Optimisation
TP - Réseau de Neurones RBF

Thomas Robert

1 Principe

Note : Je désigne ici par « couche » le « passage » d'un ensemble de neurones à un autre.

L'objectif de ce TP est d'implémenter un réseau de neurones RBF permettant à la fois de réaliser de la classification et du rejet.

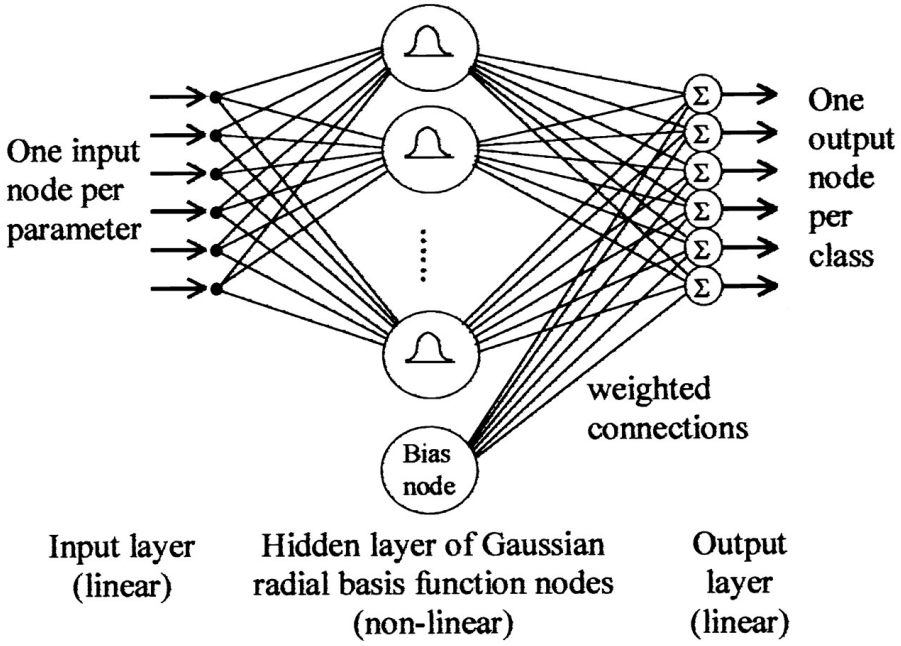

Figure 1 – Exemple de réseau RBF [1]

Le fonctionnement du réseau est relativement simple, la première couche de est une couche **RBF** cachée à n_{in} entrées et K neurones de sortie. Chaque sortie suit une loi gaussiennes multivariées (de dimension n_{in}). La liaison entre les entrées et chaque sortie est "directe", c'est à dire que le vecteur d'entrée est directement passée à la fonction gaussienne sans somme pondérée :

$$y_{RBF_i} = \phi(x_{in}|\mu_i, \Sigma_i) \qquad \forall i \in [1, K]$$

On remarque que la sortie de chaque gaussienne dépend de deux hyper-paramètres μ_i et Σ_i.

Afin de régler ces hyper-paramètres, on applique un algorithme de K-means au jeu de données afin de déterminer K classes dont on calcule les moyennes μ_i et covariance Σ_i que l'on utilisera afin de paramétrer la couche **RBF**.

1. M. F. Wilkins et al. *Identification of Phytoplankton from Flow Cytometry Data by Using Radial Basis Function Neural Networks*

Enfin, la sortie de la couche **RBF** est reliée à une couche de sortie à n_{out} neurones, un par classe que l'on veut apprendre. Cette fois, la sortie suit bien la formule usuelle :

$$y_{out} = f(y_{RBF} W_{out})$$

où f est une fonction d'activation que l'on choisira comme la fonction *softmax* afin d'obtenir une estimation de distribution de probabilités en sortie, et W_{out} une matrice de poids pondérant chaque sortie de la couche **RBF** par rapport à chaque neurone de sortie.

Si on note Y_{RBF} la matrice contenant les sorties de la couche **RBF** pour toutes les valeurs du jeu d'apprentissage, on peut estimer W_{out} par les moindres carrées :

$$W_{out} = (Y_{RBF}^\top Y_{RBF})^{-1} (Y_{RBF}^\top Y_{target})$$

où Y_{target} est la matrice des sorties voulues pour les données du jeu d'apprentissage.

On ajoute ensuite une règle de décision qui ne conserve la sortie du réseau que si le point est suffisamment probable pour au moins une des gaussiennes, ou qui rejette le point sinon.

2 Implémentation

L'implémentation est relativement simple. Le calcul des paramètres des gaussiennes utilise la fonction `kmeans` de Matlab, le calcul des probabilités à posteriori des gaussiennes utilise la fonction `mvnpdf` et `Wout` est estimé par les moindres carrés avec un `\`.

Sur les données *iris*, les matrices de covariances obtenues n'étaient pas de plein rang et il a donc été nécessaire de rajouter « un chouilla sur la diagonale » pour que les calculs fonctionnent.

3 Tests et résultats

Le code a été testé sur les données *iris*. Afin de tester le rejet, des points aberrants ont été ajoutés aux données.

Pour stabiliser les résultats, en particulier car le K-means est très aléatoire, on fait 20 itérations entre test et apprentissage (sur la même séparation apprentissage / test).

On affiche la variation de l'erreur en test en fonction de l'hyperparamètre K (nombre de neurones **RBF**) en figure 2.

On peut observer que la meilleure valeur de K en test est de 9 dans ce cas. Elle varie généralement selon le découpage du jeu de données entre 3 et 10.

Cette variabilité forte est probablement due a la faible taille du jeu de données (150 points).

On obtient, pour la meilleure valeur de K, la matrice de confusion en figure 3.

On constate que la classification fonctionne bien et que 3 des 5 erreurs sont dues au rejet de points qui n'étaient normalement pas des points aberrant, c'est à dire que leur probabilité d'appartenance aux gaussiennes était toujours inférieure à 1 %.

Encore une fois, il est possible que ce rejet trop fréquent soit dû à la taille du jeu de données, dont la partie d'apprentissage ne contenait pas de points suffisamment proche des points rejetés en test et qui n'auraient pas du l'être. Une base plus complète donnerait sans doute de meilleurs résultats.

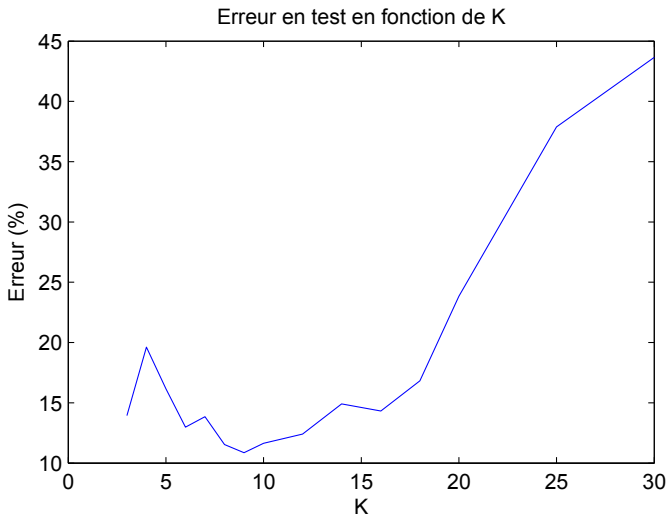

FIGURE 2 – Erreur en test en fonction de K

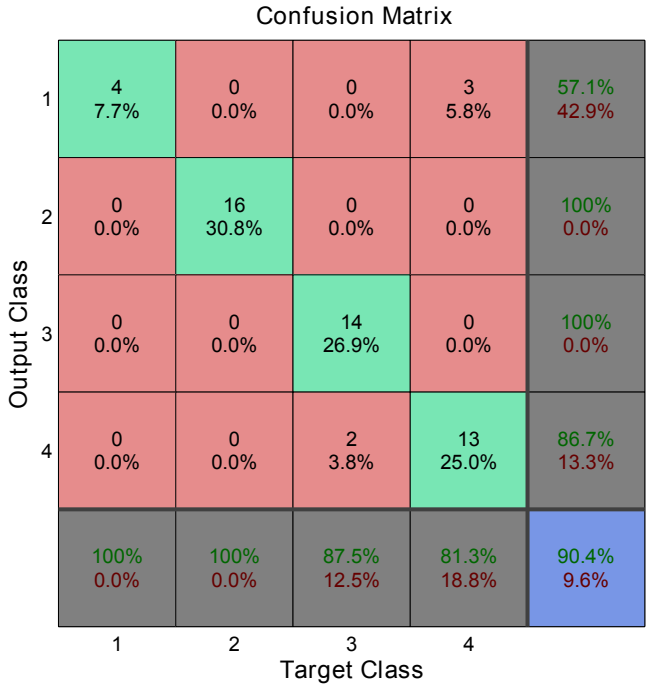

FIGURE 3 – Matrice de confusion pour $K = 9$. Classe 1 : rejet.