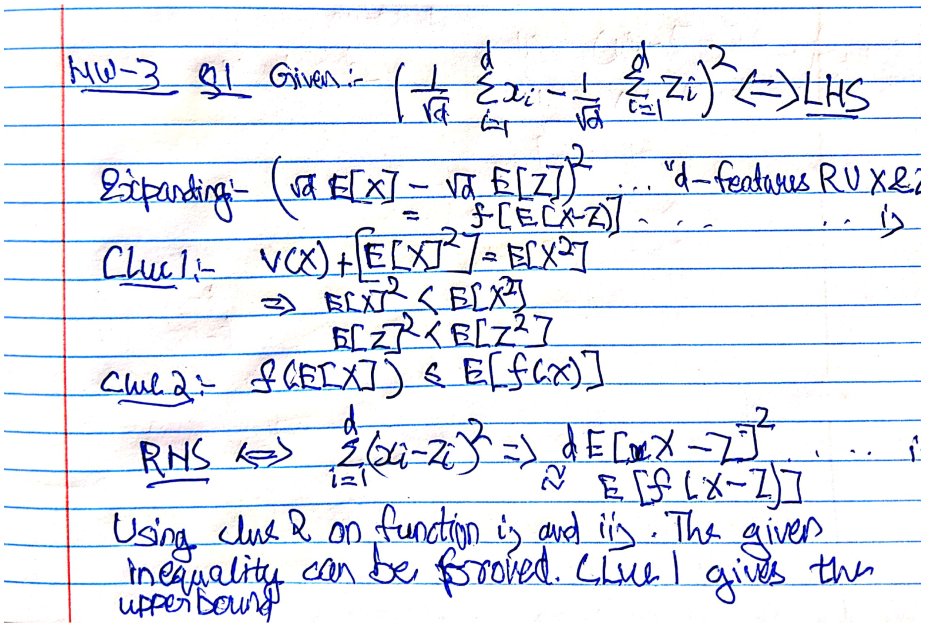
CPTS570 HW 3

Chandan Dhal

November 15, 2019

1 Analytical Part

- **Question1:** Given two high dimensional vector point x and z.



HW-3 Q1 Given:- $\left(\frac{1}{\sqrt{d}}\sum_{i=1}^{d} x_i - \frac{1}{\sqrt{d}}\sum_{i=1}^{d} z_i\right)^2 \Leftrightarrow$ LHS

Expanding:- $\left(\sqrt{d}\,E[X] - \sqrt{d}\,E[Z]\right)^2$ ... "d-features RV X&Z

$= f[E[X-Z]]$ ... i)

Clue 1:- $V(X) + [E[X]^2] = E[X^2]$

$\Rightarrow E[X]^2 < E[X^2]$

$E[Z]^2 < E[Z^2]$

Clue 2:- $f(E[X]) \leq E[f(x)]$

RHS $\Leftrightarrow \sum_{i=1}^{d}(x_i - z_i)^2 \Rightarrow d\,E[X - Z]^2$ ... ii)

$\approx E[f(X-Z)]$

Using clue 2 on function i) and ii). The given inequality can be proved. Clue 1 gives the upper bound

Figure 1: Math workout for Part a

Summary of fig 1: The equation 1, summarizes the LHS is of the functional form of Expected value of two RVs X,Z. Similarylt equation 2, summarizes that RHS is the expected value of a functional form of RVs X,Z. Thus Clue 2, jensen's equality holds. [**Sufficient**]

The upper bound can be proved by expanding equation 1 and using clue 2. [**Necessary**]

Part b: In high dimensional space computing norms for NN algorithm can be expensive. From part a, we can infer that the functional form is trying to map these high dimensional RVs to a plane and then expected value is computed. This gives a heuristic bound of how close are two given high-dimensional vectors with out extensive norm computation. Something similar to kernal concept.

- **Question2:** This article summarizes two main topics related to K-NN algorithms involving large number of examples. It discusses NN algorithms based on locality-sensitive hashing and its application in practical scenarios. Later it describes a new hashing-based algorithm involving features in large euclidian space.

  Section 2 discusses some basic definitions required for understanding LSH and also describes an algorithm in Fig 2. Section 3 summarizes various LSH families available. In section 4 discusses a new LSH family which significantly improves the methods described in section 2.

- **Question3:** Given a set of k rules $R = [R_1, R_2, R_3, ..., R_k]$, we can consider two possible cases of creating decision trees.

  Case 1: If it guarantees that during the i-th decision rule it results into one pure and one impure leafs. Then a complete decision tree can be mapped.

  Case 2: The ordering or some structure is required along with Rule set R. For example, if rule R_1 creates two leaves where rule R_2 and R_3 are applied respectively.

  Generally, we can not create the decision tree from rule set,R

- **Question4:** A generative classifier: The Naive Bayes is linear classifier using Bayes Theorem and assumes independence condition among features. It's generative because bayes theorem can help us to determine posterior or a-priori estimations.

  A descriptive classifier: Logistic regression measures the relationship between a label Y and features by using probability scores as the predicted values of the dependent variables. Posterior and apriori estimates can not be obtained.

  The articles concludes that when the training size reaches infinity the discriminative model: logistic regression performs better than the generative model Naive Bayes. However the generative model reaches its asymptotic faster (O(log n)) than the discriminative model(O (n)), i.e the generative model (Naive Bayes) reaches the asymptotic solution for fewer training sets than the discriminative model (Logistic Regression).

- **Question 5 a:** The training examples follows Naive-Bayes assumption. In this case we would expect logistic to perform better but naive bayes reaches asymptote faster than Logistic.

  Question 5 b: The training set does not follow Naive-Bayes assumption. Then, from results of Andrew NG paper, the logistic will perform better as training set approaches infinity

Question 5 c: Naive-Bayes is generative model, hence $P(x)$ can be determined.

Qustion 5 d: Logistic Regression is a discriminative model, hence $P(x)$ can not be determined.

- **Question6:** This article describes most of the key points we have seen in overfitting lectures. It describes a classifier and how classifiers can be used in learning medical diagnosis systems.It summarizes the abstract idea of ML and talks about issues to finding the optimal algorithm in hypothesis class. Since often times the optimal algorithm performs poor on the real test examples.

  If we work too hard to find the optimal classifier in the training data set, then we are neglecting the possible introduction of errors in the data. This is referred as overfitting hence our optimal objective function is not correct. It discusses more work done in fixing the objective function and attempt to predict the off-training acccuracy with on-traning set accuracy [Validation].

  In ML it is optimal to be sub-optimal!This observation was realised by the interplay of statistics and ML concepts.

- **Question7:** This paper is dealing with the optimization part of the abstract ML concept. The key idea is described in introduction section where they pose 9 questions to access the good/bad characteristics of a classifier for small sample set or large sample set. Later they considered neural network applications where questions 1,2,5 and 6 can be rephrased in expect MSE of a predictor and Questions 3,4,7,8 can be rephrased in terms of determining which classifier has smaller MSE. But there's more results described in the paper.

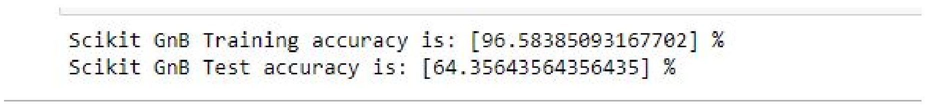
  The summary describes about how Question 8 can not be answered and hence the stastical methods described in the paper is an approximate.

2 Programming Part

- **Part 1: Naive Bayes Classifier** Given text files was processed to feature vectors. The features are the entire words collection in the train set with stop words removed

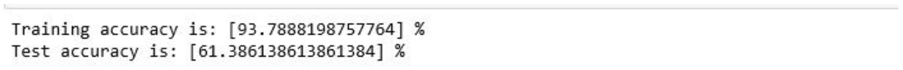
  Figure 2 depicts the accuracy results obtained from Scikit GnB function. Figure 3 depicts the accuracy results of the implemented NB classifier with laplace smoothing.

  Observations Naive Bayes Classifier is the best text based classifier. The poor testing accuracy might be result of error in feature extraction. Other tricks apart from normalization can be implemented to make the test accuracy better.



```
Scikit GnB Training accuracy is: [96.58385093167702] %
Scikit GnB Test accuracy is: [64.35643564356435] %
```

Figure 2: Scikit learn GnB test and training Accuracy



```
Training accuracy is: [93.7888198757764] %
Test accuracy is: [61.386138613861384] %
```

Figure 3: Scikit learn GnB test and training Accuracy

- **Part 2: Convolutional Neural Net**

  This was the straightforward extension of the uploaded jupyter notebook code. The last convolution layer has 32 outputs from 16 inputs and we have one linear layer that connects to 10 class labels. Thus, the input of this layer should be 32, after the adaptive average pull of all the output layers of the forth convolution layer.

  Figure 2 is the testing accuracy plot obtained by normalizing the data and using 4 convoluation layer and 1 linear layer as described respectively. Figure 3 depicts the testing accuracy for 10 epochs.

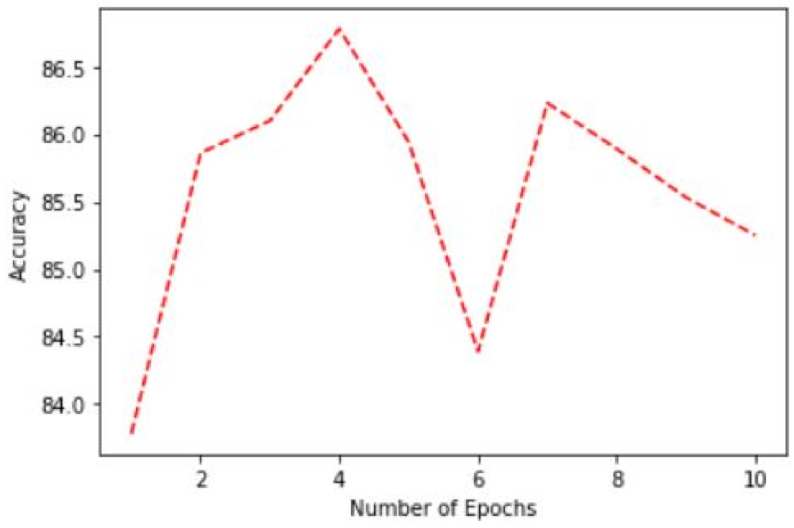
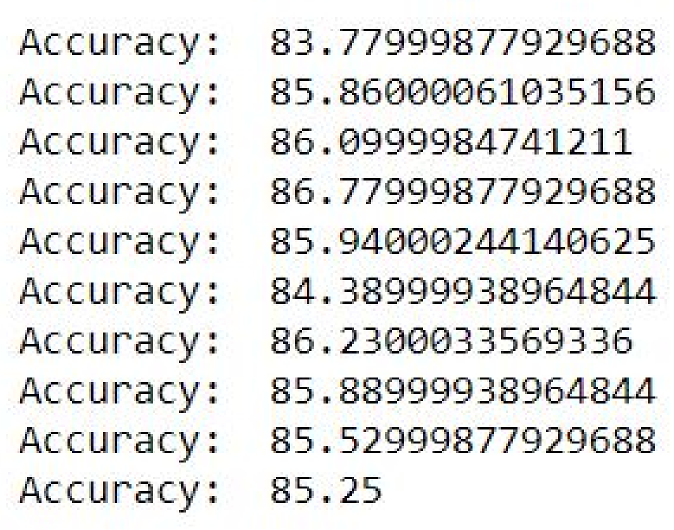


Figure 4: Convolution NN test Accuracy

For training accuracy, the same computation issues from hw2 was observed. But with PyTorch we can exploit Batch-processing for computing large number of example samples. The below plot, depicts the Training



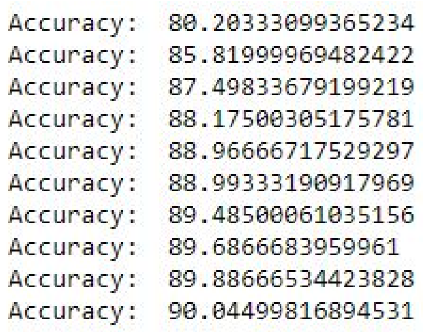
```
Accuracy:  83.77999877929688
Accuracy:  85.86000061035156
Accuracy:  86.0999984741211
Accuracy:  86.77999877929688
Accuracy:  85.94000244140625
Accuracy:  84.38999938964844
Accuracy:  86.2300033569336
Accuracy:  85.88999938964844
Accuracy:  85.52999877929688
Accuracy:  85.25
```

Figure 5: Accuracy per 10 epochs

accuracy for 10 epochs with 100 batches of examples.

Figure 7 depicts the test and training accuracy for datset processed for 100 batches [$bs = 100$]

Futher, if batch size is small then the accuracy value computation is corrupted,like figure 8



```
Accuracy:  80.20333099365234
Accuracy:  85.81999969482422
Accuracy:  87.49833679199219
Accuracy:  88.17500305175781
Accuracy:  88.96666717529297
Accuracy:  88.99333190917969
Accuracy:  89.48500061035156
Accuracy:  89.6866683959961
Accuracy:  89.88666534423828
Accuracy:  90.04499816894531
```

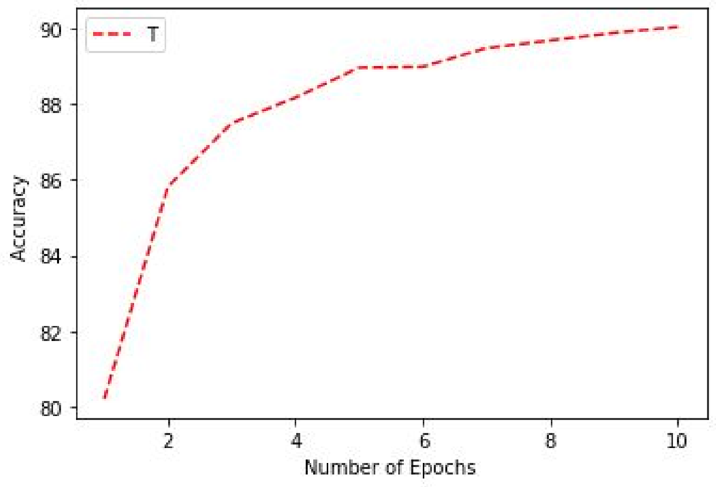


Figure 6: Accuracy per 10 epochs

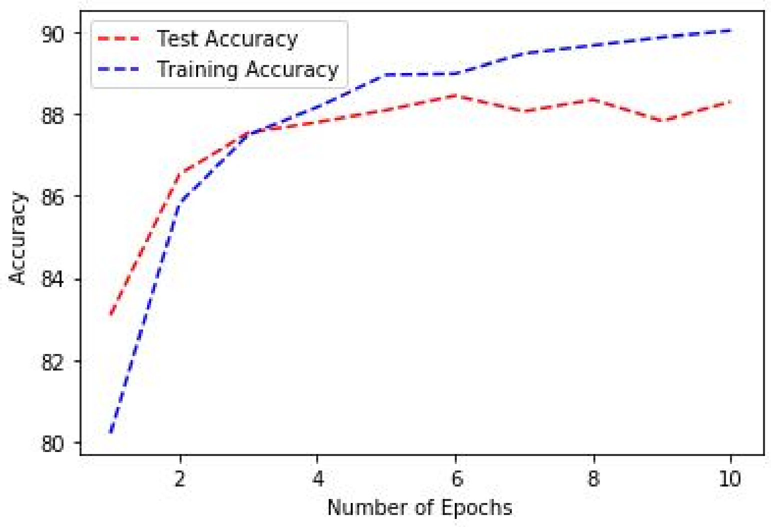


Figure 7: Accuracy per 10 epochs

```
Accuracy:  10.0
Accuracy:  10.0
Accuracy:  10.0
Accuracy:  10.0
Accuracy:  10.0
Accuracy:  10.0
Accuracy:  10.0
Accuracy:  10.0
Accuracy:  10.0
Accuracy:  10.0
```

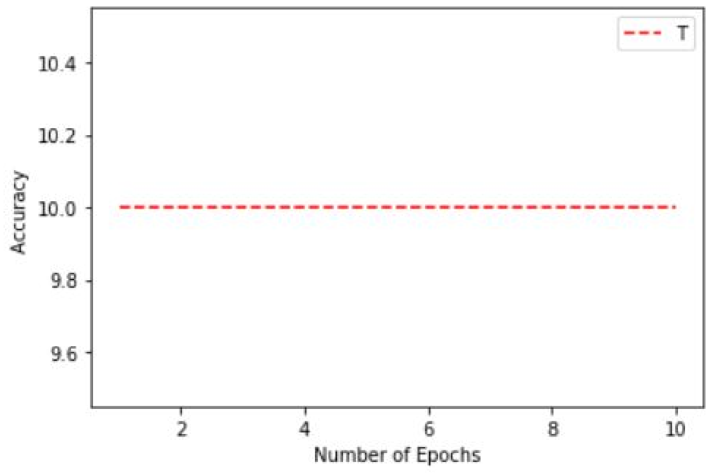


Figure 8: Accuracy per 10 epochs