RDF : TP : Classement de caractère manuscrits par Régression Logistique

binôme : Benjamin Ruytoor et Aurore Allart
date : 11 mars 2013

Introduction

Nous avons vu dans le TP précédent, 2 algorithmes de reconnaissance de forme : kNN et Naive Bayes. Dans ce TP, nous allons voir un autre algorithme Régression Logistique. La régression logistique est un modèle de régression binomiale : il s'agit de modéliser l'effet d'un vecteur de variables aléatoires sur variable aléatoire binomiale.

1. Explication du code

Ce code est divisé en 3 parties :

- Chargement des données et initialisation des variables.
- Classement des chiffres par l'algorithme.
- Prédiction de la reconnaissance.

Dans la première partie, nous allons résumer ce que fait le code.

Tout d'abord, on augmente la taille de la pile de Scilab pour éviter un arrêt du programme par manque de place. Ensuite, on charge le fichier contenant les chiffres à classer.

Puis, on calcule les différentes variables qu'on aura besoin par la suite pour afficher les 100 premières images, ainsi que pour les calculs de reconnaissance.

Enfin on affiche les 100 premières images à classer et on attend une réponse de l'utilisateur.

Voici la 2e partie du programme, celle du classement des images. La partie contenant le code permettant le classement des images par la fonction sigmoïde.

Tout d'abord, on supprime l'image affichée. Ensuite, on définit le nombre de classe différente pour le tri, par exemple la classe 1, 2, 3, etc.

Puis, on définit le théta qui sera utiliser dans la fonction 'sigmoid', et on définit le calcul de celle-ci. Après, on code une fonction 'lrCout' qui utilise la fonction 'sigmoid'.

Enfin, on fait appel dans une boucle, à fonction définit précédemment sur toutes les images.

La fonction sigmoïde est la fonction de répartition de la loi logistique. La courbe possède 2 asymptotes en y = 0 et y = 1. Si la fonction retourne 1, cela veut dire que l'image fait parti de cette classe, sinon elle fait partie d'une autres classe, qu'on ne peut déterminer tout de suite.

```
function g=sigmoid(z)
g = 1.0 ./ (1.0 + exp(-z));
endfunction
```

param :
  theta = contient les valeurs de a et b dans l'équation de la droite de Sigmoid
  ind = paramètre non utilisé. Permet à Scilab d'allouer de la mémoire à l'intérieur de son espace de travail
  X = tableau contenant les images
  new_y = tableau qui permet de prendre que les images pour le test par exemple tous les images contenant des 0 (y tableau qui contient la valeur du chiffre de l'image)

return :
  J = cout d'une prédiction
  grad = dérivé partielle de la fonction J
  ind = paramètre non utilisé. Permet à Scilab d'allouer de la mémoire à l'intérieur de son espace de travail

```
function [J, grad, ind]=lrCout(theta, ind, X, new_y)
[X_lignes,X_cols]=size(X);
lambda = 0.1;//Parametre de regularisation
taille_theta = size(theta);
oo = ones(taille_theta(1),taille_theta(2));
oo(1)=0;
J =  (1/X_lignes) * ( -new_y'*log(sigmoid(X*theta)) -(1-new_y)'*log(1-sigmoid(X*theta)))  + (lambda /(2*X_lignes)) * sum(theta.^2);
grad =  (1 / X_lignes) * (X' * (sigmoid(X * theta) - new_y) + lambda*theta.*oo);
endfunction

//boucle qui appelle la fonction précédente
for ii=1:num_labels
    new_y=1*(y==ii);
    costf=list(lrCout, X,new_y);
   [cout, bb]=optim(costf,all_theta(ii,:)',imp=2);
   all_theta(ii,:)=bb';
    //disp()
end
```

Dans la 3e partie, nous abordons l'affichage des images cibles et la prédiction du chiffre.

2. Exécution du programme

Lorsqu'on lance le programme, on a 2 parties : Entrainement et Prédiction. Dans la première partie, l'appel de la fonction optim nous permet de voir le déroulement des itérations pour chaque image.

Voici l'affichage de cette partie :

```
 iter num    1, nb calls=   1, f= 0.6931
 iter num    2, nb calls=   2, f= 0.2639
 iter num    3, nb calls=   4, f= 0.1726
….
 iter num   98, nb calls=  99, f= 0.1328E-01
 iter num   99, nb calls= 100, f= 0.1327E-01
 iter num   99, nb calls= 100, f= 0.1327E-01
***** leaves -qn code-, gradient norm=  0.1960448544058139E-03
Optim s'est arrêté : Le nombre maximum d'appels à f a été atteint.
```

A la fin des itérations, on obtient la norme du gradient, c'est à dire la dérivée partielle pour cette image.

Dans la 2e partie, on obtient le pourcentage de précision avec l'algorithme de la Régression Logistique qui est ici de 96,44 %. On obtient aussi la prédiction de chaque image.

Conclusion

Cette méthode est beaucoup plus efficace que les méthodes kNN et Naïve Bayes (environ 10 % d'erreur).