DS222 ML With Large Datasets - Assignment 1

Apoorv Umang Saxena [1]

Abstract

This report shows the results of running the Naive Bayes algorithm for document classification in 2 different setups - one locally on a single machine and the second using MapReduce on the turing cluster.

1. Naive Bayes formula used

The following formula (nai) was used to calculate class probabilities during the prediction phase for both the local and mapreduce implementations. $\hat{P}(x_i \mid \omega_j)$ is the class conditional probability which is calculated as

$$\hat{P}(x_i \mid \omega_j) = \frac{\sum tf(x_i, d \in \omega_j) + \alpha}{\sum N_{d \in \omega j} + \alpha \cdot V}$$

where

- x_i: A word from the feature vector x of a particular sample.
- $\sum tf(x_i, d \in \omega_j)$: The sum of raw term frequencies of word x_i from all documents in the training sample that belong to class $_j$.
- $\sum N_{d \in \omega j}$: The sum of all term frequencies in the training dataset for class $_j$.
- α: An additive smoothing parameter (set to 0.1)
- V:The size of the vocabulary (number of different words in the training set)

The class-conditional probability of encountering the text x can be calculated as the product from the likelihoods of the individual words (under the naive assumption of conditional independence)

$$P(\mathbf{x} \mid \omega_j) = P(x_1 \mid \omega_j) \cdot P(x_2 \mid \omega_j) \cdot \cdot P(x_n \mid \omega_j)$$

$$= \prod_{i=1}^{m} P(x_i \mid \omega_j)$$

2. Document preprocessing

The following steps were taken to reduce the noise present in the documents in the training data and test data. This helped in reducing the size of the vocabulary so that the feature vector size is reduced.

1. Removing URLs, punctuation and unicode character codes
2. Converting to lowercase
3. Stemming - using the Snowball stemming algorithm (Sno)
4. Removing stop words

3. Parameters

- Unique words (Vocabulary): 211577
- Classes: 50
- Priors for each of the 50 classes

$Totalparameters = 211577 * 50 + 50$

Hyperparameters:

α - Additive smoothing factor set to 0.1

4. Difference between local and MapReduce implementations

The local implementation uses the *nltk* (nlt) library for removal of stopwords while the MapReduce implementation uses a smaller list of stopwords. This results in a slightly increased test accuracy for the local implementation.

5. Prediction accuracies

See table 1

6. Time Taken

6.1. Local implementation

Preprocessing of the documents took the majority of the time in the local implementation (1h 15min 23 sec for 214997 documents in the training set).

	Local	MapReduce
Train	84.8	85.36
Devel	77.10	77.10
Test	80.15	80.11

Table 1. Correct class prediction accuracies

Number of Reducers	Time taken (sec)
1	62
2	40
5	18
8	14
10	12

Table 2. Training time on turing cluster

The word/class counting time (reduction time for the MapReduce implementation) was **106 seconds** for the local implementation on the full training dataset.

6.2. MapReduce implementation

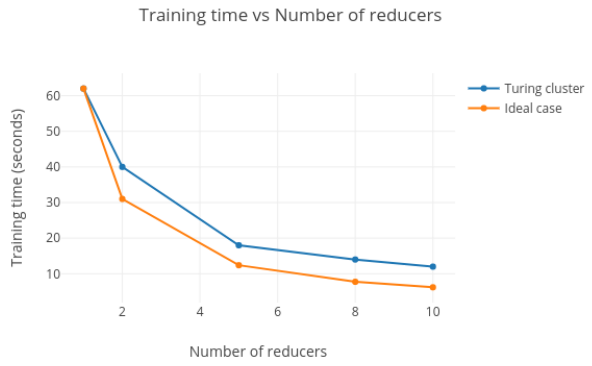

Figure 1. Training time on turing cluster vs number of reducers

7. Results

The training time is **not linear** with respect to the number of reducers.

In the ideal case, time taken should halve if the number of reducers is doubled. However, this implementation of Naive Bayes using MapReduce performs worse than the ideal case.

This is because parallelizing the reduce step involves significant overhead and increasing the number of reducers beyond a certain point will eventually make the performance worse.

References

Snowball Stemming algorithm. http://snowballstem.org/. Accessed: 2018-09-7.

Sebastian Raschka naive bayes and text classification. https://sebastianraschka.com/Articles/2014_naive_bayes_1.html. Accessed: 2018-09-7.

NLTK natural language toolkit - python library. https://www.nltk.org/. Accessed: 2018-09-7.