Parallelizing Random Lasso

Members — James Matthew Hamilton & Daniel Karasek

Note, applied for c-day.

Introduction:

Random lasso is the state-of-the-art method of regression analysis used on high dimensional genetic datasets for predicting complex gene regulatory networks (GRNs). Random lasso builds upon some of its predecessors, e.g. least squares, ridge regression, and lasso regression.

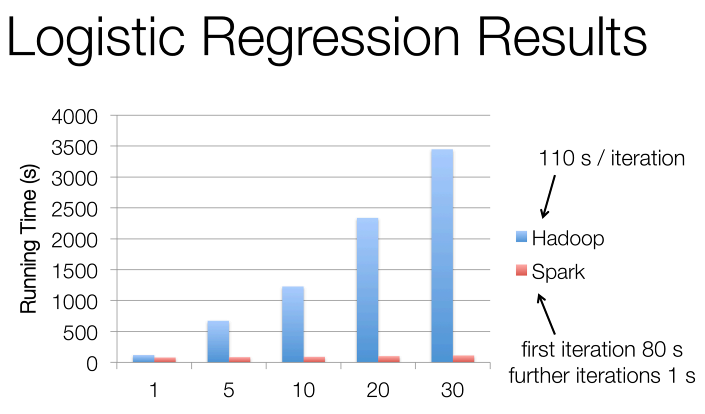
Problem Definition:

It has long been a challenge in bioinformatics to improve the accuracy and speed of algorithms that predict complex GRNs. **Runtime for any meaningful research using random lasso is often hours or days.** Parallel programing takes advantage of multiple cores within a computer to run tasks congruently, but this is only beneficial if these congruently run tasks are completely independent of each other. Random lasso has highly-independent tasking – making the prospect of parallel computing attractive.

The problems we tackled: (1) identifying areas most susceptible to parallel programing within random lasso, and (2) implement parallel computing in the most algorithmically effective was on the aforementioned susceptible areas.

Methods:

Using Apache Spark we implemented parallel programing on a random lasso algorithm – a form of regression analysis. Apache Sparks builds on Hadoop, and has been shown to outperform competitors in regression analysis:



Using **Sparklyr** and **PySpark** we can implement Spark in R and Python respectively. No packages exist for random lasso in R or Python, but one is currently in the works in the Department of Computer Science at Kennesaw State University. Researchers will often opt for less complex algorithms, so **an openly available random lasso package would benefit researchers.** We plan to implement Spark within this new random lasso package in the works.

Experiment Design:

Our methods of parallelizing random lasso focus on the highly independent bootstrapping and sampling processes with random lasso. Random lasso essentially runs lasso many time with a random sample each time. The final result is simply an average of all the previously calculated, through lasso, coefficients. Since **lasso is the most computationally burdensome step within each independent bootstrap**, we plan to parallelize these lasso bootstraps. **A computer with two cores should nearly reduce the runtime in half. We will record and compare our incremental changes to random lasso without spark.**

Data:

Random lasso is particularly beneficial in bioinformatics, but it can be applied to any dataset that meets the following conditions: (1) high dimensionality, and (2) a sample size that is significantly less than the number of features, i.e. $q << p$. We tested plan to test our implementation of random lasso on simulated genetic data provided by Doctor Kim. In this simulated data the samples are patients and the features are genes. The amount of samples and features vary, but max out at 1,000 samples and 100,000 features.

Data Preprocessing:

Some level of data preprocessing is often required. Data needs to be in matrix form with samples as rows and the features as columns. To reiterate, the amount of features should be significantly larger than the amount of samples, else simpler regression analysis methods will suffice with greater speed. Data should also be normalized with a **studentized residual**, similar to z score, as opposed to a min-max calling between 0 and 1. A studentized residual avoids a min-max assumption, which suits our needs since the subset of data we are given doesn't necessarily contain the min or max for the whole population.

C-Day Proposal

Research Question or Motivation:

We were motivated by lack of packages and implementation guidance on the state-of-the-art method, i.e. random lasso, for research in fields with high-feature low-sample datasets such as bioinformatics random lasso. We believe the public should have access to an intuitive and malleable package for research. One of the drawbacks of random lasso is its runtime, and we mitigated this drawback by implementing parallel programing on the highly independent processes of random lasso.

Materials and Methods:

<u>used the above section on methods</u>.

Preliminary Results:

Early results using elementary parallel programing options and two cores showed a reduction in runtime by nearly a half. This is because random lasso has highly independent processes. Four cores should reduce the runtime by three-fourths.

Intellectual or business merit of your project?

There exists no package for regression analysis using random lasso. Random lasso is the state-of-the-art method of regression analysis for multidimensional datasets with comparatively high features and low samples. Due to the algorithmic complexity of Random Lasso, researchers often opt for more juvenile methods of regression analysis that are less accurate, but pre-made in a package. We are releasing a public random lasso package for both R and Python, so that this algorithm becomes a more publicly accessible tool for research.

Random lasso runs lasso multiple times on multiple bootstraps of a dataset. Because of this, some researchers may shy away from using random lasso, given its time consuming and computationally burdensome nature. We alleviated some of this deterrent by adding parallel computing using Spark to the highly independent processes of the random lasso algorithm.

Actions you will take to enhance the potential of your project to benefit society?

As mentioned before, this algorithm will be publicly accessible on R and Python as a package. On top of this we are building a website that gives users a guide on how to use the package. This website will also show users on how to preprocess data for random lasso, and give general educational material on the random lasso and its predecessors.