Naive Bayes implementation Memory And Map-Reduce Assingment-01 DS-222 (Machine Learning with Large Dataset)

Chaikesh Chouragade
(SR No. 14358)

Abstract

Implemented Naive bayes clasiifier in Memory and in Map-Reduce Framework.Experimented with number of reducers and reported the accuracy and runtimes for training and testing.

1. In Memory Naive Bayes

Used python programming language and 8GB Memory laptop with no GPU or pqrallelization support.

1.1. DATA Preprocessing

Lower Case: All words converted to words with lowercase letters.

Length: All words with length less than 3 were removed.

Punctuation : All punctuation mark and digits were removed.

1.2. Training

Approach: Overall,maintained dictionaries equal to number of document classes , which is 50 in our case . Each dictionary corresponds to particular class label with words in class as key, and count of word in the respective class as values.
while training if we encounter particular key ,we increment the values corresponding to that key.In addition to this maintened two counter for prior probability calculation.it essentially counts number of toatal instances/documents and number of instances/document of particualr class.
Parameter: vocab*#class + #class =24162200
Training Time: 367 sec
Training Accuracy: 81.97%

. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1.3. Testing

Approach: during test time we calculating log probabilities for a test document to belong to a each class and assign the claas which has maximum probability.

Development Accuracy: 79.01 %
Test Time: 163 sec
Test Accuracy: 90.23 %

2. Map-Reduce naive Bayes

2.1. DATA Preprocessing

Same as Done for In Memory Naive bayes.

2.2. Training

Single Mapper and Reducer was used .description of mapper and reducer is as follows-
Parameter: vocab*#class + #class =24162200

Mapper: Input to mapper is full_train.txt ,which spill out a tuple
{word,label,1}
{ANY,label,1}
For prior calculation
{label,1}
{ANY,1}
Reducer: Reducer sums the value by sorting on the basis of keys,emitting a tuple
{word,label,counts}
{ANY,label,counts}
For prior calculation
{label,counts}
{ANY,counts}

Training Time: best with 10 reducers is 125 second
Training Accuracy: best with 10 reducers is 81.72%

Table 1. Training-time(seconds) on Map-Reduce for Naive Bayes.

REDUCERS	MAP	SHUFFLE	MERGE	REDUCE
1	104	12	8	36
2	103	10	4	18
5	107	18	1	8
8	105	18	1	6
10	105	14	1	5

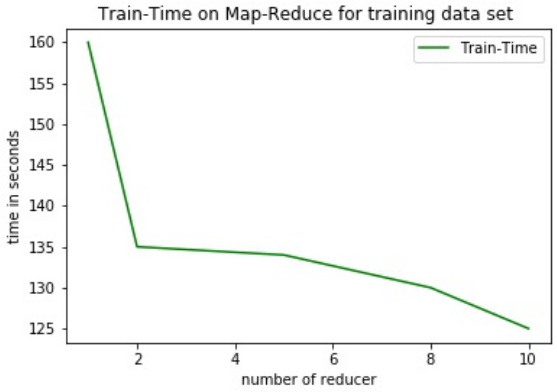

Figure 1. Historical locations and number of accepted papers for .

2.3. Testing

overall 3 mapper and 3 reducers were used at test time plus some local merging is performed, description of mappers and reducers are as follows-

Mapper1: Input to mapper is full_test.txt ,which spill out a tuple
{word,docIndex,1}

Reducer1: Reducer sums the value by sorting on the basis of keys,emitting a tuple
{word,docIndex,count}

Concatenate: in this stage we concatenate output of reducer1 below the output of training phase reducer.

Mapper2: Input to this mapper is output from previous concatenation stage.Mapper is an Identity mapper ,which spills the same tuple as input.
After MapReduce Sorting operation we get to exploit the fact that all the event counts for word are now stored **sequentially**.
The {word,label,counts} records from output of training phase precede the {word,docIndex,count} from output of

Table 2. Testing-time(seconds) on Map-Reduce for Naive Bayes.

REDUCERS	MAP	SHUFFLE	MERGE	REDUCE
1	49	13	4	281
2	29	13	0	81
5	42	12	1	43
8	45	11	1	36
10	51	11	1	26

reducer1.Essentially we have all the necessary information for probability calculation,including the hashmap we maintained by utilizing cache file.

Reducer2: In this reducer most of the calculation part takes place,and is most time consuming.Sorting is done according to {word}
Output of Reducer2 is a tuple
{docIndex,label,score}
where score is probability of a docIndex belonging to that particular label.
Mapper3: Input to this mapper is output of reducer2,again Mapper is an Identity mapper ,which spills the same tuple as input.
Sorting is done according to {docIndex,label},so that we get all docIndex sequentially.

Reducer3: Reducer sums up the score for a particular {docIndex,label}.then it compares the probability and spill out the maximum probability label corresponding to the particular docIndex

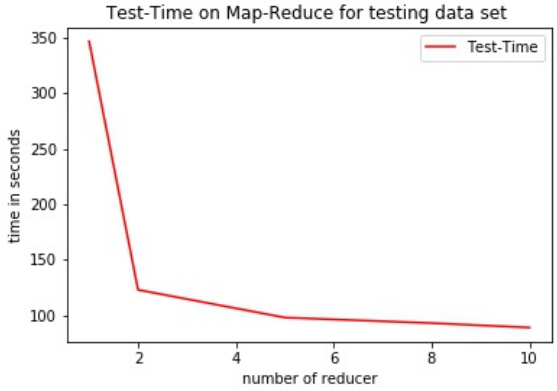

Figure 2. Historical locations and number of accepted papers for .

Accuracy: accuracy is calculated by comparing the true label and predicted label from the output of reducer3.//

Development Accuracy: 78.27%

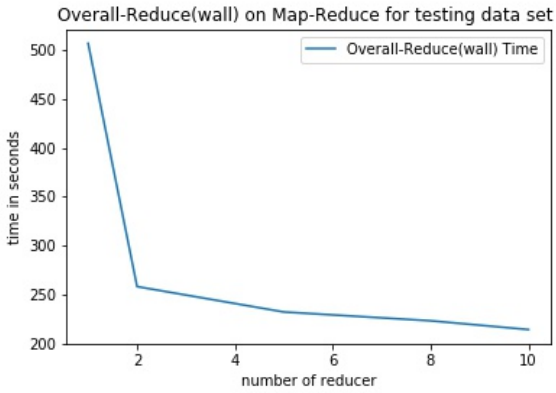

Figure 3. Historical locations and number of accepted papers for .

Test Time: best with 10 reducers 89 sec
Test Accuracy: best with 10 reducers is 80.13%

2.4. Conclusion

- The important observation by experimentation is that the both the training time and testing time does not decreases linearly.First it decreases sharply and then decreases slowly.It can be due to the fact that as number of reducers increases the workload for master node to distribute tasks and to do appropriate partition also increases.
- The accuracy in both the method is nearly same, As both the method uses the same algorithm ,but differ only in implmentation we get nearly same accuracies for train ,development and test data set.
- We get significant reduction in time. 65% reduction in training time and 45% reduction in test time and overall reduction in time is around 60%.But important point to note is that if we had larger dataset,in memory method would be useless,only option would be mapreduce.