Les perceptrons multi-couches avec R

Fabrice Rossi

30 avril 2003

1 La bibliothèque `nnet`

Le logiciel R propose la bibliothèque `nnet` qui permet d'utiliser des perceptrons multi-couches simples :

- l'architecture est limitée à une couche cachée (perceptrons à deux couches), avec la possibilité d'introduire des connexions directes entre les entrées et les sorties (i.e., un terme linéaire) ;
- l'algorithme d'apprentissage est une méthode *quasi-newton* ;
- l'erreur est soit une erreur quadratique, soit l'entropie croisée ;
- la fonction d'activation des neurones de sortie peut être linéaire, logistique ou *softmax* ;
- on peut bloquer certains paramètres lors de l'optimisation, ce qui permet de mettre à zéro certaines connexions (ce qui revient à les supprimer) ;
- le modèle peut être pénalisé grâce à un terme de *weight decay*.

Pour entraîner un PMC, on utilise la fonction `nnet`, par exemple de la façon suivante :

```
1 mlp=nnet(x,y,size=2)
```

Cet appel place dans la variable `mlp` le pmc avec 2 neurones dans la couche cachée qui modélise les données `x` et `y`. Les données doivent être proposées sous forme de deux matrices, chaque ligne correspondant à un vecteur d'observations (les variables explicatives sont dans `x` alors que les variables à expliquer sont dans `y`). La fonction `nnet` propose de très nombreuses options décrites dans la documentation.

Quand le pmc a été entraîné, on peut l'utiliser sur de nouvelles données avec la fonction `predict`. Par exemple `predict(mlp,t)` renvoie une matrice correspondant aux images des lignes de la matrice `t` par le pmc décrit par `mlp`.

2 Exercices

Exercice 2.1

On considère le modèle quadratique suivant : $Y \sim X^2 - 2X + 1 + \mathcal{N}(0, 2)$. On considère 50 points de mesure répartis uniformément sur l'intervalle $[-5, 5]$. Tracer les prédictions réalisées par les pmc optimaux à 2, 3 et 10 neurones cachés. Étudier par simulation le dilemme biais/variance sur cet exemple.

Exercice 2.2

Reprendre l'exercice précédent afin de déterminer le modèle optimal avec les méthodes suivantes :

- découpage des données
- validation croisée
- *bootstrap*

Exercice 2.3

Reprendre l'exercice précédent en considérant 10 neurones cachés et en sélectionnant le paramètre optimal pour le *weight decay* avec les méthodes proposées.

Exercice 2.4

Reprendre l'exercice précédent en effectuant un élagage par l'algorithme *Optimal Brain Surgeon* après le *weight decay* (mais avant l'évaluation finale du modèle).