Instituto Federal de Educação, Ciência e Tecnologia – IFB
Campus Taguatinga
Curso Tecnológico em Automação Industrial
Disciplina: Aprendizagem de Máquina
professor Lucas Moreira

Trabalho 03

O objetivo do segundo trabalho é familiarizar o(a) aluno(a) com a Rede Neural Artificial e seu algoritmo de treinamento.

Nesse trabalho, o(a) aluno(a) irá implementar um programa computacional que implementa uma RNA e calcula seus pesos a partir do algoritmo de treinamento *backpropagation*.

Dados

Os dados usados nesse trabalho serão os mesmos utilizados no trabalho de regressão logística: um conjunto de 5.000 imagens em tons de cinza de caracteres numéricos escritos a mão de 20 x 20 pixels de dimensão.

Esses dados estão disponíveis no arquivo "data.mat" e já estarão disponíveis na forma de vetor, ou seja, cada imagem está representada em um único vetor linha com 400 elementos, onde cada elemento do vetor corresponde ao valor numérico de cada pixel da imagem, e o conjunto total de dados de treinamento será formado pela matriz **X** de dimensões 5000 x 400.

Por ser um problema de classificação usando um algoritmo de treinamento supervisionado, as classes correspondentes a cada caracter são fornecidas no vetor **y**, incluído em data.mat. A ordem dos elementos de **y** corresponde à mesma ordem das linhas de **X**, portanto, a primeira linha de **X** conterá os pixels de uma imagem, e o primeiro elemento de **y** terá a classe ao qual essa imagem corresponde. O vetor y, portanto, terá seus elementos com os valores correspondentes a cada caracter numérico, assim, se a imagem de índice 3500 representa o caracter 7, por exemplo, então $y^{(3500)} = 7$. A única exceção são imagens correspondendo ao caracter 0 (zero), que terão o valor numérico correspondente no vetor **y** iguais a 10. Isso facilita a criação do vetor de treinamento e impede codificação extra, visto que o Matlab/Octave tem os índices de suas matrizes iniciando em 1 (um).

Problema

Deseja-se treinar uma RNA que consiga reconhecer uma imagem digital de 20 x 20 pixels em tons de cinza em diferentes valores numéricos. O número de neurônios da camada de entrada é definido pelo número de variáveis do problema, nesse caso 400 (não considerando o neurônio de *bias*). O número de neurônios da camada de saída deve ser o número de classes do problema, nesse caso 10 (os caracteres de 0 a 9).

O número de camadas ocultas, e o número de neurônios em cada uma delas, são decididos pelo projetista. No seu programa, pode-se usar somente uma camada oculta, entretanto, essa camada deve aceitar qualquer quantidade de neurônios, a ser modificado com apenas uma variável no seu script principal.

Para efeito de visualização, foi disponibilizado a função *displayData( )*, que mostrará 100 imagens da tabela de treinamento.

O trabalho deve ser constituído por 3 (três) arquivos Matlab/Octave (com extensão .m) distintos. Um deles já foi disponibilizado (*trabalho2.m*), e deve ser preenchido pelo(a) aluno(a) a fim de cumprir o requisitado.

São eles:

- O script principal *trabalho2.m*, que deverá ser executado para realizar a regressão logística;
- Um script de função para cálculo da derivada da função sigmoid;
- Um script de função para cálculo da função custo e suas derivadas. Todos os cálculos devem ser regularizados.

Dicas

1- A função custo, em resumo, é a diferença do valor esperado de saída $\boldsymbol{y}$ e o valor da hipótese (o valor de saída da sua RNA). Para calcular essa função, portanto, você deve realizar a propagação direta dos dados de treinamento, multiplicando-os pelos pesos e passando pela função de ativação das camadas seguintes.

2- A função de ativação dos neurônios é a função sigmoid, que deve receber um parâmetro de entrada e retornar um valor de saída. Caso o parâmetro de entrada seja um vetor, a função deve retornar um vetor de mesma dimensão, contendo o valor da função sigmoid de cada elemento. Para facilitar a escrita do programa, sugere-se escrever a função de ativação em um script separado.

3- A função custo não regularizada é dada por:

$$J(\theta)=-\left[\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)}\cdot\log\left(h_\Theta\left(x^{(i)}\right)\right)_k+\left(1-y_k^{(i)}\right)\cdot\log\left(1-h_\Theta\left(x^{(i)}\right)_k\right)\right]$$

Onde $h_\Theta(x^{(i)})$ é calculada pela propagação direta, lembrando que para cada matriz , deve ser inserido o neurônio de bias. No caso desse problema, K = 10, e os vetores ***y*** fornecidos são o rótulo da classe a que pertence uma determinada amostra, ou seja, *y* é um valor escalar (y = 1, 2, … , 10). Como a camada de saída terá um neurônio para cada classe, você deve transformar esse escalar $y^{(i)}$ em um vetor 10 x 1:

$$y=\begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix},\quad\begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix},\quad\begin{bmatrix}0\\0\\1\\\vdots\\0\end{bmatrix},\quad\cdots\quad,\quad ou\quad\begin{bmatrix}0\\0\\0\\\vdots\\1\end{bmatrix}$$

Ou seja, seu vetor de treinamento ***y*** será agora uma matriz de treinamento $\mathbf{Y}_{m\ x\ K}$. Implemente essa transformação dentro do script que calcula a função custo.

4- A função-custo regularizada possui um termo a mais na equação anterior:

$$\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2$$

Esse termo pode ser calculado de forma independente, e somado ao valor não regularizado. Como a RNA terá apenas uma camada intermediária, e a camada de saída possui 10 neurônios, o termo acima fica:

$$\frac{\lambda}{2m}\left[\sum_{j=1}^{s_l}\sum_{i=1}^{400}\left(\Theta_{ji}^{(1)}\right)^2+\sum_{j=1}^{10}\sum_{i=1}^{s_l}\left(\Theta_{ji}^{(2)}\right)^2\right]$$

Onde s_l é o número de neurônios na camada escondida.

5- A derivada da função sigmoid é usada durante a etapa do *backpropagation*. Ela é calculada por:

$$g'(z)=\frac{d}{dz}g(z)=g(z)(1-g(z))$$

Para debugar essa função, teste-a com valores de entrada conhecidos. Para valores com módulo muito grandes (tanto positivos quanto negativos), o valor da derivada deve ser próximo de zero. Para z = 0, a derivada deve ser exatamente 0,25. Para vetores ou matrizes, a função dever retornar o valor de cada elemento, ou seja, vetores e matrizes de mesma dimensão.

6- A função custo deve receber como parâmetros a matriz de dados de entrada **X**, o vetor de valores alvo **y** (lembrando que a RNA é um algoritmo de treinamento supervisionado), as matrizes de parâmetros **Θ** e o parâmetro de regularização λ, e deve retornar o valor do custo para esse conjunto de parâmetros, e um vetor coluna contendo todas as derivadas parciais da função. Para facilitar, e agilizar, o cálculo da função custo, é **obrigatório** o uso de operações algébricas no lugar de loops de repetição, como o *for*. Por isso, a matriz **X** deve conter uma coluna adicional na primeira posição, formada somente pelo valor 1. O nome do arquivo deve ser o mesmo nome da função, acrescida da extensão .m.

7- O início do algoritmo de treinamento exige um chute inicial para os valores dos pesos. Utilize a função *randInitializeWeights( )* para iniciar esses valores.

8- Uma vez codificada a função que retorna a função-custo e suas derivadas parciais, pode-se usar a função *fminunc( )*, para estimar os valores ótimos de **Θ**.

Entrega

Os 3 (três) arquivos citados acima, devem ser compactados em um único arquivo .zip e enviados para o endereço de e-mail lucas.moreira@ifb.edu.br até às 23h55min do dia 11 de novembro de 2018.

O trabalho é individual, e deve ser entregue um arquivo compactado, contendo os 3 (três) scripts solicitados, por aluno(a).

Cópias de trabalhos receberão nota 0 (zero).

Em caso de dúvidas, pede-se que entrem em contato pelo endereço de e-mail acima.