Les réseaux RBF avec R

Fabrice Rossi

31 mars 2003

1 Inversion de matrice

1.1 Décomposition en valeurs singulières

Soit une matrice A, $m \times n$. On rappelle que la décomposition de A en valeurs singulières est un triplet de matrices, U, D et V telles que :
- D est une matrice $n \times n$ diagonale dont les termes diagonaux (les valeurs singulières) sont positifs ou nuls et rangées par ordre décroissant
- U et V sont des matrices orthonormées
- $A = VDU^T$

La fonction `svd` de R calcule la décomposition en valeurs singulières d'une matrice paramètre et renvoie une liste contenant les matrices U et V, ainsi qu'un vecteur représentant la diagonale de D.

1.2 Exercice

Exercice 1.1

1. Écrire une fonction R `inverse` qui calcule la pseudo-inverse d'une matrice en utilisant la décomposition en valeurs singulières de celle-ci.
2. On considère la matrice M^n (n lignes et n colonnes) définie par $M^n_{ij} = \frac{1}{i+j}$. Cette matrice est très mal conditionnée et les algorithmes classiques ont beaucoup de difficultés à résoudre un système de la forme $M^n x = y$. Comparer les résultats donnés par la méthode `solve` de R et une méthode basée sur l'inversion par svd pour résoudre le système $M^n x = y^n$ avec $y^n_i = i$ pour différentes valeurs de n (par exemple n compris entre 5 et 10).

2 Réseaux RBF simples

2.1 Principe

Pour les réseaux RBF simples, on se contente de répartir uniformément les centres (i.e. les μ_i) dans le pavé de $\mathbb{R}^n$ considéré et on fixe σ à deux fois la distance moyenne entre les centres. Une fois les centres et σ fixés, on peut transformer les données et calculer les coefficients optimaux comme indiqué dans le cours (en utilisant la fonction d'inversion définie précédemment).

2.2 Exercice

Exercice 2.1

1. Écrire une fonction `rbf.design` qui, à un pavé de $\mathbb{R}^n$ (à représenter de façon simple) et un entier p, associe une liste contenant la matrice des p^n centres répartis uniformément dans le pavé et le σ correspondant.
2. Écrire une fonction `rbf` qui à des données (représentées sous forme d'une matrice d'observations X (dans $\mathbb{R}^n$) et d'un vecteur à prédire y), et un entier p, associe le réseau RBF optimal à p^n neurones pour la prédiction de y par X, sous la forme d'une liste contenant la matrice des centres, la valeur de σ et le vecteur des coefficients du réseau.

3. Tester la fonction sur différentes données, en commençant par des données artificielles, puis en passant aux données `airquality` et `swiss` de R.

2.3 Problème des grandes dimensions

La solution proposée dans les sections précédentes n'est pas satisfaisante quand la dimension de l'espace d'entrée augmente, car le nombre de neurones croit alors exponentiellement. C'est pourquoi, en général, on réalise une adaptation aux données. Une solution simple consiste à fixer le nombre de neurones et à déterminer les μ_i correspondants en les répartissant de façon adaptée aux données. Pour ce faire on utilise l'algorithme de classification *k-means*. Celui-ci est implémenté dans R par la fonction `kmeans` de la bibliothèque `mva`. Pour l'appliquer aux réseaux RBF, on détermine les μ_i comme les centres de classes découvert par l'algorithme avec comme nombre de classes le nombre de neurones.

2.4 Exercices

Exercice 2.2

1. Écrire une nouvelle fonction `rbf.design` (`rbf.design.km`) qui positionne les centres en utilisant l'algorithme *k-means*. Dans cette nouvelle version p désigne directement le nombre de neurones. Écrire une nouvelle fonction `rbf` (`rbf.km`) basée sur ce nouveau choix des centres.
2. Comparer les résultats de la nouvelle fonction à l'ancienne, sur des données réelles.

Exercice 2.3

La solution retenue dans l'exercice précédent est un peu trop simpliste car elle n'utilise pas complètement l'adaptabilité des neurones. En effet, on peut choisir σ plus efficacement en l'adaptant à chaque neurone. L'algorithme k-means affecte chaque observation à l'un des centres. Au lieu de calculer σ comme 2 fois la distance moyenne entre les centres, on peut choisir 2 fois l'écart-type de la distance entre les observations affectées à un centre et celui-ci.

Questions :

1. Écrire une nouvelle fonction `rbf.design` (`rbf.design.kms`) qui positionne les centres avec l'algorithme k-means et détermine une valeur de σ par centre, selon la méthode proposée au dessus. Écrire une nouvelle fonction `rbf` (`rbf.kms`) basée sur ce nouveau choix des centres et des σ.
2. Comparer les résultats de la nouvelle fonction à l'ancienne, sur des données réelles.

3 Sélection de modèle

Exercice 3.1

1. Écrire une fonction R qui estime la qualité d'un modèle RBF par validation croisée. Le modèle RBF sera déterminé selon la méthode proposée dans l'exercice 2.3.
2. Déterminer par cette méthode le nombre de neurones optimal pour la modélisation des données `airquality` et `swiss` de R.

Exercice 3.2

1. Modifier la fonction `rbf.kms` afin qu'elle permettre la prise en compte d'une pénalité sur la dérivée seconde de la fonction calculée.
2. Déterminer la validation croisée le meilleur modèle (en utilisant un nombre de neurones importants mais en pénalisant le modèle) pour la modélisation des données `airquality` et `swiss` de R.
3. Dans les deux cas, estimer les performances du modèle par *bootstrap*.