CS294-1 Behavioral Data Mining Spring 2012

Programming Assignment 2: A Large-Scale Linear Regression Sentiment Model

Due: Tuesday February ~~21~~ 24

The goal of this assignment is to apply linear regression at scale to a numerically-scored sentiment dataset. The data were collected by Mark Dredze and others at Johns Hopkins. The web site with original data is here: http://www.cs.jhu.edu/~mdredze/datasets/sentiment/. The data are amazon.com reviews in a variety of product domains. For the assignment, you should process at least the book review dataset which contains about 1 million reviews. While the dataset is larger than in HW1, this is not a "cluster-scale" problem. On the other hand, it will require you to focus on performance. Running a large-scale task on one machine without hitting speed bumps is an important step toward writing a resource-efficient cluster job.

To evaluate the quality of your model, you should do a 10-fold cross-validation and compute AUC and 1% lift scores (lecture 3). The model should be a linear regression model but you have several design degrees of freedom:

- L_1 or L_2 error functions.
- Ridge or Lasso regularization.
- Exact solution (assumes squared error and ridge regression) or stochastic gradient.
- Unigram, bigrams, trigrams as features
- Stemming
- Stopword removal

You don't need to explore all these options but you should try at least two or three.

Tokenized data

Tokenizing or parsing (this dataset uses XML tags but is simple enough not to require a parser) the data can consume a lot of time. To get straight to the modeling task, tokenized binary data is available at this location:

http://www.cs.berkeley.edu/~jfc/DataMining/SP12/restricted/HW2/books

The README.txt file explains the contents of the files. Other product categories will be added. It's likely that the data matrix will not fit in memory on your machine, so you will have to read the data in blocks. It should be possible to complete the data-to-model calculation in an hour or less. You will probably want to compute and save some intermediate representations (e.g. bag-of-words or bag-of-ngrams) before doing the regression so that you don't have to do everything from scratch each time you recompute a model.

Matrix Library

As we saw in the first lecture, naïve matrix arithmetic for dense matrix inversion is *much* slower than an optimized library. It should take less than half an hour to solve a 20k x 20k system with a good library but several days with a naïve implementation. You can use Matlab for this assignment which includes good implementations of sparse and dense matrix operations. If you don't have access to Matlab or choose not to go this route, you can still get reasonable execution times using sparse matrix operations and stochastic gradient. The gap between optimized and naïve sparse matrix operations is much smaller.

Writeup

Please describe the sequence of steps you took in constructing and evaluating your model. There should be enough detail to allow someone else to reproduce the results. Include graphs (e.g. ROC and lift plots, error-vs-iteration plots) , and lists of the strongest positive and negative terms. Please also compute the Gflop performance of your method. You can count the flops for each matrix operation (or just do matrix multiplies, which should dominate) based on the matrix dimensions for dense operations, or the number of non-zeros for sparse matrices. Hand in a hardcopy and send a zip file with the code and graph data to kenghao@cs.berkeley.edu by Friday Feb 24.