On va donc refaire des math YOUYOU

Mais commençons par le début les Matrices eh eh

A) Les matrices

Une matrice est un tableau a deux dimensions qui peut stocker énormément de données facilement accessible c’est super pratique

Addition et soustraction de matrice

Il faut juste qu'il est la même dimension / taille

```python
import numpy as np

# Déclaration de deux matrices de 4 par 2
matrice1 = np.array([[1, 2],
                     [3, 4],
                     [5, 6],
                     [7, 8]])

matrice2 = np.array([[8, 7],
                     [6, 5],
                     [4, 3],
                     [2, 1]])

# Addition de matrices
somme = matrice1 + matrice2

# Soustraction de matrices
difference = matrice1 - matrice2

print("\nAddition de matrices :")
print(somme)

print("\nSoustraction de matrices :")
print(difference)
```

Let's try

Transposé une matrice

On fait pivoter une matrice sur sa diagonale ligne devient colone est vice versa

Lets try

```python
import numpy as np

matrice2 = np.array([[8, 7, 4],
                     [6, 5, 8],
                     [4, 3, 17],
                     [2, 1, 1024]])

# Transposer la matrice 2
matrice2_transposee = np.transpose(matrice2)

print("\nMatrice 2 transposée :")
print(matrice2_transposee)

```

Multiplié une matrice

Il faut que les deux matrices et un nombre de ligne égales aux nombres de colonnes et vice versa

Cela consiste a prendra chaque ligne pour la multiplier avec chaque colonnes

Let's try

```python
import numpy as np

# Déclaration de deux matrices de 4 par 2
matrice1 = np.array([[1, 2],
                     [3, 4],
                     [5, 6],
                     [7, 8]])

matrice2 = np.array([[8, 7, 4, 3],
                     [6, 5, 8, 1],])

# Multiplication de matrices
resultat_multiplication = np.dot(matrice1, matrice2)

print("\nRésultat de la multiplication de Matrice 1 par Matrice 2 :")
print(resultat_multiplication)

#pour plus de compréhension prenons un exemple plus simple

matrice3 = np.array([[2, 1, 3],
                     [3, 4, 1]])
matrice4 = np.array([[3, 1],
                     [2, 3],
                     [1, 0]])

print("\n",np.dot(matrice3, matrice4))
```

B)Vectorisation des fonctions

Bien évidement je ne suis pas mathématiciens donc FF pour la démonstration du pourquoi du comment on va simplement transmuter nos fonctions pour qu'elle soit applicable a l'échelle de matrice

Commence par Z qui est le perceptron

Z = (w1*x1 + w2*x2) +b qui devient Z = X*W +b ou X représentera l'ensemble de nos données comprise dans une matrice, W l'ensemble de nos paramètres w et b qui ne change pas.

Ensuite a qui est la fonction sigmoïde et bah elle change pas car a la fin on utilise toujours z

Donc a = A = 1 / 1 + e^-z

Ensuite la fonction Gout ou LogLoss

$$LL = -\frac{1}{m}\Sigma\, y \times \log(A) + (1-y) \times \log(1-A)$$

Il est horrible

Pour finir la descente de gradient

On va mesurer l'efficacité de nos paramètres W1 W2 et b

Déjà A comprend w1 et w2 donc on aura juste $W = W - \alpha \frac{\partial LL}{\partial W}$

Et notre copain b qui sera egale a $b = b - \alpha \frac{\partial LL}{\partial b}$

C'est super pas besoin de trop comprendre le pourquoi du comment

V)Première IA

Exemple tirée de la chaine YouTube MachineLearnia qui vise a déterminer si une plante est toxique ou non avec sa longueur et sa largeur de feuilles

pip install scikit-learn
pip install numpy
pip install matplot

V6 reseaux de neurones

Fonction gout = $L = -\frac{1}{m}\Sigma y \times \log\left(A^{[2]}\right) + (1-y) \times \log\left(1 - A^{[2]}\right)$

Mise a jour des parametres = $W^{[2]} = W^{[2]} - \alpha \frac{\partial L}{\partial W^{[2]}}$

$W^{[1]} = W^{[1]} - \alpha \frac{\partial L}{\partial W^{[1]}}$

$b^{[2]} = b^{[2]} - \alpha \frac{\partial L}{\partial b^{[2]}}$

$b^{[1]} = b^{[1]} - \alpha \frac{\partial L}{\partial b^{[1]}}$