Parallel Programming Final Project Report

Detian Deng

May 13, 2015

Abstract

In this project, with application to a pain prediction problem using fMRI data, three versions of paralleled LASSO solvers based on Stochastic Coordinate Descent were implemented and their speed-up/scale-up were analyzed.

1 Background and Data Description

Functional magnetic resonance imaging (fMRI) is a functional neuroimaging procedure using MRI technology that measures brain activity by detecting associated changes in blood flow. Wager and et. al. used fMRI data as biomarkers to predict the scale and type of pain that a subject feels. [4]
The scientific goal of this project is to find better biomarkers (regions of the brain) and improve the prediction accuracy. The computation challenge is that fMRI data are often large. In this particular dataset, the outcome Y_{ij} is a continuous pain score from 0 (no pain) to 10 (extreme pain) for subject i at scan session j, where $i = 1, \dots, 40$ and $j = 1, \dots, 165$, and the covariates X_{ijk} is the preprocessed fMRI intensity of the kth voxel for subject i at scan session j, where $k = 1, \dots, 16000$. These 16000 voxels are pre-selected from the raw scan data based on meta analysis of 224 previous studies. Therefore, the data matrix is 6600×16000.
Thus in order to speed up the training procedure and gain scalability, proper parallel implementation is needed.

2 Paralled LASSO

In this project, parallel performance was only explored for LASSO regression, which serves as a benchmark, and more sophisticated prediction models can be explored in the future.
LASSO regression is a linear regression model with quadratic loss function plus $L1$ penalization on the coefficient vector β, thus the optimization problem is to minimize the following function with given regularization parameter λ:

$$\frac{1}{2}||X^T\beta - Y||_2^2 + \lambda||\beta||_1$$

Given the non-differentiability of the $L1$ term, this optimization has to be solved iteratively. Knowing that this problem can be solved by many algorithms such as LARS (Efron et al., 2004) and BBR (Genkin et al., 2007), we chose to use the Stochastic Coordinate Descent (SCD) (Shalev-Shwatz et al., 2011) [3] and its parallel version pSCD (Bradley et al., 2011)[1] on this dataset, since Shalev-Shwatz showed this method outperforms other state-of-the-art approaches on many large datasets. Briefly speaking, pSCD is a procedure that at each iteration, randomly picks P (number of threads) coordinates of β and minimize them individually using soft-threshold with multiple threads, then collectively updates β.

3 Experiments

The authors of pSCD made their C++ source code with R/Matlab API publicly available (https:// github.com/ lianos/ buckshot), however, the code has many bugs, most importantly, it actually did not implement the pSCD algorithm they described in the paper, instead, it cycles across all coordinates without sampling them, also it updates the loss function using compare-and-swap (CAS) synchronization every time immediately after a coordinate is updated, which makes it somewhere between the step-wise update as in Friedman's [2] cyclic coordinate descent,and the collective update described in their paper. Therefore, based on the published source code, I implemented three versions of the Parallel Coordinate Descent algorithm:

1. Paralleled Stochastic Coordinate Descent as described in Bradley's paper. (pSCD)
2. Cyclic Coordinate Descent with CAS synchronization as intended in the source code. (sCCD)
3. Cyclic Coordinate Descent with the inner loop of updating $X^T\beta$ computed in parallel. (pCCD)

For all experiments, the regularization parameter λ was fixed to 1.6, which is chosen as the best value based on leave-1-subject-out cross validation, and the convergence tolerance was set to 10^{-5}. And all experiments took place on the same AWS c3.4xlarge EC2 instance with 16 virtual cores. However, the pSCD and sCCD algorithms are very unstable when features are highly correlated as the voxels in fMRI data (explained more in the following section), therefore in addition to the fMRI dataset, a simulated dataset with X of size 2000×2000 and feature-wise independent was also used.

4 Results

For fMRI dataset, pSCD and sCCD failed to converge when using more than 2 threads. The reason could be explained intuitively using figure 1. When the features are correlated, the collective update makes a worse update than the serial one, which could even make the sequence diverge, and when the features are uncorrelated, the collective update makes a better update than the serial one, which suggests fewer total updates and hence speed-up. The theoretical guide line given by the paper is that the number of threads $\leq d/\rho + 1$, where d is the dimension of X and ρ is the spectral radius of $X^T X$. However, the voxels in fMRI data are intrinsically highly correlated and in this dataset the spectral radius is larger than its dimension, thus pSCD and sCCD are not good parallel solutions for our dataset. In contrast, pCCD converges well on fMRI data. It is robust to the correlation between features since it is basically the same algorithm as CCD, and the parallelism brought by updating $(X^T\beta)_i^{(k+1)} = (X^T\beta)_i^{(k)} + X_{ij}\delta\beta_j$ speeds up the optimization very well when X has large enough number of rows so that each thread would have enough work to amortize the start-up cost.

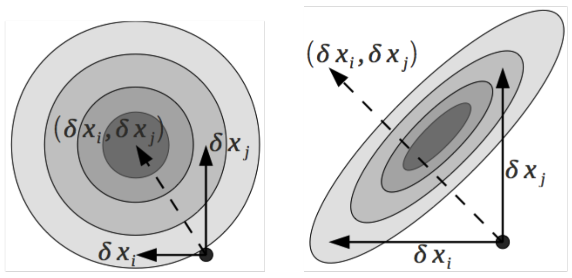

Figure 1: Intuition for parallel coordinate descent. Contour plots of two objectives, with darker meaning better. Left: Features are uncorrelated; parallel updates are useful. Right: Features are correlated; parallel updates conflict. δx means $\delta\beta$ in our context.

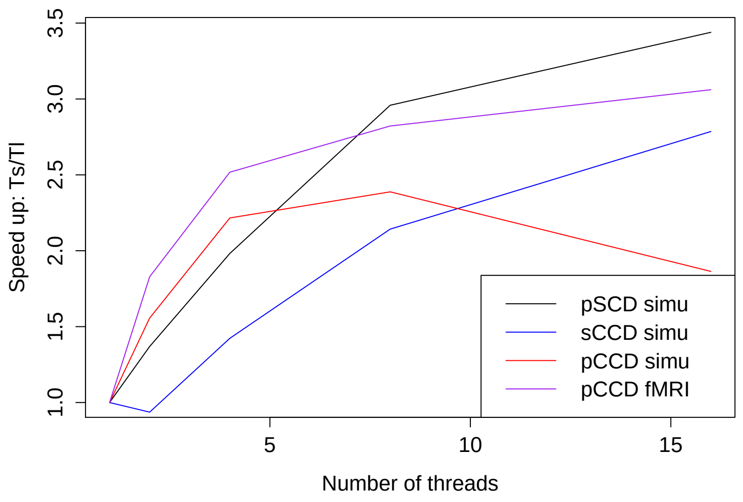

Figure 2: Speed-up comparison between three methods. Black, red, and blue line suggest application to simulated data set. Purple line suggests application to fMRI data set.

Since pSCD and sCCD failed for fMRI data, in order to compare these methods, they were also applied to a simulated data. The data were simulated by first randomly generating a matrix $X_{2000\times1000}$ and a vector $\beta_{1000\times1}$ and setting outcome $Y = X\beta + \epsilon$ where ϵ is a Gaussian white noise. Then another matrix $Z_{2000\times1000}$ were generated and column bind with X to serve as the input features. Speed-up curve of these three methods is shown in figure 2, which indicates that pSCD and sCCD do have very good speed-up property when the predictors are not correlated. But pCCD does not speed up very well on simulated data set when using more than 8 threads, probably due to the fact that the simulated data set is smaller so that there might not be enough work for each thread to amortize the data loading and thread initializing cost.

In figure 3, scale-up comparison was made based on running time on simulated data set with varying number of rows but fixed number of columns. As we can see, pSCD has in general better scale-up but the trajectory is some what unpredictable because of its stochastic nature. And sCCD scales a little better than pCCD probably because of the memory access pattern of the implementation of the sparse vector that stores the data.

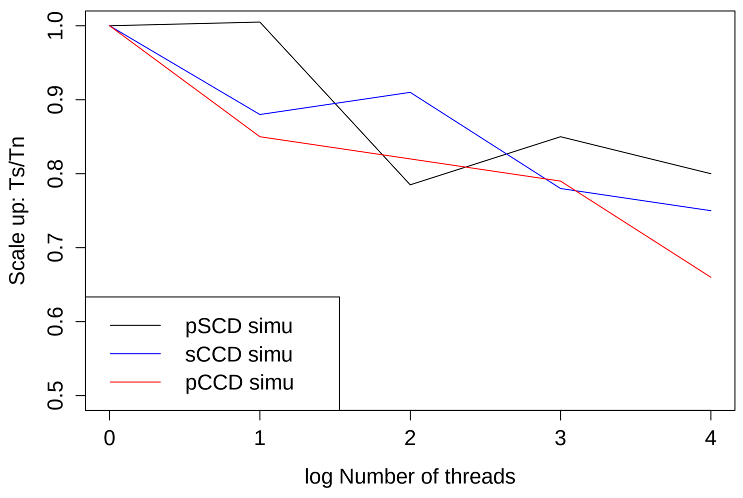

Figure 3: Scale-up comparison between three methods. Black, red, and blue line suggest application to simulated data set.

In general, pCCD is the most robust method with good accuracy but almost always the slowest if the other tow also converge. sCCD is subject to large risk of divergence for correlated features, but if features are uncorrelated, it converges faster than pCCD with good accuracy. pSCD cannot deal with correlated features either but if converge, it is always the fastest, however, since it chooses coordinates randomly, by chance, some coordinates are chosen less frequently thus its solution is a worse approximation to the optimum comparing too the other two methods.

5 References

[1] Joseph K Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for l1-regularized loss minimization. *arXiv preprint arXiv:1105.5379*, 2011.

[2] Jerome Friedman, Trevor Hastie, Holger Höfling, Robert Tibshirani, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[3] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l 1-regularized loss minimization. *The Journal of Machine Learning Research*, 12:1865–1892, 2011.

[4] Tor D Wager, Lauren Y Atlas, Martin A Lindquist, Mathieu Roy, Choong-Wan Woo, and Ethan Kross. An fmri-based neurologic signature of physical pain. *New England Journal of Medicine*, 368(15):1388–1397, 2013.