# Un neurone (ou perceptron)

avec les poids pour chaque variable donné en entré au neurone. Attention, il faut les valeurs normalisées.

On applique ensuite la fonction qui renvoie une valeur sur [0 ;1]

## Performance du modèle

On compare la probabilité d’appartenance de la classe avec la réalisation. Ainsi, la qualité de prédiction du modèle avec la classe réel pouvant être y={0 ;1} :

L’ensemble des données d’apprentissage suit une loi de Bernoulli ce qui donne: vraisemblance.

*Elle tend vers 0 plus on a de données.*

L augmente avec la qualité de prédiction du modèle et le nombre de données. Pour faciliter les calculs et la lecture, on applique la fonction log qui conserve l’ordre (croissante) :

On cherche à minimiser la fonction cout noté log loss

On applique :

* une pondération m (nbre de données) pour que la qualité ne dépende pas du nombre de données.
* un signe –

## Calcul des poids

L’optimisation du modèle passe par l’ajustement des poids. On cherche à minimiser le log L

Cela passe par le calcul du gradient cad des dérivés partielles.

avec un pas d’apprentissage.

où avec X la matrice chaque ligne correspond aux valeurs d’une variable

Avec A la matrice colonne des

# Réseau de neurones