Genetic Analysis via Iterative Hard-Thresholding

Collab with Open Mendel using the Julia programming language

Benjamin Chu & Kevin Keys & Janet Sinsheimer & Kenneth Lange

Department of Biomathematics

Contacts: biona001@ucla.edu, kevin.keys@ucsf.edu, JanetS@mednet.ucla.edu, klange@ucla.edu

Repository: https://github.com/klkeys/IHT.jl

Background - Some Problems in Statistical Genetics

At the DNA level, modern humans are 99.9% identical. The remaining 0.1% of genetic variations drive many trait and disease differences observed in humans. Individual variations in the DNA sequence are termed **single nucleotide polymorphisms** (SNPs) and biologists can identify them via **Genome Wide Association Studies** (GWAS). These studies aim to answer one main question:

Q: Which genetic variants are associated with a trait?

A few Obstacles...

- GWAS datasets are often **super big** (100+ GB). **Q: How to analyze them efficiently?**
- SNPs are often **rare** with **small effect size**. **Q: How to separate *weak signals* from *noise*?**
- Genetic data processing is **unnecessarily complicated**. **Q: Can we not have a centralized platform providing a streamlined pipeline for genetic analysis?**

Features of IHT.jl

1. **Integration with Open Mendel:** Prepare 1 input file to not only run IHT, but also 11 more genetics analyses, such as simulation, fitting variance component model, linkage analysis...etc
2. **Doubly sparse group projection:** Optionally estimates sparse β with at most J active groups and k active predictors per group.
3. **Scale SNPs using prior weights:** User can optionally scale each SNP's weight based on known information (frequency, data quality, candidate gene/pathway approach...etc)
4. **GPU acceleration(*):** Greatly improves computational performance (only for numeric data).
5. **Cross Validation(*):** Automatically determines optimal number of features

(*) = *only available when interfacing with PLINK.jl* = *future work*

Key Performance Optimizations

- **Computation on raw genotypes.** IHT.jl interfaces with SnpArrays.jl to perform the following linear algebra directly on raw genotype data: `A_mul_B!`, `At_mul_B!`, `A_mul_Bt!`, `At_mul_Bt!`, `Ac_mul_B!`, `A_mul_Bc!`, `Ac_mul_Bc!`. For suitably sized matrices, these routines are faster than analogous BLAS routines because they avoid file swapping with the hard disk (see Figure 2).
- **Standardization on the fly.** We can precompute mean minor allele count vector $(\mathbf{u})$ and reciprocal normalizing constant vector $(\mathbf{v})$. Then perform matrix/vector computations directly as:

$$\mathbf{X}_{standardized} = \left(\mathbf{X}_{raw} - \mathbf{1}_n \mathbf{u}^T\right) diag(\mathbf{v})$$

- **Streaming only necessary columns of genotypes.** At each iteration, the estimate of β is sparse with none-zero entries denoted by β_k, and corresponding columns of $\mathbf{X}$ denoted by $\mathbf{X}_k$. Multiplying the 0 entries of β with dense $\mathbf{X}$ is redundant. Therefore, we have:

$$\nabla f(\beta) = -\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = -\mathbf{X}^T(\mathbf{y} - \mathbf{X}_k\beta_k)$$

- **Multi-threaded geno-matrix computation (future work)** To speed up the gradient computation, we can make our matrix/vector multiplication multi-threaded. That will be done in the near future.

(Group) Iterative Hard-Thresholding

Iterative hard-thresholding (IHT) is one of the most scalable algorithms for *feature selection.* It has great convergence guarantees, and empirically performs better than LASSO and MCP for genetic and numeric data [Keys 2017] in terms of model selection. Below we outline the algorithm and discuss our theoretical contributions.

Mathematical Section

For continuous phenotype $\mathbf{Y}$, iterative hard-thresholding minimizes the residual sum of squares subject to different (non-convex) l_0 "norm" constraints:

$$f(\beta) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\beta||_2^2, \quad \text{subject to} \begin{cases} \text{Regular IHT:} & ||\beta||_0 \leq k \\ \text{Group IHT:} & ||G_i(\beta)||_0 \leq k, \quad i \in \{1, ..., J\} \end{cases} \tag{1}$$

The sparsity constraints k, J and group memberships G are assumed to be known, but in practice they have to be determined via cross-validation. Note that when $J = 1$, group IHT reduces to regular IHT. Update to β is accomplished *iteratively* by first taking the negative gradient step, then project so at most J active groups and k active predictors per group remains:

$$\beta^+ = P_{S_{J,k}}\left[\beta - \mu\nabla f(\beta)\right], \quad \mu_m = \frac{||\beta^m||_2^2}{||X\beta^m||_2^2} \tag{2}$$

The operator $P_{S_{J,k}}$ finds the J largest group in l_2 magnitude first, leaves the k largest entry of each group alone, and sets everything else to 0. For normal IHT, convergence is guaranteed if $\mu_m \leq \omega_m$ where:

$$\omega_m \leq (1 - c)\frac{||\beta^{m+1} - \beta^m||_2^2}{||X(\beta^{m+1} - \beta^m)||_2^2} \tag{3}$$

for some constant $0 < c \ll 1$.

Algorithm 1: (Group) Iterative hard-thresholding

Input : genotype matrix $\mathbf{X}$, response vector $\mathbf{y}$, membership indicator vector $\mathbf{G}$, scalars J, k

Output: β with at most J active groups and k active predictors per group

Initialize: $\beta \equiv \mathbf{0}$.

while *not converged* **do**

  Gradient step: $\tilde{\beta} = \beta^m - \mu\nabla f(\beta)$

  Projection: $\beta^{m+1} = P_{S_{J,k}}(\tilde{\beta})$

  (Optional) Debiasing: $\beta^{m+1} = \arg\min f(\beta^{m+1})$ such that $supp(\beta^{m+1})$ satisfies constraints described in (1)

end

Results

- IHT recovers predictors that passes the Bonferroni correction under the scheme of univariate analysis, but also proposes additional predictors that does not pass the cutoff.
- Predictors **might not** be picked even though it has a higher p-value.
- On a dataset with 2200 people and 10000 SNPs (roughly 200MB uncompressed), IHT with J = 1 and k = 10 converges in 1.511 seconds and uses 10.5 MB.

Future goals: Enable analysis on half a million samples, whose data size is roughly between 30GB and 1TB. Afterwards, we will focus on proving convergence guarantees for doubly sparse projection and enabling logistic IHT.

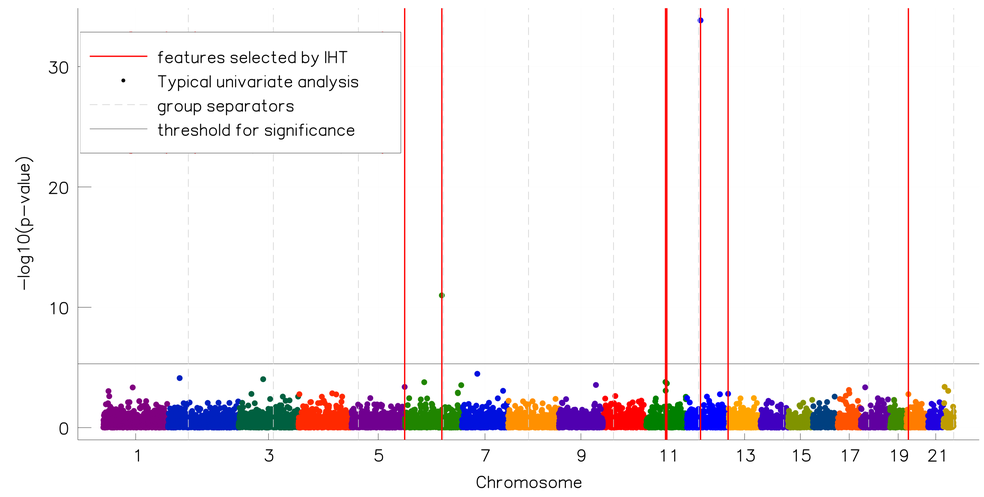

Figure 1: IHT (J=4, k=2) Superimposed on Traditional p-value Graph

Table 2 lists a few milestones of our package tested on a 200 MB (uncompressed) dataset:

Date	Speed	Memory	Key improvement
June 24	1.896 s	573.3 MB	Prototype code; converts genotype data to float64 matrix
July 4	2.579 s	1.230 GB	removed StatsBase dependency, added extra functions
July 18	2.207 s	244.1 MB	Enabled matrix subsetting, compute on raw data
July 20	1.511 s	10.50 MB	Wrote efficient std computation

Figure 2 below compares matrix-vector multiplication speed of BLAS and SnpArrays. For big enough matrices, SnpArrays is faster:

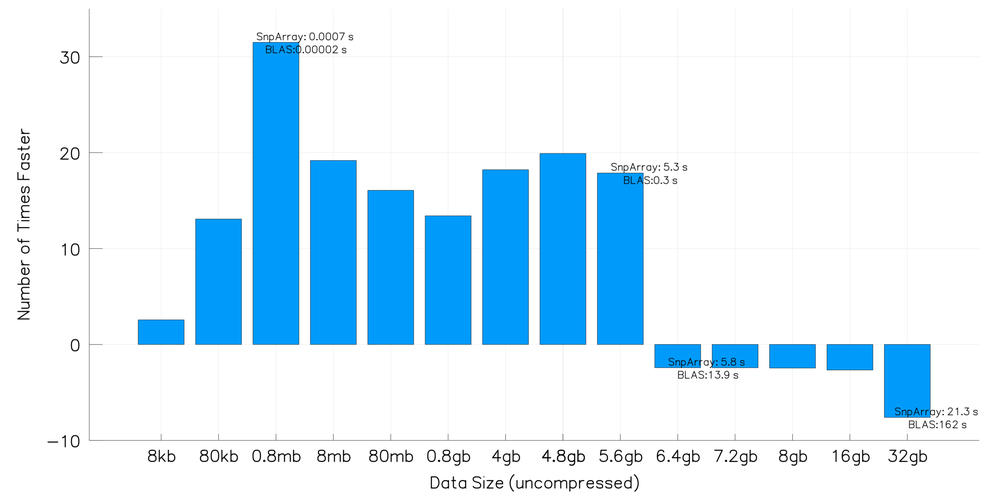

Figure 2: Linear Algebra in SnpArrays is faster than BLAS for Large Matrices

Acknowledgements

- This research was supported by NIH Training Grant in Genomic Analysis and Interpretation T32HG002536
- This research was supported by Google Summer of Code 2018 with NumFOCUS.
- This trip was sponsored by Julia Computing.
- Special thanks to Kevin Keys for being the best mentor!