Parallelizing Semi-Supervised Machine Learning Algorithms

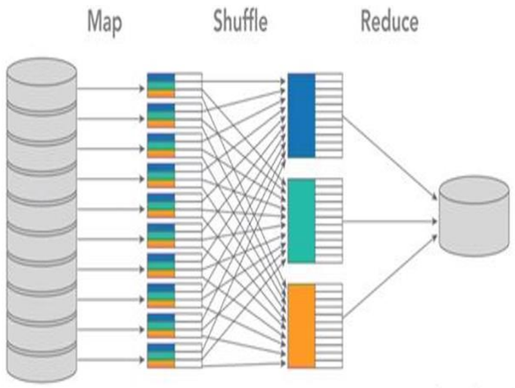
Nicholas Gauthier
Rakesh M. Verma, University of Houston
ReDAS Lab

Background

I am aiming to implement a quick and efficient way of using semi-supervised learning (SSL) methods with big data. These algorithms do not typically scale well, and become slow as well as expensive when working with copious amounts of data.

SSL methods are advantageous compared to supervised or unsupervised methods because of its improved learning accuracy with labeled and unlabeled data as well as ability to work with large amounts of unlabeled data.

I plan to achieve this by implementing these SSL algorithms on Hadoop's MapReduce framework. MapReduce is designed for working with big data and distributing the processing across many compute nodes. It is typically broken into two functions: Map and Reduce.



Objectives

1. Identify which SSL algorithms have been previously parallelized.
2. Study the algorithms that have yet to be parallelized, pick a model to use with each algorithm, and implement a serial implementation of the algorithms with their respective models.
3. Convert the serial implementation of the algorithms into a correctly functioning parallel version in MapReduce.
4. Test the MapReduce implementation on multiple datasets.
5. Compare the output and runtime of the MapReduce implementations vs. the serial implementations.

Models and Algorithms

Models and Algorithms Chosen (Model - Algorithm):

1. Multinomial Naïve Bayes (MNB) - **Semi-supervised Expectation Maximization (SS-EM)**
2. To Be Determined - **Transductive Support Vector Machine (TSVM)**
3. To Be Determined - **Self-Training**

Methods

I am currently working on parallelizing MNB with SSEM. MNB is being used as a model for Text Classification. I expect the MapReduce version to be slower when comparing the two with this dataset as it is quite small. It should also be noted that I did not write the serial implementation I am working with, and the code is open source.

Understanding the serial implementation

Researching several libraries and subfunctions used such as: Sci-kit learn, Scipy, and Numpy.

Identify what can be parallelized

I wrote pseudocode to help me visualize which portions of the program could be parallelized in order to save computation time.

Learning Hadoop

Learning how to use HDFS, command line arguments, and config files.

I am also utilizing the 'MRJob' API because it simplifies job submission, I/O tasks, and Hadoop configuration.

Pre-processing the Data

The data sets used in this program have been pre-processed in the sci-kit learn library. I had to locate the original dataset, 20newsgroups, and re-process it so that it can be used in MapReduce.

Discussion

The parallel implementation of Multinomial Naïve Bayes (MNB) with Semi-supervised Expectation Maximization (SS-EM) is still a work in progress, but the following are a couple of the implications that I expect will happen at the end of this project:

- Building the initial classifier (Before the EM iterations) will be significantly quicker as it involves quite a bit of matrix multiplication.
- To find that the parallel implementation of MNB with SS-EM will be drastically slower on small data sets, but significantly faster with big data.

Future Work

- Finish parallelizing the Multinomial Naïve Bayes and Semi-supervised EM program.
- Testing the program on several datasets, and comparing it to the serial implementation if applicable.
- Determine models for TSVM and Self-Training
- Parallelize TSVM and Self-Training on MapReduce.
- Test TSVM and Self-Training on several datasets comparing it to the respective serial implementations if applicable.

Acknowledgements

This research was made possible by a NSF grant to the University of Houston Computer Science Department (NSF CNS-1551221).

References

Langit, Lynn. *Introducing MapReduce*. 5/20/2015. *Lynda.com*. Accessed 8/5/19. https://www.lynda.com/NoSQL-tutorials/Introducing-MapReduce/368756/387721-4.html