CSE/ECE 343/543: Machine Learning

Assignment-2 Decision Tree,Random Forest,MLP
Max Marks: 100 (Programming:85, Theory:15) Due Date: 30/09/2021, 9.00 PM

Instructions:

• You are allowed to discuss but the final answer should be your own. Keep collaborations at high level discussions. Copying/Plagiarism will be dealt with strictly.

• Late submission penalty: As per course policy.

• Your submission should be a single zip file AdmissionNo_HW2.zip. Including only the relevant files arranged with proper names. A single .pdf report explaining your codes with details of EDA and pre-processing, relevant graphs, visualization and solution to theory questions. Anything not in the report will not be considered for evaluation. The structure of submission should follow:

```
AdmissionNo HW1
        |− Q1.py
        |− Q2.py
        |− Report.pdf
        |− Weights (folder)
        |− plots (folder)
```

• Restrict to using only Python for coding assignments.

• You are free to use math libraries: Numpy, Pandas; and use Matplotlib library for plotting.

• Use of inbuilt function for any evaluation metric is not allowed. Each of the metric needs to be implemented from scratch.

• Remember to turn in after uploading on Google Classroom. No excuses or issues would be taken regarding this after the deadline.

• Start the assignment early. Resolve all your doubts from TAs in their office hours two days before the deadline.

• Document your code. Lack of comments and documentation would result in loss of 20% of the obtained score.

• The assessment will be done on basis of the following components:
– Working codes

– Analysis and clarity of results (drawing comparisons across different parts) clarity of the report
– Understanding the theoretical concepts/viva

Questions

Programming[35+50=85marks+10bonus*]

Q1: Decision Tree and Random Forest [35 marks]

For this question, you can use the decision tree classifier from sklearn.

Dataset: PM2.5 Data UCI Archive

Target Variable: Month

You will have to handle null values in the data. (Remove ”No” column as that is an index)
Split the data into a training, validation, and testing set (70:15:15 ratio) using the custom-designed train test split method. Use the same training set for training the following models. (You can not use sklearn for splitting the dataset.)

(a) Train a decision tree using both the Gini index and the Entropy. Don’t change any of the other default values of the classifier. In the following models, use the criteria which give better accuracy on the test set.**[3 marks (1.a) + 2 marks (Data processing) = 5 marks]**

(b) Train decision trees with different maximum depths [2, 4, 8, 10, 15, 30]. Find the best value of depth by using testing and training accuracy. Plot the curve between training and testing accuracy and depth to support your analysis. **[5 marks]**

(c) Ensembling is a method to combine multiple not-so-good models to get a better performing model. Create 100 different decision stumps (max depth 3). For each stump, train it on randomly selected 50% of the training data, i.e., select data for each stump separately. Now, predict the test samples’ labels by taking take a majority vote of the output of the stumps. How is the performance affected as compared to parts 1. (a) and 1. (b) **[7 marks]**

(d) Now, try to tune the decision stumps by changing the max-depth [4, 8, 10, 15, 20, best achieved from (b)] and the number of trees. Analyze the effect on the training, validation, and testing accuracy. Use majority vote for final prediction on the test data. **[7 marks + 1 marks = 8 marks]**

(e) Another popular boosting technique (also covered in the Tutorial) is Adaboost. Use the sklearn Adaboost algorithm on the above dataset and report the testing accuracy. Use the

Decision tree as the base estimator and with a number of estimators as [4, 8, 10, 15, 20]. Compare RF and Adaboost results. **[10 marks]**

Q2: MLP [50 marks + 10 marks Bonus]

You have to implement a general algorithm for Neural Networks. You can only use the NumPy library and the Pandas library for reading the dataset. Make a class named **MyNeuralNetwork** and implement all your functions over there. **[29 marks]**

1. The network should have the following parameters **[6 marks]**
    a. N_layers: Number of Layers(int)
    b. Layer_sizes: array of size n layers which contains the number of nodes in each layer. (array of int)
    c. activation: Activation function to be used (string)
    d. learning rate: the learning rate to be used (float)
    e. weight init: initialization function to be used
    f. batch size: batch size to be used (int)
    g. num epochs: number of epochs to be used (int)
2. Implement the following activation functions with their gradient calculation too: ReLU, Leaky ReLU, sigmoid, linear, tanh, softmax. **[7 marks]**
3. Implement the following weight initialization techniques for the hidden layers: **[6 marks]**
    a. zero: Zero initialization
    b. random: Random initialization with a scaling factor of 0.01
    c. normal: Normal(0,1) initialization with a scaling factor of 0.01
4. Implement the following functions with bias=0 and cross-entropy loss as the loss function. You can create other helper functions too. **[10 marks]**
    a. fit(): accepts input data & input labels and trains a new model
    b. predict proba(): accepts input data and returns the class wise probability
    c. predict(): accepts input data and returns the prediction using the trained model
    d. score(): accepts input data and their labels and returns the accuracy of the model

Use the **MNIST Handwritten dataset** for training and testing of the neural network model created Make a 7:2:1 train test Val split (implemented from scratch). Show all the preprocessing steps for the dataset. Use the training dataset for calculating the training error and the test dataset for calculating the testing error. **[31 marks]**

1. Train your neural network with the 4 hidden layers with layer sizes = [256,128,64,32] (excluding the input and output sizes) and learning rate = 0.08 and the number of epochs = 150. Use normal weights initialization for the parameters. Save the weights of the trained model separately for each activation function defined above and report the test accuracy. **[10 marks]**
2. Plot the training v/s epochs and validation v/s epochs for each activation function. Which activation function works best according to you and Why? Give your analysis and comparisons for each. **[3 marks]**

3. In every case, what should be the activation function for the output layer? Give reasons to support your answer. **[2 marks]**
4. Now, use sklearn with the same parameters (from part 1) and report the test accuracy obtained using ReLU, sigmoid, linear, and tanh activation functions. Comment on the differences in accuracy with respect to your implementation, if any. **[6 marks]**

5. Now, train your neural network with learning rates = [0.001, 0.01, 0.1, 1]. Choose the appropriate number of epochs and the activation function. Which learning rate performs best according to you? Compare and give your analysis [**BONUS: 10 marks**]

Theory [5 * 3 marks = 15 marks]

1. A badminton player, during the practice session, fell down and hurt his ankle. Based on the x-ray results, the doctor thinks that he has a fracture with a probability of 0.2. So, the question is, should the player play in the tournament tomorrow? If he plays, he thinks he'll win with a probability of 0.1. If his leg is broken and he plays with a broken leg, then he'll damage it further. So, his utilities are as follows: if he wins the race and his leg isn't broken, +100; if he wins and his leg is broken, +50; if he loses and his leg isn't broken 0; if he loses and his leg is broken -50. If he doesn't play, then if his leg is broken your utility is -10, and if it isn't, it's 0.

1. Draw the decision tree for this problem.
2. Evaluate the tree, indicating the best action choice and its expected utility
3. Compute the expected value of perfect information about the state of your leg.
4. Compute the expected value of perfect information about whether you'll win the race.
5. Is it possible to use a decision tree in the case that the probability that you'll win the tournament depends on whether your leg is broken?

2. Prove that any function taking a bit-vector to a boolean value, of the form:
$f : \{0, 1\}^d \rightarrow \{0, 1\}$ can be represented as a Neural Network with just 1 hidden layer. You can make use of an appropriate activation function.

3. Create a Neural Network with 2 input nodes and 1 hidden layer that the output of the neural network is the binary logistic linear regression classifier given by:
$Y = argmax_y P(Y = y|X)$, where

$$Y = P(Y = 1|X) = \frac{exp(\beta_1 X_1 + \beta_2 X_2)}{1 + exp(\beta_1 X_1 + \beta_2 X_2)}$$

$$Y = P(Y = -1|X) = \frac{1}{1 + exp(\beta_1 X_1 + \beta_2 X_2)}$$

Specify the weights of all connections between NN nodes along with the activation function used for each node.