Que faire avec un neurone ?

Collonville Thomas

Version 0.1.0-SNAPSHOT, 20/11/2018

Plan

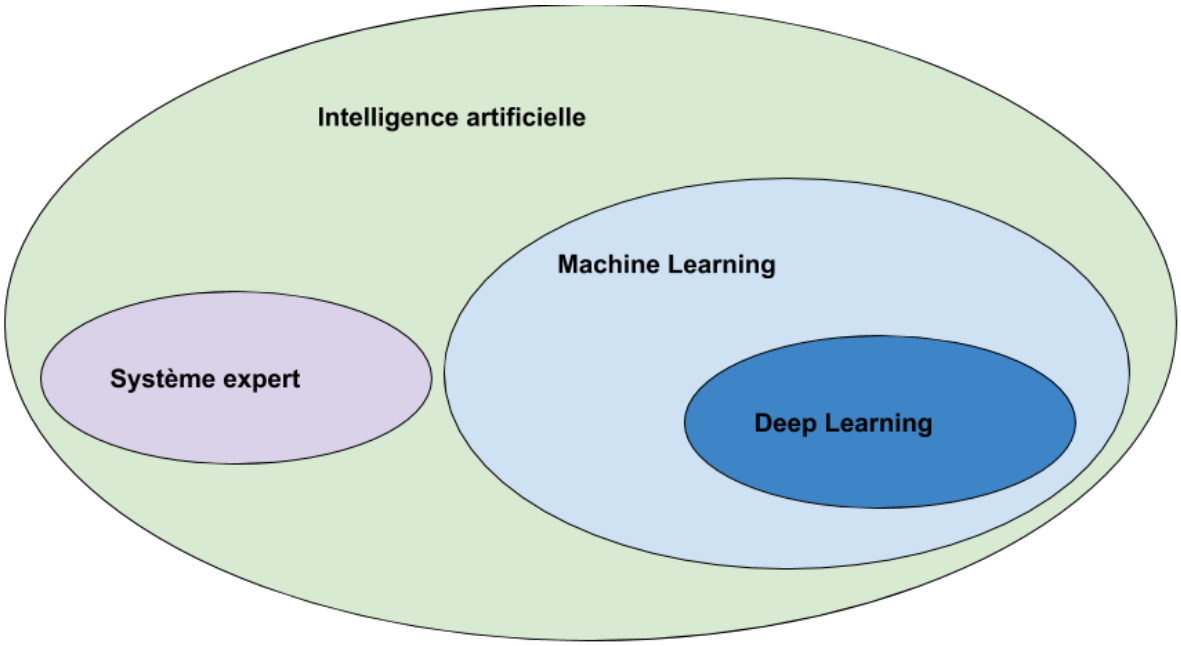
> When one teach, two learn
>
> — Robert Heinlein

Ou un peu d’espoir pour certains d’entre nous…

1. Sommaire

- Qu’est ce que l’IA
- Historique
- Le neurone
- Processus de traitement
- Exemple

2. Qu’est ce que l’IA



3. Historique

- 1940 : Alan TURING : Machine de Turing
- 1943 : Warren McCULLOCH & Walter PITTS Modèle formel de neurone.
- 1949 : Donald HEBB : Mémoire associative, premières règles d’apprentissage.
- 1960 : Franck ROSENBLATT et Bernard WIDROW, Perceptron et Adaline.
- 1980 : Stephen GROSSBERG et Teuvo KOHONEN Auto-organisation des réseaux et adaptation

3.1. Historique

- 1986 : Paul Smolenski : Machine de BOLTZMANN
- 1997: Deep Blue
- 2011 : Watson
- 2014 : LeCun Deep Learning
- 2015 : Alpha Go
- 2018 : Alexa

4. Les modeles

4.1. Non-supervisé

- Foret aléatoire
- Clustering et réduction de dimension
- SVD
- Analyse par composante principale (a noyau ou non)
- Règles

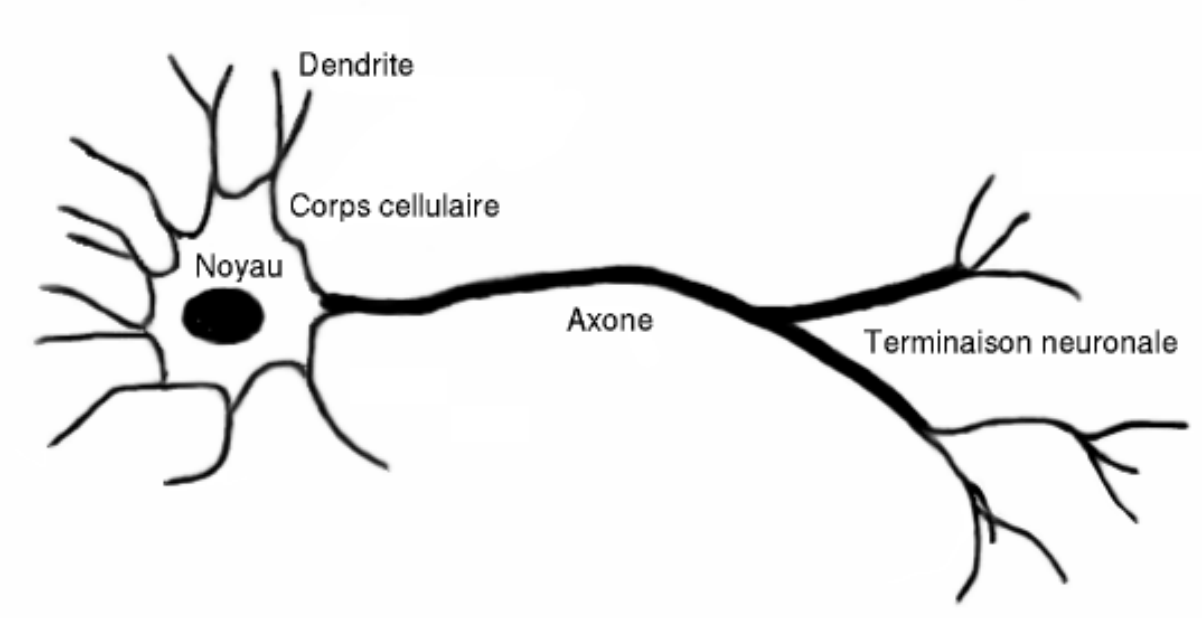
4.2. Supervisé

- Régression linéaire ou logistique
- Descente de gradient
- Régression polynomiale
- Séparation a vaste marge (SVM)
- Arbre de descision
- Classification linéaire ou non linéaire
- Réseau de neurone
- Naïve bayes

5. Le neurone

- Outil mathématique
- Formalisé par McCULLOCH et PITTS

5.1. Le neurone biologique



5.2. Constitution

- d'un noyau : le cœur de la cellule neuronale
- de dendrites permettant d'agréger les informations entrantes venant des synapses
- d'axones fournissant la réponse neuronale
- de synapses : interconnexion entre les axones et les dendrites permettant le transfert de l'influx nerveux

5.3. Quelques nombres

- 100 Milliards de neurones
- 10000 Synapses par neurone
- 10^15 Synapses dans le cerveau humain

5.4. Utilité

- Mémoire et persistance des données dans le temps
- Réflexion, élaboration des idées, associer des concepts et des stratégies
- Sens, Analyse des données, traitements des sons, des images, du touché
- Construction d'une réponse moteur, l'équilibre, l'orientation, la marche, dextérité

5.5. Le neurone formel

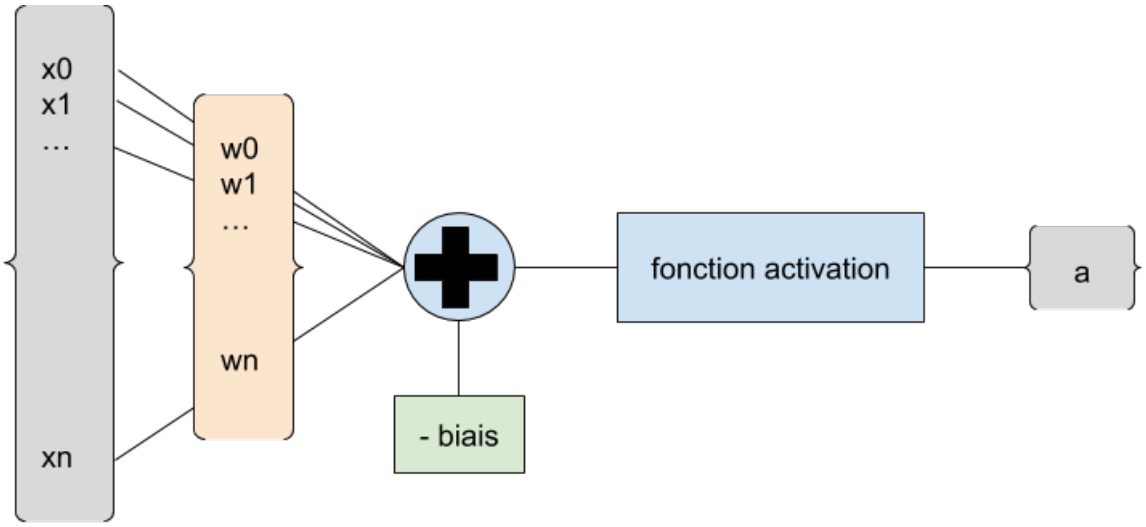
$$a = f(\sum_i^n (x_i.w_i) - b)$$

$$a = f(W^T.X - b)$$

- a la sortie du neurone

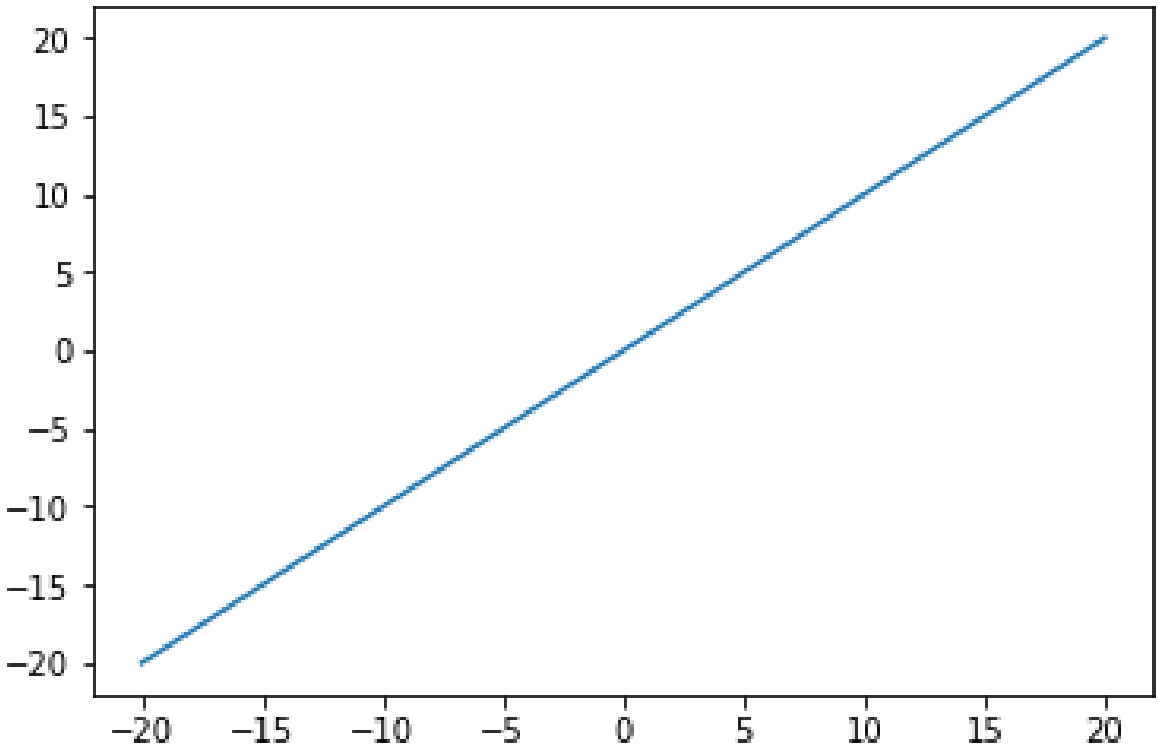
- xi, le signal d'entré
- wi, le poid de ponderation
- biais, une constante de pondération
- f, la fonction d'activation

5.6. Le neurone formel

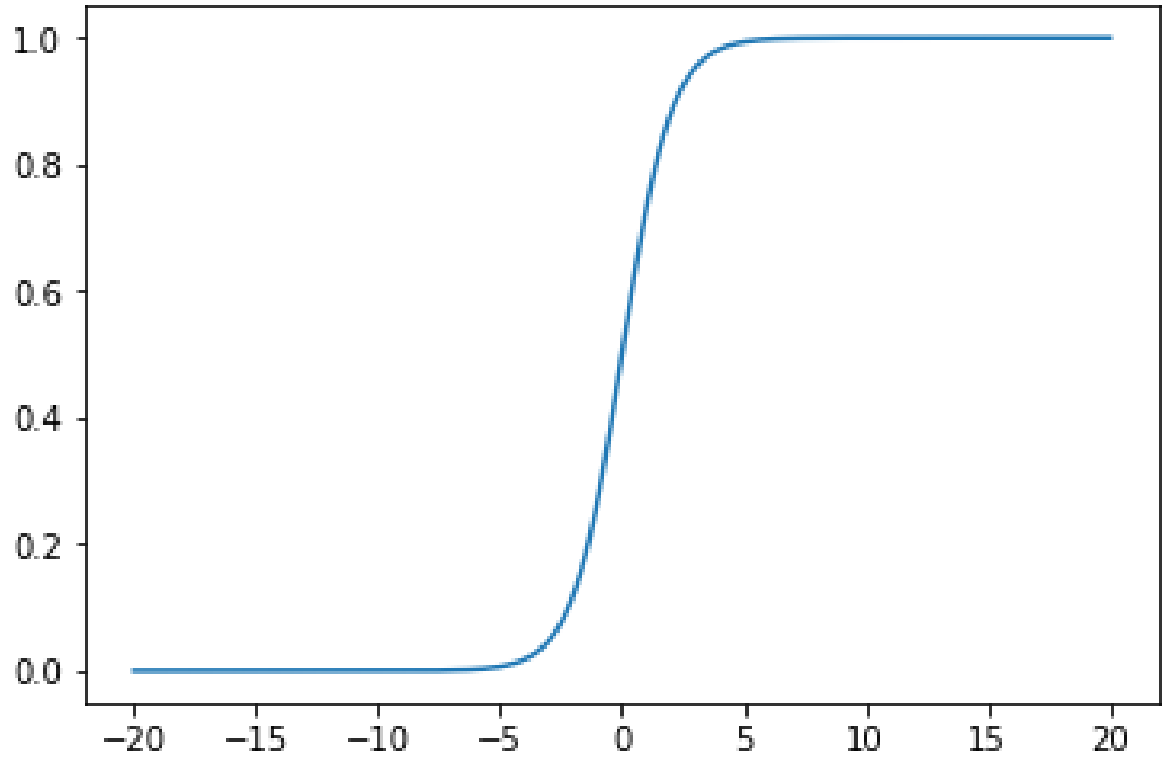


5.7. La fonction d'activation

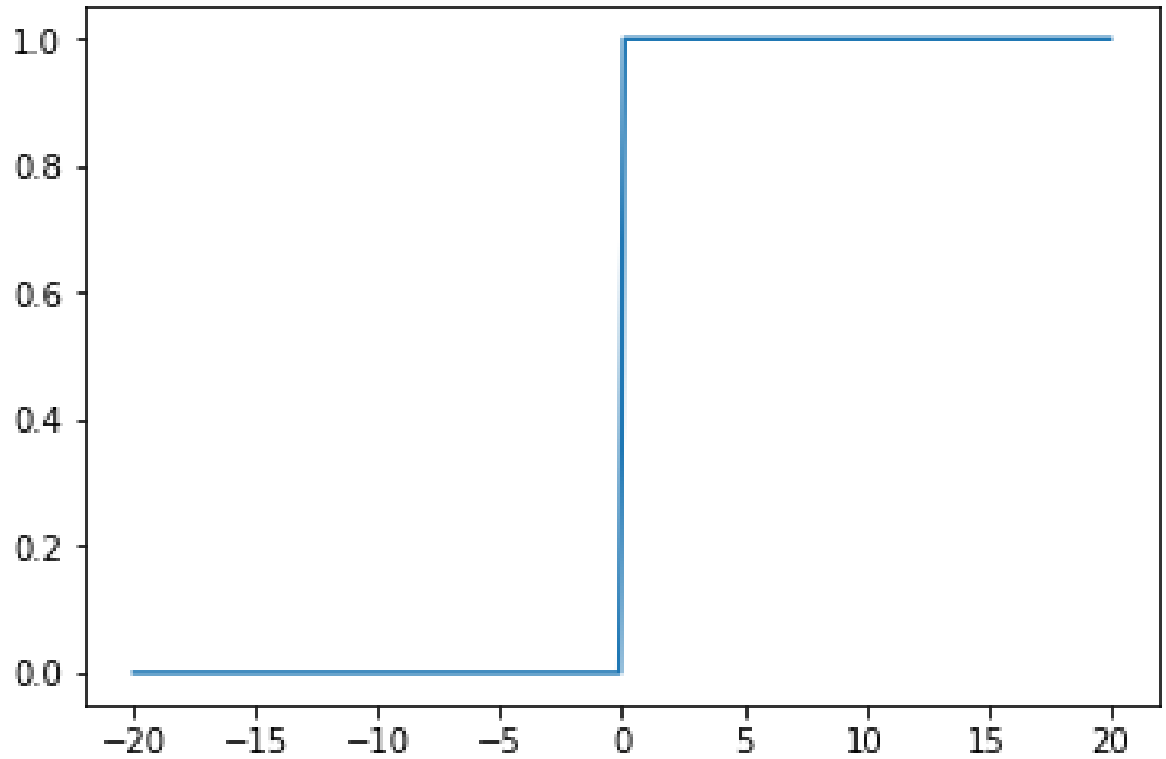
5.8. Lineaire



5.9. Sigmoire



5.10. Limiteur



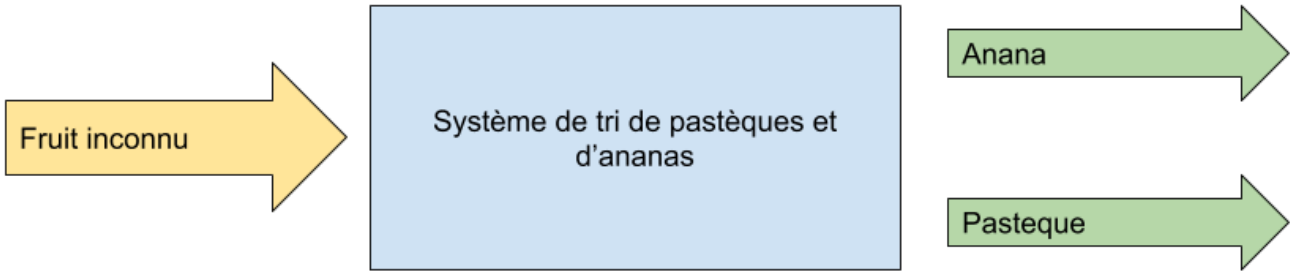
6. Processus

- Analyse du probleme
  - Nettoyage des données
  - Visualisation des données
  - Jeux de test
  - Jeux d'entrainement
- Definition d'un modele
- Apprentissage
- Mesure d'efficacité
- Mise en exploitation

7. Exemples

- Problemes de classification
- Approche Linéaire
- Approche Sigmoide
- Probleme de regression
- Approche Linéaire

8. Probleme de tri



- a rugosité → 0 lisse a 1 rugeux
- la couleur → 0 bleu a 1 rouge
- la forme → 0 rond a 1 alongé
- le poid → 0 (20gr) à 1 (2000gr)

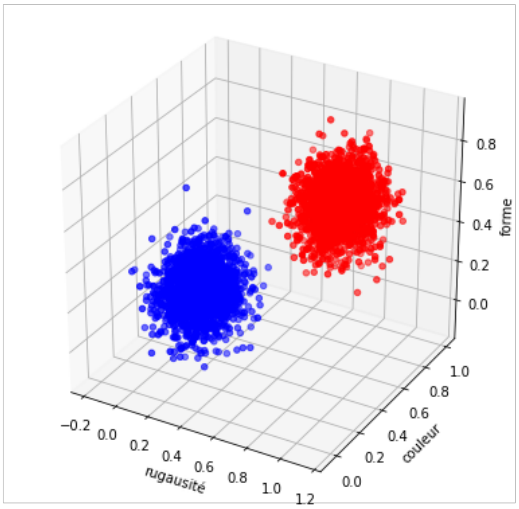
8.1. Les données

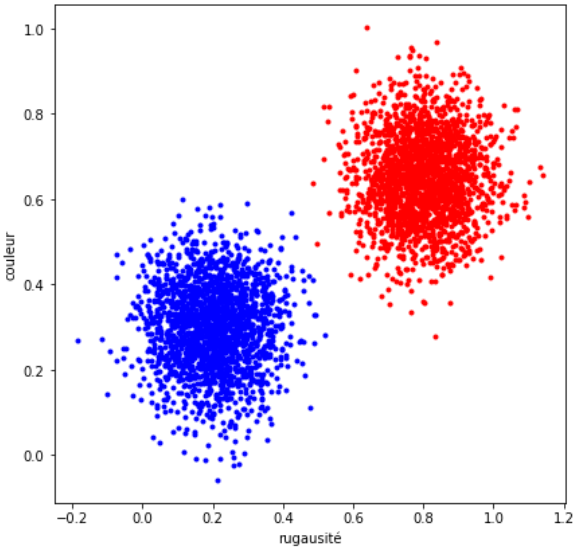
```
def generateSet(prototype,nbrEchantillon,coef):
    rand_value=np.random.randn(len(prototype),len(prototype[0]))/coef
    #print(rand_value)
    rand_set=prototype+rand_value
    if nbrEchantillon == 0 :
        return prototype
    else:
        return np.concatenate((rand_set,generateSet(prototype,nbrEchantillon-1,coef)))
```

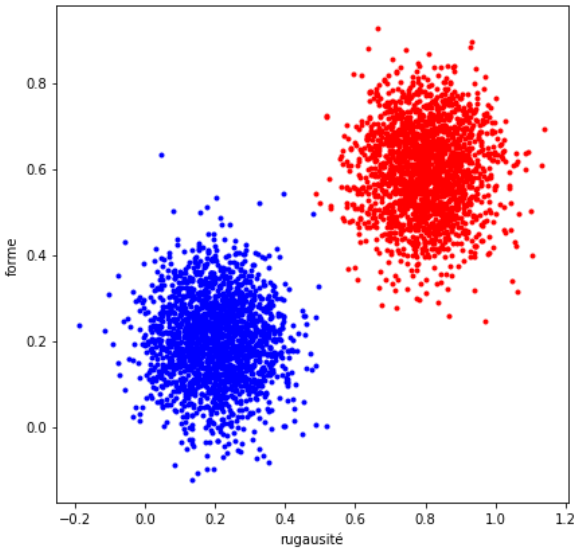
8.2. Les données

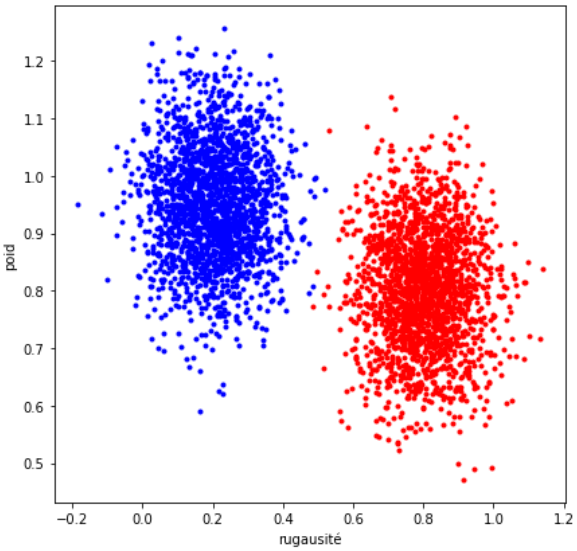
```
pasteque=np.array([[0.2, 0.3, 0.2, 0.95]])
anana=np.array([[0.8, 0.65, 0.6, 0.8]])

pasteques=generateSet(pasteque,1999,10)
ananas=generateSet(anana,1999,10)
# 10 -> pour separer les ensembles
```

8.3. Les données









8.4. Profil moyen

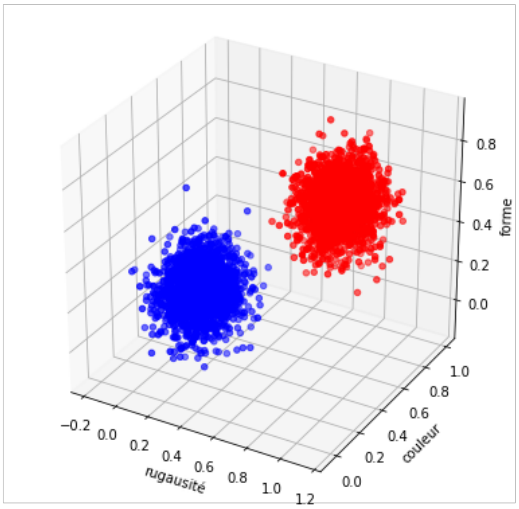
```
pastèque [0.2, 0.3, 0.2, 0.95]
anana [0.8, 0.65, 0.6, 0.8]
```

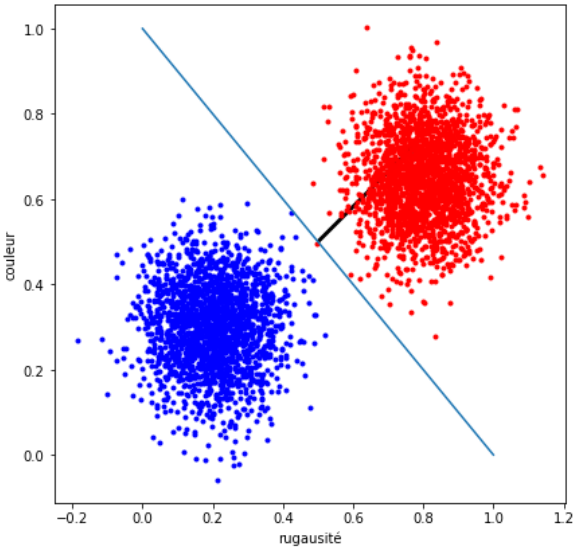
9. La classification lineaire

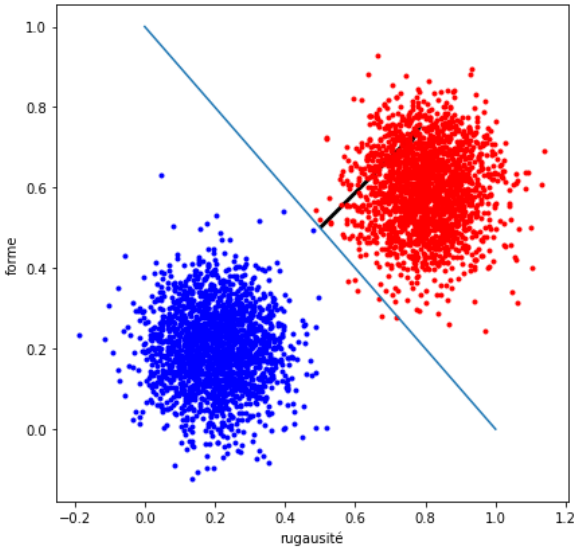
9.1. Solution adhoc

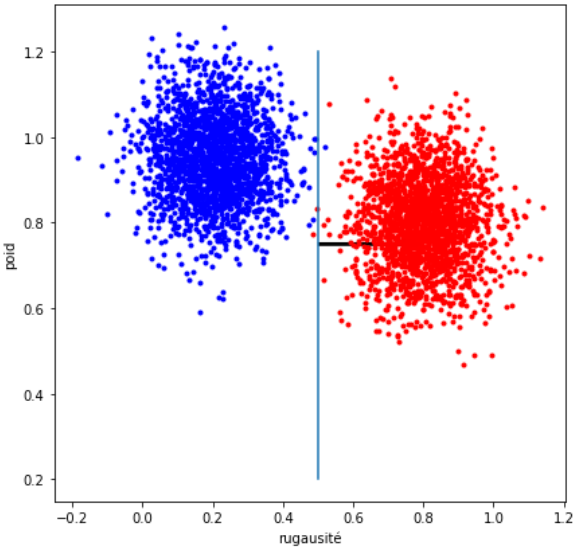
- W=[1;1;1;0]
- biais 1,5
- Verification analytique
  - limiteur((Wt.pasteque)-biais)= limiteur(0.4- 1.5)= limiteur(-1.1)= 0
  - limiteur((Wt.anana)-biais)= limiteur(2.35- 1.5)= limiteur(0.85)= 1

9.2. Pourquoi ca marche









9.3. Solution logicielle

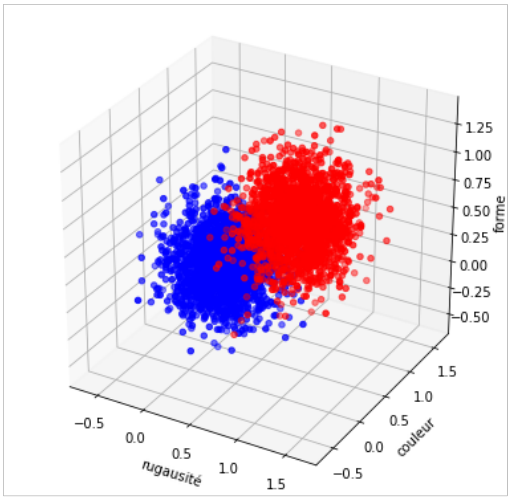
```
def neuroneLim(entre,W,biais):
    a=np.dot(entre,W.T)-biais
    #print("a neurone:",a)
    if a > 0:
        return 1
    return 0
```

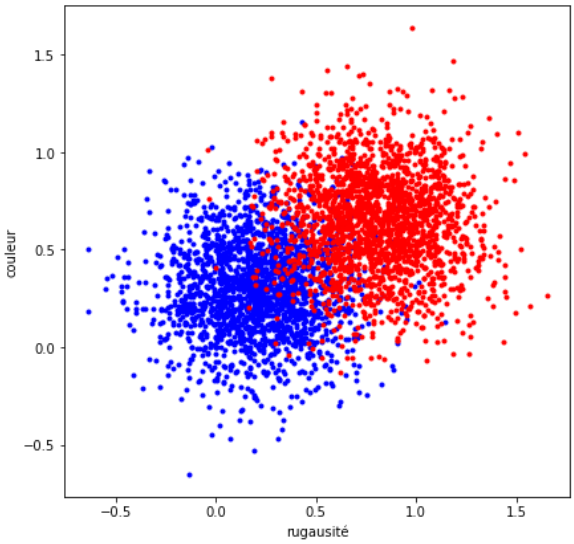
9.4. Mesure de la performance

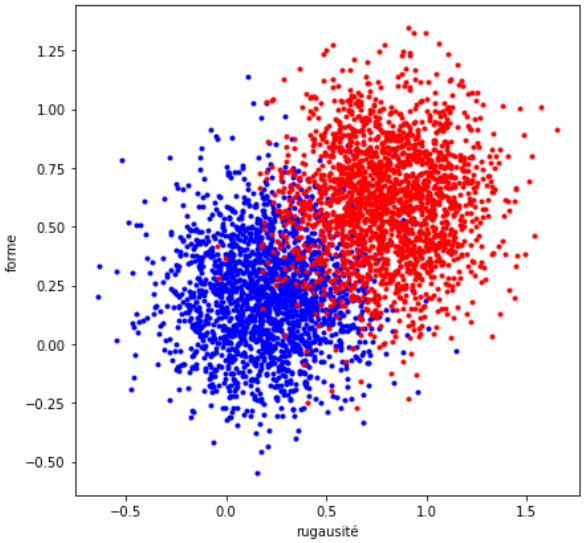
- Calcul du cout
  - Ratio des bonnes reponses par rapport au mauvaise
- pasteques: 71 somme: 96.49824912456228

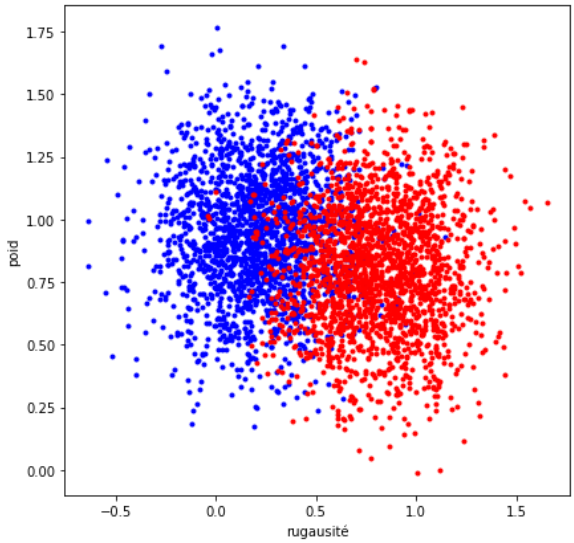
- ananas: 1815 somme: 90.79539769884943

9.5. Superposition









Outil plus precis?

9.6. Matrice de confusion

$$\begin{pmatrix} reel\,/\,prediction & pastequesP & ananasP \\ pastequesR & 1938 & 62 \\ ananasR & 202 & 1798 \end{pmatrix}$$

9.7. Interpretation

- 1938 Vrai Positif
- 1798 Vrai Négatif
- 62 Faux Négatif
- 202 Faux Positif

9.8. Precision et rappel

- Précision : VP/(VP+FP)= 1938/(1938+202) = 0.90
  - capacité à détecter des pastèques en présence d'ananas
  - 0.90 de chance que le modèle réponde que le fruit est un ananas
- Rappel ou sensibilité : VP/(VP+FN)= 1938/(1938+62) =0.97
  - capacité à réellement détecter une pastèque dans un ensemble ne contenant que de pastèques

9.9. Apprentissage

- supervisés → on indique la bonne reponse
- non supervisé → le modele interprete la reponse (approche par clustering)
- semi-supervisé

```
si etiquete - sortie > 0 alors W=W+data
si etiquete - sortie < 0 alors W=W-data
si etiquete - sortie = 0 alors W
```

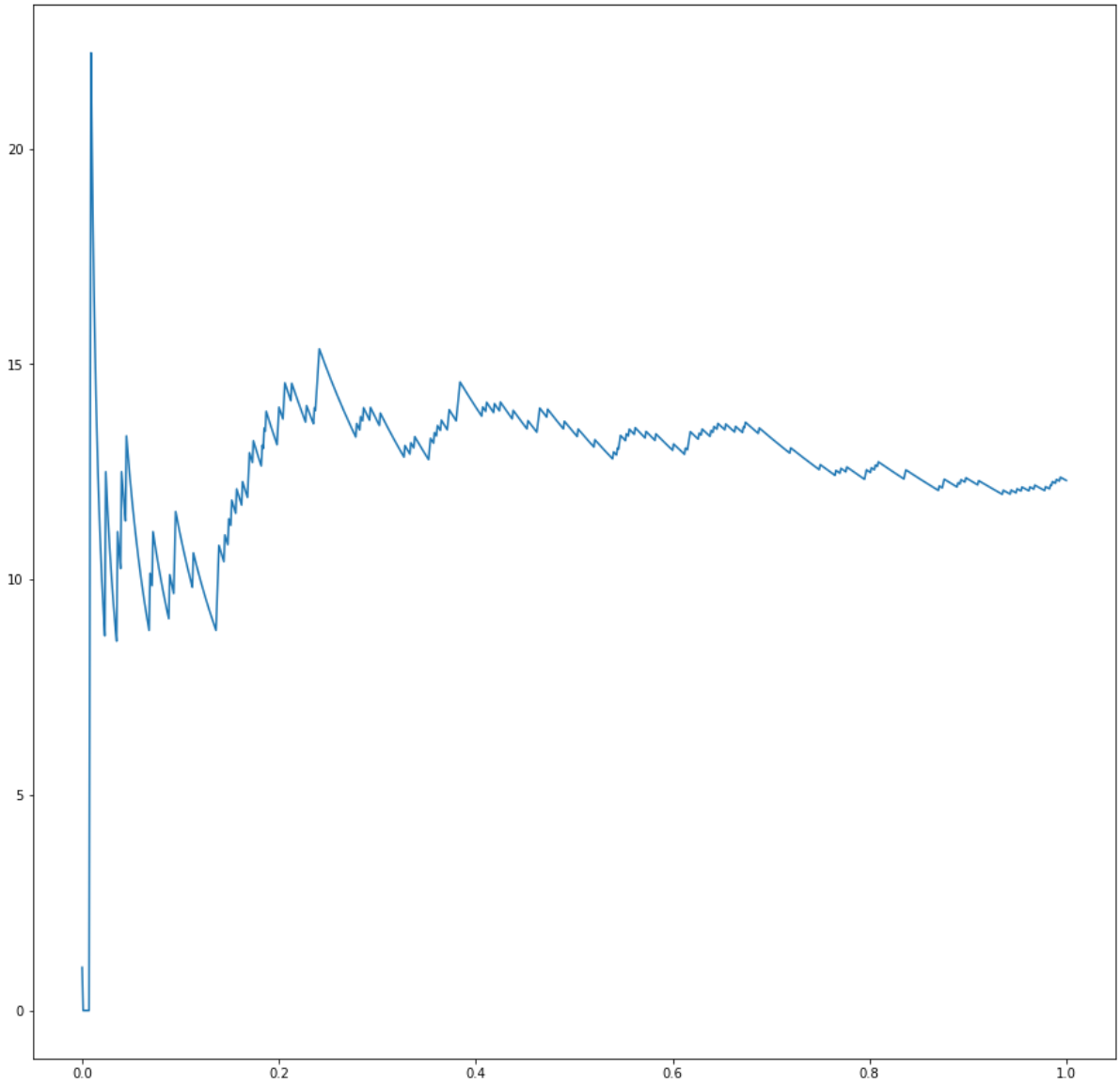
9.10. Apprentissage

```
def majW(W, sortie, etiquette,entree):
    return W+(etiquette-sortie)*entree

for (val,etiquete) in datasApprentissage:
    sortie=neuroneLim(val,W,biais)
    W=majW(W, sortie, etiquete,val)
```

9.11. Test de l'apprentissage

```
for (val,etiquete) in datasTest:
    sortie=neuroneLim(val,W,biais)
    #print(sortie,etiquete)
    if sortie != etiquete:
        erreur.append(erreur[len(erreur)-1]+1)
    else:
        erreur.append(erreur[len(erreur)-1])
```

9.12. Test de l'apprentissage



10. La classification sigmoide

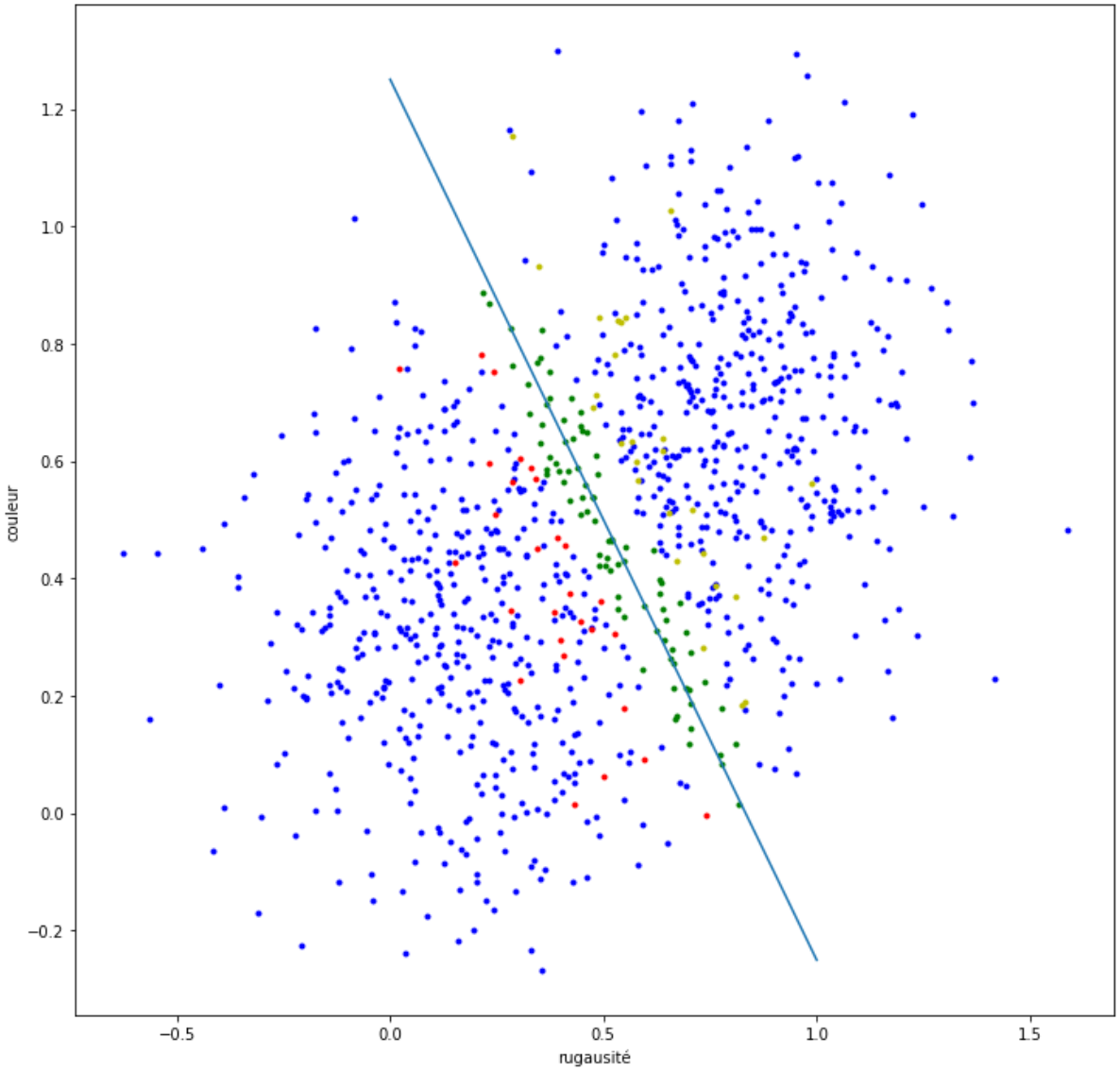
10.1. Modele Sigmoide

```
def neuroneCore(entre,W,biais):
    return np.dot(entre,W.T)-biais

def sigmoid(a):
    return 1 / (1 + math.exp(-a))

def neuroneSig(entre,W,biais):
    a=neuroneCore(entre,W,biais)
    return sigmoid(a)
```

10.2. Resultat

- 15 données indécidables
- 471 pasteques qui sont bien des pasquetes!
- 452 ananas qui sont bien des ananas!
- 40 pasteques qui se prennent pour des ananas
- 22 ananas qui se prennent pour des pastèques

10.3. Interpretation



11. La regression lineaire

11.1. Problematique

[800] | *donnéesBruite.png*

11.2. Estimateur

$$\widehat{y} = W^T.X + b$$

```
Equation Normale
```

$$\Theta = (X^T.X)^{-1}.X^T.Y$$

11.3. Inference

```
public Double linearInfer(Double[] stepInputs)
{
    Stream.Builder<Double> sum=Stream.<Double>builder();
    for(int i=0;i<dendrites.length;i++)
    {
        if(i<stepInputs.length)
            sum.add(dendrites[i]*stepInputs[i]);
        else
            sum.add(dendrites[i]);
    }
    return sum.build().reduce((x,y)-> x+y).get();;
}
```

12. Conclusion