# Week 2 :

Je commence en implémentant un réseau de neurone simple pour le début.

Source : https://www.youtube.com/watch?v=kft1AJ9WVDk&list=PLcTRC8diSLLB5ViyPGiwZfAi3t7bhwjQv

## Premier code python :

Utilisation de numpy.

**Fonction sigmoid :**

Elle est utilisée pour introduire de la **non-linéarité** dans le modèle, ce qui permet au réseau de neurones de mieux capturer les relations complexes dans les données.

Fonctionne dans un cas de binarité.

On utilise données d’entrées, sorties et weights fictifes pour tester.

.T permet de mettre l’array en vecteur colonne.

Np.dot fais un scalaire de mes matrices.s

Je calcule sur papier pour vérifier mes résultats.

Le résultat est calculé en fonction de la fct sigmoid :  
si j’obtiens -0.99999 la fonction sigmoïde me renverra 0.26.

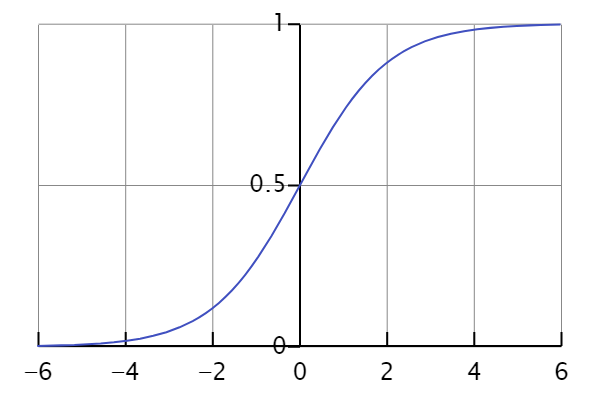


Figure / Sigmoid fonction (Wikipédia)

**Training process :**

1. Take the input for training and see the output.
2. Calculate the error / différence entre ce qu’on a eu et ce qu’on voulait.
3. En fonction de la gravité de l’erreur réadapter les poids.
4. Repeat

Ajuster les poids en faisant : error.input.’sigmoid(output)

Sigmoid() = 1/1+exp(-x) ---------- Sigmoid’() = x.(1-x)

**Ajustement des poids**: adjustements =  error \* sigmoid\_derivatives(outputs)

# Interrogation

Pourquoi la dérivée de la fonction sigmoïde ?

Pourquoi la fonction sigmoïde ?

Pourquoi erreur \* sigmoide’ ??