A Scalable and Portable Framework For Massively Parallel Variable Selection in Genetic Association Studies

Gary K. Chen

Division of Biostatistics, Department of Preventive Medicine, 2001 North Soto Street Los Angeles, CA 90089-9237

ABSTRACT

Summary:
The deluge of data emerging from high throughput sequencing technologies poses large analytical challenges when testing for association to disease. We introduce a scalable framework for variable selection, implemented in C++ and OpenCL, that fits regularized regression across multiple Graphics Processing Units (GPUs). Open source code and documentation can be found at a Google Code repository under the URL http://code.google.com/p/parallel-lasso/.

Contact: gary.k.chen@usc.edu

1 INTRODUCTION

As the cost of sequencing continues to drop exponentially, it will soon be practical to test all variation in the genome for association to disease using data from thousands of individuals. There are obvious computational challenges in analyzing data sets on this scale. Regularized regression methods such as the LASSO (Tibshirani, 1996) and other extensions are appropriate tools for selecting important variables in problems where variables far exceed observations. Programs like glmnet (Friedman et al., 2010) are computationally efficient for small to moderately sized datasets, but do not scale to extremely large datasets due to memory burden. We introduce an object-oriented framework that scales across nodes on GPU clusters yet shields users from the underlying complexities of a distributed optimization algorithm, allowing them to easily implement custom Monte Carlo routines (e.g. permutation testing, bootstrapping, etc.). Practical use of our framework is demonstrated by an application to real and simulated data.

2 IMPLEMENTATION

Our C++ framework, named gpu-lasso, implements the mixed L1 and L2 penalized regression model of Zhou et al. (Zhou et al., 2010) on datasets with any arbitrary number of variables. L1 penalties enforce sparsity while L2 penalties enable correlated predictors within groups (e.g., genes, pathways) to enter the model as well. gpu-lasso exploits the optimization scheme of greedy coordinate descent (GCD) which, upon estimating regression coefficients across all variables, updates the single variable leading to the greatest improvement to the likelihood with its new coefficient. In general this requires more iterations to converge than cyclic coordinate descent (CCD), which updates each regression coefficient as it cycles through variables. However, this disadvantage diminishes for sparser models. More importantly, greedy coordinate descent exposes parallelism across subjects and variables which makes it both a better fit for GPU processors and a more scalable algorithm compared to cyclic coordinate descent, which only exposes parallelization at the subject level. Since GPU memory is far more limited than host memory, for larger datasets such as whole-genome sequence data it is essential to coordinate optimization across two or more GPUs. MPI routines in our framework handle the this coordination, enabling GPUs to be distributed across a network. Our GPU kernels are implemented in OpenCL which assure maximum portability across either ATI or nVidia GPU devices.

We compared run time behavior of our program across multiple configurations and to glmnet (Friedman et al., 2010). We were also interested in scalability properties as optimization is split across nodes. Our host was configured with a pair of nVidia Tesla C2050s, 24 Xeon X5650 cores, and 48 GB of RAM. We generated datasets of various sizes by extracting genotypes from the first 250,000 SNPs and 1 million SNPs (ordered by position) of a large GWAS (see Application). Table 1 shows that due to its implementation in the R environment, glmnet has a much heftier memory requirement compared to our framework.

Table 1. Computational requirements

<table>
<thead>
<tr>
<th>Method</th>
<th>Time per Iteration</th>
<th>Memory per node</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>250,000 variables</td>
<td></td>
</tr>
<tr>
<td>glmnet</td>
<td>2m20s</td>
<td>46GB</td>
</tr>
<tr>
<td>gpu-lasso (1 CPU)</td>
<td>54s</td>
<td>415MB</td>
</tr>
<tr>
<td>gpu-lasso (1 GPU)</td>
<td>1.85s</td>
<td>415MB</td>
</tr>
<tr>
<td>gpu-lasso (2 GPU)</td>
<td>.93s</td>
<td>208MB</td>
</tr>
<tr>
<td></td>
<td>1 million variables</td>
<td></td>
</tr>
<tr>
<td>glmnet</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>gpu-lasso (1 CPU)</td>
<td>3m47s</td>
<td>1.7GB</td>
</tr>
<tr>
<td>gpu-lasso (1 GPU)</td>
<td>7.7s</td>
<td>1.7GB</td>
</tr>
<tr>
<td>gpu-lasso (2 GPU)</td>
<td>3.8s</td>
<td>863MB</td>
</tr>
</tbody>
</table>

7,000 subjects in all analyses

© The Author(s) 2012. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/2.5/uk/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
controls FDR at the .05 level. The first two SNPs listed in the table replicate significant findings in earlier studies of prostate cancer (Murabito et al., 2007; Schumacher et al., 2011) while the last SNP appears to be a genuinely novel risk variant as we have recently replicated this finding in an independent Stage 2 analysis (Haiman et al., 2011).

4 DISCUSSION

We describe our scalable framework gpu-lasso which can be particularly useful for fitting sparse models in high-dimensional settings. To demonstrate how one can carry out Monte Carlo routines with our framework, we provide full source code listing for the C++ class that implements Stability Selection in our Supplementary document. We should stress that our choice of GCD as our optimization routine may not be ideal in other contexts, particularly when large models need to be estimated, such as exploration of the entire LASSO path over a grid of values for the optimal penalty parameter. In this case, cyclic coordinate descent may be preferable as first, the increased number of iterations for GCD may swamp out gains from limited parallel resources, and second, GCD may potentially converge to models that overestimate sparsity. Alternatively, one could conceivably constrain the search to a set of candidate (sparse) models by adding a BIC penalty for example. For smaller datasets, software such as glmnet can be more practical, since efficient routines like the LARS algorithm (Efron et al., 2004), which solves the LASSO path without exploring a penalty parameter grid, are already bundled. As datasets increase in sample size, LARS and related approaches can lose their edge over a simple (parallelized) penalty grid search since such methods require inversion of a covariance matrix with dimension bounded by the number of samples (i.e. O(n)).

ACKNOWLEDGEMENT

I would like to thank Kenneth Lange and Marc Suchard for their insights, Kai Wang and Paul Thomas for early feedback, the USC Epigenome Center for GPU computing resources, and the African American Prostate Cancer Consortium, listed in (Haiman et al., 2011), for contributing data in the GWAS application.

REFERENCES