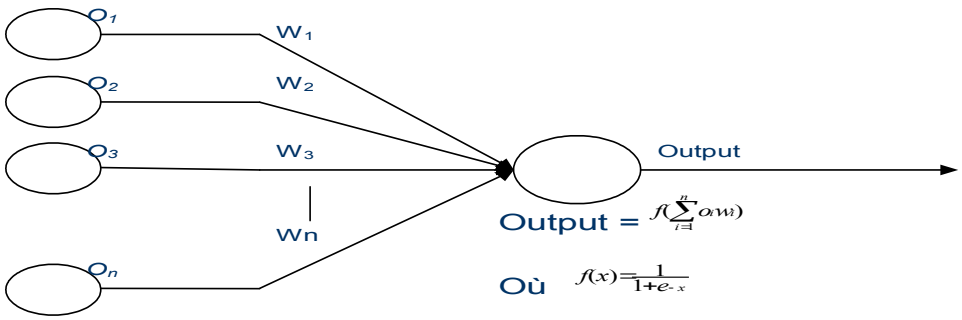
Data Mining : Apprendre ? Les réseaux neuronaux

Le neurone artificiel est modélisé ainsi (par exemple). La fonction d'activation proposée f est un filtre sur l'intensité (la tension) électrique reçue (voir tableau ci-dessous). Si on veut vraiment voir apparaître une fonction affine comme argument de la fonction f on ajoute souvent une entrée o_0 comme un biais/ un seuil / un offset (pour revenir à l'équation d'une droite générale : $w_1*o_1+w_2*o_2+w_0=0$.



avec des fonctions d'activations au choix : (Rq. : la fonction identité est la fonction linéaire par excellence).

Pas unitaire		$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$
Sigmoïde		$f(x) = \frac{1}{1+e^{-\beta x}}$
Linéaire Seuillée		$f(x) = \begin{cases} 0 & \text{if } x \leq x_{min} \\ mx+b & \text{if } x_{max} > x > x_{min} \\ 1 & \text{if } x \geq x_{max} \end{cases}$
Gaussienne		$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$
Identité		$f(x) = x$

Réflexion préliminaire.

1. Que donne le modèle avec deux entrées seulement plus l'entrée de biais. Dessinez-le.

2. Faites le lien avec la notion de droite séparatrice dans l'espace des deux entrées.

Tutoriel. On va en Suisse.

Rendez vous sur le serveur de l'Université de Lausanne suivant :
http://lcn.epfl.ch/tutorial/french/

Hélas la technologie évolue. Pour des raisons de sécurité, les applets java ne se lancent plus facilement du browser sur le web. En 2018, installer 'sudo apt-get install icedtea-plugin' ou 'icedtea-7-plugin' sous linux permettait de s'en sortir pour exécuter ces applets (ou changer les sites autoriser pour java sous macos dans les paramètres systèmes. Vous pouvez toujours essayer. Sinon télécharger les archives *.tar* (http://helios.mi.parisdescartes.fr/~lomn/Cours/BC/Data/) correspondantes sur mon site, extraire ces archives. Puis afficher le fichier *html* d'instruction dans un browser web et lancer l'applet du répertoire principale avec la commande `'appletviewer html/index.html'`.

1. **Neurone Artificiel** :

En utilisant la fonction d'activation de pas unitaire, on va essayer de déterminer **sur papier** un ensemble de poids (et une valeur de seuil) qui va produire la classification suivante :

x1	x2	sortie
-0.2	0.5	0
0.2	-0.5	0
0.8	-0.8	1
0.8	0.8	1

- Représenter dans un repère, le problème de classification.
- Représentez le neurone artificiel correspondant.
- En utilisant la fonction d'activation linéaire (identité ici), déterminez un ensemble de poids (et une valeur de seuil) qui pourrait produire la classification exigée. Est-ce possible de façon exacte ?
- En utilisant la fonction d'activation échelon ou pas (fonction linéaire seuillée pour $x_{min}=x_{max}=0$), déterminez visuellement un ensemble de poids (et une valeur de seuil) qui va produire la classification exigée.
- Testez l'ensemble de ce qui précède grâce à l'applet http://lcn.epfl.ch/tutorial/french/aneuron/html/index.html .
- Seriez-vous capable de tenir le même raisonnement que précédemment avec la fonction d'activation sigmoïde ($\beta=1$ ou $\beta=10^3$) ?. Qu'apporte une telle fonction pour le problème de classification ?
- Sur l'ensemble de classification suivant, ce modèle permettrait-t-il de répondre à la question ? Si non, proposez un modèle neuronal possible.

x1	x2	sortie
-0.2	0.5	0
0.2	-0.5	1
0.8	-0.8	0

0.8	0.8	1

- Testez l'ensemble de ce qui précède grâce à l'applet http://lcn.epfl.ch/tutorial/french/aneuron/html/index.html .
- Quel comportement différent induit le changement de fonction d'activation ?

2. **Apprentissage sur perceptron simple** :
   - Lancer l'applet correspondante : http://lcn.epfl.ch/tutorial/french/perceptron/html/index.html
   - Modifier les paramètres de Learning Rate, Number of Iterations et Error Threshold pour chaque fonction d'activation pour apprendre la configuration (0,0,0), (1,0,0), (0,1,0) et (1,1,1) avec (x,y,classe). Commentez.
   - Même question pour le problème du XOR. (N'utilisez pas la case à cocher SOLVE XOR).

On donne le principe de l'algorithme d'apprentissage par rétro-propagation de l'erreur (bien entendu cet algorithme dépasse largement le cadre de cette initiation, c'est informatif en terme de concision du pseudo code associé) :

Box 6. The Back-Propagation Training Algorithm

The back-propagation training algorithm is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a multilayer feed-forward perceptron and the desired output. It requires continuous differentiable non-linearities. The following assumes a sigmoid logistic non-linearity is used where the function $f(\alpha)$ in Fig. 1 is

$$f(\alpha) = \frac{1}{1 + e^{-(\alpha - \theta)}}$$

Step 1. Initialize Weights and Offsets

Set all weights and node offsets to small random values.

Step 2. Present Input and Desired Outputs

Present a continuous valued input vector $x_0, x_1, \ldots x_{N-1}$ and specify the desired outputs $d_0, d_1, \ldots d_{M-1}$. If the net is used as a classifier then all desired outputs are typically set to zero except for that corresponding to the class the input is from. That desired output is 1. The input could be new on each trial or samples from a training set could be presented cyclically until weights stabilize.

Step 3. Calculate Actual Outputs

Use the sigmoid nonlinearity from above and formulas as in Fig. 15 to calculate outputs $y_0, y_1 \ldots y_{M-1}$.

Step 4. Adapt Weights

Use a recursive algorithm starting at the output nodes and working back to the first hidden layer. Adjust weights by

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i'$$

In this equation $w_{ij}(t)$ is the weight from hidden node i or from an input to node j at time t, x_i' is either the output of node i or is an input, η is a gain term, and δ_j is an error term for node j. If node j is an output node, then

$$\delta_j = y_j(1 - y_j)(d_j - y_j),$$

where d_j is the desired output of node j and y_j is the actual output.

If node j is an internal hidden node, then

$$\delta_j = x_j'(1 - x_j') \sum_k \delta_k w_{jk},$$

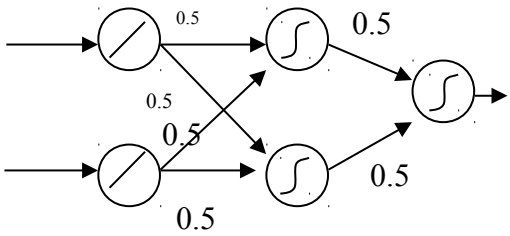
where k is over all nodes in the layers above node j. Internal node thresholds are adapted in a similar manner by assuming they are connection weights on links from auxiliary constant-valued inputs. Convergence is sometimes faster if a momentum term is added and weight changes are smoothed by

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i' + \alpha(w_{ij}(t) - w_{ij}(t-1)),$$

where $0 < \alpha < 1$.

Step 5. Repeat by Going to Step 2

- Sur l'ensemble d'apprentissage suivant, et à partir du réseau à une couche cachée initial suivant, intuitivement comment va procéder l'algorithme d'apprentissage selon vous ?



x1	x2	sortie
0	1	1
0	0	0
1	1	0
1	0	1

3. Applet **Adaline, Perceptron et Rétro-propagation**
http://lcn.epfl.ch/tutorial/french/apb/html/index.html

- **Cas idéal:** placez 10 points rouges (classe 1) et 10 points bleus (classe 0) dans deux groupes similaires, distincts et linéairement séparables et apprenez avec l'algorithme de rétropropagation. Quelles valeurs de taux d'apprentissage (learning rate) donne les meilleurs résultats ?

- **Différentes dispersions de groupe**: Placez 20 points rouges (1) dans un très petit groupe (points fortement corrélés) et 5 points bleus (0) dans un très large groupe de manière à ce que les deux classes soient linéairement séparables et apprenez avec l'algorithme de rétropropagation. Quelles valeurs de taux d'apprentissage (learning rate) donne les meilleurs résultats ?

- **Séparation non parfaite:** Placez 10 points rouges (1) et 10 points bleus (0) dans deux groupes similaires et linéairement séparables. Placez ensuite des points bleus supplémentaires à l'intérieur du groupe de points rouges et apprenez avec l'algorithme de rétropropagation. Quelles valeurs de taux d'apprentissage (learning rate) donne les meilleurs résultats ?

4. Applet Perceptron multi-couche (sortie des neurones: {0;1})
http://lcn.epfl.ch/tutorial/french/mlp/html/index.html

- Résolvez le problème du XOR.
- Placez trois groupes de points en ligne : un rouge avec 3 points, un bleu avec 6 points et encore un rouge avec 3 points. Est-ce que le perceptron multi-couche le plus simple est capable de résoudre ce problème ? Si ce n'est pas le cas, quelle est la structure minimale qui permet de résoudre ce problème ? Précisez le momentum et le taux d'apprentissage (learning rate).
- Placez un groupe de points rouges au centre, entouré par des points bleus. Quel structure de réseau et quels paramètres (momentum et taux d'apprentissage) permettent de résoudre un tel problème ?