Ayush Garg
PUID: 0033794803

Homework 3

CS 57800

1. Questions

1. VC dimension of the given hypothesis space is 2k.
   Prove:

   (a) VC $\geq$ 2k
   Let A be an arbitrary set of elements with arbitrary labellings. By going from first to last member of A, we can place all adjacent 1's in one interval and only need to use another interval when there is 0 between 1's. We need k intervals to classify k groups of 1 correctly. Hence, a set of 2k elements has atmost k isolated 1's and thus we need atmost k intervals.

   (b) VC < 2k+1
   For any set with 2k+1 elements, there will be a labelling in which we will get k+1 isolated 1's, which cannot be covered by k disjoint intervals.

   Hence, we proved that the VC dimension of the above hypothesis space is 2k.

2. (a) Learning rate and number of epochs are the hyperpaprameters in the gradient descent algorithm.

   (b) A regression model that uses L1 regularization technique is called Lasso Regression and model which uses L2 is called Ridge Regression.
   Ridge regression (L2) adds "squared magnitude" of coefficient as penalty term to the loss function. This technique works very well to avoid over-fitting issue.
   L1 regularizer adds "absolute value of magnitude" of coefficient as penalty term to the loss function. It shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features.

   (c) The loss function with L2 regularization looks like this:

$$L = min \quad (\frac{\lambda}{2}||W||_2^2 + \sum_{i=i}^{N} max(0, 1 - y_i f(x_i)))$$

   The gradient for this loss function looks like this:

$$\begin{aligned} \frac{\partial L}{\partial w_i} &= \lambda|w_i| - y_i f(x_i) \quad if \quad y_i f(x_i) < 1 \\ &= \lambda|w_i| \qquad if \quad y_i f(x_i) \geq 1 \end{aligned}$$

where f(x_i) = wx_i
The gradient algorithm looks like this:

- Intialize W^0 randomly
- for i = 0 ... T: (num of iterations)
  - $\Delta w = (0, 0, ..., 0)$
  - for every training example d = 1....D:
    - for every component of w j = 0,1...N:
      - $\Delta w_j = \frac{\partial L}{\partial w_j}$

    $w^{i+1} = w^i - \alpha \Delta w$
    return w^{i+1} when it has converged

(d) The loss function with L1 regularization looks like this:

$$L = min \quad (\lambda ||W||_1 + \sum_{i=i}^{N} max(0, 1 - y_i f(x_i)))$$

The gradient for this score function looks like this:

$$\begin{aligned} \frac{\partial L}{\partial w_i} &= \lambda - y_i f(x_i) \quad if \quad y_i f(x_i) < 1 \text{ and w} > 0 \\ &= \lambda \qquad if \quad y_i f(x_i) \geq 1 \text{ and w} > 0 \\ &= -\lambda - y_i f(x_i) \quad if \quad y_i f(x_i) < 1 \text{ and w} < 0 \\ &= -\lambda \qquad if \quad y_i f(x_i) \geq 1 \text{ and w} < 0 \end{aligned}$$

where f(x_i) = wx_i
The gradient algorithm looks like this:

- Intialize W^0 randomly
- for i = 0 ... T: (num of iterations)
  - $\Delta w = (0, 0, ..., 0)$
  - for every training example d = 1....D:
    - for every component of w j = 0,1...N:
      - $\Delta w_j = \frac{\partial L}{\partial w_j}$

    $w^{i+1} = w^i - \alpha \Delta w$
    return w^{i+1} when it has converged

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. This represents the actual size of vocabulary used for training purpose. Number of features generated during training:

   - Unigram : 3879
   - Bigram : 6468
   - Trigram : 2443

   This represents the total possible token combinations which could have been there in the n-gram model. Size of the feature space:

   - Unigram : 26949
   - Bigram : 26949^2
   - Trigram : 26949^3

3. For the logistic regression model, I am using unigrams for the feature extraction along with L2 regularization.

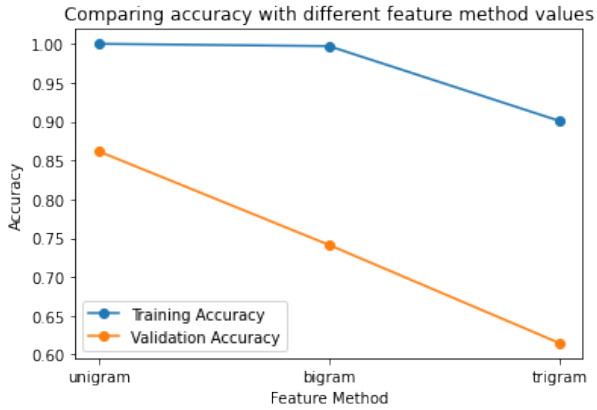

Figure 1: Accuracy plot for feature selection method

From this plot, we can see that the training accuracy of the model keeps on decreasing as we increase the value of n in n-gram model. This shows that the model tends to over-fit on the data on which it is trained and hence it is unable to generalise well on the unseen dataset.

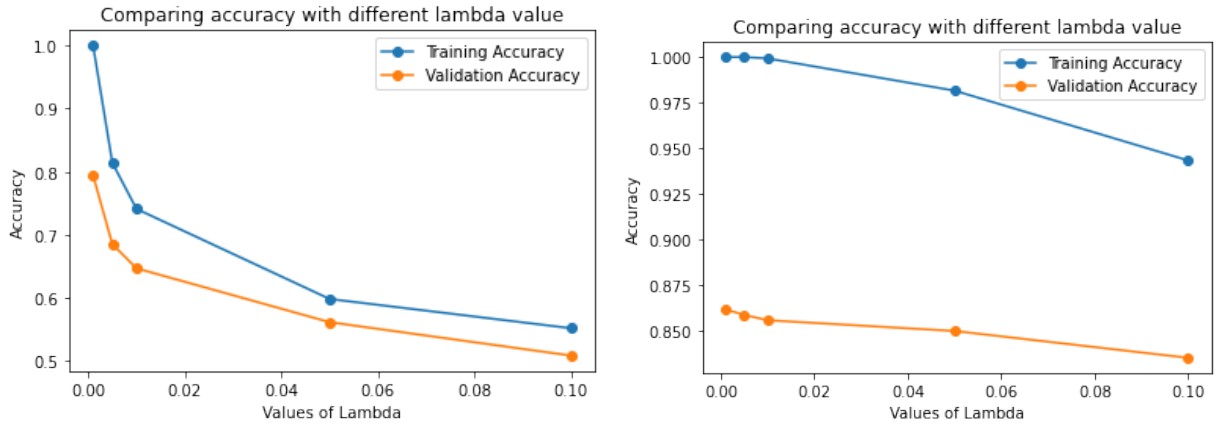

Figure 2: Accuracy plot for regularization method

From these plots (Figure 2), we can see the effect of L1 and L2 regularization on the logistic regression model for different values of lambda. We can clearly infer from the graphs that we obtain better results when we use L2 regularization.
These are the optimal values of the hyperparameters being used in the logistic regression model. 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:

- Learning rate : 0.001
- Max Epochs : 20
- Lambda (Coefficient of the regularizer) : 0.001

Here are the respective plots obtained for each of the hyperparameter

From this plot (Figure 3), we can see that there is small increase in the test accuracy near the value of 0.001 and afterwards, there is not much change in the accuracy as we change the value of learning rate. Different learning rates considered for the analysis are: 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.02, 0.05.

From this plot (Figure 4), we can see that there is an increase in the train as well as test accuracy when max epochs reaches to value 5 and afterwards, there is not much change in the accuracy as we change the value of learning rate. We achieve the highest accuracy at max epochs = 20. Different epoch values considered for the analysis are: 1,5,10,20,30,50,100,200.

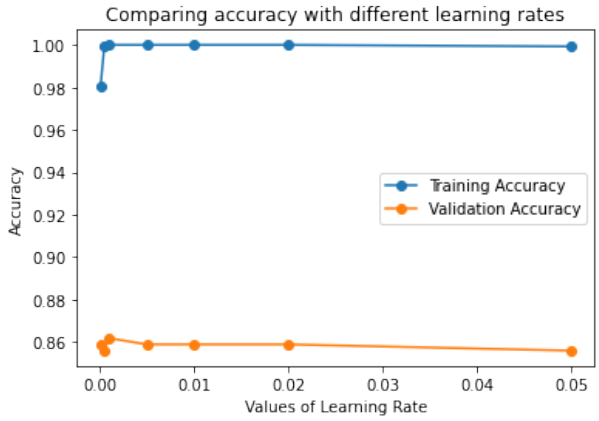

Figure 3: Accuracy plot for learning rate

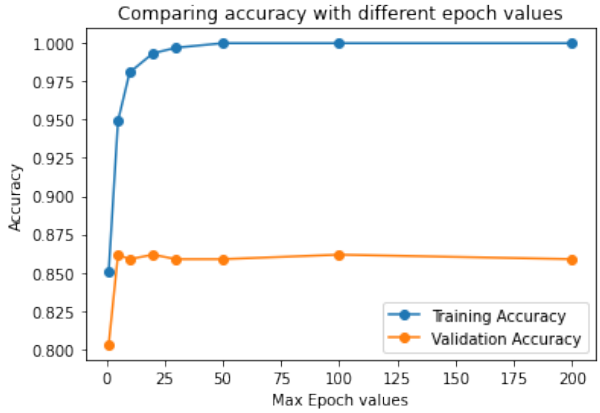

Figure 4: Accuracy plot for epoch values

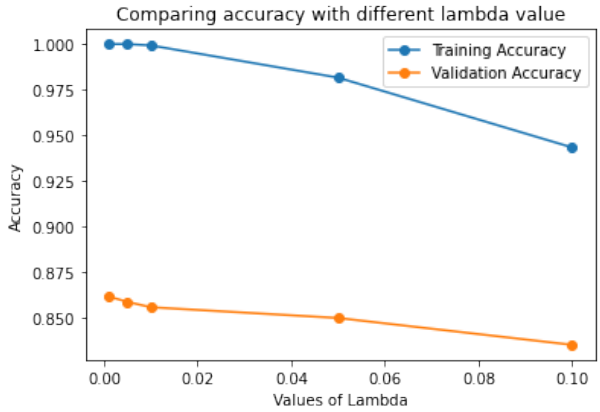

Figure 5: Accuracy plot for lambda values

From this plot (Figure 5), we can see that there is constant decrease in the train as well as test accuracy as we keep on increasing the value for lambda. This trend indicates that we should take a small value to achieve a better performance with our model. We achieve the highest accuracy with lambda value as 0.001. Different lambda values considered for the analysis are: 0.001, 0.005, 0.01, 0.05, 0.1, 1

4. For the ensemble model, first of all, I generate the vocabulary based on the feature method which was found optimal for the logistic regression model. Afterwards, I do the feature extraction for the dataset to generate the corresponding features for training the model. The hyperparameter used in the ensemble model is the number of classifiers. Afterwards, we train the logistic regression model equal to the number of classifiers selected and store those models in order to get the prediction on the other unseen datasets. For doing the prediction, we take the majority vote approach. In this approach, basically, we take the values predicted for each data point in our dataset and take the majority value (value predicted by most number of classifiers) as our final value for the ensemble model.
The optimal hyperparameter value for the number of classifier is 31. This can be inferred from the following graph. Different values considered for the analysis are: 1,5,11,31,51,101.

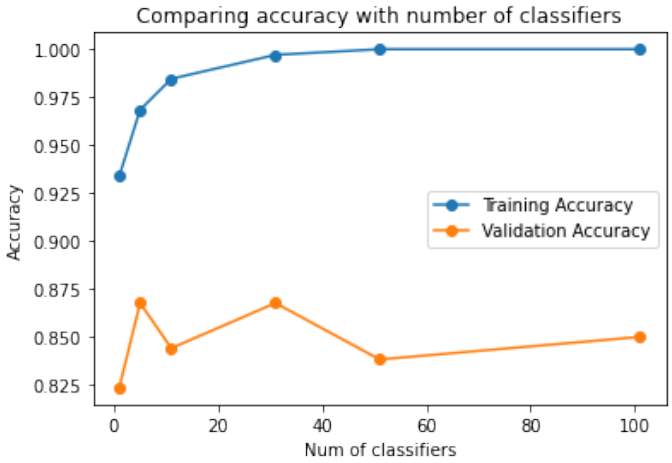

Figure 6: Accuracy plot for variation in number of classifiers

5. The optimal values for the hyperparameters are as follows:

   - Learning rate : 0.001
   - Max Epochs : 20
   - Lambda value : 0.001

The values can be easily inferred from the graphs shown in the previous part.

Here are the training and testing accuracies obtained for Logistic Regression as well as Ensemble model

```
----------Logistic Function Performance----------

Training Accuracy Result!
***************
Accuracy: 0.9933823529411765
***************

Testing Accuracy Result!
***************
Accuracy: 0.861764705882353
***************

Unseen Test Set Accuracy Result!
***************
Accuracy: 1.0
***************

----------Ensemble Method Performance----------


Training Accuracy Result!
***************
Accuracy: 0.9941176470588236
***************

Testing Accuracy Result!
***************
Accuracy: 0.8647058823529412
***************

Unseen Test Set Accuracy Result!
***************
Accuracy: 1.0
***************
```

The test accuracy obtained in HW2 was 80% for logistic regression model whereas in this assignment after using L2 regularization with 'unigram' as feature selection method , I am getting 86.1% test accuracy. This shows that adding a regularizer in the cost function helps us to fine tune our model and provide better performance as compared

to a non-regularized model.
The ensemble method also increases the accuracy a bit as it trains a number of logistic classifiers and then takes the majority vote for each data point. However, the increase in not too much. This is probably because the logistic regression classifier does a good job for this dataset and hence using an ensemble method does not help us that much in this case.