DETECTING EPISTASIS EFFECT IN GENOME-WIDE ASSOCIATION STUDIES BASED ON PERMUTATION TESTS AND ENSEMBLE APPROACHES

by

BENJAMIN PHILIP HORSTMAN

Submitted in partial fulfillment of the requirements For the degree of Master of Science

Thesis Advisor: Dr. Jing Li

Department of Electrical Engineering and Computer Science
CASE WESTERN RESERVE UNIVERSITY

May, 2010
CASE WESTERN RESERVE UNIVERSITY
SCHOOL OF GRADUATE STUDIES

We hereby approve the thesis/dissertation of

Benjamin Philip Horstman

candidate for the
Master of Science

(degree *)

(signed) Jing Li
(chair of the committee)

Robert C. Elston

Andy Podgurski

(date) 3-31-10

*We also certify that written approval has been obtained for any proprietary material contained therein.
**Table of Contents**

Table of Tables .......................................................................................................................... v

Table of Figures .......................................................................................................................... vi

Acknowledgements ..................................................................................................................... viii

Glossary ......................................................................................................................................... ix

1. Introduction ............................................................................................................................. 2

   1.1. Problem Statement .............................................................................................................. 2

   1.2. Three Fundamental Difficulties in GWAS ........................................................................ 2

2. Literature Review ..................................................................................................................... 3

   2.1. Problems with Classical Feature Selection Techniques .................................................... 4

   2.2. Classical Statistics and Brute Force .................................................................................... 5

   2.3. Machine Learning Methods ............................................................................................... 8

   2.4. Statistical Estimation and Inference .................................................................................. 10

3. Methods ................................................................................................................................... 12

   3.1. Permutation Tests ............................................................................................................... 12

   3.2. Decision Trees ................................................................................................................... 13

      3.2.1. Gini Impurity .............................................................................................................. 15

   3.3. Adaboost ........................................................................................................................... 16

      3.3.1. Algorithm Adaboost.M1 (courtesy\textsuperscript{21}) ..................................................... 17
3.4. Feature Importance Estimation .................................................................................. 18
  3.4.1. Feature Importance Calculation .......................................................................... 19
3.5. Implementation Details ............................................................................................ 20
4. Simulation Disease Models .......................................................................................... 22
  4.1. Two-Locus Models ................................................................................................. 22
  4.2. Three-Locus Models .............................................................................................. 24
  4.3. A Higher Order Interaction Model .......................................................................... 26
  4.4. Methods of Parameter Setting ................................................................................ 26
    4.4.1. Parameter Setting Using Marginal Effect Size $\lambda$ ........................................ 27
    4.4.2. Drawbacks to parameter setting using $\lambda$ ...................................................... 30
    4.4.3. Parameter Setting using $\chi^2$ ........................................................................... 31
  4.5. Construction of Simulated Models .......................................................................... 33
5. Simulation Results ......................................................................................................... 35
  5.1. Two-locus Models .................................................................................................. 35
  5.2. Three-locus Models .............................................................................................. 45
  5.3. Higher-order Interaction Model .............................................................................. 49
  5.4. Computational Resources ...................................................................................... 50
6. Application to WTCCC Affymetrix SNP data ............................................................... 51
  6.1. Bipolar Disease ...................................................................................................... 52
6.2. Coronary Artery Disease ................................................................. 53
6.3. Chron's Disease ........................................................................... 54
6.4. Hypertension ............................................................................... 54
6.5. Rheumatoid Arthritis ................................................................. 55
6.6. Type-I Diabetes ........................................................................... 55
6.7. Type-II Diabetes ......................................................................... 56
7. Conclusion and Future Work ............................................................ 57
7.1. Reduced power at low MAF ....................................................... 59
7.2. High Variance Partitioning (HVP) ............................................... 60
7.2.1. The HVP Problem Statement ................................................. 60
7.2.2. How HVP algorithms transform the marginal effect ............... 63
7.2.3. Developing a robust HVP algorithm ....................................... 64
7.3. Alternative Scoring Functions For Adaboost ............................... 66
7.4. Adaboost Selection in two-stage algorithms ............................... 67
7.5. Including Expert Knowledge ..................................................... 68
Bibliography ....................................................................................... 70
Table of Tables

Table 1: Two-locus Additive Model ................................................................. 23
Table 2: Two-locus Threshold Model .............................................................. 23
Table 3: Two-locus Epistasis Model ............................................................... 23
Table 4: Three-locus Additive Model ............................................................ 24
Table 5: Three-Locus Threshold Model ......................................................... 24
Table 6: Three-locus Epistasis (TSym) Model ................................................ 25
Table 7: Four-locus Model consisting of two independent subgroups, one like Table 2 and one like Table 3, with additive effect when they act jointly .................................................. 26
Table 8: Expected counts of various genotypes for three-locus threshold model, assuming 50/50 N=4000 case-control study .................................................................................. 32
Table 9: Bipolar Disease top 15 SNPs .............................................................. 53
Table 10: Coronary Artery Disease top 15 SNPs .............................................. 53
Table 11: Chron's Disease top 15 SNPs ........................................................... 54
Table 12: Hypertension top 15 SNPs ............................................................... 55
Table 13: Rheumatoid Arthritis top 15 SNPs ................................................... 55
Table 14: Type-I Diabetes top 15 SNPs ........................................................... 56
Table 15: Type-II Diabetes top 15 SNPs .......................................................... 56
Table 16: A HVP set of partitions for N=8 that has 7 partitions into two all of N/2 average pair-wise distance ......................................................................................... 65
Table 17: Greedy HVP approach vs. number of partitions at optimal pair-wise average distance ........................................................................................................... 65
Table of Figures

Figure 1: Decision Tree example with SNPs, individual count and gini impurities. The gini gain of the split at SNP1 can be calculated as (0.5 - 0.1*0.25 - 0.3 * 0.33 - 0.6 * .16) = .28, assuming the input data has an even number of both classes (see section 3.4.1) ................................................. 14

Figure 2: TSym Model geometric representation. Each axis is interpreted as one SNP; red dots represent relative increase in disease risk................................................................. 25

Figure 3: Parameters vs. effect size for three-locus additive model. λ+1 is the vertical axis.

Figure courtesy Yixuan Chen........................................................................................................ 28

Figure 4: Parameters vs. effect size for three-locus threshold model. λ+1 is the vertical axis.

Figure courtesy Yixuan Chen........................................................................................................ 29

Figure 5: Parameters vs. effect size for two-locus epistasis model. λ+1 is the vertical axis.

Figure courtesy Yixuan Chen........................................................................................................ 30

Figure 6: Excerpt from Zhang and Liu 18, showing power vs. MAF (.05, .2, .3, .5) for BEAM, Stepwise B-statistic, stepwise logistic regression, and Pearson’s chi-square test. This model is analogous to our two-locus additive model, though simulation details differ .................... 31

Figure 7: Two-Locus Additive Model with theta=0.2................................................................. 36

Figure 8: Two-Locus Additive Model with theta=0.35............................................................. 37

Figure 9: Two-Locus Additive Model with theta=0.5............................................................... 38

Figure 10: Two-Locus Threshold Model with theta=0.4........................................................... 39

Figure 11: Two-Locus Threshold Model with theta=0.5............................................................ 40

Figure 12: Two-Locus Threshold Model with theta=0.6............................................................ 40

Figure 13: Two-Locus Threshold Model with MAF=0.1 .......................................................... 41
Figure 14: Two-Locus Epistasis Model with theta=0.3 ................................................................. 42
Figure 15: Two-Locus Epistasis Model with theta=0.35 ............................................................... 43
Figure 16: Two-Locus Epistasis Model with theta=0.4 ................................................................. 43
Figure 17: Two-Locus Epistasis Model with MAF=0.4,0.5 ............................................................. 44
Figure 18: Three-locus Threshold Model with theta=0.5 ............................................................. 45
Figure 19: Three-locus Threshold Model with theta=1.0 ............................................................. 46
Figure 20: Three-locus Threshold Model with theta=1.5 ............................................................. 46
Figure 21: Three-Locus TSym Model with theta=1................................................................. 47
Figure 22: Three-Locus TSym Model with theta=2................................................................. 48
Figure 23: Three-Locus TSym Model with theta=3................................................................. 48
Figure 24: Average Pearson's $\chi^2$ test value for the four loci in the higher-order interaction model ................................................................. 49
Figure 25: Higher-order Interaction Model with theta=0.8......................................................... 50
Acknowledgements

I would like to thank my friends in the Case Western Reserve University Computational Biology Club, especially my advisor Dr. Jing Li, who came up with the original idea for this thesis topic and was instrumental in teaching me the required concepts to achieve its completion, and my partner Yixuan Chen who greatly assisted me in selecting and creating disease models for algorithmic validation. Additionally, I would like to acknowledge support from NIH/NLM grant LM008991, NIH/NCRR grant RR03655, NSF grant CRI0551603, the Ohio Board of Regents Choose Ohio First Scholarship and Tomas & Evelyn James Fellowship, and an allocation of computing time from the Ohio Supercomputer Center.
Glossary

- attribute - see locus.
- epistasis - In the context of disease models, epistasis is broadly defined as a deviation from additivity; for example see Tang et al\(^1\). Typically it is used to describe situations with heavy epistasis, that is, with many heavily interacting components.
- feature - see locus.
- genetic marker - A piece of genetic information from an individual’s genome. Often a SNP. see also locus.
- GWAS - Genome-Wide Association Study. Uses naive list of markers (e.g., SNP) across entire genome in a case-control setting to detect association between markers and class labels (e.g. disease status).
- Linkage Disequilibrium (LD) - The tendency for SNPs with close genetic positions to be correlated because of the small likelihood of recombination between them.
- locus - The smallest unit of feature data (aka feature, attribute). In our context usually a genetic marker such as a SNP.
- marginal effect - In the context of disease models, the portion of disease risk that is explained by a single locus acting alone.
- minor allele frequency (MAF) - The minor allele is the nucleotide in a SNP that is less common in the population. MAF then refers to the frequency of this allele.
- odds ratio - A method of representing the effect size, with 1 being none and higher numbers representing higher odds of effect. See also penetrance.
• penetrance - The probability that a genotype will cause disease effect, from 0 to 1. See also odds ratio.

• population prevalence - The fraction of the population affected by a condition. For our disease models, this is controlled by the baseline η, the effect size θ, and the MAF.

• SNP - Single-Nucleotide Polymorphism. A SNP represents a single base-pair difference in a genome. Since humans are diploid, each human SNP is written with two characters (e.g. AA, AC, CC).

• WTCCC - Welcome Trust Case-Control Consortium\textsuperscript{2}. Created a publicly available dataset that is the benchmark for large human GWAS data.
Detecting Epistasis Effect in Genome-Wide Association Studies
Based on Permutation Tests and Ensemble Approaches

Abstract

by

BENJAMIN PHILIP HORSTMANN

Most complex disease involve multiple genes, their interactions, environmental factors, and gene-environment interactions. Although genome-wide association studies (GWAS) have shown some success for identifying genetic variants underlying complex diseases, most existing studies are based on limited single-locus approaches, which detect SNPs essentially based on their marginal association with the phenotype. In this paper, we propose an ensemble approach based on Boosting to study gene-gene interactions. We extend the basic Adaboost algorithm by incorporating an intuitive importance score based on Gini impurity to select candidate SNPs. Permutation tests are used to control the statistical significance. We perform extensive simulation studies across five models and at realistic GWAS sizes to evaluate the efficacy of our approach and compare it to an existing epistasis algorithm called BEAM. Our results indicate that our approach is valid, efficient for GWAS, and on disease models with epistasis has more power than $\chi^2$ genotypic test and BEAM.
1. Introduction

1.1. Problem Statement

In the simplest form, our problem statement can be summarized as follows.

Given:

- \( N \) individual phenotypes, \( Y_1 \ldots Y_N \), which are binary and always known
- \( L \) markers per individual \( i, X_{i,1} \ldots X_{i,L} \), which are categorical (usually two-or-three-valued, e.g. 0,1,2 with an additional option 3 for unknown)

An algorithm must determine which markers, if any, are associated with the phenotype.

Additional information might be included such as the genomic distance between the \( L \) markers, the actual base pairs of a marker instead of numeric values (AA, AT, TT versus 0, 1, 2), prior probabilities for association, etc. For simplicity, we will focus on the simplest case, despite the fact that additional information can add additional power.

Other variations on this problem statement exist, and I briefly discuss one in section 7.4.

1.2. Three Fundamental Difficulties in GWAS

GWAS is computationally interesting because of three fundamental difficulties that preclude the use of classical algorithms. The first is effectively the Curse of Dimensionality, except worse. The curse of dimensionality recognizes that the search space of the problem grows exponentially with the number of dimensions or features.
In addition to using up to one million SNPs, GWAS studies also have limited data points due to genotyping costs. Thus, in addition to the Curse of Dimensionality, we also have far fewer examples than features (in WTCCC\textsuperscript{2}, approximately 5000 individuals and 500,000 SNPs). Second, since complex diseases are by definition caused by multiple interacting vectors, the exponential search space above is not necessarily navigable by heuristic algorithms such as a greedy search. Finally, we do not always know what SNPs do functionally, if anything, or which SNPs interact. It is also difficult to compute this computationally, due to the large number of SNPs. Thus, it is usually impossible to perform useful feature space transformations. Since we cannot easily determine all of the interacting SNPs, feature space reduction is dangerous because it could remove SNPs with higher order correlation to the disease or reduce the signal of their effect.

2. Literature Review

While discussing my literature review, I will structure it as follows. First, I will briefly discuss classical machine learning feature extraction techniques and why they are not sufficient for use in GWAS. Then, I will discuss proposed methods by grouping them by school of thought into three groups: classical statistics and brute force, machine learning, and statistical estimation and inference. These groups are somewhat subjective and usually derived at least partially from the background of the algorithm creators. Sometimes statistical writers will split the last two groups along different lines, into greedy search and stochastic search. This distinction is misleading, since most algorithms have features of both (consider greedy tree building and stochastic
bootstrapping from Random Forests in section 2.3 or greedy path finding and stochastic sampling of paths from SNPHarvester in section 2.4). Therefore I do not follow their treatment.

2.1. Problems with Classical Feature Selection Techniques

Since the fundamental difficulties of GWAS were outlined in section 1.2, it is easy to see why classical feature selection techniques have difficulty. Usually they are not designed for situations where there are one million features and only a few thousand data points. This typically manifests as overfitting\(^3\) if the implementation is able to handle the dataset size at all (\(1,000,000 \times 5000 = 4.7\) Gigabytes, assuming a conservative 1 byte per SNP). Then, the unknown complex interaction means that linear methods, such as Principal Components Analysis (PCA)\(^4\), are not usable. Thus, complex new methods must be attempted, such as Kernel PCA\(^5\). Unfortunately, these methods still have their flaws. Kernel PCA uses the kernel trick to map features into a higher dimensional space; since the space is already one million SNPs this creates obvious problems. For example, Kernel PCA requires \(O(L^2)\) memory and \(O(L^3)\) running time, where \(L\) is the number of SNPs\(^5\).

To summarize, classical feature selection techniques usually either expect fully linear (PCA), fully pair-wise (KPCA, others), or fully interacting (MDR, see section 2.2) levels of data interaction. Additionally, they are often intended for floating point features (KCPA, others) and cannot take advantage of the categorical nature of SNPs, which reduces power and increases running time and memory usage.
2.2. Classical Statistics and Brute Force

Despite the problems noted in section 1.2, classical statistics such as Pearson's single-locus chi-square test are by far the most common form of GWAS analysis used in leading papers\(^2\). There are several reasons for this. First, these tests are easy to understand as they have extensive statistical basis. Second, they are fast, easily implemented, and have widespread mature implementations that can handle GWAS scale data\(^6\) which means they have a low barrier to entry. Finally, they are often used as an initial test in Biology or Genetics focused papers, leaving further computationally intensive study to others.

Pearson's chi-squared test is defined in equation 1; it uses a chi-square distribution to test whether the observed distribution of genotypes is consistent with the null hypothesis of no disease association:

\[
\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}
\]

Other single-locus tests are sometimes used, but for brevity Pearson's chi-square will be assumed due to its use in\(^2 7\). Worse than assuming a single-locus test, Pearson's chi-square test is usually evaluated with Bonferroni Correction for multiple testing. This conservative procedure limits the power of the test to detect even single-locus effects.

In addition to the single-locus version, it is possible to use Pearson's test to include two-way interactions with a two-locus chi-square test. This is feasible when the number of SNPs is small, but since the number of two-way SNP interactions is quadratic in the
number of SNPs, it can become computationally expensive on the most interesting, large scale datasets. Conservative Bonferroni Correction must also be quadratic, and permutation tests (see section 3.1) are not usually a viable option due to the already large search space. Consider a dataset with 1,000,000 SNPs, 5000 individuals, and 1000 permutations. Then the 2-locus permutation search space is $5 \times 10^{18}$.

Computational Methods to improve the efficiency of two-locus tests is one avenue of research; an example is Wei Wang et. al.’s COE algorithm. This algorithm shows that common statistics such as Pearson’s chi-square test are convex functions and then uses the data relations between tests on that dataset to prune most SNP pairs while still giving globally optimal results. Thus, it allows the use of common multi-locus statistics with well understood properties yet less computational cost than the naive approach. This approach can reduce the computational complexity of when permutation tests are used by at least two orders of magnitude. Unfortunately, how the algorithm performance compares to Bonferroni Corrected two-locus tests is unclear in their presentation of the algorithm and could possibly be worse. I make this guess since COE is still quadratic with respect to the number of SNPs, $O(NL^2)$, where N is number of individuals and L is number of loci, and likely has higher constant overhead than a simpler genome-wide two-locus scan. Thus it is still computationally prohibitive for most GWAS.

One other machine learning algorithm that uses brute force to detect interactions and perform dimensionality reduction is Multifactor Dimensionality Reduction (MDR). MDR
exhaustively considers full interaction between all features, and therefore can reduce the feature space without fear of losing important associations. It does this by selecting a set of k loci, representing all k-way relations with a k-locus contingency table, and then marking entries as high risk if those cells have a sufficiently perturbed case-control ratio.

Of course, this power comes with a cost. Considering all interactions is hugely computationally expensive, so MDR is commonly used with 9-500 SNPs. Thus, MDR is not directly usable for GWAS and is more intended for analysis of SNP subsets after GWAS. Still, the approach has somewhat widespread use, and so merits mention here.

Finally, no discussion of the use of classical statistics in learning would be complete without logistic regression. Logistic regression is a generalized linear model, which means it uses feature space transformations to represent nonlinear interactions. In GWAS, this usually means the addition of terms for 2-way, or k-way SNP interaction. Effectively, the log odds or logit of the class label is modeled as a linear combination of transformed features: $\text{logit}(y) = \beta_0 + \beta_1 x_1 + \cdots + \beta_T x_T$. The parameters $\beta$, which function as weights, are determined using maximum likelihood estimation.

The main problem here is that the number of k-way interactions is exponential in k. Thus, the running time for single and two-way interactions becomes $O(NL^2)$, where N is the number of individuals and L is the number of SNPs.

To combat this, logistic regression is usually used in a form called stepwise logistic regression\(^9\). First, each feature is tested with a quick single locus test for marginal association with diseases. Then, the top p percent of markers are chosen and all k-way
(k=2,3) associations are tested. This reduces the computational burden by a couple of orders of magnitude, but can miss markers with small marginal effect. Despite this drawback, stepwise logistic regression is still one of the more popular and promising classical techniques due to its well understood nature and comparatively higher ability to detect more complex effects in GWAS.

2.3. Machine Learning Methods

This section describes usages of machine learning methods, such as tree-based methods and support vector machines (SVM), to tackle GWAS.

Random Forests, created by Breiman (who is also the author of CART, the quintessential book on decision tree learning), are a modern classifier with many desirable properties. For example, the algorithm determines feature importance, detects feature interactions, can handle large feature spaces, and does not normally require special handling to reduce overfitting. Thus, it has been proposed for use in GWAS. Random Forests are an ensemble system; they use many weak learners, specifically modified decision trees (see section 3.2 which discusses our implementation of decision trees) grown in special ways, together to formulate a robust picture of the data. Random Forests works by first taking M bootstrap samples. Then, each bootstrap sample is used to grow a modified decision tree. Random Forests uses feature space partitioning to introduce increased heterogeneity into the decision trees. This means that each node in the decision tree only considers m randomly chosen features. Since literature suggests $m \approx \sqrt{L}$, where L is the number of features, this means that the
decision trees are much faster to grow than normal. Thus, the number of trees M and the number of features in a partition m control the running time.

However, there are drawbacks to Random Forests. For one example, the initial selection of a feature requires some marginal effect. In the absence of marginal effect, Random Forests must rely upon the bootstrap resampling and feature space partitioning to initially find disease loci. While the feature space partitioning reduces complexity, it seems especially disingenuous when there is gene-gene interaction. Assuming an epistasis disease model like the one in Table 3 with low marginal effect, the only time Random Forests can make an informed decision is when both risk loci are in the same tree. Since there is little marginal effect, the possibility approaches that of random chance, which is approximately $\sqrt{1/L}$, where L is the number of features. Thus, much of the work Random Forests does is spurious, since both necessary disease loci cannot be present. A final drawback is that while Random Forests attempts to estimate variable importance, these estimates are only statistically valid if the trees in the forest are sufficiently independent. Due to the interrelated nature of GWAS data, we have found this to violate the expected controlling of type-I error extensively. Thus, alternative approaches for significance must be proposed.

Our own approach described in section 3 was primarily motivated by solving the problems with Random Forests by using more applicable machine learning machinery. Another machine learning algorithm that has been extremely popular recently is Support Vector Machines (SVM). Thus, SVM have also been proposed as a potential
GWAS algorithm\textsuperscript{17}. SVM operate by attempting to find a hyperplane in the feature space that optimally separates the individuals. Optimal in this sense means that the Euclidean distance from any individual to the hyperplane is large (maximized in the ideal case). Since real data is often not linearly separable, SVM usually use kernel functions to map the feature space into a higher dimensional space where a linear separation can be made. Since real data is often noisy, finding a good decision plane is an inexact process and creates its own selection problem. Additionally, the power of SVM in typical GWAS situations with likely disease models has not been extensively tested in simulation, making currently published work less validated.

2.4. Statistical Estimation and Inference

Since modern machine learning is basically multivariate statistics, it makes sense that statistically oriented algorithms would also be proposed as solutions to the same problems. These algorithms typically use approaches such as Markov Chain Monte Carlo (MCMC) to infer an indicator variable that represents the loci that are in epistasis.

BEAM\textsuperscript{18} is probably the seminal algorithm of this type and later algorithms in this category largely adapt it. BEAM uses MCMC sampling to attempt to infer whether each locus is a disease locus, a jointly affecting disease locus, or a background (uncorrelated) locus. This begins by assigning each locus to each group according to a prior distribution (perhaps uniform, but could include prior knowledge). Then many rounds of MCMC are run using the Metropolis-Hastings algorithm. This roughly amounts to randomly moving loci between groups. Thus, the number of MCMC rounds is the primary parameter that
mediates runtime. After the MCMC they use a special statistic, called the B-Statistic, to infer statistical significance from the hits sampled in MCMC. This approach avoids computing all interactions, but can still theoretically find high order interactions significant.

BEAM has two main unaddressed problems. The first is that the suggested number of MCMC rounds, m, is $O(L^2)$ where L is the number of SNPs. This either requires BEAM to take a very long time, or have reduced power to sample a good indicator. Thus, BEAM has limited ability to scale to large datasets, such as those created by the WTCCC. The second unaddressed problem is that BEAM and the B-statistic effectively assume Bonferroni Correction for multiple testing. Thus, the algorithm fails to utilize the non-randomness of SNP data and makes conservative estimates of significance.

Despite its drawbacks, BEAM is the major baseline we compare our approach to in section 5 due to its notoriety and vastly different approach.

epiMODE\textsuperscript{1} is a small improvement to BEAM that accounts for multiple interaction subgroups and linkage disequilibrium between markers. This should increase the power (assuming the underlying disease has multiple interaction groups and the assumptions about loci actually being in LD are valid), but it comes at a cost. epiMODE must use Reversible Jump MCMC (RJ-MCMC) to infer these additional model parameters, increasing the complexity of an already expensive process.

SNPHarvester\textsuperscript{16} is another algorithm that is similar to BEAM but takes a different algorithmic approach to determining marker partitioning. It is listed here due to its
similarity to BEAM and the fact that it is labeled as a stochastic search algorithm like the others in this section. It uses a deterministic algorithm called PathSeeker to make a path of M k-SNP groups by swapping SNPs in and out of the active set if the swap improves the score (each group formed along the way that passes a threshold is returned). SNPHarvester then harvests these groups for different k. This alone does not evaluate statistical significance; it requires another step such as logistic regression or use of the B-Statistic like in BEAM.

3. Methods

In this section I will describe all of the components of our approach to GWAS. I will begin by describing permutation tests for controlling type-I error, a general technique that can be used to achieve tighter significance bounds for any statistic that can formulate a numerical importance value for a feature (this includes common tests like Pearson's chi-square). Then, I will describe our algorithm in three parts. First, I will describe decision trees and how we use them as our weak learner. Second, I will describe the Adaboost algorithm we use to create an ensemble system. Finally, I will describe how to calculate the statistic for variable importance estimation.

3.1. Permutation Tests

A permutation test is a general technique for controlling type-I error that is broadly applicable. Our formulation exactly follows that of Zhang et al. First, create M permutations of the class labels (without replacement) and label these \( \{Y_1, \ldots, Y_M\} \). For permutation \( Y_m \), let \( T_{Y_m} \) be the maximum test statistic among all possibilities in that
permutation. Then, the distribution of $T_{Y_m}$ will be used as the null distribution of the statistic. So, to determine a statistic threshold for a controlled false positive rate, say $\alpha = 0.05$, take the $\alpha * M$ th largest value from the list of $T_{Y_m}$.

The statistical advantage to the permutation test is the tighter control for type-I error and multiple testing. For example, in our simulation studies detailed in section 4, we use a simulated dataset derived from the WTCCC data$^2$ with $N=4000$ individuals and $L=40220$ SNPs. Using a $\alpha = 0.05$ significance level and Bonferroni Correction for multiple testing, this amounts to a Pearson's chi-square test threshold of 27.2. This is higher than the permutation derived threshold of 25.8. The permutation test allows the similarity in data to be used to set a lower threshold while still controlling the false positive rate at the desired level.

The major downside to permutation tests is that the technique greatly increases the amount of computation that must be done. While a threshold value derived by Bonferroni Correction can be computed trivially, the permutation test modifies the running time of the underlying algorithm adding a factor linear in the number of permutations. For simple statistics such as single-locus Pearson's chi-square test, this is not terribly limiting, but for two-locus or higher tests, it is prohibitively computationally expensive.

### 3.2. Decision Trees

Decision trees are a very simple algorithm that can be used for supervised learning. While they are naive and are flawed on many types of data, they are especially useful as
smaller components of larger systems\textsuperscript{19}. This section will describe how we implement decision trees, which will be used as a component in our whole algorithm. Additionally, the decision trees described here are similar to those used in the Random Forests algorithm discussed above.

Decision trees are grown (trained) in a top down manner (see Figure 1). At each node, a feature is selected. Then, based on the values of that feature, the individuals are partitioned into subgroups. Usually decision trees are built with binary splits, where individuals with one value of the feature are placed into one group, and the remainder into the other. Since SNP data is three valued, we extend this to do a ternary split. This means that one split encapsulates all of the information about a SNP, instead of only a fraction like a binary split would. Each subgroup is then handled recursively until some stopping criterion is met. For Random Forests, one continues until class homogeneity is reached. We impose a 5-depth limit on our trees, but this value is arbitrary and changing it seems unimportant.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{decision_tree.png}
\caption{Decision Tree example with SNPs, individual count and gini impurities. The gini gain of the split at SNP1 can be calculated as \((0.5 - 0.1*0.25 - 0.3 * 0.33 - 0.6 * .16) = .28\), assuming the input data has an even number of both classes (see section 3.4.1).}
\end{figure}
Decision trees typically use marginal effect to select features for nodes. This can be any single-locus statistic; Kuncheva defines others\(^\text{19}\). Our approach and Random Forests use the Gini Impurity defined in section 3.2.1.

Despite only using marginal effects to select features, decision trees can still detect some interaction. Because of the recursive partitioning, lower nodes are effectively conditioned on the value of their parents. However, because of the focus on marginal effects during feature selection features with a high level of epistasis might never be selected, and therefore missed.

### 3.2.1. Gini Impurity

Equation 2 defines the calculation of Gini Impurity. There, \( p_i \) is the probability of class \( i \), \( n_i \) is the number of class \( i \) individuals at a node, and \( N_{\text{node}} \) is the total number of individuals at a node. A node that minimizes the Gini impurity only contains individuals of one class.

\[
GI = 1 - p_0^2 - p_1^2 = 1 - \left( \frac{n_0}{N_{\text{node}}} \right)^2 - \left( \frac{n_1}{N_{\text{node}}} \right)^2
\]

(2)

Decision trees use equation 3, a weighted average of impurities, to select the most pure split at every node (thus, the best value is the smallest, e.g. 0). There, \( d.n \) represents the number of individuals at a child node node, \( N \) the number of individuals at the parent, the sum ranges over the children of the node being split, and GI is the Gini impurity of each child.
The Gini impurity statistic itself does not account for missing data. Random Forests handles this by trying to impute what is missing. Another approach that we used for the Adaboost is to modify equation 3 such that individuals who are missing the genotype at that attribute effectively go into all of the child bins. This lowers the impurity of features containing missing values and naturally biases the statistic against such attributes.

3.3. Adaboost

Adaboost\textsuperscript{20} is a popular algorithm among a class of supervised learners called ensemble systems, which also includes Random Forests and Bagging. See Polikar\textsuperscript{21} for an overview of this type of supervised learning. Ensemble systems operate on the principle of the wisdom of crowds. Instead of trying to create a monolithic learner or model of the system, ensemble systems attempt to create many heterogeneous versions of simpler learners, called weak learners. The opinions of these heterogeneous experts are then combined to formulate a complete picture of the data. Boosting is a general technique developed by Schapire\textsuperscript{22} that attempts to decrease the error of a weak learning algorithm using clever resampling of the training data. Adaboost is the most popular Boosting algorithm and I will detail it in section 3.3.1. We use the classical algorithm without modification. A detailed description of Adaboost follows.
The core idea of Adaboost is to draw bootstrap samples to increase the power of a weak learner. This is done by weighting the individuals when drawing the bootstrap sample. When a weak learner instance misclassifies an individual, the weight of that individual is increased (and increased more if the weak learner instance was otherwise accurate). Thus, hard to classify individuals are more likely to be included in future bootstrap samples. In the end, the ensemble votes for class labels weighting the weak learner instances by training set accuracy.

While Adaboost was designed to decrease training set error, some have argued that instead it primarily reduces weak learner variance. This is disputed; the modern consensus is that Boosting and many other approaches can be reformulated in terms of margin theory. This maximum margin theory is also the basis of Support Vector Machines. The goal of this approach is to maximize the distance from the class decision boundary and the training set. This improves generalization over other algorithms that have similar test set error.

3.3.1. Algorithm Adaboost.M1 (courtesy)

Input:
- Sequence of N examples \( S = \{x_i, y_i\}, i = 1 \ldots N \)
- Weak Learning algorithm \texttt{WeakLearn}
- A number of iterations T

Initialize:
\[
D_1(i) = \frac{1}{N}, i = 1 \ldots N
\]
Do for $t = 1, 2, \ldots, T$:

1. Select training subset $S_t$ proportionally from $D_t$
2. Train WeakLearn with $S_t$, receive hypothesis $h_t$
3. Calculate the error of $h_t$:
   \[
   \epsilon_t = \sum_{i: h_t(i) \neq y_i} D_t(i) \quad \text{if } \epsilon_t > \frac{1}{2} \text{ abort}
   \]
4. Set: $\beta_t = \frac{\epsilon}{1-\epsilon_t}$
5. Update distribution: $D_t$:
   \[
   D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \beta_t^{I(h_t(x_i) = y_i)}
   \]
   Where $I()$ is the identity function which is 1 when its argument is true and 0 otherwise, and $Z_t$ is a normalization constant.

Test: given a new instance $x$

1. Total vote for each class is:
   \[
   V_j = \sum_{t=1}^T \beta_t \log \frac{1}{\beta_t} , j = 1, \ldots, C
   \]
2. Chose the class that receives the highest vote.

3.4. Feature Importance Estimation

Adaboost does not naturally have a feature importance score, which probably explains why it has not been used in this context before. In this section I describe our importance score which is adapted from that of Breiman in his book Classification and Regression Trees$^{12}$.

The key observation is that the Gini impurity of a split already encodes information about how "pure" a node is. Thus, by recording the gain in purity (Gini gain), we can see how effective the split is. Then, by weighting by the number of individuals in the node
and the weight of the weak learner instance, we can formulate a general picture of how effective the split was. The hope is that variables that are chosen often and make good splits in effective weak learners are important. Section 3.4.1 details this calculation.

This method of importance calculation has several drawbacks, such as bias when features take on vastly different ranges\textsuperscript{26}. Luckily for us, the simplicity of SNP data (three valued categorical) means that these theoretical weak points are irrelevant; this gives us a simple, efficient, and powerful importance estimate.

### 3.4.1. Feature Importance Calculation

As stated above, the first step is to calculate the Gini gain of the split (relative to the parent node). Equation 4 gives the formula for this, where d.n represents the number of individuals in a node and d.GI represents the Gini impurity of a node's split.

\[
\text{node.gain} = \text{node.GI} - \sum_{d \in \text{node.children}} \frac{d.n}{\text{node.n}} d.GI
\]

Armed with this formula, it is simple to generalize it to include information from our entire ensemble, as shown in equations 5 and 6. While we will use a simple version, tweaking this formula could improve ability to detect epistasis; this will be briefly discussed in section 7.3.

\[
\text{score}[m] = \sum_{\text{tree } \in \text{ensemble}} \text{tree.wt} \sum_{\text{node } \in \text{tree if node.attr = m}} \frac{\text{node.n}}{\text{tree.n}} \text{node.gain}
\]
\[ \text{tree.wt} = \log \left( \frac{1 - e_t}{e_t} \right) \]  

(6)

The main assumptions of this formula are the linear dependence of Gini gain, node weight, and tree weight. These assumptions are made for simplicity rather than optimality. Since Adaboost already weighs each weak learner instance internally (parameter \( \beta \) in section 3.3.1), we adapt that weight for our scoring function. The resulting scores are arbitrarily scaled and depend on algorithm parameters such as tree pruning depth and number of Adaboost iterations; higher is better.

3.5. Implementation Details

We created a custom implementation of our approach using Python, scipy, and C++. The primary drivers behind this decision were the performance of existing machine learning algorithm implementations such as decision trees and the need to adapt them to include the importance score calculation. If a software program is not carefully optimized, it can easily be rendered useless for GWAS. For example, representing a marker in a data type larger than a byte can be disastrous, as memory usage could become unfeasible for standard desktop PCs. Additionally, without access to the internals of the decision tree and Adaboost implementation, it would be impossible to implement our importance estimation calculation. Since there are no off-the-shelf implementations of this calculation, this point is a deal breaker.

We chose the Python language due to its wide availability and ease of rapid prototyping. Thus, the first implementation of Decision Trees and Adaboost was completed very
quickly and runs on multiple machines and on both the CWRU and OSC cluster architectures. Additionally, Python has many useful libraries, such as numpy. Numpy gives Python access to quick multidimensional arrays that we use to efficiently store the SNP data, giving Python Matlab-like functionality. Calculations on numpy arrays also run at C speeds, meaning the calculations are much faster than the Python equivalent. Scipy is a collection of scientific algorithms that allowed us to use off-the-shelf implementations for parts of our algorithm. Furthermore, scipy allows just-in-time compilation of C++ code to be linked with the Python interpreter.

The main performance bottlenecks are the size of the data set and the calculation for the Gini importance statistic. Numpy byte arrays solve the former problem, although a small improvement could be made with a half-byte per SNP implementation, and the later problem is solved by just-in-time compilation of C++ code. We use a heavily optimized C++ implementation of the Gini importance calculation that can operate directly upon the underlying structure of numpy arrays. We also created a GPGPU (General-purpose computing on graphics processing units) implementation using nVidia's CUDA architecture, but the performance increase was less than four-way multithreading the C++ implementation. The rest of our code is straight Python 2.4+. Despite this, the significantly more efficient C++ function for Gini importance calculation still takes over 65% of our CPU time in a short profiling test. Thus, it is still the main bottleneck. This makes sense as it is a memory bound computation accessing megabytes of data; options to further optimize it are limited.
4. Simulation Disease Models

Assessing performance of GWAS algorithms effectively is extremely difficult. Most algorithms deny a theoretic basis for determination of power due to the problem's complexity. Because the true disease vectors for most complex diseases are unknown, real world data is not especially useful for assessing performance. Even presuming an archetypical example existed, algorithm performance on that example is not very useful since it mostly tests whether the algorithm can memorize that dataset, not whether it can generalize well. Thus, broad, robust, blind simulation studies are an integral part of objectively assessing algorithm performance. This section will detail the two, three, and four-locus models we have created (section 4.1 to section 4.3), why we have created them (section 4.1 to section 4.3), and how we have chosen model parameters to intimately examine algorithm performance (section 4.4).

We define our models in terms of the penetrance for each genotype combination and minor allele frequency (MAF), an approach popular in Biology and Genetics. Because of this, theorizing about model properties is non-obvious and we extensively discuss our approach in section 4.4.

4.1. Two-Locus Models

Two-locus models represent the simplest forms of epistasis. They will allow us to get a baseline before we examine more complex and harder to understand models that might more accurately reflect real world complex diseases. In general, 9 parameters are necessary to specify the needed penetrances, since there are 9 two-way genotype
combinations. We simplify this slightly by using only two parameters, η for the baseline (background or genotyping error) effect and θ for regulating the effect size of disease risk genotypes. Our two-locus models are similar to the ones proposed by Marchini et al.

<table>
<thead>
<tr>
<th>Xi = 0</th>
<th>Xi = 1</th>
<th>Xi = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xj = 0</td>
<td>η</td>
<td>η(1+θ)</td>
</tr>
<tr>
<td>Xj = 1</td>
<td>η(1+θ)</td>
<td>η(1+2θ)</td>
</tr>
<tr>
<td>Xj = 2</td>
<td>η(1+2θ)</td>
<td>η(1+3θ)</td>
</tr>
</tbody>
</table>

Table 1: Two-locus Additive Model

To begin, we define a two-locus additive model. This simple model represents two disease loci, with additive risk for each minor allele present and additive risk between the two SNPs. It is intended for use as a baseline, since this model contains no epistasis and is extremely simple.

<table>
<thead>
<tr>
<th>Xi = 0</th>
<th>Xi = 1</th>
<th>Xi = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xj = 0</td>
<td>η</td>
<td>η</td>
</tr>
<tr>
<td>Xj = 1</td>
<td>η</td>
<td>η(1+θ)</td>
</tr>
<tr>
<td>Xj = 2</td>
<td>η</td>
<td>η(1+θ)</td>
</tr>
</tbody>
</table>

Table 2: Two-locus Threshold Model

The threshold model is a model with small amounts of epistasis. Basically, either locus is a dominant disease model, but both loci must have at least one minor allele for increased disease risk to be present. This model might also be described as a dominant model union another dominant model.

<table>
<thead>
<tr>
<th>Xi = 0</th>
<th>Xi = 1</th>
<th>Xi = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xj = 0</td>
<td>η</td>
<td>η</td>
</tr>
<tr>
<td>Xj = 1</td>
<td>η</td>
<td>η(1+θ)</td>
</tr>
<tr>
<td>Xj = 2</td>
<td>η(1+2θ)</td>
<td>η</td>
</tr>
</tbody>
</table>

Table 3: Two-locus Epistasis Model
Table 3 details what we will refer to as the two-locus epistasis model, which has been studied for years\textsuperscript{27}. This model has extreme epistasis between two loci. When the MAF is 0.5, this model has no marginal effect, meaning it is impossible to detect unless the GWAS algorithm can detect epistasis. This model biologically corresponds to two genes determining the phenotype, which also regulate each other. Thus, only if an individual has only two alleles of one risk SNP will they have a large chance of disease. If one disease allele of each SNP is present, then the effect size is smaller, and if both SNPs have minor alleles and one has at least two present, the effect reverts back to baseline.

4.2. Three-Locus Models

<table>
<thead>
<tr>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_h=0$</td>
<td>$\eta$</td>
<td>$\eta(\theta)$</td>
<td>$\eta(2\theta)$</td>
<td>$\eta(3\theta)$</td>
<td>$\eta(4\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
</tr>
<tr>
<td>$X_h=1$</td>
<td>$\eta(\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(2+\theta)$</td>
<td>$\eta(3+\theta)$</td>
<td>$\eta(4+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
</tr>
<tr>
<td>$X_h=2$</td>
<td>$\eta(2\theta)$</td>
<td>$\eta(3\theta)$</td>
<td>$\eta(4\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
</tr>
</tbody>
</table>

Table 4: Three-locus Additive Model

The three-locus additive model is a straightforward extension of the two-locus one. One important feature of higher order extensions of these simple models is that, despite each locus having significant marginal effect, interaction effects become important. Because each locus alone only explains one third of the disease risk, methods that only examine disease loci individually will have reduced power to find associations significant.

<table>
<thead>
<tr>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_h=0$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta(\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
</tr>
<tr>
<td>$X_h=1$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta(\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
</tr>
<tr>
<td>$X_h=2$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta(\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta(1+\theta)$</td>
</tr>
</tbody>
</table>

Table 5: Three-locus Threshold Model

The three-locus threshold model is exactly analogous to the two-locus one.
The two-locus epistasis model has a flaw that prevents it from being adequately extended to three loci. To achieve high epistasis situations, it requires the high-risk genotypes to be both major or minor alleles for all markers. Additionally, it cannot be aligned to the direction of the minor alleles (consider mirroring Table 3 horizontally), because the MAF effects would interfere with the ability to develop a high epistasis model. This would lead to one locus having high marginal effect, meaning the model effectively degrades back into a two-locus epistasis model with added noise. Thus, a more creative approach must be taken.

Table 6: Three-locus Epistasis (TSym) Model

<table>
<thead>
<tr>
<th></th>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
<th>$X_i=0$</th>
<th>$X_i=1$</th>
<th>$X_i=2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_h=0$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta(1+2\theta)$</td>
<td>$\eta$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta$</td>
<td>$\eta(1+2\theta)$</td>
<td>$\eta$</td>
<td>$\eta$</td>
</tr>
<tr>
<td>$X_h=1$</td>
<td>$\eta$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta$</td>
</tr>
<tr>
<td>$X_h=2$</td>
<td>$\eta(1+2\theta)$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta(1+\theta)$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta$</td>
<td>$\eta(1+2\theta)$</td>
</tr>
</tbody>
</table>

Table 6 details a three-locus epistasis model I will call TSym, short for totally symmetric. If one visualizes the three-locus contingency table as a cube, TSym is created by wrapping 4 two-locus epistasis models around the faces of the cube. Because this distributes risk symmetrically between all three loci, a high epistasis model can be formed. As in the two-locus case, this model has no marginal effect with MAF of 0.5.

Figure 2: TSym Model geometric representation. Each axis is interpreted as one SNP; red dots represent relative increase in disease risk.
Additionally, it is even harder to detect in this case because it has no two-locus effects between any two pairs of its loci. Thus, this model is an extremely strong epistasis model that is very difficult to detect.

4.3. A Higher Order Interaction Model

Since the number of genotype combinations grows exponentially with respect to the number of interacting loci, detailing higher order interaction models explicitly becomes difficult, and their meaning can become incomprehensible. We take an approach used by others and combine smaller models to formulate higher order ones, with additive risk between the so-called epistasis modules.

<table>
<thead>
<tr>
<th>X_i=0</th>
<th>X_i=1</th>
<th>X_i=2</th>
<th>X_i=0</th>
<th>X_i=1</th>
<th>X_i=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>η</td>
<td>η</td>
<td>H</td>
<td>η</td>
<td>η(1+2θ)</td>
</tr>
<tr>
<td>H</td>
<td>η(1+θ)</td>
<td>η(1+θ)</td>
<td>H</td>
<td>η</td>
<td>η(1+θ)</td>
</tr>
<tr>
<td>H</td>
<td>η(1+θ)</td>
<td>η(1+θ)</td>
<td>H</td>
<td>η</td>
<td>η(1+2θ)</td>
</tr>
</tbody>
</table>

Table 7: Four-locus Model consisting of two independent subgroups, one like Table 2 and one like Table 3, with additive effect when they act jointly.

4.4. Methods of Parameter Setting

While developing models is one step of creating an exhaustive simulation, controlling model parameters is even more important to ensure the results test the desired properties. For example, the two-locus epistasis model in Table 3 is not extremely interesting at low MAF. This is because MAF changes the distribution of the genotypes simulated, and the result is that each locus has reasonably high marginal effect. Thus, methods to control the parameter settings must be developed.
First, I'll describe a calculation for a value called $\lambda$, which has been used by others such as Zhang et al. Then, I will describe the drawbacks of this method. Finally, I will detail our proposed solution using the expected Pearson's $\chi^2$ Test value.

### 4.4.1. Parameter Setting Using Marginal Effect Size $\lambda$

One method of controlling parameters is to directly control the marginal effect size in the models. This is done by calculating a statistic $\lambda$. The exact calculation can vary based on normalization, aligning with zero, etc., but the general concept is to get a ratio of the effect size increase. Here I will follow the lead of Marichini et al and Zhang et al in describing $\lambda$. See equation (8).

\[
\text{odds}(g_i) = \frac{P(D|g_i)}{P(D|\overline{g_i})}
\]  

(7)

\[
\lambda = \frac{\text{odds}(Aa)}{\text{odds}(aa)} - 1
\]  

(8)

For example, for the three-locus additive model above, one can calculate lambda as follows. First, calculate the penetrance at the aa genotype as $\eta(1 + 2p_2 \theta + 2p_3 \theta)$. Here $p_2$ and $p_3$ are the MAF of locus two and locus 3. Thus, the odds(aa) is $\frac{\eta(1+2p_2 \theta +2p_3 \theta)}{1-\eta(1+2p_2 \theta +2p_3 \theta)}$. Using a similar calculation for the Aa genotype, we arrive at a lambda of $\frac{\eta(1+\theta+2p_2 \theta +2p_3 \theta)}{1-\eta(1+\theta+2p_2 \theta +2p_3 \theta)} / \frac{\eta(1+2p_2 \theta +2p_3 \theta)}{1-\eta(1+2p_2 \theta +2p_3 \theta)} - 1$.

Using this formula, a graph of the effect size vs. parameters can be generated. Some examples for three-locus models follow.
In Figure 3 we show how $\lambda$ approximates the power of the model; it increases with theta as expected.
Figure 4: Parameters vs. effect size for three-locus threshold model. $\lambda+1$ is the vertical axis. Figure courtesy Yixuan Chen

Figure 4 shows a similar graph for the three-locus threshold model. We can see a similar effect with theta, but how MAF effects can change based on the model.

Using this sort of analysis, one can pick a MAF, baseline $\eta$, "marginal effect" $\lambda$, and desired population prevalence, and decide how much effect size is necessary.
λ has some useful properties. For the two-locus epistasis model, we have already stated that the marginal effect is zero when the MAF is 0.5. Figure 5 shows that changing theta does not increase λ for MAF 0.5 in this case, just as expected. Thus, in this sense at least λ is a good approximation for determining if there is zero marginal effect.

4.4.2. **Drawbacks to parameter setting using λ**

As section 4.4.3 shows, we did not use λ to set the effect size of our models. Previously, authors had effectively defined "marginal effect size" and λ to mean the same thing. That is, high λ meant high effect size. Yet this is not the whole story. Consider Figure 6.
The additive model, intended to be a model control consisting of no epistasis, clearly has varied marginal effect size with constant $\lambda$. This makes it difficult to understand how even simple tests like chi-square will perform when considering just $\lambda$. This is because $\lambda$ is only concerned with the increase of risk between genotype combinations. As such, it only accounts for MAF with respect to normalizing for other loci. Thus, MAF also has a huge effect on the actual marginal effect of a model, since it works to weight the penetrance table of the model. Thus, we reject $\lambda$ as a suitable choice for controlling the power of our model parameters.

4.4.3. Parameter Setting using $\chi^2$

Instead of the $\lambda$ approach, we will use the expected Pearson's chi-square test value to set parameters. This means that we control marginal effect size very explicitly, in relation to a popular statistic that only captures marginal effects.

<table>
<thead>
<tr>
<th></th>
<th>prob</th>
<th>E</th>
<th></th>
<th>prob</th>
<th>E</th>
<th></th>
<th>prob</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(aa</td>
<td>D)</td>
<td>0.16</td>
<td>640.27</td>
<td>P(bb</td>
<td>D)</td>
<td>0.16</td>
<td>640.27</td>
<td>P(cc</td>
</tr>
<tr>
<td>P(Aa</td>
<td>D)</td>
<td>0.28</td>
<td>1119.78</td>
<td>P(Bb</td>
<td>D)</td>
<td>0.28</td>
<td>1119.78</td>
<td>P(Cc</td>
</tr>
<tr>
<td>P(AA</td>
<td>D)</td>
<td>0.06</td>
<td>239.95</td>
<td>P(BB</td>
<td>D)</td>
<td>0.06</td>
<td>239.95</td>
<td>P(CC</td>
</tr>
<tr>
<td>P(aa</td>
<td>C)</td>
<td>0.25</td>
<td>1017.75</td>
<td>P(bb</td>
<td>C)</td>
<td>0.25</td>
<td>1017.75</td>
<td>P(cc</td>
</tr>
<tr>
<td>P(Aa</td>
<td>C)</td>
<td>0.20</td>
<td>808.91</td>
<td>P(Bb</td>
<td>C)</td>
<td>0.20</td>
<td>808.91</td>
<td>P(Cc</td>
</tr>
<tr>
<td>P(AA</td>
<td>C)</td>
<td>0.04</td>
<td>173.34</td>
<td>P(BB</td>
<td>C)</td>
<td>0.04</td>
<td>173.34</td>
<td>P(CC</td>
</tr>
<tr>
<td>total</td>
<td>1.00</td>
<td>4000.00</td>
<td>total</td>
<td>1.00</td>
<td>4000.00</td>
<td>total</td>
<td>1.00</td>
<td>4000.00</td>
</tr>
</tbody>
</table>
Table 8: Expected counts of various genotypes for three-locus threshold model, assuming 50/50 N=4000 case-control study

From Table 8, we can see that it is easy to calculate the expected Pearson's chi-square value for a particular model and parameter set. Table 8 can be calculated from the penetrance table, η and θ, N, and MAF directly using Bayes formula. Thus, we control model parameters so that Pearson's chi-square detection power varies from 0 to 1, and therefore get a broad view of the disease model. Ideal GWAS algorithms should roughly track this power in simple cases, but should be able to detect epistasis and beat this power on suitable models.

As stated above, we can use Bayes form to calculate the genotype frequencies as follows.

\[
P(g_i|D) = \frac{P(g_i)P(D|g_i)}{\sum_j P(g_j)P(D|g_j)}
\]  
(9)

Using these genotype frequencies and an assumed sample size we can calculate the expected chi-square statistic as follows.

\[
\chi^2 = N \sum_i \frac{(P(g_i|D) - P(g_i|\bar{D}))^2}{P(g_i|D) - P(g_i|\bar{D})}
\]  
(10)

To compute the power of this test, we must know the distribution of the Pearson's \( \chi^2 \) Test statistic under the alternative hypothesis. This is a non-central chi-square distribution \( \chi^2(k, s) \) where \( k \) is the degrees of freedom and \( s \) is the non-centrality parameter. However, when the number of cases and controls is equal, the non-
centrality parameter is exactly the chi-square test value. Therefore, given the numeric values of the expected genotype frequencies in cases and controls, the significance level, and the degree of freedom, we can numerically obtain the “expected” power of the $\chi^2$ test.

### 4.5. Construction of Simulated Models

Of course, disease models and parameters alone are insufficient for testing GWAS algorithms. We must embed the signal into background noise somehow. There are two common approaches to this, the first being to simulate the noise simultaneously with the disease model, and the second being to embed the simulated signal into separately generated noise. While the first can more easily include complexities such as simulated linkage disequilibrium between signal SNPs and noise SNPs, the second is simpler and can have a more realistic background. To keep things simple, we adopt the second approach.

To generate one hundred replicates of a simulated dataset, first we input the disease model and parameters into the gs program\textsuperscript{29}. Gs is a program developed by Yixuan Chen and Jing Li which, among other things, can simulate disease SNPs defined by models such as the ones in this section. Next, we take a subset of the WTCCC data\textsuperscript{2} for background noise. Since this data is merely being used as background, we selected 4000 individuals total, some from each of the Bipolar Disorder, National Blood Service, and Type-I Diabetes datasets. For statistical expediency, we limit our noise to only the first chromosome, or 40220 SNPs, but make no attempt to clean the data in any manner (we used the false positive rate of the simulations in section 5 to ensure this did not adversely affect the results). While ideally GWAS simulations should run at sizes
equivalent to real studies, algorithm performance and computational resource restrictions prevent this from being possible. 40220 SNPs is already two orders of magnitude above the size of simulations done in other papers\textsuperscript{18}, so we hope to better examine algorithm performance on large datasets. This noise is then merged with the signal by randomly replacing noise SNPs with the values from the signal. It is important to randomize the SNP index and the order of the individuals in this step to avoid interactions between the background signal and simulation signal. A key is saved at this step so that simulations can be performed blind, but the true results can be discovered later. Finally, the merged dataset is saved in a format applicable to the algorithm being run.

There are several other considerations in formulating the simulation study. One is the number of replicates. As stated above, we use only 100 replicates to deal with random chance skewing the results. While this is more than used by others (some are as low as twenty\textsuperscript{18}), it is not the 1000 suggested for rigorous statistical properties\textsuperscript{7}. We considered 100 a good tradeoff between reasonably low noise while still being computationally feasible. It does mean that small deviations in the results in section 5 will not be statistically significant. Another consideration is linkage disequilibrium (LD). In real data, usually disease SNPs are correlated with those around them. Some algorithms have attempted to use this fact to increase statistical significance\textsuperscript{1}, but for simplicity we ignore this (since it seems on shaky statistical basis). LD also could be a factor in a different way, namely that the underlying risk SNP might not be typed, but a marker in LD could be. This basically has the effect of increasing the noise in the model
(and reducing the power), so we did not test this in our initial experiments. We do wish to include this analysis in the future, see section 7.

5. Simulation Results

This section will detail results from our simulation study using WTCCC data as background noise and signal from our models in section 4. We compare four algorithms: classical Pearson's chi-square test with Bonferroni correction for multiple testing, chi-square test with permutation test derived significance threshold, our own Adaboost based machine learning approach with 1000 iterations and 546 individuals in a bootstrap sample (this number was chosen by accident; typically 4000 would be used but the difference should be marginal), and finally the BEAM algorithm, which is an archetypical statistical inference method using Bayesian interference and Markov Chain Monte Carlo (MCMC) sampling. These algorithms were chosen to allow us to use fundamentally different algorithms while respecting our limited computational resources. All algorithm powers presented are derived from the power on one hundred replicates with a false positive rate of $\alpha=0.05$. BEAM was run with all other parameters set to their defaults, except the number of MCMC rounds was set to 16,176,484, or 1% of the number of markers squared, due to time constraints.

5.1. Two-locus Models

The Two-Locus Additive Model is used as a baseline to ensure algorithms are working properly. We expect Bonferroni corrected $\chi^2$ test and BEAM to have similar power. Similarly, we expect permutation threshold $\chi^2$ test and our Adaboost approach to have
similar power to each other. The latter two approaches should have slightly higher power than the former on non-epistasis models, as they exploit internal structure in the data to give a tighter threshold while still controlling type-I error.

![Graph showing power comparison]

Figure 7: Two-Locus Additive Model with \( \theta = 0.2 \)

Figure 7 shows the lower signal \( \theta = 0.2 \) additive model. All algorithms have roughly similar power, with the two permutation test algorithms slightly ahead. Unexpectedly, our approach has slightly more power than the permutation threshold \( \chi^2 \) test. There are at least two possible causes for this. One is that our approach is more stable with respect to the type of data we are dealing with, allowing the permutation threshold to be tighter. The second, and in our opinion more likely, cause is due to the recursive partitioning. An additive model does not have direct interaction between the loci’s effects, but even so, the presence of multiple effect loci means that the data looks noisier to a single locus test. Recursive partitioning partially shields lower splits from
this noise by condition on the split feature's values. Thus, for the additive model, the test at the second locus will get done three times (each on roughly one third of the data), but the distribution of class labels in these nodes has been perturbed such that the cases tend to avoid the controls. This effectively removes much of the noise for the second test.

Figure 8: Two-Locus Additive Model with theta=0.35
Figure 8 and Figure 9 are higher signal versions of Figure 7. As such, they are very similar and have roughly the same characteristics. However, there is an important feature in the MAF=0.1 sections of these graphs. As can be seen, our approach actually has lower power than even the classical Bonferroni corrected $\chi^2$ test. This is a current limitation of our algorithm that will be addressed a little in section 7.1. This reduced power is not an extremely important factor as it happens on the easiest parameter settings when there is little epistasis. Thus, it should be trivial to modify our approach to fall back to a more standard algorithm in this case. Still, it is an important limitation of the approach as formulated.
The threshold model is the first model with some degree of epistasis. Our approach outperforms all others across varying minor allele frequencies in Figure 10 and is the only algorithm to detect both loci in a nontrivial fraction of the replicates. Additionally, we can see that BEAM’s power roughly tracks that of Bonferroni corrected $\chi^2$ test until the MAF=0.5 case. Whether it is detecting interaction effects, has a tighter threshold due to formulation (like the permutation threshold $\chi^2$ test), or a combination of both, is unclear, but it still manages some power in the MAF=0.5 case.

The following graphs will examine this threshold model at higher signal levels.
Figure 11 and Figure 12 have very similar trends. As before, our approach out-performs all other algorithms across varying MAF. BEAM continues to be beaten by the permutation threshold $\chi^2$ test, even in the MAF=0.5 case, showing that its ability to
declare features significant using interaction effects is not significantly stronger than the ability of a permutation test to lower the importance threshold.

The MAF=0.5 cases of the threshold model show an "easy" interaction case. In these easier cases, either slightly increasing the power using the underlying structure of the data or detecting interaction effects are effective, as Figure 11 and Figure 12 show using the permutation threshold $\chi^2$ test and BEAM, respectively.

Since the threshold model has inherently lower signal at MAF=0.1, we increased the theta some. Most of the approaches perform similarly here, with our approach having reduced power compared to above due to the aforementioned trouble with disease models with lower MAF. Due to the lack of epistasis, every approach has very similar power.
The epistasis model is intended as an extreme epistasis case, when two disease genes additionally regulate each other. Nevertheless, unless the MAF is close to 0.5, there are still non-trivial marginal effects in the model. As before, lower MAF reduces any epistasis effect and causes the disease model to decay into a predominately marginal signal. Figure 14, Figure 15, and Figure 16 examine these cases.
As before, all approaches perform similarly except ours. Ironically, our approach has relatively lower power in the easier cases, and relatively higher power as the interaction increases.
Figure 17 examines the epistasis model at high MAF and therefore high interaction, which makes detection increasingly difficult. Our approach is exceptionally effective in the intermediate range of MAF=0.4, with a maximum advantage in ability to detect both SNPs of 0.8 in the theta=1 case (over the permutation threshold $\chi^2$ test, no less). BEAM has slightly increased power compared to the Bonferroni corrected $\chi^2$ test, showing it is sometimes detecting some epistasis effect, but as before it cannot even beat the permutation threshold $\chi^2$ test.

The MAF=0.5 case is special because it is the only disease model we examined with absolutely zero marginal effect. Thus, the expected chi-squared test value at a disease locus is exactly zero. Therefore, interaction effects are extremely important as they are the only effective way to detect association. The difficulty of this problem is evident, as the only approach to post any results was ours with a measly 0.05 power.
5.2. Three-locus Models

First, we examine the three-locus extension of the threshold model.

As before, our approach is more effective than the competitors at high MAF. BEAM is beaten by the permutation threshold \( \chi^2 \) test. The low signal prevents this graph from being especially illuminating.
In Figure 19 and Figure 20, we can better examine algorithmic performance. Our approach continues to perform well in the MAF=0.3 cases, and the MAF=0.5 cases have sufficient signal that the results begin to flatten out. No algorithm has any power...
whatsoever in the MAF=0.1 case, despite the increase in theta to 1.5, showing that the three-locus threshold model requires an additional boost in power, analogous to the two-locus version.

Figure 21: Three-Locus TSym Model with theta=1

Figure 21 shows the performance on our tough three-locus model. As is plainly evident, the high interaction of this model causes it to become very difficult with moderate MAF. Additionally, our approach continues to have trouble with lower MAF, and BEAM, along with our approach, is routinely beaten by the permutation threshold $\chi^2$ test. This shows the difficulty in detecting many-loci epistasis disease models.
Figure 22 and Figure 23 show the effect of increasing signal size. Since there is little epistasis at low MAF, this increases the power of all approaches to 1. At higher MAF, more interesting effects are shown. Our approach has a significant lead over all others.
in the high epistasis MAF=0.3 case and is the only algorithm to detect all three disease loci a nontrivial fraction of the time.

5.3. **Higher-order Interaction Model**

For the Higher-order Interaction Model, we set the MAF such that both groups of loci had roughly the same marginal effect.

![Figure 24: Average Pearson’s $\chi^2$ test value for the four loci in the higher-order interaction model](image)

*xs value*
Once again we see that our approach is more powerful than all other approaches. Beam is roughly even with Pearson's Bonferroni Corrected $\chi^2$ test, showing that it is unable to significantly detect the interactions in this model.

5.4. Computational Resources

On the Ohio Supercomputing Center Glenn Cluster nodes, BEAM requires about 5 hours 40 minutes to complete one replicate (one instance from the 100 replicates of any single model, MAF, and theta above) of our simulation studies, while our approach takes only about 30 minutes. Additionally, because loading the data onto the cluster node takes a nontrivial amount of time, our approach spends a smaller fraction of time doing actual
computation. Finally, BEAM has extreme variance in running time, and can occasionally take up to 10 hours to complete because the MCMC sampler does not converge quickly.

Both implementations use efficient data storage and take little extra overhead. BEAM uses around half of the memory of our implementation, possibly due to inefficiencies in our data loading process.

6. Application to WTCCC Affymetrix SNP data

After proving out method's validity and efficiency on simulation data, we applied it to the WTCCC Affymetrix SNP data, case-control data of over 14,000 individuals across seven diseases. We use the WTCCC's list of excluded SNPs for data cleaning. The criteria for generating this list is MAF greater than 1%, missing rate less than 1% and some SNPs included due to genotype calling problems noticed by visual inspection. This leaves 469,412 SNPs.

Due to a bug in a library we used, our method currently can't handle datasets larger than 2GB. Thus, for the initial examination we created smaller datasets by forming a 1:1 ratio of cases and controls by excluding any extra controls present. This meant data sizes of roughly 3,800 to 4,000 individuals.

The cleaned WTCCC data has many loci where Pearson's $\chi^2$ test is not applicable. This is because the test only approximates a chi-square distribution. If cells in the contingency table have extremely low values (near zero), this approximation becomes invalid. In
generating their results, the WTCCC does not include genotypic test values for these loci, but the exact threshold for exclusion is not mentioned in their paper.

While our approach does not assume an underlying distribution, we find significance threshold controlling to be very incorrect on the WTCCC data. Specifically, our method reports extreme scores for many loci in the data. However, repeating the permutation test on the full number of SNPs confirms our earlier permutation threshold. If these case-control ratios were random chance they should have increased the permutation threshold. The exact meaning of our results must be carefully interpreted.

Preliminary tables of the top reported SNPs follows. These tables have only had a cursory comparison with the WTCCC as we have not finished that analysis. A * indicates an instance where a SNP or gene was reported as associated in the WTCCC paper. Base position (bp) of zero indicates that the WTCCC data does not include position information for that locus. There could be more overlap between our results and the WTCCC reported genes. This is because the Affymetrix chip is so dense that analyzing individual SNPs can be impractical; analyzing at the gene level is much simpler for validating results. LD is also high between SNPs, meaning that an associated gene often creates many significant SNPs. These tables also contain gene information from dbSNP.

### 6.1. Bipolar Disease

<table>
<thead>
<tr>
<th>rs#</th>
<th>chr</th>
<th>bp</th>
<th>gene if reported in wtccc</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs17480050</td>
<td>8</td>
<td>19426197</td>
<td>CSGALNACT1</td>
<td></td>
</tr>
<tr>
<td>rs1048194</td>
<td>16</td>
<td>79622203</td>
<td>*</td>
<td>CENPN</td>
</tr>
<tr>
<td>rs4922060</td>
<td>8</td>
<td>19452788</td>
<td>CSGALNACT1</td>
<td></td>
</tr>
<tr>
<td>SNP_A-1992585</td>
<td>8</td>
<td>19454071</td>
<td>CSGALNACT1</td>
<td></td>
</tr>
</tbody>
</table>
Table 9: Bipolar Disease top 15 SNPs

<table>
<thead>
<tr>
<th>SNP</th>
<th>Chromosome</th>
<th>Position</th>
<th>Gene</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs16998540</td>
<td>X</td>
<td>36167018</td>
<td>CXorf30</td>
</tr>
<tr>
<td>rs11784776</td>
<td>8</td>
<td>19396011</td>
<td>CSGALNACT1</td>
</tr>
<tr>
<td>rs17479749</td>
<td>8</td>
<td>19404858</td>
<td>CSGALNACT1</td>
</tr>
<tr>
<td>rs16887854</td>
<td>8</td>
<td>38853285</td>
<td>unknown</td>
</tr>
<tr>
<td>rs7699261</td>
<td>4</td>
<td>145491340</td>
<td>unknown</td>
</tr>
<tr>
<td>rs9967923</td>
<td>20</td>
<td>23901078</td>
<td>unknown</td>
</tr>
<tr>
<td>rs10850424</td>
<td>12</td>
<td>113911241</td>
<td>unknown</td>
</tr>
<tr>
<td>rs11227044</td>
<td>11</td>
<td>105825863</td>
<td>unknown</td>
</tr>
<tr>
<td>rs9972717</td>
<td>16</td>
<td>52408805</td>
<td>FTO</td>
</tr>
<tr>
<td>rs2092590</td>
<td>6</td>
<td>172385</td>
<td>unknown</td>
</tr>
<tr>
<td>rs16822398</td>
<td>3</td>
<td>154351388</td>
<td>unknown</td>
</tr>
</tbody>
</table>

The SNP rs1048194 was also reported as significant in the WTCCC's data's single locus analysis.

6.2. Coronary Artery Disease

Table 10: Coronary Artery Disease top 15 SNPs

<table>
<thead>
<tr>
<th>SNP</th>
<th>Chromosome</th>
<th>Position</th>
<th>Gene</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs2241572</td>
<td>11</td>
<td>129298658</td>
<td>PRDM10</td>
<td></td>
</tr>
<tr>
<td>rs3734068</td>
<td>11</td>
<td>129320230</td>
<td>PRDM10</td>
<td></td>
</tr>
<tr>
<td>rs3785579</td>
<td>17</td>
<td>62472963</td>
<td>CACNG1</td>
<td></td>
</tr>
<tr>
<td>rs17480050</td>
<td>8</td>
<td>19426197</td>
<td>CSGALNACT1</td>
<td></td>
</tr>
<tr>
<td>rs9892020</td>
<td>17</td>
<td>62460904</td>
<td>LOC100289587</td>
<td></td>
</tr>
<tr>
<td>rs2159210</td>
<td>17</td>
<td>62466422</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs4629211</td>
<td>2</td>
<td>138994974</td>
<td>SPOPL</td>
<td></td>
</tr>
<tr>
<td>rs9923888</td>
<td>16</td>
<td>76996600</td>
<td>WWOX</td>
<td></td>
</tr>
<tr>
<td>rs8065894</td>
<td>17</td>
<td>30882043</td>
<td>SLFN12L</td>
<td></td>
</tr>
<tr>
<td>rs10495687</td>
<td>2</td>
<td>18580060</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs736097</td>
<td>X</td>
<td>28701790</td>
<td>IL1RAPL1</td>
<td></td>
</tr>
<tr>
<td>rs8111044</td>
<td>19</td>
<td>49741978</td>
<td>FLJ41856</td>
<td></td>
</tr>
<tr>
<td>rs12417428</td>
<td>11</td>
<td>26779037</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs7574222</td>
<td>2</td>
<td>139056521</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs13439520</td>
<td>8</td>
<td>80983708</td>
<td>unknown</td>
<td></td>
</tr>
</tbody>
</table>

In our preliminary analysis, none of the SNPs we report match those mentioned in the original WTCCC report. However, further analysis might reveal overlaps.
6.3. Chron's Disease

<table>
<thead>
<tr>
<th>rs#</th>
<th>chr</th>
<th>bp</th>
<th>reported in wtccc</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs157613</td>
<td>4</td>
<td>16387385</td>
<td></td>
<td>LDB2</td>
</tr>
<tr>
<td>rs1553460</td>
<td>4</td>
<td>17804959</td>
<td>unk</td>
<td></td>
</tr>
<tr>
<td>rs12078461</td>
<td>1</td>
<td>117273572</td>
<td>PTGFRN</td>
<td></td>
</tr>
<tr>
<td>rs2416472</td>
<td>5</td>
<td>117033845</td>
<td>unk</td>
<td></td>
</tr>
<tr>
<td>rs4471699</td>
<td>16</td>
<td>30227808</td>
<td>SMG1</td>
<td></td>
</tr>
<tr>
<td>rs11938418</td>
<td>4</td>
<td>17804041</td>
<td>unk</td>
<td></td>
</tr>
<tr>
<td>rs207683</td>
<td>4</td>
<td>16381403</td>
<td>LDB2</td>
<td></td>
</tr>
<tr>
<td>rs150261</td>
<td>4</td>
<td>16417600</td>
<td>LDB2</td>
<td></td>
</tr>
<tr>
<td>rs6431654</td>
<td>2</td>
<td>233826508</td>
<td>*</td>
<td>ATG16L1</td>
</tr>
<tr>
<td>rs189988</td>
<td>4</td>
<td>16379333</td>
<td>LDB2</td>
<td></td>
</tr>
<tr>
<td>rs150260</td>
<td>4</td>
<td>16386273</td>
<td>LDB2</td>
<td></td>
</tr>
<tr>
<td>rs6752107</td>
<td>2</td>
<td>233826187</td>
<td>*</td>
<td>ATG16L1</td>
</tr>
<tr>
<td>rs3792106</td>
<td>2</td>
<td>233855479</td>
<td>*</td>
<td>ATG16L1</td>
</tr>
<tr>
<td>rs287961</td>
<td>4</td>
<td>16399413</td>
<td>LDB2</td>
<td></td>
</tr>
<tr>
<td>rs302925</td>
<td>9</td>
<td>134563217</td>
<td>unk</td>
<td></td>
</tr>
</tbody>
</table>

Table 11: Chron's Disease top 15 SNPs

Here, we report several loci from the ATG16L1 gene which WTCCC also reports.

However, they are not strongly associated. We also report lots of loci from the LDB2 region, for reasons unknown.

6.4. Hypertension

<table>
<thead>
<tr>
<th>rs#</th>
<th>chr</th>
<th>bp</th>
<th>reported in wtccc</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs825148</td>
<td>1</td>
<td>221661863</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs1528356</td>
<td>7</td>
<td>120908691</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs10843660</td>
<td>12</td>
<td>30259724</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs16837871</td>
<td>4</td>
<td>5992013</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs10743704</td>
<td>12</td>
<td>30249349</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs4131463</td>
<td>6</td>
<td>99364552</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs9669515</td>
<td>12</td>
<td>30260529</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs7664229</td>
<td>4</td>
<td>5993536</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs7691494</td>
<td>4</td>
<td>5993550</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs1157480</td>
<td>12</td>
<td>30264271</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs7664218</td>
<td>4</td>
<td>5993514</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs1553460</td>
<td>4</td>
<td>17804959</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs6840033</td>
<td>4</td>
<td>141448311</td>
<td>SCOC</td>
<td></td>
</tr>
</tbody>
</table>

54
The WTCCC did not report any significant loci for the hypertension dataset, and largely our results seem similar with few SNPs of known function reported.

### 6.5. Rheumatoid Arthritis

<table>
<thead>
<tr>
<th>rs#</th>
<th>chr</th>
<th>bp</th>
<th>notes</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs3785579</td>
<td>17</td>
<td>62472963</td>
<td>CACNG1</td>
<td></td>
</tr>
<tr>
<td>rs9892020</td>
<td>17</td>
<td>62460904</td>
<td>LOC100289587</td>
<td></td>
</tr>
<tr>
<td>rs2159210</td>
<td>17</td>
<td>62466422</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs6457620</td>
<td>6</td>
<td>0</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs7539166</td>
<td>1</td>
<td>15368492</td>
<td>TMEM51</td>
<td></td>
</tr>
<tr>
<td>rs2121526</td>
<td>10</td>
<td>55356769</td>
<td>PCDH15</td>
<td></td>
</tr>
<tr>
<td>rs13425033</td>
<td>2</td>
<td>105443057</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs12010932</td>
<td>X</td>
<td>120783817</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs17665418</td>
<td>3</td>
<td>71931855</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs4959053</td>
<td>6</td>
<td>0</td>
<td>PSORS1C1</td>
<td></td>
</tr>
<tr>
<td>rs9268402</td>
<td>6</td>
<td>0</td>
<td>C6orf10</td>
<td></td>
</tr>
<tr>
<td>rs1440065</td>
<td>2</td>
<td>223557854</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs851235</td>
<td>X</td>
<td>46056459</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs6457617</td>
<td>6</td>
<td>0</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>rs851236</td>
<td>X</td>
<td>46056396</td>
<td>unknown</td>
<td></td>
</tr>
</tbody>
</table>

Table 13: Rheumatoid Arthritis top 15 SNPs

In our preliminary analysis, none of the SNPs we report match those mentioned in the original WTCCC report. However, further analysis might reveal overlaps.

### 6.6. Type-I Diabetes

<table>
<thead>
<tr>
<th>rs#</th>
<th>chr</th>
<th>bp</th>
<th>reported in wtccc</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs9273363</td>
<td>6</td>
<td>0</td>
<td>*</td>
<td>downstream of HLA-DRB1</td>
</tr>
<tr>
<td>rs9275418</td>
<td>6</td>
<td>0</td>
<td>unknown?</td>
<td></td>
</tr>
<tr>
<td>rs9391858</td>
<td>6</td>
<td>0</td>
<td>C6orf10</td>
<td></td>
</tr>
<tr>
<td>rs3177928</td>
<td>6</td>
<td>0</td>
<td>*</td>
<td>HLA-DRA</td>
</tr>
<tr>
<td>rs3805006</td>
<td>3</td>
<td>4775769</td>
<td></td>
<td>ITPR1?</td>
</tr>
<tr>
<td>rs4254200</td>
<td>13</td>
<td>82213653</td>
<td></td>
<td>unknown</td>
</tr>
<tr>
<td>rs3135377</td>
<td>6</td>
<td>0</td>
<td>unknown</td>
<td></td>
</tr>
</tbody>
</table>

55
Table 14: Type-I Diabetes top 15 SNPs

The major histocompatibility complex is located on chromosome 6 and includes genes related to the immune system. This area, especially the HLA genes are known to be linked with type-I diabetes and we show strong association. Possibly many of the unknown SNPs of unknown location from chromosome six are also from this region.

6.7. Type-II Diabetes

<table>
<thead>
<tr>
<th>rs#</th>
<th>chr</th>
<th>bp</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs17480050</td>
<td>8</td>
<td>19426197</td>
<td>CSGALNACT1</td>
</tr>
<tr>
<td>rs4922060</td>
<td>8</td>
<td>19452788</td>
<td>CSGALNACT1</td>
</tr>
<tr>
<td>rs3777582</td>
<td>6</td>
<td>46022006</td>
<td>CLIC5</td>
</tr>
<tr>
<td>rs11705626</td>
<td>22</td>
<td>23765959</td>
<td>KIAA, mentioned in T1D section</td>
</tr>
<tr>
<td>rs16837871</td>
<td>4</td>
<td>5992013</td>
<td>unknown</td>
</tr>
<tr>
<td>rs11042656</td>
<td>11</td>
<td>10171031</td>
<td>SBF2</td>
</tr>
<tr>
<td>rs11784776</td>
<td>8</td>
<td>19396111</td>
<td>CSGALNACT1</td>
</tr>
<tr>
<td>SNP_A-1992585</td>
<td>8</td>
<td>19454071</td>
<td>CSGALNACT1</td>
</tr>
<tr>
<td>rs17479749</td>
<td>8</td>
<td>19404858</td>
<td>CSGALNACT1</td>
</tr>
<tr>
<td>rs226261</td>
<td>21</td>
<td>26953847</td>
<td>unknown</td>
</tr>
<tr>
<td>rs7818368</td>
<td>8</td>
<td>19483828</td>
<td>CSGALNACT1</td>
</tr>
<tr>
<td>rs16998540</td>
<td>X</td>
<td>36167018</td>
<td>unknown</td>
</tr>
<tr>
<td>rs7691494</td>
<td>4</td>
<td>5993550</td>
<td>unknown</td>
</tr>
<tr>
<td>rs3777583</td>
<td>6</td>
<td>46022068</td>
<td>CLIC5</td>
</tr>
<tr>
<td>rs11104004</td>
<td>12</td>
<td>76089203</td>
<td>unknown</td>
</tr>
</tbody>
</table>

Table 15: Type-II Diabetes top 15 SNPs

In our preliminary analysis, none of the SNPs we report match those mentioned in the original WTCCC report. However, further analysis might reveal overlaps.
7. Conclusion and Future Work

We have defined the problem of epistasis detection in GWAS, examined proposed approaches in the literature, formulated our own approach based on the ensemble algorithm Adaboost, Decision Trees, and a custom variable importance measure, and tested our approach versus classical approaches and the epistasis mapping algorithm BEAM.

In models with high epistasis, we saw that our approach was the most effective and was vastly more powerful than single-locus approaches. However, we saw a limitation for models with low MAF, even when this meant that there was little epistasis and a single-locus test like Gini importance should be applicable. This limitation of our approach is further discussed in section 7.1, and other possible improvements to our approach are detailed after.

BEAM uses a MCMC sampling procedure to attempt to determine which loci are in epistasis. However, our simulations show that the algorithm often is not more powerful than a permutation threshold $\chi^2$ test. We hypothesize that this is because of the larger number of SNPs in our simulation. Since the number of suggested MCMC rounds is dependent on the square of the number of SNPs, BEAM is very sensitive to that number. Our simulations include more than an order of magnitude more SNPs than those in the original BEAM paper. To deal with this, we ran BEAM with far fewer MCMC rounds than suggested. This presumably results in a quantifiable power reduction; nevertheless BEAM still used significantly more time than our approach. Furthermore, algorithms like
BEAM, which have an implicit quadratic dependency on running time, need to prove they can be more effective that the two-locus chi-square test. This effectively means being able to detect loci from our TSym model with MAF=0.5, since that model will not have any two-locus marginal effect. As BEAM was unable to detect any replicates of our single-locus model without marginal effect and was routinely beaten by the permutation threshold $\chi^2$ test, it is exceedingly unlikely that it can do this.

In the disease model with no marginal effect, no approaches were able to detect a nontrivial fraction of replicates. This is important, because one of the main arguments against the classical stepwise logistic regression approach is that it might miss some SNPs with low marginal effects. Yet, proposed algorithms also fail on this difficult case on GWAS scale data sets. Future studies should work on real GWAS sized data and examine whether proposed approaches are actually capable of detecting epistasis at a level that would not pass the single-locus screen of stepwise logistic regression.

Previous studies have only considered this for small number of SNPs where doing a full two-locus search of the search space is easily done.

One simple extension to our disease models we would like to examine is the effect of Linkage Disequilibrium (LD). Instead of embedding disease SNPs themselves, we would simulate what would happen if the functional vector was not genotyped, but other close by SNPs were. These SNPs would be in varying levels LD with the disease locus, meaning they could be detected as associated with the disease, with reduced power depending on the level of LD.
7.1. **Reduced power at low MAF**

As noted several times, our algorithm, despite performing comparatively well on medium to high epistasis models, was outperformed on models with low MAF and low epistasis, which should be easy for an approach based largely on single-locus statistics. We hypothesize that this could be due to the nature of the Gini impurity statistic. In this section I will explain how differing test statistics coupled with low MAF could produce this problem.

We compared our approach to Pearson's Chi-square test and BEAM, which degrades to Pearson's test in cases of no epistasis. Thus, we expect these algorithms to have very similar performance without epistasis, which is confirmed in the original paper and our own simulations.

Consider an additive model like the one described in Table 1. At a low MAF, the high effect entries, especially the one with all four minor alleles, are rarely present, even at a 50/50 case/control ratio. You might only expect 40 with those rare alleles. This will be dwarfed by the number of individuals with other combinations. Different single-locus statistics can interpret deviations from the null hypothesis differently when comparing the rare and common groups. Pearson's chi-square test assumes a chi-square distribution and the calculation of the statistic is additive between multiple events (genotypes). Our Gini impurity is just a comparison of class frequencies. Then the decision to choose a variable is based on the weighted average of the three child nodes' impurity. This causes the Gini impurity calculation to have different characteristics than
a chi-square based test, leading to differing performance as compared to the chi-square test with respect to varying allele frequencies. Thus, modifying this calculation to remove this bias could improve performance. Alternatively, a different split characteristic, such as information gain, might have more applicable properties. It is important to note that this difference between Gini impurity and chi-square is fundamentally a single-locus effect, and therefore should be independent from of approach's ability to detect epistasis.

7.2. High Variance Partitioning (HVP)

High Variance Partitioning (HVP) is a general technique that tries to optimize the task of bootstrap resampling for detecting epistasis effects. HVP optimizes by creating bootstrap samples that are very different from each other, but still based on the idea of a random sample. Since variance in weak learners can be achieved by variance in bootstrap samples, this can be an effective approach to increasing power. In some initial testing, we observed a rudimentary HVP algorithm that can sometimes achieve similar power in 9 half-trees (equivalent in time to 18 normal trees) as an Adaboost learner with 200 trees. In section 7.2.1 I will describe the HVP problem and a simple HVP algorithm. Then, in section 7.2.2 I will discuss how this transforms our view of marginal effects.

7.2.1. The HVP Problem Statement

Consider all the partitions of a set \( N \) of size \( N \). There are \( B_N \) of these partitions, where \( B_i \) is the Bell number. We want to select partitions that are "different" from other
partitions, e.g., to have lots of varied partitions. Additionally, we want partitions of roughly equal size. So, to begin with we will only consider partitions $p$ of the set into two. The number of such partitions is given by the $B_{N,2}$ Bell Polynomial, Specifically, the $x_{N/2}^2$ term. This is $\binom{N}{N/2}/2$.

**Distance Between Two Partitions**

Define the distance $\delta$ between two partitions $a, b$ ($|b| = |a|$) as follows: For each group $b_i$ in $b$, map it to a group $a_i$ such that for all groups $a_i$, the total number of elements that differ between $a_i$ and its partner is minimized.

For example, consider the following:

$a = \{(1,3,5,...), \{2,4,6,...\}\}$

$b = \{(1,2,5,6,9,10,...), \{3,4,7,8,11,12,...\}\}$

Here, we can match $a_0$ with $b_0$ or $b_1$. Assuming it maps to $b_0$, exactly half of the elements of $b_0$ are in $a_0$ (the integers $1$ mod 4). Similarly, half of the elements of $b_1$ are in $a_1$ (the integers $0$ mod 4). Since each $b_0$ and $b_1$ is of size $N/2$, and half of the elements are different, there are $N/2$ total differences; therefore $\delta(a,b) = N/2$. Note that $N/2$ is the maximum difference of two infinite partitions due to the minimization mentioned above.

**Maximizing Differences Between Half Partitions**
Now, we can pose some questions about partitioning in half:

1. How to select $k$ half-partitions such that the total distance between all pairs of partitions is maximized ($\max \delta_{N,k}$).

2. What is an algorithm that takes $k$ chosen half-partitions, and finds another partition $p$ such that the total distance between all pairs of the set of the chosen partitions $+p$ is minimized.

One notices that question 2 sounds a lot like a greedy formulation to this problem. 1 is really hard in the sense that there are $\binom{N/2}{k}$ ways to do that, so brute forcing them will not work.

Here is an algorithm that builds partitions which satisfy questions 1 and 2 (sort of) when $k < \log_2(N)$:

```python
for skip in (2**kk for kk in range(k)):
    partition = {0:[], 1:[]}
    for n in range(N):
        t = int(n/skip) % 2
        partition[t].append(n)
```

The intuition is pretty simple. First, pick every other element to be in $a$. Then, the elements $0, 1 \mod 4$. Then, the elements $0, 1, 2, 3 \mod 8$, etc. Each of these sets is has a
difference of \( N/2 \) to each other, meaning the average pair-wise distance is maximized at \( N/2 \).

7.2.2. **How HVP algorithms transform the marginal effect**

Recall that our method detects interaction effects in two ways: through recursive partitioning in decision trees and bootstrap resampling in Adaboost. HVP algorithms attempt to "optimize" the bootstrap resampling not by changing the underlying distribution of the bootstrap sample, but by choosing high variance samples. Rather than slowly searching the whole space around a traditional bootstrap sample (weighted by probability), we quickly hit many representative samples. This makes HVP algorithms potentially much faster than Adaboost and could increase their power in a permutation test, since fewer noise trees would be present.

In a disease with high epistasis, such as the one described by our two-locus epistasis model with MAF 0.5, the original distribution of the data should not have any detectable marginal effect. Thus, the Gini impurity of a split will be roughly the impurity of the current distribution, e.g. 0.5. Thus, a split will never be made on the disease locus! Bootstrap resampling helps reduce this problem by providing many similar distributions near the original data. By tweaking the data slightly like this, the perfect balance of epistasis can be upset, and therefore a marginal effect is again present. Thus, in this way resampling can help single-locus based Gini impurity decision trees detect an epistasis effect. Recursive partitioning in the trees allows a powerful weak learner to be
grown once one of the two disease loci has been selected (since it effectively conditions on features after selecting them).

High Variance Partitioning allows us to create lots of half-sized partitions that are very different from each other while still simulating a normal resample. Instead of each resampling being independent, it deliberately introduces a dependency such that samples that are too similar to something we have already examined will not have time wasted on them. How much is too much is open to debate, and it seems possible that a HVP approach might decay to be equivalent to normal bootstrap resampling as the number of samples approaches infinity.

7.2.3. Developing a robust HVP algorithm

The simple HVP algorithm presented in section 7.2.1 above has at least two large drawbacks.

First, it can only find log(N) partitions. It can be shown that for some N this is very inefficient. For example, for N that is a power of two, we have created an algorithm that finds N-1 partitions that have N/2 average pair-wise distance from each other. This can be done by assuming the following set of partitions for N=8:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td></td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td></td>
<td>4</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td></td>
<td>3</td>
<td></td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 16: A HVP set of partitions for N=8 that has 7 partitions into two all of N/2 average pair-wise distance

To generate a table of size N=16, duplicate rows 1-7, and fill in the new columns 8-15 with first a replica of the column in that row in the original, and the inverse pattern in the duplicated row. This gives us \((N-1)*2 = 14\) partitions. The 15th is just selecting the entire first half of the data. This approach works for any N that is a power of 2. We are unsure if this approach is even optimal, but it is surely much better than the naive approach. Unfortunately, extension to non-power of \(2^N\) is difficult. Because the table cannot always have the necessary symmetry (this approach effectively just splits recursive partitions in half), the result will be sub-optimal. It seems that a different algorithm would be needed for other N. Even worse, since the space of possible partitions grows like a double factorial, computing optimal ones might be computationally prohibitive (of course, once found a partition set is applicable to any data set with that N).

<table>
<thead>
<tr>
<th>Greedy number of partitions</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>12</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

Table 17: Greedy HVP approach vs. number of partitions at optimal pair-wise average distance

Table 17 shows the performance of a greedy HVP algorithm. This algorithm works by enumerating all half-partitions, picking one, and then continuing to pick more based on which remaining partitions are the maximum distance from the first. It terminates when no more partitions at that distance remain. Note that despite being greedy, this algorithm still has factorial runtime (because it must enumerate all the half-partitions).
This table shows how the maximum number of partitions has odd behavior based on numbers: 6 has far more than normal because each half partition in that set is only of distance 1 from the others (since 1.5 cannot be achieved). Other numbers also have odd behavior (especially primes). Thus, to extend the algorithm above, one would need to deal with this and find out how to solve the problem for prime N.

The second drawback of the log(N) algorithm is that it assumes we need optimality. In reality, we want these partitions to guide our machine learning approach. Thus, a set of partitions that has average pair-wise distance almost at the maximum is roughly as good as an equivalently sized optimal average pair-wise distance one. Furthermore, we might be able to find exceptionally more partitions that are reasonably different if we examine partitions that are highly variant but not optimally variant. This could allow broader weak learners to be grown and further increase algorithm power.

7.3. **Alternative Scoring Functions For Adaboost**

Recall our Adaboost feature importance scoring function described in section 3.4. It was meant as a straightforward adaptation of a similar method by Breiman in the context of CART\textsuperscript{12}. There are several obvious problems and improvements that present themselves.

First, the current scoring function could be conservative with respect to controlling false positive rate. This is because of the competitive nature of our scoring. By being selected, a feature takes a potential slot away from the rest of the features, because the total number of splits is constant (modulo algorithm parameters). Thus, if there is a
strong signal in one feature, control of the false positive rate will be conservative for the rest of the features.

Second, the given scoring function ranks tree weight equally with Gini gain. This is intended to improve the score for SNPs that do not make great splits, but do enable a conditional distribution which does. However, it could be that this places too much importance on the tree’s score over the SNP’s score (or vice-versa).

7.4. Adaboost Selection in two-stage algorithms

As noted in section 2.2, a two-stage approach using a single-locus screen followed by a higher-order algorithm is a classical approach to dealing with the difficulties of GWAS. We would also propose that our approach could be useful as an alternative first stage. Our algorithm is fast enough to be run trivially on GWAS data (overnight on a desktop PC). Additionally, it captures single-locus effects while potentially including interaction effects. Thus, it has the potential to out-perform a purely single-locus screen such as Pearson's chi-square test. For example, consider the TSym simulation study with MAF of 0.3 and theta of 2. In this test, no approach can detect all three loci significant in any replicate; the power is zero. However, if you use our approach as a screen, take the top 50 SNPs by score, and then do a 50 versus rest two-locus test, the power becomes 95% to detect all three while still controlling type-I error.

BEAM can also operate in a Bayesian sense and be used in this manner. Future work could compare classical chi-square, our approach, and a low iteration BEAM algorithm as a screen for a more highly-interacting model. Due to our ability to declare epistatic
SNPs as significant more often than chi-square, we hypothesize that it would also make a more effective screen than other proposed approaches. Using a high variance partitioning algorithm in this manner could even achieve similar performance to our Adaboost approach while running at speeds within an order of magnitude of Pearson's chi-square test.

7.5. Including Expert Knowledge

One of the difficulties of GWAS is due to the lack of prior knowledge about how SNPs may interact. Thus, we are forced to consider a huge feature space on limited data, which limits how effective or efficient algorithms can be. This section will explore ways to use various types of expert knowledge to make the problem easier. BEAM's Bayesian formulation gives an obvious example of how to include expert knowledge. We can adjust the prior probabilities to give the algorithm a potentially more efficient start. In the extreme case, we might only consider the interaction effects between an exceptionally small group of SNPs (e.g., 4).\textsuperscript{30}

One potential way to reduce the problem space would be to use a protein-protein interaction network to reduce the number of potential SNP interactions. Instead of considering each SNP individually interacting with all other SNPs, we partition the SNPs into groups based on genes. Then, we can use the protein-protein interaction network to get SNP pairs to evaluate for further interaction effects. This should greatly reduce the problem space. The main problems with this method are that not all SNPs might be near a gene, the gene might be unknown, or the gene-gene interaction effect might not
be known. Then the reduction would preclude the possibility of finding the correct answer. Nevertheless, if there are truly diseases with more than three disease loci in heavy epistasis, this kind of heuristic will be vital in making blind epistasis mapping feasible.
Bibliography


27. Culverhouse R, Suarez BK, Lin J, Reich T. A Perspective on Epistasis: Limits of

