Machine Learning

Assignment 2: Neural Networks

Due Date: Indicated on eLearning

Instructions

- There are two parts to this assignment. The first part requires you to solve some theoretical/numerical questions, and the second part requires you to code a neural network.
- For the programming part, please use parameters and not hard coded paths or values. All instructions for compiling and running your code must be placed in the README file.
- All work submitted must be your own. Do not copy from online sources. If you use any references, please list them.
- Please write the names and netids of all students on the front page.
- 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 four days have been used up, 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.**
- Please ask all questions on Piazza, not via email.

1 Theoretical Part (40 points)

For the following, please show all steps of your derivation and list any assumptions that you make. You can submit typed or **legible** hand-written solutions. If the TA cannot read your handwriting, no credit will be given.

1.1 Gradient Descent

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 $x_1, x_2, \ldots, x_n$ are the inputs, $w_1, w_2, \ldots, w_n$ are the corresponding weights, and w_0 is the bias weight. Show all steps of your derivation and the final result for weight update. You can assume a learning rate of η.

1.2 Comparing Activation Function

Consider a neural net with 2 input layer neurons, one hidden layer with 2 neurons, and 1 output layer neuron as shown in Figure 1. Assume that the input layer uses the identity activation function i.e. $f(x) = x$, and each of the hidden layers and output layer use an activation function $h(x)$. The weights of each of the connections are marked in the figure.

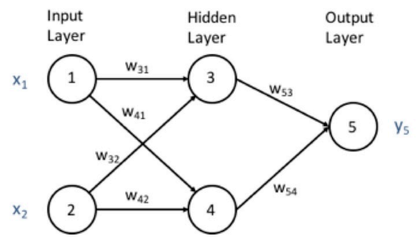

Figure 1: A neural net with 1 hidden layer having 2 neurons

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

b. Now suppose we use vector notation, with symbols defined as below:

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

$$W^{(1)} = \begin{pmatrix} w_{3,1} & w_{3,2} \\ w_{4,1} & w_{4,2} \end{pmatrix}$$

$$W^{(2)} = \begin{pmatrix} w_{5,3} & w_{5,4} \end{pmatrix}$$

Write down the output of the neural net in vector format using above vectors.

c. Now suppose that you have two choices for activation function $h(x)$, as shown below:
Sigmoid:

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

Tanh:

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

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

Hint: First compute the relationship between $h_s(x)$ and $h_t(x)$ and then show that the output functions are same, with the parameters differing only by linear transformations and constants.

2 Programming Part (60 points)

In this part, you will write code to optimize the performance of a neural net by trying various combination of hyper-parameters and evaluating their results.

A template code has been provided for your reference. You do not need to follow it exactly, if you can come up with a better solution. The class **NeuralNet** contains the code necessary for this assignment.

Below are the requirements and suggested steps of the program

- In the main method, the program passes path to the data file. Be sure to point that to a network path that your program can access.
- In the **preprocess** method, you are required to do standard pre-processing tasks such as handling null values, ensuring data integrity, and standardization of attributes.
- Complete the **train_evaluate** method. Perform at least the following steps:
  - Create different neural networks with all possible combination of hyperparameters specified. You are free to use any library or package that creates the neural network for you using the chosen combination of hyperparameters.
  - Keep track of model history i.e. model performance (accuracy) vs number of epochs in every case. Plot the model history for all the cases on a single plot. If that becomes too congested, you can break it up into two or three parts, and plot each part on the same plot.
  - Output a table of results containing following columns: model hyperparameters, training and test accuracies, and training and test errors (e.g. mean squared error)

Dataset

You can use **any one** dataset from the UCI ML repository:
`https://archive.ics.uci.edu/ml/datasets.php`
Note: If the above direct link does not work, you can just Google the UCI ML repository.

What to submit:

You need to submit the following for the programming part:

- Link to the dataset used. *Please do not include the data as part of you submission.*

- Your source code and a README file indicating how to run your code. Do not hardcode any paths to your local computer. It is fine to code any public paths, such as AWS S3.
- Model history plots for every model. Model history is a plot of accuracy against the number of epochs.
- Output for your dataset summarized in a tabular format for different combination of parameters.
- A brief report summarizing your results. For example, which activation function performed the best and why do you think so.
- Any assumptions that you made.