AS - TP3 - Non-linéarité

Ludovic Denoyer et Nicolas Baskiotis

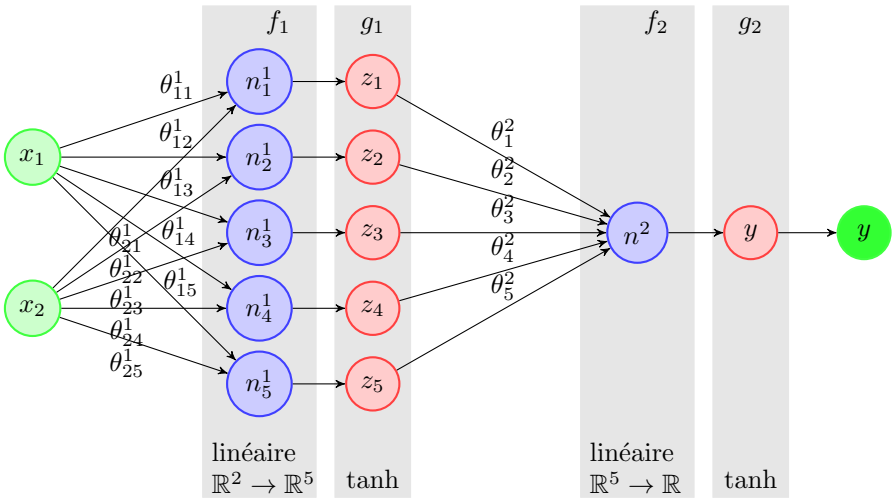
4 oct. 2016

Préambule - prérequis

Vous aurez besoin de la fonction `accuracy(y,out)` codée lors du dernier tp.
Créez un jeu de données artificiel binaire de 4 guaussiennes, deux d'étiquette positive centrées en $(-1,-1),(1,1)$ et deux d'étiquette négative centrées en $(-1,1),(1,-1)$ (problème du *XOR*). Faites en particulier un jeu d'apprentissage et un jeu de validation.
Indications : vous pourrez en particulier vous inspirer du code du premier TP.

Un réseau de neurone a la mano

Le modèle linéaire peut-il bien classifier le *XOR* ? Une première solution est d'utiliser le kernel trick (voir cours). Dans ce TP, nous allons étudier une autre solution qui consiste à enchaîner des modules linéaires et des transformations non-linéaires. Nous utiliserons comme fonction non-linéaire la tangente hyperbolique. Sous Torch, le module `nn.Tanh` permet de faire la transformation nécessaire.



— Comment s'exprime la sortie y en fonction de $\mathbf{x} = (x_1, x_2)$? Donnez le code qui permet de calculer la sortie.
— Comment faire une descente de gradient sur cette architecture ? Quelles dérivées partielles sont nécessaires ? (regardez le petite guide la backpropagation sur le site de l'ue). Donnez le code.
— Expérimentez sur les données précédentes. Visualisez les frontières de décisions (utilisez le code du premier TP). Regardez également comment se comporte la sortie de g_1 sur quelques dimensions.
— Regardez la documentation du module `Sequential`. Testez le même réseau avec ce module.