**Distributed Logistic Regression Report**

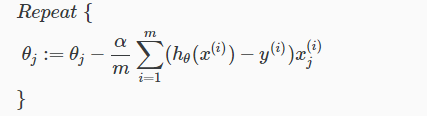
Peng Rui, 14331226, School of Data and Computer Science, Sun Yat-Sen University

**Concepts and Techniques**

**Simplified Cost Function**

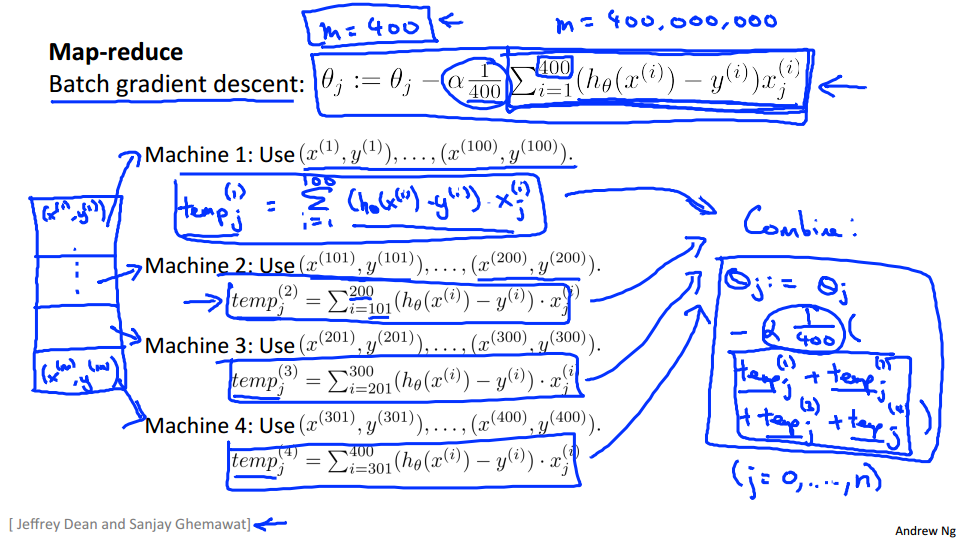


**Gradient Descent**



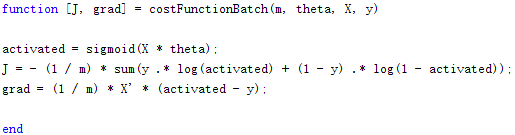
**Map-reduce and data parallelism**

We can divide up batch gradient descent and dispatch the cost function for a subset of the data to many different machines so that we can train our algorithm in parallel.



**Simulating Distributed Logistic Regression**

**Implementing Cost Function and Gradient Using Vectorization**

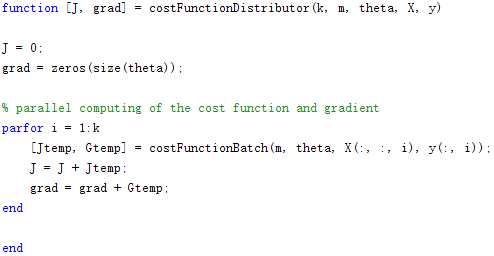


**Distributing Cost Function and Gradient**

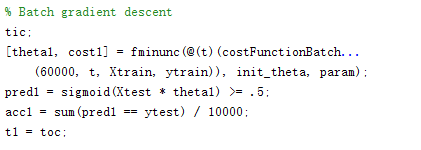
*parfor loopVar = initVal:endVal; statements; end executes for-loop iterations in parallel on workers in a parallel pool.*

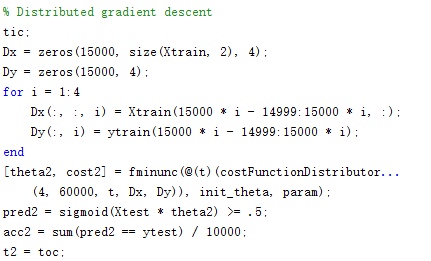
*MATLAB® executes the loop body commands in statements for values of loopVar between initVal and endVal. loopVar specifies a vector of integer values increasing by 1. If you have Parallel Computing Toolbox™, the iterations of statements can execute on a parallel pool of workers on your multi-core computer or cluster. As with a for-loop, you can include a single line or multiple lines in statements.*

Here, Parallel Computing Toolbox is used to simulate distributed logistic regression.

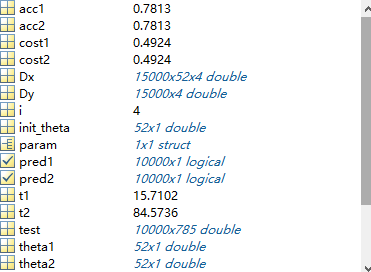


**Test with Simplified MNIST Database**





After running both version of logistic regression, we can conclude that the results of both method are very similar, with the testing accuracy of about 0.7813 and cost about 0.4924. However, the distributed version cost more time. Running logistic regression without paralleling costs only 15.7102 seconds, while the same algorithm with paralleling costs 84.5736 seconds.



**Reasons of the Worse Efficiency of Distributed Logistic Regression**

*Vectorizing code allows you to benefit from the built-in parallelism provided by the multithreaded nature of many of the underlying MATLAB libraries. However, if you have vectorized code and you have access only to local workers, then parfor-loops may run slower than for-loops. Do not devectorize code to allow for parfor; in general, this solution does not work well.*

Here, my implementation of the cost function and gradient are vectorized, and the vectorized implementation benefits from the built-in parallelism. And this is probably the reason why my parallel version of logistic regression runs much slower. Also devectorizing to allow parfor is deprecated. Vectorizing code in MATLAB helps to make it run faster.

**Conclusion**

The materials and references of this report are mainly from the course [Machine Learning](http://www.ml-class.org) and [MATLAB documentation](https://www.mathworks.com/help/matlab/index.html). A learning algorithm is Map-reduceable if it can be expressed as computing sums of functions over the training set. Linear regression and logistic regression are easily parallelizable.

However, in MATLAB, vectorized code has already benefited from the built-in parallelism provided by the multithreaded nature of many of the underlying MATLAB libraries. As a result, if you want to make code run faster, first try to vectorize it.

**Appendix**

To reduce the size of the submitted zip flie, the data sets of this project are not included. You can download it from [the MNIST Database](http://yann.lecun.com/exdb/mnist/). The C++ code of initializing the database and MATLAB code of running this algorithm are provided in the src folder.