Un neurone (ou perceptron)

$z = w_1x_1 + w_2w_2 + b$ avec w_i les poids pour chaque variable x_i donné en entré au neurone. Attention, il faut les valeurs normalisées.

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

$$a(z) = \frac{1}{1 + e^{-z}}$$

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} :

$$p(Y = y) = a(z)^y \times (1 - a(z))^{1-y}$$

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

$$L = \prod_i^n {a_i}^{y_i} \times (1 - a_i)^{1-y_i}$$

Elle tend vers 0 plus on a de données.

L augmente avec la qualité de prédiction du modèle et le nombre de données. Pour facilité 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

$$L = -\frac{1}{m}\sum_{i=1} y_i \,.\, log(a_i) + (1 - y).\, log(1 - a_i)$$

On applique :

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

Calul des poids

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

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

$w_{i+1} = w_i - \alpha \frac{\partial L}{\partial w_i}$ avec α un pas d'apprentissage.

$\frac{\partial L}{\partial w_i} = \frac{1}{m} X^T.(A - y)$ où avec X la matrice chaque ligne correspond aux valeurs d'une variable $X = \begin{matrix} x_1 & x_2 & x_3 \end{matrix}$

Avec A la matrice colonne des a_i

$$b = b - \alpha \frac{\partial L}{\partial b}$$

$$\frac{\partial L}{\partial b} = \frac{1}{m}\sum (A - y)$$

Réseau de neurones