Instituto Federal de Educação, Ciência e Tecnologia – IFB
Campus Taguatinga
Curso ABI em Ciências da Computação
Disciplina: Inteligência Computacional
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 arquivos de dados train.csv e test.csv contêm imagens em tons de cinza de dígitos desenhados à mão, de zero a nove. Cada imagem tem 28 pixels de altura e 28 pixels de largura, totalizando 784 pixels no total. Cada pixel tem um único valor de pixel associado a ele, indicando o tom desse pixel, com números mais altos significando mais escuro. Esse valor de pixel é um inteiro entre 0 e 255, inclusive.

O conjunto de dados de treinamento, (train.csv), tem 785 colunas. A primeira coluna, chamada "label", é o dígito que foi desenhado pelo usuário. O restante das colunas contém os valores de pixel da imagem associada.

Cada coluna de pixel no conjunto de treinamento tem um nome como pixelx, onde x é um inteiro entre 0 e 783, inclusive. Para localizar esse pixel na imagem, suponha que tenhamos decomposto x como x = i * 28 + j, onde i e j são inteiros entre 0 e 27, inclusive. Em seguida, o pixelx está localizado na linha i e na coluna j de uma matriz 28 x 28 (indexação por zero).

Visualmente, se omitirmos o prefixo "pixel", os pixels comporão a imagem da seguinte forma:

000 001 002 … 026 027
028 029 030 … 054 055
| | | … | |
728 729 730 … 754 755
756 757 758 … 782 783

Para avaliar a eficiência do algoritmo, pode-se usar o conjunto de dados de teste, (test.csv), que possui o mesmo formato que o conjunto de treinamento, exceto que ele não contém a coluna "label".

Problema

Deseja-se treinar uma RNA que consiga reconhecer uma imagem digital de 28 x 28 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 784 (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 (aluno).

Para o desenvolvimento do algoritmo, poderá ser usado as bibliotecas TensorFlow (www.tensorflow.org) ou PyTorch (www.pytorch.org). Nesse caso, o código-fonte entregue deve conter a explicação completa, em forma de comentário, das funcionalidades da função utilizada.

Dicas

1- A função custo, em resumo, é a diferença do valor esperado de saída ***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:

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

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 sugerido 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 valores aleatórios (pequenos).

8- Uma vez codificada a função que retorna a função-custo e suas derivadas parciais, deve-se utilizar uma função de gradiente descendente para estimar os valores ótimos de **Θ**.

Entrega

Todos os arquivos necessários para a correta execução do algoritmo, 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 16 de junho de 2019.

Serão aceitos grupos de até 2 (dois) alunos(as).

Cópias de trabalhos receberão nota 0 (zero).

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