Práctica 3: Regresión logística multi-clase y redes neuronales

Grupo 13:
David Ortiz Fernández.
Andrés Ortiz Loaiza.

1-Regresión logística multi-clase

En esta primera parte de la práctica hemos desarrollado el algoritmo de regresión logística multiclase y a continuación se lo hemos aplicado al conjunto de datos, el cual consiste en 5000 imágenes de un tamaño 20x20 pixeles con un código de color para cada pixel.

Para ello hemos comenzado cargando los datos, y seguidamente hemos visualizado 100 datos aleatorios. A continuación, hemos usado la función sigmoide para el cálculo de la hipótesis. Para continuar hemos desarrollado una función que se encarga de calcular el coste y realizar el descenso de gradiente dado unos valores de entrada theta y lambda.

También hemos desarrollado una función que se encarga de realizar el enfoque uno contra todos, ya que nos encontramos ante un problema de clasificación multiclase. Esta función, devolverá un vector theta para cada clase.

Flujo principal:

```
clear ;
close all;
%1.1. Visualización de los datos
load('ex3data1.mat');
% almacena los datos leídos en X, y
%m son 5000 elementos
m = size(X, 1);

% Selecciona aleatoriamente 100 ejemplos
rand_indices = randperm(m);
sel = X(rand_indices(1:100), :);

displayData(sel);

printf('Presione enter para continuar.\n');
pause;
%1.2. Vectorización de la regresión logística
%Entrenamiento
num_etiquetas = 10;
lambda = 0.1;

[all_theta] = oneVsAll(X, y, num_etiquetas, lambda);

printf('Presione enter para continuar.\n');
pause;

%1.3. Clasificación de uno frente a todos
%Parte prediccion
m = size(X, 1);
num_labels = size(all_theta, 1);

p = zeros(size(X, 1), 1);
X = [ones(m, 1) X];

%calcular para cada ejemplo de entrenamiento cuál es la probabilidad de que
%pertenezca a cada una de las clases
predict = fsigmoide(X*all_theta');

%&asignamos la etiqueta para la que se obtenga el valor máximo
[predict_max, index_max] = max(predict, [], 2);
prob = index_max;

printf('Predicción: %f\n', mean(double(prob == y)) * 100);
```

Vectorización de la regresión logística regularizada

```
%Función que devuelva el coste y el gradiente de la regresión logística
%regularizada sin utilizar bucles
function [J, grad] = lrCostFunction(theta, X, y, lambda)

m = length(y);
J = 0;
grad = zeros(size(theta));

% calculo de la hipotesis usando funcion sigmoide
h = fsigmoide(X*theta);

% regularizacion de theta eliminando primer valor
%thetaReg = [0;theta(2:end, :);];
theta(1) = 0;
% Vectorización de la función de coste
%J = (1/m)*(-y'* log(h) - (1 - y)'*log(1-h))+(lambda/(2*m))*thetaReg'*thetaReg;
J = (1/m)*sum(-y.*log(h) .-(1.-y).*log(1.-h)) + ((lambda/(2*m))*sum(theta.^2));

%Vectorización del gradiente
grad = (1/m)*(X'*(h-y)+lambda*theta);
```

Función sigmoide:

```
function g = fsigmoide(z)

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

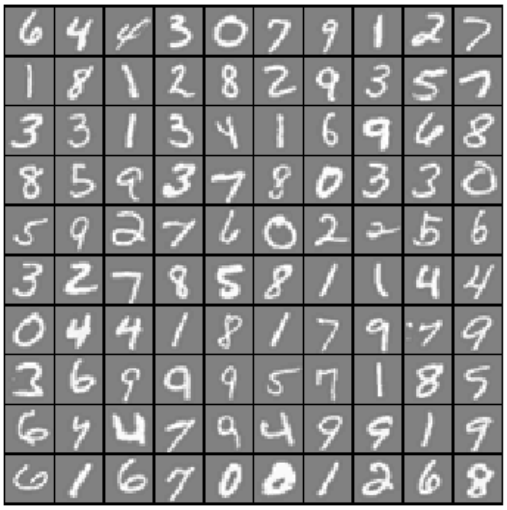
Clasificación de uno frente a todos.

Se implementa usando “fmincg” para aumentar la eficiencia.

```
%función que devuelva una matriz T ? RK×(N+1) donde cada fila de T corresponde
%a los parámetros aprendidos para el clasificador de una de las clases
function [all_theta] = oneVsAll(X, y, num_etiquetas, lambda)
%ONEVSALL entrenavarios clasificadores por regresión logística y devuelve
% el resultado en una matriz all_ t h e t a , donde la fila i-ésima
% corresponde al clasificador de la etiqueta i-ésima

m1 = size(X, 1);
m2 = size(X, 2);

all_theta = zeros(num_etiquetas, m2 + 1);

X = [ones(m1, 1) X];

initial_theta = zeros(m2 + 1, 1);
options = optimset('GradObj', 'on', 'MaxIter', 50);

  for c = 1:num_etiquetas,

   [theta] = fmincg(@(t)(lrCostFunction(t, X, (y==c), lambda)), initial_theta, options);
    all_theta(c, :) = theta';

  endfor

endfunction
```

Resultados:

Visualización por pantalla al utilizar la función displayData:



El resultado de la predicción tras haber entrenado el clasificador es de: 94.960000 %.
Se puede apreciar que el porcentaje de acierto es bastante elevado, siendo muy reducida la cantidad de elementos clasificados incorrectamente.

2 - Redes neuronales

En esta segunda parte de la práctica tendremos una primera toma de contacto con las redes neuronales, para el cálculo de la precisión de la red utilizaremos la función max, como se sugiere en el guion de la práctica, tras aplicar la propagación hacia delante.

```
close all;
clear ;

load('ex3data1.mat');
m = size(X, 1);
% El ficherochero ex3weights.mat contiene las matrices T(1) y T(2) con el
%resultado de haber entrenado la red neuronal
load('ex3weights.mat');
% Theta1 es de dimensión 25 x 401
% Theta2 es de dimensión 10 x 26

%computar el valor de h?(x(i)) paracada ejemplo i. De la misma forma que en la
%regresión logística, interpretaremos que la clase asignada por la red neuronal
%a un ejemplo es la correspondiente a la salida de la red con el máximo valor
X = [ones(size(X,1),1) X];

h = fsigmoide(X * Theta1');
h = [ones(size(h,1),1) h];

ma = fsigmoide(h * Theta2');
[x, pre] = max(ma');

pre = pre';
printf('La precisión de la red neuronal es: %f\n', mean(double(pre == y))*100);
```

Resultados:

La precisión de la red neuronal es: 97.520000

Se observa que la precisión de la red neuronal es superior incluso el de la regresión logística.

Para el cálculo de la precisión se ha utilizado la función mean.