Práctica 4

En esta práctica seremos nosotros mismos los que entrenaremos la red neuronal. Para ello necesitamos una serie de funciones:

Función sigmoide:

```
function [g] = sigmoide(z)
g = 1 ./ ( 1 + e .^-z) ;
endfunction
```

Función para hallar la derivada de la sigmoide:

```
function [res] = derivSig (z)
res = sigmoide(z) .* (1-sigmoide(z));
endfunction
```

Función para la inicialización aleatoria de los vectores de pesos:

```
function W = pesosAleatorios ( L_in, L_out )
ini = 0.12;
W = rand(L_out, L_in+1) .*(2*ini) - ini;
endfunction
```

Función de coste y gradiente:

```
function [ J, grad ] = costeRN ( params_rn , num_entradas , num_ocultas , num_etiquetas , X, y ,
        lambda )
Theta1 = reshape(params_rn(1:num_ocultas*(num_entradas+1)), num_ocultas, (num_entradas+1) );
Theta2 = reshape(params_rn((1+( num_ocultas*(num_entradas+1))):end), num_etiquetas,
        (num_ocultas+1) );
m = rows(y);
X = [ ones(m,1) X ];
z2 = X * Theta1';
a2 = sigmoide(z2);
a2 = [ ones(rows(a2),1) a2];
hTheta = sigmoide(a2 * Theta2');
Y=zeros(m,num_etiquetas);
for i=1:m
  Y(i,y(i))=1;
endfor
J = sum(sum( (-Y) .* log(hTheta)  -  (1-Y) .* log(1-hTheta) ));
J = (J + (lambda/2)*(sum(sum(Theta1(:,2:end))) + sum(sum(Theta2(:,2:end)))))/m;
Theta2(:,1)=0;
sigma3 = hTheta - Y;
delta2 = sigma3' * a2;
grad2 = (delta2/m) + (lambda/m) * Theta2;

Theta1(:,1)=0;
sigma2 = (sigma3 * Theta2)(:,2:end) .* derivSig(z2); #sigma 2
delta1 = sigma2' * X; #delta 1
grad1 = (delta1/m) + (lambda/m) * Theta1;

grad= [grad1(:);grad2(:)];

endfunction
```

Función para entrenar la red usando fmincg:

```
function [ precision ]= entrenaRed(X,y,lambda)

num_entradas = 400;
num_ocultas = 25;
num_etiquetas = 10;

Theta1=pesosAleatorios(num_entradas,num_ocultas);
Theta2=pesosAleatorios(num_ocultas,num_etiquetas);
all_theta=[Theta1(:);Theta2(:)];

options = optimset('MaxIter', 50);
coste = @(p) costeRN(p, num_entradas, num_ocultas, num_etiquetas, X, y,lambda);
[params_rn, cost] = fmincg(coste, all_theta, options);

Theta1 = reshape(params_rn(1:num_ocultas*(num_entradas+1)), num_ocultas, (num_entradas+1) );
Theta2 = reshape(params_rn((1+( num_ocultas*(num_entradas+1))):end), num_etiquetas,
        (num_ocultas+1) );

levelOne = sigmoide(X * Theta1(:,2:end)');
levelOne = [ ones(rows(levelOne),1) levelOne];
probabilities = sigmoide(levelOne * Theta2');

[trash , prediccion] = max(probabilities, [], 2);

precision=mean(double(prediccion== y)) * 100;
endfunction
```

Ejecutando varias veces esa función podemos ver como los porcentajes de acierto fluctúan entre el 95% y el 96%, dada la aleatoriedad de la inicialización de las matrices de pesos.

Daniel Bastarrica Lacalle