Ayush Garg
PUID: 0033794803

Homework 2

CS 57800

1. Questions

1. (a) For boolean function:
   We can say that there are $\binom{5}{3}$ possibilities to select any 3 variables out of the given 5 variables. Hence, it can be represented as a disjunction of 10 clauses which represent these 10 possibilities. It is represented as below:
   $(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_4) \vee (x_1 \wedge x_2 \wedge x_5) \vee (x_1 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_3 \wedge x_5) \vee (x_1 \wedge x_4 \wedge x_5) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_3 \wedge x_5) \vee (x_2 \wedge x_4 \wedge x_5) \vee (x_3 \wedge x_4 \wedge x_5)$

   (b) For linear function:
   It can be represented in this manner:
   $x_1 + x_2 + x_3 + x_4 + x_5 \geq 3$

2. The number of monotone conjunctions defined over n boolean variables is 2^n as every variable has two choices, either it is present or not.

3. The size of CON_L would be equal to the size of CON_B. This is because every monotone conjunction can be expressed in the form of linear function too. Hence, it's size is also 2^n.

4. A is called a mistake bound algorithm on class C if $M_A(C)$, the maximum number of mistakes the algorithm A makes on any sequence of examples S for any concept f in the class C is polynomial in the complexity parameter of the target concept.

5. Let the function f to learn belongs to the class C of conjunctions of 2n variables where n variables represent x_1 ... to ... x_n and the remaining variables $\bar{x}_1$ ... to $\bar{x}_n$ represent their negations.
   Now, in the algorithm, we predict using this hypothesis function h(x) which consists of the $\wedge$ of these 2n variables as described above.
   If the prediction is correct, then we do not change the state. If the prediction is false and the label is true, then we remove all the literals from the hypothesis function which are false in h(x). We keep on repeating this process.
   We can infer from the above approach that the first mistake will help us to reduce the number of variables in our h(x) from 2n to just n. Afterwards, each mistake will remove atleast one variable, hence, we make atmost n+1 mistakes.
   As the maximum number of mistakes which can be made is polynomial in the complexity parameter(n) of the class, we say that this is a mistake bound algorithm.

6. (a) As the dataset is linearly separable, we can say that both the classifiers will converge as per Perceptron Convergence Theorem.

   (b) The training error would be zero for both the approaches as the Perceptron will correctly classify the data, given that it is linearly separable as per Perceptron Convergence Theorem.

7. We want to get a kernel function which considers all conjunctions of size atmost k. Considering all the conjunctions of size atmost k that satisfy both x and y can be represented by this kernel function:

$$K(x,y) = \sum_{i=0}^{k} \binom{same(x,y)}{i} \tag{1}$$

This kernel function basically takes into account all the conjunctions of size 0, size 1 .... upto size k which is being asked in the problem. This is the process how the normal formula for n conjunctions is also derived through as the summation of the above given kernel upto n would give us $2^{same(x,y)}$

2. Programming Assignment

1. I split the dataset in the ratio of 80:20, keeping 80% of the total dataset for training purposes and the remaining 20% for the testing purpose to see how my model generalises on the unseen data.
   This split gives us the optimal training and testing dataset sizes because if we reduce the training set size then we won't be able to learn the required features properly and the model might underfit. Also, if the testing dataset is too small then we won't be able to generalize the performance of our model on the unseen data.

2. I tried two approaches for feature engineering process:
   One of them is Bag of Words approach, wherein I create a dictionary of all the words present in the dataset. I reduce the size of the vocabulary dictionary by removing the stopwords,numbers and characters other than the english alphabets. Thereafter, I create a feature vector for each of the text sample based on whether that word from vocabulary is present in that sample or not. When we get some words in the test sample which are not present in our vocabulary, we skip them and try to predict based on what we know from the vocabulary which we used while training the model.
   The other method used by me was Bi-gram approach. It follows a similar approach as that of Bag of Words for preprocessing. The difference comes in the words which are stored in our dictionary. Instead of mongrams, we have a pair of words in our dictionary and then while converting a text sample to a feature vector, we look for whether that pair is present in the sample or not. This approach also does not take into account the pair of words which are not present in the vocabulary and just predict based on what features the model knows from the training phase.
   Finally, I went with Bag of Words approach because with it, I was able to achieve a better accuracy for both the models.

3. The first figure shows the plots obtained for the Perceptron Model.
   The different plots denote the different max_epochs considered in training the model.

Each plot displays the average training and test accuracy achieved after 5-fold cross validation of the train data which is obtained from the split function from Execution.py file. Different learning rates considered for the analysis are: 0.001, 0.005, 0.01, 0.05, 0.1, 0.2.
We can easily infer from the plots that we obtain the best test accuracy in the second plot, i.e with 10 epochs when the learning rate is 0.005

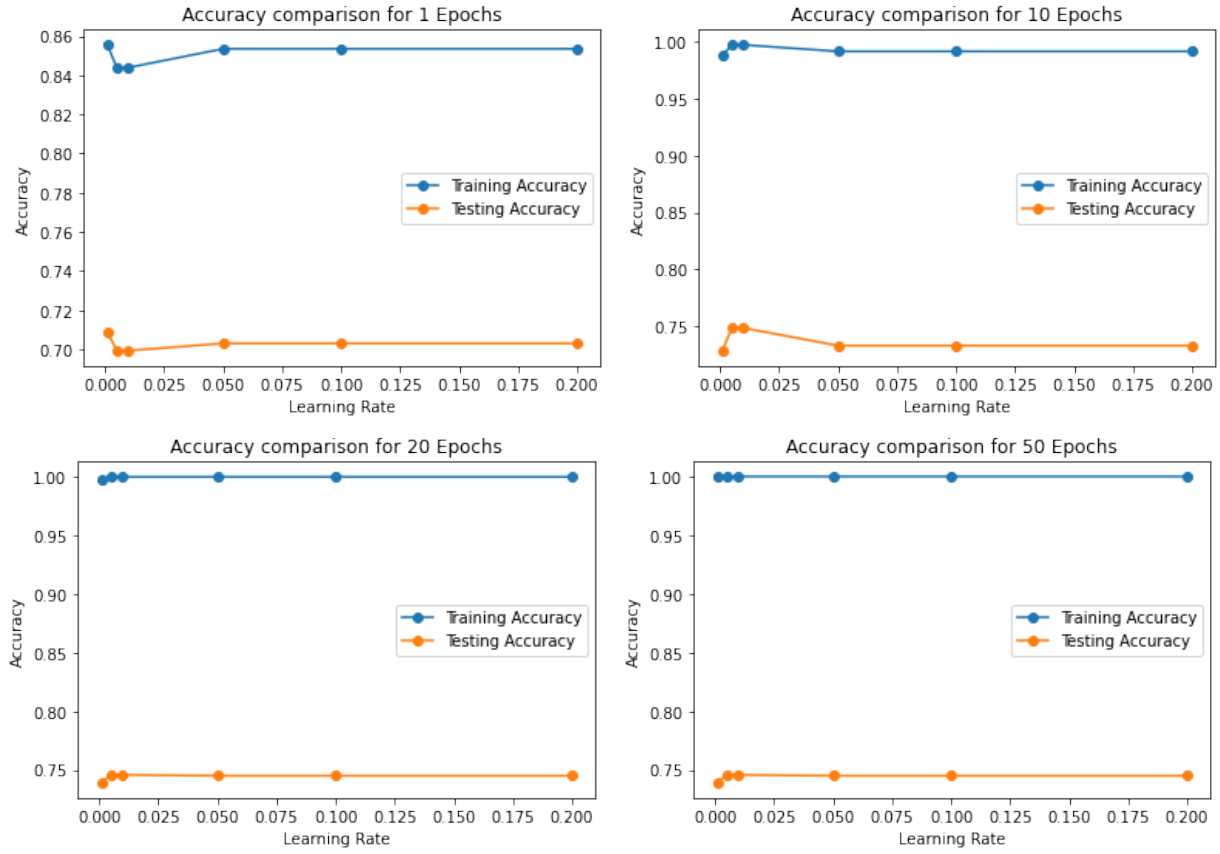

Figure 1: Validation Accuracy for Perceptron

This figure shows the plots obtained for the Logistic Model.
The different plots denote the different max_epochs considered in training the model. Each plot displays the average training and test accuracy achieved after 5-fold cross validation of the train data which is obtained from the split function from Execution.py file. Different learning rates considered for the analysis are: 0.001, 0.005, 0.01, 0.05, 0.1, 0.2.
We can easily infer from the plots that we obtain the best test accuracy in the third plot, i.e with 20 epochs when the learning rate is 0.001

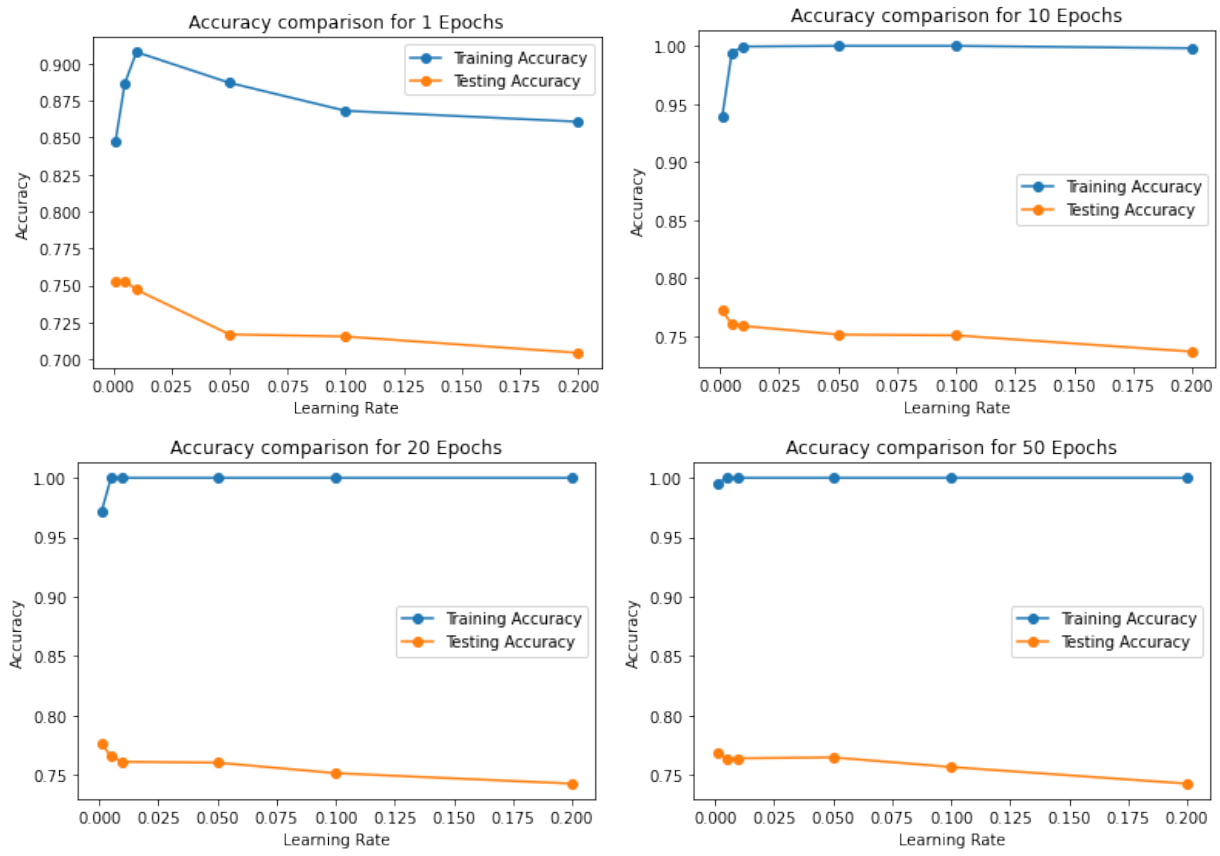

Figure 2: Validation Accuracy for Logistic

4. The optimal value for the hyperparameters for the Perceptron model are:
   learning rate = 0.005 and max epochs = 10
   These optimal values have been inferred by analysing the average test accuracies for the 5 fold cross validation over the dataset. The accuracies received have been shown in the plot. The values can be easily inferred from the graphs shown in the previous part.
   The optimal value for the hyperparameters for the Logistic model are:
   learning rate = 0.001 and max epochs = 20
   These optimal values have been inferred by analysing the average test accuracies for the 5 fold cross validation over the dataset. The accuracies received have been shown in the plot. The values can be easily inferred from the graphs shown in the previous part.
   With the above selected values for the hyperparameters, the accuracies obtained are shown below:
   Thus, we can that the training accuracy is slightly better for the Perceptron model as

Figure 3: Validation Accuracy

compared to the Logistic model whereas the Logistic model gives a better performance

on the test data than the Perceptron model.