CS 6375- ASSIGNMENT 3

Please read the instructions below before starting the assignment.

- This assignment consists of written problems (Part I) and programming part (Part II). Please create separate folders named **parti** and **partii** and zip them together for submission.

- You should use a cover sheet, which can be downloaded at: http://www.utdallas.edu/~axn112530/cs6375/CS6375_CoverPage.docx

- You are allowed to work in pairs i.e. a group of two students is allowed. Please write the names of the group members on the cover page.

- You have a total of 4 free late days for the entire semester. You can use at most 2 days for any one assignment. After that, there will be a penalty of 10% for each late day. The submission for this assignment will be closed 2 days after the due date.

- **For part I:**
  - You can submit a scanned handwritten or typed solution. If handwritten, it should be legible and clear.
  - Show your calculations and steps clearly.

- **For part II:**
  - Before starting the assignment, please read about perceptron, ANN, and backpropagation algorithm from chapter 4 of Tom Mitchell's textbook and other links mentioned at the end.

  - You are free to use any of the following programming languages for Part II - Java, Python, C++, Ruby. <u>You cannot use any machine learning packages or libraries, but are free to use other math or stats, or data parsing libraries.</u>
  The instructions for compiling and running your code must be specified in the README file.

  - Please include a report that briefly describes your approach, your experiments, and the best results obtained.

- **<u>Your code will be checked for plagiarism. Do not copy code from online resources.</u>**

CS 6375 – ASSIGNMENT 3

Part I – Questions (30 points)

Solve the following questions:

1. In class we had derived the backpropagation algorithm for the case where each of the hidden and output layer neurons used the sigmoid activation function:

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

Revise the backpropagation algorithm for the case where each hidden layer neuron uses the *tanh* activation function

$$f(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

and output layer neuron uses the identity function:

$$f(x) = x$$

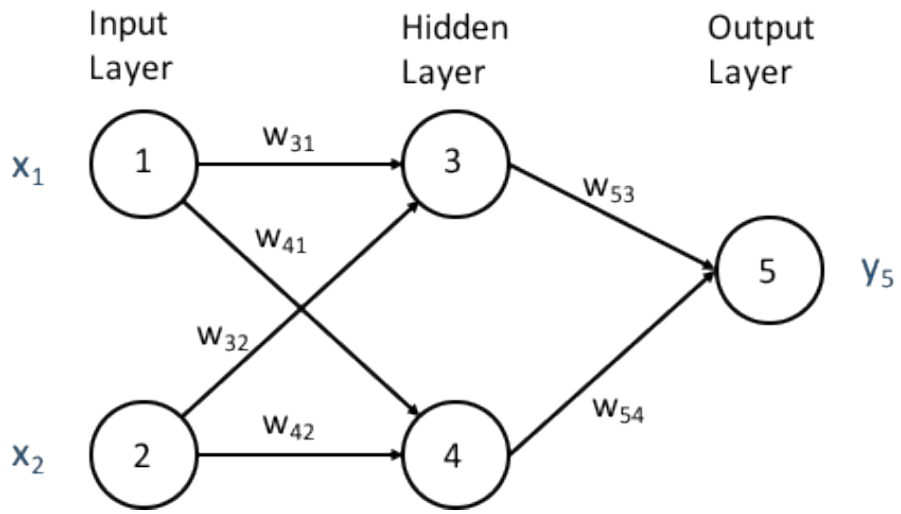
Show all steps of your derivation.

2. Derive a gradient descent training rule for a single unit neuron with output o, defined as:

$$o = w_0 + w_1(x_1 + x_1^2) + \cdots + w_n(x_n + x_n^2)$$

where x_1, x_2, …, x_n are the inputs and w_1, w_2, …, w_n are the corresponding weights. You can assume an identity activation function i.e. f(x) = x.
Show the steps in your derivation.

3. Consider a neural net with 2 input layer neurons, one hidden layer with 2 neurons, and 1 output layer neuron as shown below:



Assume that the input layer uses the identity activation function i.e. f(x) = x, and each of the hidden layers uses an activation function h. The weights of each of the connections are marked above.

a. Write down the output of the neural net (y_5) in terms of the weights and inputs x_1 and x_2, and activation function h.

b. Suppose we use vector notation as follows:

$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$W^{(1)} = \begin{pmatrix} w_{31} & w_{32} \\ w_{41} & w_{42} \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} w_{53} & w_{54} \end{pmatrix}$$

Write down the output of the neural net in terms of the above vectors and the activation function.

c. Now suppose that there are two choices for h:

Sigmoid: $h_1(x) = \frac{1}{1+e^{-x}}$

Squashing function: $h_2(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Show that neural nets created using the above two activation functions can generate the same function.

Hint: First compute the relationship between h_1 and h_2 and then show that the output functions are same, with the parameters differing only by linear transformations and constants.

4. Solve question 4.10 from Tom Mitchell's textbook

Part II - Programming Assignment (70 points)

In this assignment you will implement the backpropagation algorithm for Neural Networks. and test it on various real world datasets. Before running the algorithm, you will have to ensure that all the features of the dataset are properly scaled, standardized, and categorical variables are encoded using numerical values.

The two steps of this assignment are listed below. You have to create separate class for each of the steps.

1. Pre-processing:

Pre-processing involves checking the dataset for null or missing values, cleansing the dataset of any wrong values, standardizing the features and converting any nominal (or categorical) variables to numerical form. This step is essential before running neural net algorithm, as they can only accept numeric data and work best with scaled data.

The arguments to this part will be:

- complete input path of the raw dataset
- complete output path of the pre-processed dataset

Your pre-processing code will read in a dataset specified using the first command line argument and first check for any null or missing values. You will remove any data points (i.e. rows) that have missing or incomplete features.

Then, you will perform the following for each of the features (independent variables):

- If the value is numeric, it needs to be standardized, which means subtracting the mean from each of the values and dividing by the standard deviation.
See here for more details: https://en.wikipedia.org/wiki/Feature_scaling#Standardization

- If the value is categorical or nominal, it needs to be converted to numerical values.

For example, if the attribute is gender and is encoded as "male" or "female", it needs to be converted to 0 or 1. You are free to figure out specifics of your encoding strategy, but be sure to mention it in the report.

For the output or predicted variable, if its value is numerical or continuous, you are free to convert it into categorical or binary variables (classification problem) or keep it as it is (regression problem). You have to specify this clearly in your report.

After completing these steps, you need to save the processed dataset to the path specified by the second command line argument and use it for the next step.

2. Training a Neural Net:

You will use the processed dataset to build a neural net. The input parameters to the neural net are as follows:

- input dataset – complete path of the post-processed input dataset
- training percent – percentage of the dataset to be used for training
- maximum_iterations – Maximum number of iterations that your algorithm will run. This parameter is used so that your program terminates in a reasonable time.
- number of hidden layers
- number of neurons in each hidden layer

For example, input parameters could be:

ds1 80 200 2 4 2

The above would imply that the dataset is ds1, the percent of the dataset to be used for training is 80%, the maximum number of iterations is 200, and there are 2 hidden layers with (4, 2) neurons. Your program would have to initialize the weights randomly. Remember to take care of the bias term (w_0) also.

While coding the neural network, you can make the following assumptions:
- the activation function will be sigmoid
- you can use the backpropagation algorithm described in class and presented in the textbook
- the training data will be randomly sampled from the dataset. The remaining will form the test dataset
- you can use the mean square error as the error metric
- one iteration involves a forward and backward pass of the back propagation algorithm
- your algorithm will terminate when either the error becomes 0 or the max number of iterations is reached.

After building the model, you will output the model parameters as below:

Layer 0 (Input Layer):
    Neuron1 weights:
    Neuron 2 weights:
    ..
Layer 1 (1^{st} hidden layer):
    Neuron1 weights:
    Neuron 2 weights:
    ..
Layer n (Last hidden layer):
    Neuron1 weights:
    Neuron 2 weights:
    ..

Total training error = ….

You will also apply the model on the test data and report the test error:

Total test error = ….

Testing your program

You will test both parts of your program, first the pre-processing part and then the model creation and evaluation, on the following datasets:

1. Car Evaluation Dataset
https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

2. Iris dataset
https://archive.ics.uci.edu/ml/datasets/Iris

3. Adult Census Income dataset
https://archive.ics.uci.edu/ml/datasets/Census+Income

For each of the above 3, you have to run your code and report the analysis and results in a separate folder. The analysis should contain your pre-processing strategy, your best set of parameters, and the best results in terms of accuracy. You should also keep a log of all your experiments, which should list how many different combination of parameters you tested.

Helpful Links:

1. https://visualstudiomagazine.com/articles/2014/01/01/how-to-standardize-data-for-neural-networks.aspx
2. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
3. Data pre-processing
https://www.mimuw.edu.pl/~son/datamining/DM/4-preprocess.pdf
4. http://neuralnetworksanddeeplearning.com/chap2.html