Department of Electrical and Computer Engineering
University of Delaware

FSAN/ELEG815 Analytics I: Statistical Learning

Homework #7, Fall 2019

Name: ______________________________________

1. **Feedforward Propagation and Prediction**. Implement feedforward propagation for a sigmoidal neural network and return neural network's prediction for digit recognition. In 'Weights.mat', you are given a set of parameters of a neural network with 3 layers: an input layer, a hidden layer with 25 units and an output layer with 10 units corresponding to the 10 digits classes. Your inputs are 20×20 pixel values of digit images i.e. 400 input units (excluding the extra bias). **The prediction from the neural network will be the label that has he largest output**. Report the accuracy of the neural network using the data set 'DatasetDigit.mat'. Note: use activation function $\theta(s) = \frac{1}{1+e^{-s}}$.

   Dataset description: $\mathbf{X}$ contains 20×20 pixel values of 5000 data samples. $\mathbf{y}$ contains the digit label of the 5000 data samples.

Forward propagation to compute $h(\mathrm{x})$:

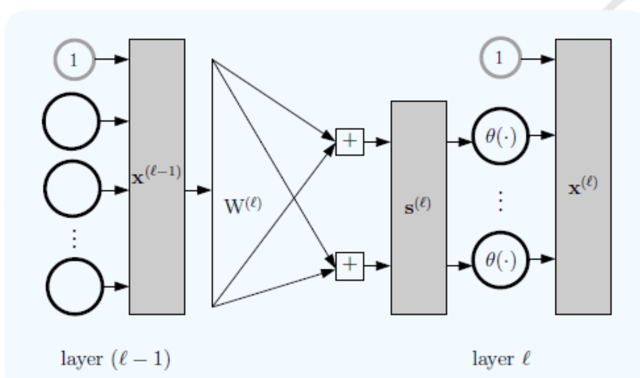
$\mathrm{x}^{(0)} \leftarrow \mathrm{x}$ **[Initialization]**

for $\ell = 1$ to L **do** **[Forward Propagation]**

  $\mathrm{s}^{(\ell)} \leftarrow (\mathrm{W}^{(\ell)})^{\mathrm{T}} \mathrm{x}^{(\ell-1)}$

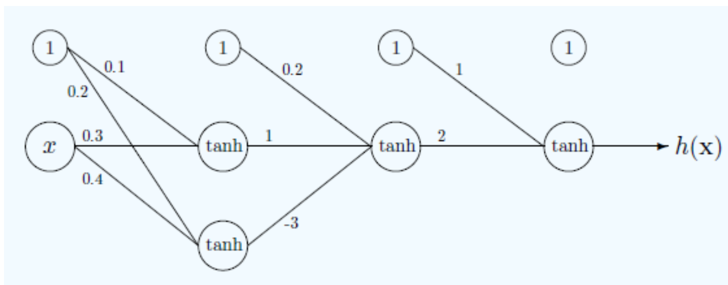
  $\mathrm{x}^{(\ell)} \leftarrow \begin{bmatrix} 1 \\ \theta(\mathrm{s}^{(\ell)}) \end{bmatrix}$

$h(\mathrm{x}) = \mathrm{x}^{(L)}$ **[Output]**



2. **Backpropagation**. Implement the backpropagtion algorithm to compute the partial derivatives that are needed for the gradient. Test your algorithm using the Example 7.1 of the e-chapter. In this exercise, there is a single input $x = 2$, $y = 1$ and the weight matrices are:

$$W^{(1)} = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix}; W^{(2)} = \begin{bmatrix} 0.2 \\ 1 \\ -3 \end{bmatrix}; W^{(3)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$



Backpropagation to compute sensitivities $\delta^{(\ell)}$.
Input: a data point (x, y).

0: Run forward propagation on x to compute and save:

$$\begin{array}{ll} \mathrm{s}^{(\ell)} & \text{for } \ell = 1, \ldots, L; \\ \mathrm{x}^{(\ell)} & \text{for } \ell = 0, \ldots, L. \end{array}$$

1: $\delta^{(L)} \leftarrow 2(x^{(L)} - y)\theta'(s^{(L)})$ **[Initialization]**

$$\theta'(s^{(L)}) = \begin{cases} 1 - (x^{(L)})^2 & \theta(s) = \tanh(s); \\ 1 & \theta(s) = s. \end{cases}$$

2: **for** $\ell = L - 1$ to 1 **do** **[Back-Propagation]**

3: Let $\theta'(\mathrm{s}^{(\ell)}) = \left[1 - \mathrm{x}^{(\ell)} \otimes \mathrm{x}^{(\ell)}\right]_1^{d^{(\ell)}}$.

4: Compute the sensitivity $\delta^{(\ell)}$ from $\delta^{(\ell+1)}$:

$$\delta^{(\ell)} \leftarrow \theta'(\mathrm{s}^{(\ell)}) \otimes \left[\mathrm{W}^{(\ell+1)} \delta^{(\ell+1)}\right]_1^{d^{(\ell)}}$$

3. (**Gradient Descent**) Revisit the handwritten digit recognition problem. Separate digit 1 from all the other digits, using intensity and symmetry as your inputs variables like you did before. Using the gradient descent, learn the parameters of a neural network with one hidden layer and 10 hidden nodes (41 weights) on 500 randomly chosen data points. Use activation function $\theta(s) = \tanh(s)$ for the hidden layer and the identity $\theta(s) = s$ for the output. Use the same data set sent to you before,

- Run gradient descent for 10000 iterations and record the time.
- Report your final weights.
- Show a plot of E_{in} and E_{out} for each iteration of the gradient descent.
- Report final E_{in} and E_{out} using all data points.
- Plot your original data set and your predictions. Compare your results for the training and testing data set.

Note: You are not allowed to use prebuilt functions.

Algorithm to Compute $E_{\text{in}}(\mathrm{w})$ and $\mathrm{g} = \nabla E_{\text{in}}(\mathrm{w})$.
Input: $\mathrm{w} = \{\mathrm{W}^{(1)}, \ldots, \mathrm{W}^{(L)}\}$; $\mathcal{D} = (\mathrm{x}_1, y_1) \ldots (\mathrm{x}_N, y_n)$.
Output: error $E_{\text{in}}(\mathrm{w})$ and gradient $\mathrm{g} = \{\mathrm{G}^{(1)}, \ldots, \mathrm{G}^{(L)}\}$.

Initialize: $E_{\text{in}} = 0$ and $\mathrm{G}^{(\ell)} = 0 \cdot \mathrm{W}^{(\ell)}$ for $\ell = 1, \ldots, L$.
for Each data point (x_n, y_n), $n = 1, \ldots, N$, **do**
  Compute $\mathrm{x}^{(\ell)}$ for $\ell = 0, \ldots, L$. [forward propagation]
  Compute $\delta^{(\ell)}$ for $\ell = L, \ldots, 1$. [backpropagation]
  $E_{\text{in}} \leftarrow E_{\text{in}} + \frac{1}{N}(\mathrm{x}^{(L)} - y_n)^2$.
  for $\ell = 1, \ldots, L$ **do**
    $\mathrm{G}^{(\ell)}(\mathrm{x}_n) = [\mathrm{x}^{(\ell-1)}(\delta^{(\ell)})^{\mathrm{T}}]$
    $\mathrm{G}^{(\ell)} \leftarrow \mathrm{G}^{(\ell)} + \frac{1}{N}\mathrm{G}^{(\ell)}(\mathrm{x}_n)$

$G^{(l)}(\mathbf{x}_n)$ is the gradient on data point $\mathbf{x}_n$. Weight update for a single iteration of the gradient descent is:

$$W^{(l)} \leftarrow W^{(l)} - \eta G^{(l)} \quad \text{for} \quad l = 1, ..., L$$

4. (**Stochastic Gradient Descent**). Repeat exercise 4 but use stochastic gradient descent instead.

   - Run stochastic gradient descent for 10000 iterations and record the time.
   - Report your final weights.
   - Report final E_{in} and E_{out} using all data points. Compare to the error obtained in Exercise 4.
   - Plot your original data set and your predictions. Compare your results for the training and testing data set.

   Note: You are not allowed to use prebuilt functions.