UNIVERSIDADE FEDERAL DE MATO GROSSO
CAMPUS DE VÁRZEA GRANDE
INSTITUTO DE ENGENHARIA

Inteligência Artificial – 2017/2
Prof. Raoni F. S. Teixeira

Atividade Prática 5: Redes Neurais
Discente: Ricardo Gonçalves de Aguiar

1 Introdução

Esta atividade foi desenvolvida na disciplina de inteligência artificial. O principal objetivo da atividade foi implementar um algoritmo para reconhecimento de dígitos em imagens utilizando uma rede perceptron multicamadas.

2 Resumo

A primeira fase da aplicação é denominada de (Forward)"Propagação para frente", na qual os sinais ($x_1 ... x_{400}$) de uma amostra do conjunto de treinamento são inserida na entrada da rede e são propagado camada a camada até a produção das respectivas saídas. Portanto, a aplicação desta fase vista somente para obter as respostas da rede, levando-se em consideração que os parâmetros, os quais permaneçam inalterado durante cada execução desta fase.

A Rede neural artificial implementada constitui-se de três camadas (Entrada, oculta, saída). A camada de entrada recebe cada um dos pixels de uma determinada imagem mais o valor do bias 1, totalizando 401 valores na entrada. A camada intermediária recebe 25 entradas mais o valor do bias, totalizando 26 na camada intermediária. A camada de saída constitui 10 unidades, uma para cada digito. Os parâmetros(pesos) da rede são representados por Θ_1 e Θ_2. A camada oculta recebe os dados da camada de entrada multiplicado com seu respectivo parâmetro Θ_1. A camada de saída recebe os dados da camada oculta multiplicado por seu respectivo parâmetro Θ_2.
Observe o Pseudocódigo abaixo.

Algorithm 1 Forward

function [E,A1,Z1,A2,Z2,] = FORWARD(Theta1,Theta2,X)
e(b;X) ▷ dados da entrada + bias
a1 = Theta1 * e
z1 = sigmoid(a1) ▷ threshold
s(b;z1)
a2 = Theta2 * s
z2 = sigmoid(a2) ▷ threshold
end function

O Pseudocódigo acima representa a propagação dos sinais já ponderados com seus respectivos parâmetros.

A função de Threshold utilizada na saída de cada neurônio da rede. Trata-se da função sigmoid, definida como:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

Logo em seguida, as respostas produzidas pelas saídas da rede são comparadas com as respectivas respostas desejada, se as respostas não forem iguais as saídas obtidas, são produzidos os desvios(erros) entre as respostas desejado e aquela produzida pelos neurônios da saída, os quais serão subsequentemente utilizados para ajustar os pesos de todos os seus neurônios.

Assim, em função desses desvios(erros), aplica-se, em seguida, a segunda fase do método, denominada de "propagação reversa"(Backpropagation).

Denotamos por δ^l o erro acumulado na l-ésima camada. Assim, podemos denotar por:

$$\delta^{(3)} = S^{(3)} - Y \tag{2}$$

Onde S^3 representa as resposta produzida pela saída da rede e Y a respectiva resposta desejada. Logo, os erros das camadas anteriores são calculados da seguinte forma:

$$\delta^{(2)} = (((\theta^{(2)})^T . * \delta^{(3)}) * \sigma'(z^{(2)})) \tag{3}$$

Onde $z^{(2)}$ é a saída da camada oculta sem o bias e sem aplicação da função sigmoid, ou seja, $z^{(2)} = \theta^{(1)} * X^{(1)}$, sendo $X^{(1)}$ os dados de entrada. A derivada da função sigmoid foi implementada utilizando o seguinte Pseudocódigo.

Algorithm 2 sigmoidGradient

function [D] = SIGMOIDGRADIENT(z)
    s = sigmoid(z)
    D = s.*(1 - s)
end function

Os erros são acumulados e o gradiente constituinte é obtido:

$$\Delta^{(l)} = \delta^{(l+1)} * (a^{(l)})^T \tag{4}$$

$$\frac{\partial J}{\partial \Theta^{(l)}} = \frac{1}{m}\Delta^{(l)} \tag{5}$$

O código abaixo representa os passos descrito acima.

```
% backpropagation

[~,y3,z2,entrada,y2] = predict(Theta1,Theta2,X);

E3 = (y3 - Y');
ax = (Theta2')*E3;
ax = ax(2:end, :);
% size(ax);
% size(sigmoidGradient(z2));
E2 = ax.*sigmoidGradient(z2)';
% size(E2);

Theta1_grad = (1/m)*(E2*entrada');
Theta2_grad = (1/m)*(E3*y2');
```

Figura 1

3 Resultados e Discussões

Os resultados obtidos são exibidos abaixo.

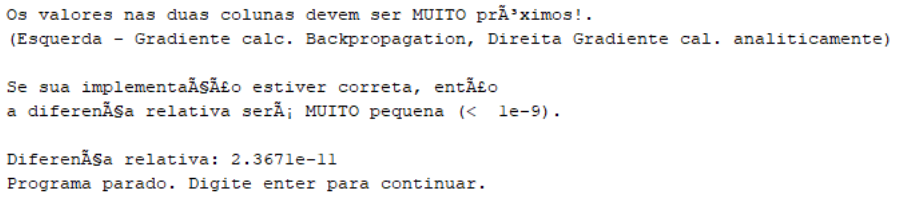

Figura 2

A figura acima representa o teste realizado com o Backpropagation sem utilizar a regularização.

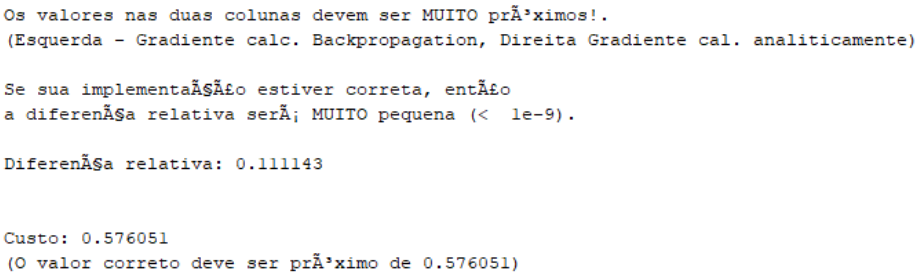

Figura 3

A figura acima representa o teste realizado com o Backpropagation utilizando a regularização.

Observa-se que o valor de custo é idêntico ao seu valor esperado, o que mostra que a implementação do custo está correta.

Logo em seguida é testado a implementação do algoritmo Backpropagation. Os testes executados foram: sem a regularização ($\lambda = 0$) e com a regularização ($\lambda = 3$). Podemos notar na (figura 2) que o Backpropagation sem a regularização obteve bons resultados, sendo o objetivo obter uma diferença relativa inferior a 10^{-9}. Já o Backpropagation com a regularização (figura 3) não obteve o resultado esperado, uma das possíveis causas, mal implementado.