Distributed Naive Bayes Classifier using Hadoop Framework

DS 222: ML with Large Datasets (2018)
Assignment 1
Swapnil Gupta, 14890

1. Naive Bayes Classifier

Naive Bayes classifier belongs to the family of generative models which assigns the posterior probability in the following manner. Given a training example $\mathbf{X} = [x_1, ..., x_n]$

$$P(Y_i/\mathbf{X}) \alpha P(\mathbf{X}/Y_i) * P(Y_i) \tag{1}$$

$$P(\mathbf{X}/Y_i) = \prod_{j=1}^{n} P(x_j/Y_i) \tag{2}$$

In the above formulation the model parameters are $P(x_j/Y_i), j\epsilon(1, ..., n), i\epsilon(1, ..., C)$ and $P(Y_i)$ which are learned during the training phase.

2. Dataset

Given is a multi-label document classification task with the following characteristics:

- No. of training set samples: 214998
- No. of validation set samples:
- No. of testing set samples: 29997
- No. of distinct classes: 50
- Vocabulary Size(after preprocessing): 392585

For the above dataset the total no. tuning parameters is $(No.of classes) * Vocab_size + (No.of classes) = 50 * 392585 + 50 = 20,021,835$.

2.1. Data Preprocessing

The following preprocessing steps have been performed on the data prior to training:

- Removing punctuation marks attached to the words.
- Removing words comprise of less than 4 characters.
- Removing non-alpha-numeric words.

3. In Memory Naive Bayes

3.1. Training Phase

After Pre-processing the following values are hashed in the memory:

- Count of each word appearing documents against each class. $C(w_j and Y_i)$
- Total no. of words in documents belonging to a class. $C(ANY and Y_i)$
- Total no. of documents belonging to a class. $C(Y_i)$
- Total vocabulary size. [vsize]
- Total no. of documents shared across all the documents. [TotalY]

Using the above counts the training parameters are computed in the following manner:

$$\begin{aligned} P(w_j/Y_i) = (C(w_j and Y_i) + smooth) \\ /(C(ANY and Y_i) + smooth * Vsize) \end{aligned} \tag{3}$$

$$P(Y_i) = C(Y_i)/TotalY \tag{4}$$

Here, the parameter ***smooth*** is a model **hyper-parameter** and is tuned on validation set.

Training phase wall clock time: **54.2 seconds**

3.2. Testing Phase

During the testing phase posterior probabilities of each class is computed as discussed in 1. and the sample is assigned to the class with the highest score.
Testing Phase wall clock time: **92.3 seconds**
Accuracy on development set data: **83.5%** Accuracy on testing data: **78.3%**

4. Distributed memory Naive Bayes

4.1. Training Phase

One mapper and a reducer is used in the training phase.
Mapper1:
This mapper sequentially reads the training data sample by sample and writes the following messages to the memory: **[(word,class),1]**, **[(ANYLABEL,class),len(sample)]**, **[(ANYWORD,word),1]**, **[(APRIOR,class),1]**

Reducer1:
The reducer receives the above messages in sorted order and output a text file with aggregated counts of the above messages to get counts of each word for each class, total count of words in each class, vocabulary of words and total no. of documents in each class respectively.

4.2. Testing Phase

A total of 3 mapreduce phases are run in testing phase sequentially using 2 distinct mappers and 3 distinct reducers.
MapReduce phase 1
It uses the same reducer as the training reducer and a different mapper.
Mapper2:
Reads the test file sample by sample and writes the following messages in memory: **[(word,sampleID),1]**.
Prior to MapReduce phase 2 the output file of training phase and MapReduce phase 1 are concatenated in order to make sure that both the training data and the test data are present in the same text file.
MapReduce phase 2
It is here the key computation for the class scores corresponding to each id happens.
Mapper3:
It is an identity mapper used only for sorting the concatenated text file.
Reducer2:
This reducer takes the output text file of training phase as a cache file. Through this file we develop a dictionary which stores total no. of words in each class. Now, operating on the sorted concatenated file it outputs a text file containing for each sample id the scores of all the classes.
MapReduce phase 3
The aim of this phase is to output the predicted class corresponding to each sample. It takes as input the output file of MapReduce phase 2. It uses the same identity mapper as used in MapReduce phase 2.
Reducer3:
This reducer takes the output of the training phase output file as a cache file in order to store a dictionary of total no. of documents corresponding to each class to add the prior information to the scores of each class. Now using this information the reducer outputs predicted class(class with maximum score) corresponding to each sample.

5. Results and time analysis

Classification Accuracy of test set: **82.4% Classification Accuracy of test set**: **78.1%**

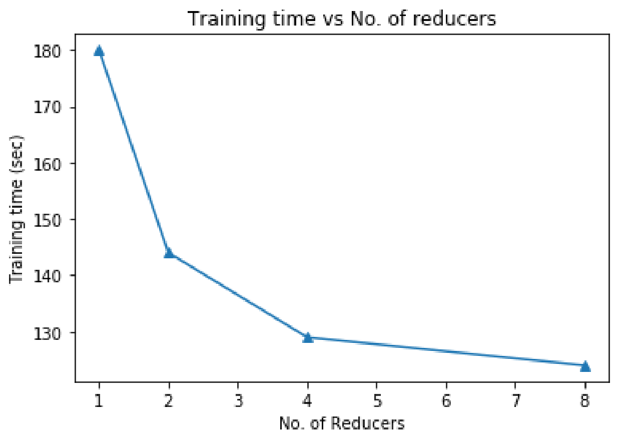

Figure 1. Training Phase time vs. No. of reducers

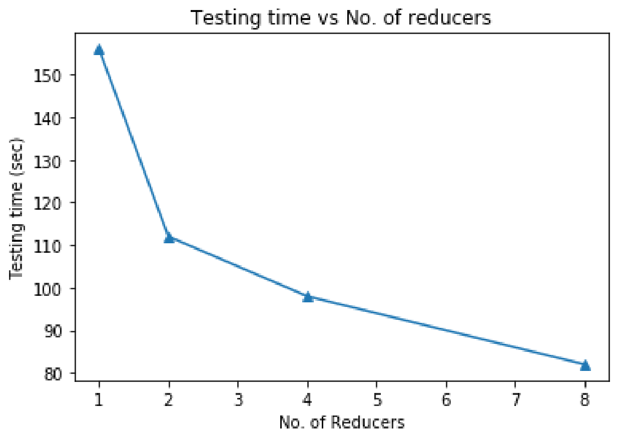

Figure 2. Testing Phase time (3 MapReduce Phases) vs. No. of reducers

6. Conclusions

In this assignment we developed some experience in implementing machine learning algorithms in a distributed environment. We implemented Naive Bayes classifier using Hadoop Framework and compared it against its in memory implementation. Experiments suggest that the in memory implementation slightly outperforms distributed implementation in terms of total time taken by the experiment to run which is **146 sec** for in memory implementation to **212 sec** in Hadoop largely due to higher training phase time. This can be attributed to the fact that the dataset is not large enough and the rigid *(key,value)* structure of MapReduce can limit the optimization in implementation. Though, it graphs suggest that increasing the no. of reducers sharply decreases the execution time.