Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Miko Kari

A Parallel Forward Selection Wrapper for Genome Wide Association Studies

Master's Thesis
Espoo, June 8, 2016

Supervisor: Professor Harri Lähdesmäki, Aalto University
Advisor: Lu Cheng Ph.D.

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF MASTER'S THESIS

Author:	Miko Kari		
Title: A Parallel Forward Selection Wrapper for Genome Wide Association Studies			
Date:	June 8, 2016	**Pages:**	68
Major:	Information and Computer Science	**Code:**	T-61
Supervisor:	Professor Harri Lähdesmäki		
Advisor:	Lu Cheng Ph.D.		

Genome wide association studies attempt to explain variations in the observed traits of organisms in terms of variations in their DNA. Many complex human diseases are believed to be associated with interactions of these single point variations within the genome. Moreso, recent research suggests that many diseases are likely to be caused by rare mutations. This demands the scanning of the entire genome as opposed to the continued scrutiny of its commonly assayed regions.

With the declining cost of whole genome sequencing, the amount of high dimensional data available to genome wide association studies can be expected to rise rapidly. At the same time, many formerly used analysis techniques are starting to show signs of weakness and new, more powerful algorithmic solutions are needed to analyze these larger data sets.

Feature selection techniques constitute a methodology that, when applied, can alleviate the computational burden faced by the analysis tools. More importantly, they can help discover the genetic markers that are most strongly associated with a phenotype and help direct future research effort to the further study of those particular factors.

This thesis presents a novel feature selection technique that scales well to high dimensional feature spaces. The method is a forward selection type wrapper that operates search paths in parallel and involves a heuristic to reduce the computational load of searching for the optimal feature subset.

The results suggest that the proposed method is better than the tested alternative standard feature selection techniques in the analysis of genetic variants and data with small concentrations of relevant features. Furthermore, using a linear regression model and the novel speedup heuristic, the parallelizable feature selection method scales to the genome wide scale given appropriate computational resources.

Keywords: parallel, feature selection, wrapper, SNP, GWAS

Language: English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Miko Kari		
Työn nimi: Monipolkuinen eteenpäin suuntautuva piirrevalintakääre genominlaajuiselle assosiaatioanalyysille			
Päiväys:	8. kesäkuuta 2016	**Sivumäärä:**	68
Pääaine:	Tietojenkäsittelytiede	**Koodi:**	T-61
Valvoja:	Professori Harri Lähdesmäki		
Ohjaaja:	Filosofian tohtori Lu Cheng		

Genominlaajuinen assosiaatioanalyysi tutkii eliöiden genomissa esiintyvien pistemutaatioiden ja havaittavien piirteiden välistä yhteyttä. Näiden yhden emäsparin pistemutaatioiden sekä niiden vuorovaikutusten uskotaan olevan yhteydessä useisiin sairauksiin. Nykytutkimus osoittaa lisäksi useiden sairauksien olevan pikemminkin harvinaisten kuin tavanomaisesti tutkittujen yleisten mutaatioiden aiheuttamia, mikä edellyttää laajempaa analyysiä myös genomin vähemmän tutkituilla alueilla.

Genominlaajuisen sekvensoinnin yleistyessä korkeadimensioisen geneettisen datan tarjonnan voidaan olettaa kasvavan räjähdysmäisesti, mikä puolestaan edellyttää uusien tehokkaiden analyysimenetelmien kehittämistä. Nykyiset laajalti käytetyt menetelemät ovat nimittäin yleisesti liian tehottomia koko genomin laajuisen analyysin suorittamiseksi.

Piirrevalintamenetelmät auttavat keventämään analyysityökalujen laskentataakkaa karsimalla epäolennaiset muuttujat datasta. Lisäksi ne johtavat merkittävien muuttujien löytymiseen ja mahdollistavat kansainvälisten tutkimusresurssien ohjaamisen niiden tarkempaan jatkotutkimukseen.

Tämä työ esittelee uuden kääretyyppisen piirrevalintamenetelmän, joka skaalautuu hyvin korkeadimensioisen datan käsittelyyn. Esitetty menetelmä on eteenpäin suuntautuva, piirteitä iteratiivisesti osajoukkoon lisäävä kääre, joka mahdollistaa useamman etsintäpolun ja hyödyntää laskentataakkaa keventävää heuristista ratkaisua.

Työn tulokset viittaavat siihen, että tämä uusi menetelmä on sen kanssa vertailtuja tunnettuja menetelmiä parempi korkeadimensioisen ja erityisesti suuria merkityksettömien muuttujien pitoisuuksia sisältävän datan käsittelyssä. Nopeutusheuristiikkaa hyödyntäessään tämä rinnakkaistettava menetelmä skaalautuu myös genomin laajuiseen tutkimukseen.

Asiasanat:	rinnakkainen, piirrevalinta, kääre, monimuotoisuus, genomi, assosiaatio
Kieli:	Englanti

Acknowledgements

I acknowledge the computational resources provided by the Aalto Science-IT project. This thesis involved computations that would have been unworkable without the use of a computer cluster.

Furthermore, I would like to extend my gratitude to my supervisor Professor Harri Lähdesmäki for this interesting research topic. Thanks also to my colleagues and advisor for their time.

Espoo, June 8, 2016

Miko Kari

Contents

Chapter 1

Introduction

Genetic factors play a role in the onset of many human diseases affecting people worldwide. Some genetic disorders, such as sickle cell anemia and cystic fibrosis, are directly caused by mutations in certain genes. Meanwhile, other mutations exist that do not directly cause a disease but are considered risk factors that increase an individual's susceptibility to a given disease. A branch of genetics has evolved to study these risk factors by means of computing levels of association between these point mutations and a given disease expression.

These risk factors predispose individuals to diseases of the kinds of diabetes, rheumatoid arthritis and many neurological disorders. Through decades, large parts of the human genome have been searched for variation that could explain the emergence of such common diseases. Still, much of this study has concentrated on the analysis of commonly occurring variants. This has been partly due to the hypothesis that common diseases should be caused by common variants and partly due to the expense of collecting genotypic data for rare mutations. While common variants have been able to explain the development of certain diseases, research indicates that many of the studied diseases could be the result of interaction among rare mutations occurring throughout the genome. These rare mutations then need to be first identified before they can be tested for association with any given disease phenotype, which is a physical trait that appears as a result of interaction between an organism's genes and the environment.

To discover these rare mutations, the whole genome of a species needs to be scanned which results in extremely high dimensional data. This type of data collection is currently made possible by the economically feasible genome wide sequencing technology. The only drawback is that with the increase in the dimensionality of the data, many of the tools and algorithms commonly used in genome wide association studies are showing signs of debility. Hence, it is becoming clear that newer, more powerful algorithms are needed to process the large data sets before conventional analysis techniques can be successfully applied.

This underlines the importance of feature selection, which is a methodology concerned with the elimination of irrelevant features from data. Thus, this thesis introduces a novel feature selection method for processing high dimensional genetic data. The presented approach is aimed to be a general purpose tool that can be run with any classification or regression model to identify relevant genetic variants. This way no assumptions are made regarding the underlying structure of the genetic interactions, which allows for the detection of a wider spectrum of SNP interactions beyond the previously heavily studied linear associations. The found relevant variants can then be subjected to further research by the scientific community.

The proposed feature selection method is based on existing methods but attempts to give more robust results than its antecedents. Additionally, a new heuristic has been tailored specially for the proposed feature selection method to ease its scaling to high, genome wide feature spaces. The results of this work will show that the proposed feature selection algorithm works better than the tested, standard feature selection techniques in the analysis of high dimensional genetic data. More specifically, the proposed method excels in the identification of the central features and converges faster than the other tested wrapper type method.

The structure of the thesis is as follows. Chapter 2 gives an overview of the topics relevant to the biological setting and the proposed feature selection algorithm. These include the essential concepts and practices of bioinformatics and machine learning, as well as the review of several common modeling algorithms. After this theory driven segment, Chapter 3 delves into the practical realities of research in the field of genetics. The issues and challenges are highlighted to motivate the development of a scalable feature selection technique. Then, Chapter 4 presents the proposed feature selection method. The chapter describes in detail the assumptions, arguments and tools employed in its design. Next, Chapter 5 briefly describes the implementation of the proposed method before presenting the results of experiments assessing its performance and comparing it to rival feature selection methods. Chapter 6 discusses some of the relevant implications of the performed research. Finally, in closing, Chapter 7 recapitulates the findings of this work.

Chapter 2

Background

This chapter presents theories and concepts which the later sections build on. The first section will briefly discuss the biological background on genetic association studies. The following sections then present the necessary data analysis techniques and models used with the proposed feature selection method.

2.1 Genome Wide Association Study

Genome Wide Association Study (GWAS) examines the associations of genotypes with observed phenotypes. Common phenotypes, such as the expression of certain diseases, are attempted to be explained by single point mutations in DNA, called *Single Nucleotide Polymorphisms* (SNPs). These SNPs are variations in a single base pair in the genome of a certain organism that occur at a particular minimum frequency in the population. Hence, not all variants are counted as SNPs but only those that have sufficient prevalence to be interesting predictors of common diseases. Consequently, any estimate on the number of SNPs in, for instance, the human genome is conditional on the used threshold value for the minimum frequency. For the commonly used 1% threshold the expected number of human SNPs is approximately 11.0 million (Kruglyak and Nickerson, 2001). Single nucleotide polymorphisms are believed to be significant factors in explaining an organism's predisposition to genetic disease, although the role of single SNPs in disease modeling functions is largely uncertain.

In the past, typical GWAS studies have concentrated on the identification of common SNPs with small effect-sizes but recently more emphasis has been placed on the analysis of the previously less explored rare SNPs with larger effect-sizes (Liu and Leal, 2010). While most common diseases are explained to some extent by common variants, the possible role of rare variants still requires assessment as it cannot be stated that the genetics of common diseases are affected just by common alleles (Bush and Moore, 2012). Moreover, the number of rare variants in the human genome is estimated to be significantly larger than that of common variants (Tennessen et al., 2012). Hence it is difficult to analyze their associations with complex disease phenotypes unless more powerful algorithms are developed that can scale to the identification and extraction task in these extremely high dimensional feature spaces. The following sections will thus move on to discuss currently available feature selection methods that serve as a starting point for resolving this issue.

2.2 Feature Selection

Feature selection techniques reduce the dimensionality of data by selecting a subset of its original features according to some criteria. Unlike other dimensionality reduction techniques that project or transform data to lower dimensional feature spaces, feature selection techniques do not affect the interpretability of the data. This interpretability is especially important in genetic studies where the identification of the actual causal genetic factors matters.

Generally the motivation for using feature selection includes (1) the improved model prediction accuracy on test data that results from the elimination of irrelevant features, (2) the reduced computational costs of using a simpler model trained on just the relevant features, and (3) the heightened understanding of the underlying process obtained through the more accurate and simpler models. In genetic association studies, the motivation for performing feature selection is particularly the identification of causal factors in disease mechanisms and the obtained clearer understanding of the disease mechanisms.

One commonly used taxonomy on feature selection techniques is their division into filter, wrapper, and embedded techniques according to their interaction with classification or regression models. The following sections will discuss these three types, starting with the commonly used simple and quick filter techniques.

2.2.1 Filter Type Feature Selection

The class of simpler selection techniques — filter methods — are popular for their speed and ease-of-use. These techniques rank available features according to an evaluation criterion so that a chosen number of the highest scoring features can be selected for modeling. Information gain, χ^2, and correlation-based feature selection are some filter techniques used in bioinformatics (Saeys et al., 2007).

The robustness and reliability of feature selection techniques are of major concern and they apply especially to filter techniques that are commonly characterized by their one-sided reliance on a single metric in the ranking of features. This concern has lead to the development of so called ensemble methods that combine the results of several different filters. Seeing that a single given filter may not perform well in all different situations, the combined results of several filters should provide more trustworthy feature selections.

Nevertheless, one notable characteristic of filter techniques is their independence of the classifier or regression model that is later used to analyze the reduced data. For this reason, the other methods of wrapper and embedded techniques tend to be favored when the optimality of the feature subset to the used classifier or regression model is concerned.

2.2.2 Wrapper Type Feature Selection

Wrapper methods are ad hoc approaches where feature selection is made on the basis of feedback from a chosen classifier, in the case of classification problems, or a regression model, in the case of regression studies (later both referred to as *classifiers*). In these methods, several different candidate feature subsets are considered in turn and a given preselected classifier is trained and tested on their features. Feature subsets are then ranked and considered based on scores received on the output of the classifier. This type of selection tends to require considerably greater computational resources than filter techniques because wrappers fit entire models to these subsets with every evaluation. However, due to this setup, the obtained feature subset can be considered optimal for the particular classifier.

The main issue for wrapper techniques is that the number of feature subsets grows factorially with the size of the feature subset according to the formula for binomial coefficients $\binom{n}{k}$, where k is the size of the subset and n is the total number of features. When larger feature subsets are considered, their numbers quickly become overwhelming. This computational complexity may prevent wrappers from considering all possible candidate subsets in high dimensional feature spaces and they may consequently fail to identify many relevant feature subsets. While heuristics are used to limit the numbers of considered subsets, the elimination of the least promising features altogether would greatly reduce the workload of the wrapper procedure with little if any deterioration in the quality of results.

With that said, due to the noted advantages of filter methods, their use is well founded as a preliminary feature elimination process in collaboration with wrapper type feature selection. Namely, the considerable speed at which filter techniques perform their selection makes them an ideal initial pruning step to eliminate the least important features so that the more laborious wrapper process can concentrate on the relevant features. The only condition is that the applied filter is expressly prudent in the removal of features and only eliminates those features that show the strongest signs of being redundant or irrelevant.

Wrapper techniques are a central focus of this thesis as the main objective is to develop a novel wrapper technique that is scalable to the genome wide scale and provides robust, near optimal results. Hence, the following subsections will delve deeper into the subject matter by introducing a few known wrappers on which the proposed method is based. The first example is the *forward selection* method.

2.2.2.1 Forward Selection

Forward selection is a wrapper method that starts with an empty set and adds features to it iteratively until the performance score of the used classifier starts to decline. The method is a greedy search algorithm in that it always amends the feature subset with the feature that yields the highest validation score from the classifier. However, the algorithm only advances in one direction and is sensitive to local optima. That is, it is liable to select features that initially improve the validation score but that do not necessarily lead to the globally optimal solution of a feature subset. This form of feature selection favors those features that perform well initially with few selected features and those that perform well later on with the already selected features.

One attempt to remedy this bias towards certain features and the sensitivity to local optima is to introduce a backtracking feature to the algorithm. This approach incorporates an additional step that allows the algorithm to remove one feature at the end of an iteration given it is beneficial. The advantage of backtracking over a separate pruning stage at the end of the forward selection process is that it can save considerable computational resources by avoiding search paths that prove unpromising early on.

Nevertheless, forward selection is suited to those cases where a few features are to be selected from a large feature set. On the other hand, a *backward elimination* method is a more appropriate choice if the data is likely to contain only few irrelevant features.

2.2.2.2 Backward Elimination

Backward elimination is the reverse process of forward selection in that here one starts with a full set of features, iteratively evaluates the relevance of each feature and considers the removal of the least relevant feature. The removal is judged against some predetermined threshold value and the process is continued until a predefined number of features have been deleted or once the further elimination of features would cause the performance of the classifier to degrade.

While forward search suffers from a bias towards strong, independent features, backward elimination may lead to the removal of relevant features with small effect sizes. Still, other selection methods exist that are capable of moving in both directions. One of these is the *forward backward search.*

2.2.2.3 Forward Backward Search

While forward selection adds features and backward elimination removes them one at a time, a so called forward-backward search method considers all available one-bit changes to its current binary vector of selected features at every iteration. It therefore evaluates all potential removals and selections of one feature for its current set and moves to the direction that gives the best improvement on the score from the classifier.

While this type of selection is more flexible than the two previous methods, it may be slow to converge to a solution and may end up performing cyclical search patterns. Yet one last wrapper, *genetic algorithms*, will be discussed in the following section. This method advances in both search directions but does so randomly while imitating natural evolution.

2.2.2.4 Genetic Algorithms

As described by Siedlecki and Sklansky (1989), genetic algorithms are feature selection techniques that employ a population of solutions. Each of these solutions is a candidate feature subset, called a chromosome, that evolves through mutations and cross-overs with the other chromosomes, mimicking natural evolution. In crossover, the chromosomes in the population form pairs that produce two new chromosomes that inherit different parts of their parents, whereas mutations randomly change one or more bits in a chromosome. Additionally, these events only take place at given probabilities determined by a fitness function and a mutation rate.

Genetic algorithms involve the update of the population through generations and the process is continued either for a predetermined number of these generations or until the quality of results is satisfactory. The new offspring will either replace the previous generation or be combined with it depending on the way the population size is maintained. However, being a nondeterministic algorithm, there is no guarantee of the optimality of the obtained solutions (Pudil et al., 1994) and as such this type of feature selection is not suitable for all use cases.

With the overview of these wrappers, it is time to look at the last class of feature selection algorithms, the embedded techniques. These are generally preferable to wrappers as they have the same positive qualities as wrapper techniques but are commonly more efficient.

2.2.3 Embedded Type Feature Selection

Embedded feature selection techniques are different from wrappers in that they have access to the classifier's internal variables. Thus, embedded feature selection methods can base their feature selection on values computed by the classifier and they do not have to compute additional, external metrics to assess the relative importance of the features. For this reason, embedded methods tend to show a reduced time complexity compared to similar wrapper methods.

Regularization methods, such as LASSO, are a common example of embedded methods. These methods penalize complex models by placing different constraints on their parameters, effectively leading to sparse models where only a subset of the original features participate in prediction tasks. From a feature selection perspective this means that the classifier conducts its own, internal feature importance assessment and selection.

This concludes the discussion on the different classes of feature selection techniques. The following section moves on to present some basic concepts of machine learning. These include the central concepts of model selection, data partitioning, and cross validation.

2.3 Model Selection

The central idea in machine learning is to present data to a learning algorithm that can learn the underlying model behind the data and then use that model to make predictions on other related data for some value of interest. There are two kinds of learning cases: supervised and unsupervised learning.

In supervised learning, we have training data for which the target classification label or regression response variable value are known and this data is used to fit a particular model. In contrast, unsupervised learning involves the clustering of training samples into clusters based on their relative similarities and the clusters are then assigned class labels. The objective in both learning cases is to classify future data samples into their correct classes or to obtain their target response values via regression analysis. Unsupervised learning is, however, out of the scope of this thesis since the target phenotype is always known in GWAS data. Unsupervised learning will hence not be discussed further and any later discussion on learning algorithms refers to supervised learning.

Model selection in supervised learning is the process of considering different candidate models and testing them on data to find the model that best reproduces the studied class labels or response values. Here supervised learning algorithms provide straightforward ways of scoring learning algorithms and selected feature subsets via the comparison of predicted class labels or response values to their true, known labels and responses.

More specifically, in model selection the data is first divided into a training and a test set. The training set is used to train the candidate models while the test set is used to test the predictive performance of each model. This testing is achieved by calculating each test sample's predicted class label or response value with a trained model and then measuring the deviation of these predictions from the samples' true, known class labels or target response values. The model whose predictions deviate the least from the ground truth is considered the best model and is selected. Finally, before making predictions on new data, the selected model is retrained using both the original training and test data sets. This, however, is not the entire picture and the following subsections will move on to discuss some general best practices relating to model selection, namely the concepts of data partitioning and cross validation.

2.3.1 Data Partitioning

When training a model for prediction tasks, the performance of the trained model needs to be evaluated in order to have a sensible measure of the accuracy of the predictions. One cannot simply use the entire available data both for training a model and testing its performance since this way the obtained performance score would be over-optimistic. Namely, the fitted model is likely to perform better with the training data than on completely new data. The obtained score would then serve no purpose in reporting the classifier's expected performance on future, unseen data.

Thus, the data available for supervised prediction tasks is often divided first into a training set and a test set so that a determined portion, for instance 75 %, of the available samples are placed in the training set and the remaining form the test set. This way the model whose prediction performance we wish to estimate can be trained on the training set and tested on the test set. The testing is conducted by predicting the test set's labels using the trained model and comparing them to the known correct labels of the test set. However, this type of data partitioning is not enough if one wishes to perform any model selection, such as the tuning of the classifier's parameters. In these cases the training set samples are further divided into training and validation folds according to the chosen *cross validation* approach, the concept of which is the topic of the following section.

2.3.2 Cross Validation

In cross validation the training data samples are divided into groups, called *folds*, and each fold serves in turn as the validation set while the rest of the folds are used for training the classifier. The results on the validation fold are then scored and the scores over the different validation sets averaged to get the overall validation score for the given model.

Cross validation is commonly used in model selection to avoid selecting a model that over-fits to the training data or when the amount of training data is small compared to the complexity of the model. Over-fitting refers to the condition where learning results in a complex model that fits well to the training data but fails to generalize and performs poorly on other data. In the first case, cross validation ensures that the parameters of the resulting model are selected so as to optimize the model's ability to generalize over the different partitions of the training data instead of learning the specific structure of the entire training set. In the second case, cross validation improves the performance of the trained model by releasing more of the training data for use in the training stage; instead of detaching a fixed portion of the training data to serve in model validation, all of the samples will be used to train the model over the course of the routine. This will help make maximal use of the scant training samples, leading to more accurately learned models.

One commonly used cross validation scheme is k-fold cross validation, in which the training data is first divided into k folds and the model is trained and validated k-times so that $k-1$ folds serve as the training set and the remaining fold is used for validation. The sought after model selection score is then the average of these cross validation scores.

However, when feature selection is paired with model selection, the aforementioned cross validation routine is no longer sufficient to ensure the obtained validation scores are objective and that the model, and consequently the features, have been selected properly. In these cases one must use *nested* *cross validation.* In nested cross validation, the data is first divided into training and validation folds as described above, but each training fold is further divided into training and validation folds to first select the most appropriate hyperparameter values for that particular fold. Each outer validation fold then ends up testing the candidate feature subset on a classifier specifically tuned for that fold. This ensures that the used hyperparameter values do not favor any particular features but have been selected objectively based on the data.

This encapsulates the central machine learning concepts referred to in the later sections of this thesis. The following section will now go over the class of linear classifiers and describe some of those relevant to this work.

2.4 Linear Models

This section starts off by presenting the general linear model and reviewing the concepts of regularization, sparsity and mixed models, which are essential to the understanding of the more advanced linear methods presented at the end of this section. The relevant advanced methods include the regularized linear model, LASSO, and the Bayesian sparse linear mixture model, BSLMM. LASSO, being a commonly used method, and BSLMM, for its complexity, are ideal candidates for benchmarking the proposed feature selection method.

2.4.1 Linear Regression Model

Linear regression is a method used to estimate a linear relationship between a dependent variable, also called a response variable, and one or more independent variables, also known as predictors. Univariate models involve one predictor for the predicted response, whereas multivariate models have several predictors. The linear model is of the form

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + \varepsilon \tag{2.1}$$

where y is the dependent variable, the x_i for $i \in \{1, ..., k\}$ are the independent variables, β_j for $j \in \{0, ..., k\}$ are the regression coefficients and ϵ is the error term representing the variation not captured by the regression terms.

The model that best fits to the given set of data consisting of the dependent and independent variables is often determined according to the minimization of the sum of square differences of the true target values and their estimates obtained from the fitted model. By the least square objective function, the estimates for the regression coefficients are obtained from the following optimization task:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{argmin} \; \|\mathbf{y} - \mathbf{X}^T\boldsymbol{\beta}\|_2^2 \; , \tag{2.2}$$

where $\mathbf{y}$ is a vector of the true values of the response variable, $\mathbf{X}$ is a matrix containing the values of independent variables on the rows for each observation, $\boldsymbol{\beta}$ is a vector consisting of the regression coefficients, and $\|\cdot\|_2^2$ is the square of the L2 norm. The above optimization task can be solved, for instance, using differentiation on the cost function and solving the resulting system of partial differential equations to obtain estimates for the regression coefficients.

In the above linear model, the error term is there to capture any and all of the variation that is not explained by the actual regression terms. However, in the presence of repeat measurements from the same subjects or groups of subjects, the regular linear model can no longer sufficiently explain the variation between the subjects because much of this variation is the result of individual or group specific characteristics and not the direct result of the independent variables. For instance, data from different sample cohorts or from different sources may contain group specific variation or systematic measurement errors that cannot be explained by a regular linear model's fixed effect terms. Instead, this variation could only be accounted for as random error and would increase the magnitude of the reported ε term.

However, instead of having the model explain all of this apparently random variation with the random error term ε, we can extract a random effect term from it to model the group or subject specific mean. In fact, one of the key assumptions of a linear models is inter-sample independence, which fails to actualize in the presence of repeat measurements. By introducing a random effect term, some of the random variation can be explained as cohort or subject specific variation. Consequently the samples can once again be considered independent and the fixed effect terms are better able to explain the inter-sample variation as deviations from each sample's or group's respective mean.

So, in summary, mixed models incorporate random effects in addition to fixed effects found in the linear model of Equation 2.1. The linear model in its mixed form with one random effect term would look like

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_k x_k + b_1 z_1 + \varepsilon \; . \tag{2.3}$$

Here a random effect term z_1 with its own coefficient b_1 has been extracted from the error term ε. The random effect term z_1 here can be used to model the random effects of some hierarchical variable. Overall linear models may work well in regression problems with many relevant features. However, a linear model will attempt to fit all independent variables into the model and sometimes there are only few relevant features in the data. The following section presents a sparse regression algorithm that regulates model complexity with sparsity favoring constraints.

2.4.2 Regularized Models and the LASSO

Sparse models involve only a fraction of the original features present in the available data. The use of the sparse models assumes that a given response variable can be successfully explained by a model with only few parameters. LASSO, or the Least Absolute Shrinkage and Selection Operator, is a linear regression method that uses the L1-norm of the regression weights to penalize large regression weights and to induce sparsity. The regression coefficient estimators can be obtained from the objective function

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \; \frac{1}{2N}\|\mathbf{y} - \mathbf{X}^T\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{2.4}$$

where the symbols are the same as above in Equation 2.2 and $\|\cdot\|_1$ is the L1 norm on the regression coefficients with the non-negative λ being the regularization parameter. With $\lambda = 0$ the objective function reduces back to the ordinary least squares optimization task seen in Equation 2.2 while larger λ values result in internal feature selection through the penalization of non-zero regression weights. In the following section, sparsity and mixed models are combined with the Bayesian framework to yield a more adaptive GWAS algorithm.

2.4.3 Bayesian Inference and the BSLMM

Bayesian inference is a statistical method where a posterior distribution of model parameter values is inferred from a prior distribution of those same parameters and the distribution of some observations conditional on the model parameters. In other words, it is a technique that incorporates prior information and evidence to obtain an updated distribution of model parameters. These concepts revolve around the Bayes' formula

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)} \; , \tag{2.5}$$

where $P(\theta|X)$ represents the *posterior probability* after observing samples X, $P(X|\theta)$ is the *likelihood* of the data given some model parameters θ, $P(\theta)$ is the *prior* distribution of the model parameters, and $P(X)$ is a scaling term called the *evidence*. Often, the term $P(X)$ is skipped as a scaling term and thus among posterior probabilities concerning the same observed values X, Equation 2.5 reduces to

$$P(\theta|X) \propto P(X|\theta)P(\theta) \ , \tag{2.6}$$

where it is enough to evaluate the product of likelihood and prior to obtain the updated posterior distribution. The objective with Bayesian inference is to learn the distribution of model parameters from data. There are two alternative point estimates for the model parameters: the *Maximum a Posteriori* and the *Maximum Likelihood Estimate*.

2.4.3.1 Parameter Estimation

The model parameters for the likelihood distribution can be estimated from data using the *Maximum Likelihood Estimate* or *MLE*. The idea is to find those model parameter values that maximize the likelihood function for the observed data or, more formally,

$$\hat{\theta}_{ML} = arg\underset{\theta}{max}\ P(X|\theta) \ . \tag{2.7}$$

The likelihood function can be expressed as the product of individual distributions of the data $x_i \in X$ if we assume the observations are independent and identically distributed. The above equation then becomes

$$\hat{\theta}_{ML} = arg\underset{\theta}{max} \prod_{x_i \in X} P(x_i|\theta) \ . \tag{2.8}$$

This is often easier to solve in its logarithmic form, which, due to logarithmic function being monotonically increasing, obtains its maximum at the same point as the original form, and so the above can be changed to

$$\begin{aligned}\hat{\theta}_{ML} &= arg\underset{\theta}{max}\ \log \prod_{x_i \in X} P(x_i|\theta) \\ &= arg\underset{\theta}{max} \sum_{x_i \in X} \log P(x_i|\theta) \ . \end{aligned} \tag{2.9}$$

In the case that the likelihood function is differentiable, the *maximum likelihood* estimate can be solved from Equation 2.9 by setting its first derivative equal to zero and solving for θ, which will give the estimate $\hat{\theta}$. The likelihood maximizing solution can then be distinguished from its counterpart by checking that the sign of the second derivative at that point is negative. Otherwise, if the likelihood function is non-differentiable, different optimization algorithms such as gradient descent, for instance, need to be applied.

The other mentioned estimate, the *Maximum a Posteriori* or *MAP*, follows the same line of reasoning but, instead of maximizing the likelihood, we now maximize the posterior probability in Equation 2.6

$$\hat{\theta}_{MAP} = \underset{\theta}{arg\,max}\ P(\theta|X)\ . \tag{2.10}$$

Through the same reasoning as above with the *MLE*, the estimate becomes

$$\begin{aligned}\hat{\theta}_{MAP} &= \underset{\theta}{arg\,max} P(X|\theta)P(\theta)\\ &= ...\\ &= \underset{\theta}{arg\,max} \sum_{x_i \in X} \log P(x_i|\theta)P(\theta)\ .\end{aligned} \tag{2.11}$$

Then, again, the *maximum a posteriori* estimate can be solved by means of differentiation or optimization.

2.4.3.2 Bayesian Sparse Linear Mixed Model

The *Bayesian Sparse Linear Mixed Model* or BSLMM is a hybrid between a linear mixed model and a sparse regression model that is capable of catering to two different assumptions. The first of these assumes normally distributed effect terms for all variants, whereas the second assumes that only a small proportion of the features explain the phenotype with large effect sizes. (Zhou et al., 2013)

The BSLMM is a linear mixed model with an additional random effect term

$$\mathbf{y} = \mathbf{1}_n \mu + \mathbf{X}\tilde{\beta} + \mathbf{u} + \boldsymbol{\varepsilon} \tag{2.12}$$

where random effect terms $\mathbf{u}$, error terms $\boldsymbol{\varepsilon}$ and regression coefficients $\tilde{\beta}$ have the following priors:

$$\mathbf{u} \sim MVN_n(0, \sigma_b^2 \tau^{-1} \mathbf{K}) \tag{2.13}$$

$$\boldsymbol{\varepsilon} \sim MVN_n(0, \tau^{-1} \mathbf{I}_n) \tag{2.14}$$

$$\tilde{\beta}_i \sim \pi N(0, \sigma_a \tau^{-1}) + (1 - \pi)\delta_0 \tag{2.15}$$

Here $\mathbf{K}$ is the covariance matrix of $\mathbf{X}$ and π, σ_a, σ_b, and τ are hyperparameters. The idea in BSLMM is to estimate the values of these parameters using Markov chain Monte Carlo sampling on their posterior distributions. This is to adjust the parameters to the actual data and to allow the method to find a balance between the two contained models. For more technical details, one should refer to Zhou et al. (2013).

The BSLMM is one of the three linear models used in coordination with the proposed feature selection method. The following section will move on to look at two non-linear models that are likewise used in testing the novel method.

2.5 Non-linear models

In addition to the previously discussed linear models, this thesis tests the proposed wrapper's performance with non-linear methods. *Random Forests* and *Support Vector Machines* (SVMs) are two of the prevailing methods when it comes to classification tasks in the life sciences (Touw et al., 2012). In comparisons, finely optimized SVMs have generally outperformed Random Forests by a slight margin but the latter method has some advantages that make it a viable candidate over SVM in certain applications (Touw et al., 2012). The following two sections present the key concepts of these two classifiers.

2.5.1 Random Forest Regression

Random Forests are, as the name suggests, ensembles of decision trees. Each of these trees is trained using a random subset of samples and features from the data, which accounts for the randomness in the model. The final classification or prediction made by the model is then the mode class or mean of the predictions of individual trees.

Given data with N samples and M features, the trees are constructed by first determining the tree-specific training sets consisting of N samples selected with replacement. Then, for each node in the tree, a predefined subset of m features are selected randomly from among the available attributes M. The algorithm then determines the attribute in m and a value for that attribute that results in the optimal split of the training samples in the node. Then, new nodes are defined until all training samples have been correctly separated into cases and controls.

Random forests have several advantages over other methods: (1) resistance to over-fitting, (2) robustness to noise and outliers, and (3) a short

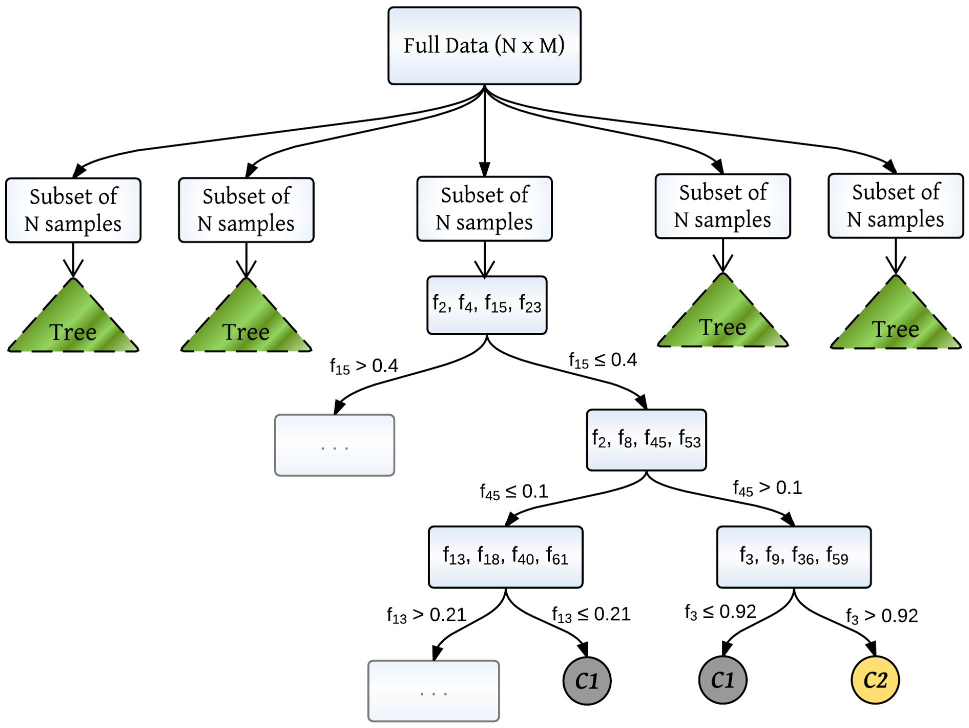

Figure 2.1: **Random Forest.** The depicted forest consists of *five* decision trees, each of which has been randomly assigned an N-sample training set by sampling with replacement from the full data set of N samples and M features. The middlemost tree has been expanded to show the internal structure of the trees. Namely, each node is randomly assigned $m = 4$ features from the full set of M features and each node is split by finding an optimal value for one of the given features. For instance, feature f_{15} has been used to split the root node. Eventually the splitting of nodes leads to leaf nodes that classify samples as either cases or controls. In the figure these have been labeled classes $C1$ and $C2$. Predictions are then made by classifying new samples into cases or controls using all trees and the final solution of the Random Forest model for a given sample is the majority vote or average response value of all trees.

training time. Even more importantly, Random Forests are well able to discern the pervasive epistasis interactions present in genetic data, such as interactions between SNPs. (Touw et al., 2012) In addition, there is generally no need to cross validate the few hyperparameters in the model as the default values already lead to sound performance (Touw et al., 2012). Some of the parameters that can be specified are the number of trees in the model, the number of variables used in each node to split the samples, the maximum depth of trees, and the function used to measure the quality of the splits. Since the samples are selected randomly for each tree, only certain samples are used to train a specific tree. The remaining samples can then be used to validate the model without the need for an separate validation set. Therefore cross validation is not needed for assessing the model's predictive performance but a score can be obtained by averaging the scores obtained for the unused tree-specific samples, the out-of-bag samples.

Besides being used for classification and regression tasks, Random Forests — employing embedded feature selection — can also output feature importance scores for variable selection tasks. In fact, three quarters of the studies reviewed by Touw et al. (2012) made use of the method's feature importance metrics. However, very few of the studies utilized these importance scores in iterative feature selection routines to attempt to refine the feature set to include only the most relevant features. Those studies that used feature selection also reported improved classification accuracies. This thesis uses Random Forests in the proposed wrapper technique for method assessment. The other used non-linear method is the Support Vector Machine.

2.5.2 Support Vector Regression

A Support Vector Machine (SVM) is fitted by finding a hyperplane in the feature space of available training data that separates the training samples into their correct classes with a maximal margin of separation to the hyperplane. The sample points closest to the hyperplane are called the support vectors and they are the samples that are the most difficult to classify as they lie close to the decision surface. Also, it so happens that these support vectors end up being the data samples that fully specify the resulting learned decision function that is used to classify new samples or to obtain their prediction response values. This results from the objective of maximizing the margin of separation between the candidate hyperplane and the training samples, which leads to the minimization of the feature weight vector in the learned model. Consequently, only features relevant to the support vectors end up receiving non-zero weights.

More elaborately, the hyperplane that is a decision surface is of the form

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.16}$$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ the input vector and b is a bias term. Therefore, samples that fall on one side of the decision boundary belong to one class and vice versa. Since the values of $\mathbf{w}$ and b are completely arbitrary, the two boundaries for the training samples defined by the support vectors — that are the ends of the margin — can be expressed as

$$\mathbf{w}^T\mathbf{x}_i + b = +1 \quad when \quad \mathbf{y}_i = +1 \tag{2.17}$$

$$\mathbf{w}^T\mathbf{x}_i + b = -1 \quad when \quad \mathbf{y}_i = -1 \tag{2.18}$$

where $\mathbf{y}_i$ are the response values of the respective sample input values $\mathbf{x}_i$. The following Figure 2.2 illustrates the concepts of hyperplane and margin as they pertain to Support Vector Machines.

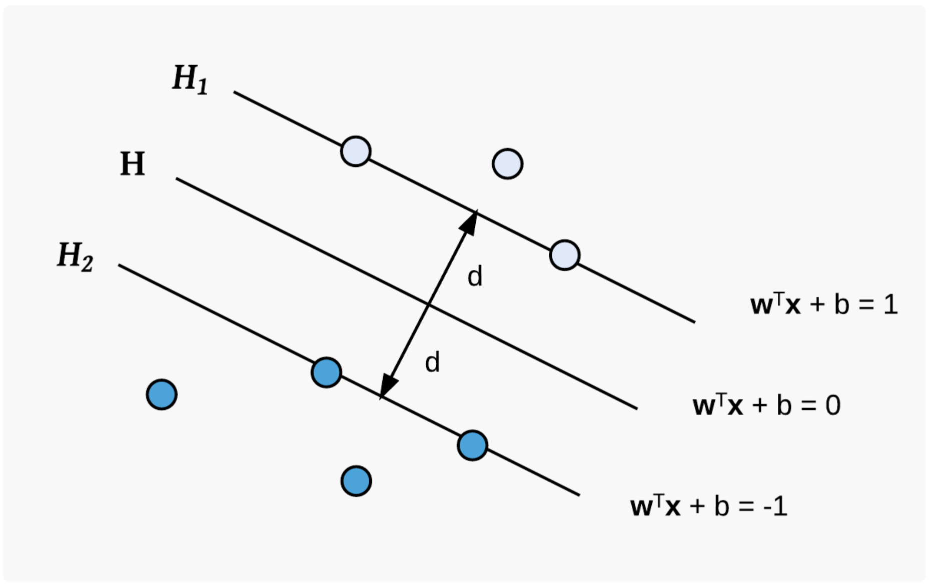

Figure 2.2: The hyperplane **H** and margin $\boldsymbol{H_1} - \boldsymbol{H_2}$ of the SVM: hyperplanes $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$ mark the edges of the margin and are defined by the support vectors that lie within it or on its edges. Here d denotes the distance of either margin from the hyperplane **H**.

The perpendicular distance from a point $\mathbf{x}$ to a hyperplane can be obtained with the distance formula

$$d = \frac{|\mathbf{n}^T\mathbf{x} + b|}{\|\mathbf{n}\|} \tag{2.19}$$

where $\mathbf{n}^T\mathbf{x} + b$ is the equation of the hyperplane and $\mathbf{n}$ is its normal vector. For a point $\mathbf{x}_i$ on the median hyperplane $\mathbf{H}$, the distance to either margin with the normal vector $\mathbf{n} = \mathbf{w}$ is

$$d = \frac{|\mathbf{w}^T\mathbf{x}_i + b|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|} \tag{2.20}$$

The width of the margin from one end to the other is then twice this distance. So, to maximize the margin, we minimize the norm of the weight vector $\mathbf{w}$. In other words, the objective function is

$$min \quad \|\mathbf{w}\| \tag{2.21}$$

Given that the two classes are linearly separable, the hyperplanes in Equations 2.17 and 2.18 yield the constraints

$$\mathbf{w}^T\mathbf{x}_i + b \geq +1 \quad when \quad \mathbf{y}_i = +1 \tag{2.22}$$

$$\mathbf{w}^T\mathbf{x}_i + b \leq -1 \quad when \quad \mathbf{y}_i = -1 \tag{2.23}$$

which can be combined into the single constraint

$$y_i(\mathbf{w}^T\mathbf{x}_i) \geq 1 \tag{2.24}$$

The resulting constraint optimization problem can then be solved for instance using the method of Lagrange multipliers. For the details one can refer to Berwick (2008).

However, often the perfect linear separability of the case and control samples is not a reality. This situation is often resolved by defining slack variables that allow the samples to transcend the margin by some small amount. Such treatment of the issue then gives rise to a hyperparameter C, or the cost parameter. A larger value for this parameter allows the model to use more support vectors to define the classification boundary. Consequently larger values of the C parameter will make the model more complex and may lead to overfitting. On the other hand, too small C threshold values can make the model too lenient towards classification errors which may result in poor prediction performance. (Mittag et al., 2012)

In the event that the samples are not at all linearly separable, an SVM can use a nonlinear kernel, such as a *radial basis function* or a *polynomial* kernel to first transform the samples to a higher dimensional feature space where they become linearly separable. This thesis makes use of SVMs as representatives of nonlinear classifiers in method assessment. Therefore only nonlinear kernels are considered and, for simplicity in model selection, only RBF kernels are used. Thus polynomial or linear kernels are not discussed further.

The RBF kernel on the other hand involves a hyperparameter *gamma* that controls the area of influence of each support vector. Smaller values of gamma increase it, resulting in more linear decision boundaries, while larger values reduce the influence of each support vector and make the classifier resemble k-nearest neighbor clustering where the classification contour follows the majority vote of the closest support vectors. In addition, the optimal gamma value depends on the choice of the parameter C and should be cross validated for the optimal performance of the SVM. (Mittag et al., 2012)

The current state of GWAS research is the main topic of the following section. Special emphasis is placed on the challenges posed by the new research direction and the increase in the dimensionality of genetic data produced thanks to the cheapening of whole genome sequencing. Finally, a novel wrapper method is proposed as a possible solution to these challenges.

Chapter 3

Computing Environment

There is currently a notable switch from the analysis of common variants to the complete sequencing of previously uncharted regions of the genome and the mapping of complex diseases to rare variants. This change in focus results from discoveries indicating that rare variants tend to have stronger phenotypic effects than common variants (Liu and Leal, 2010). In fact, according to certain geneticists, GWAS studies indicate that the heritability of complex phenotypes is for the most parts caused by rare variants with large effect sizes as opposed to common variants with low to medium effect sizes (Koboldt et al., 2013). In their review article, Cirulli and Goldstein (2010) provide a detailed commentary on these conflicting views on the origin of common diseases and provide evidence for the notable role of rare variants in the disease mechanisms. With that said, these biological details on the origin of common diseases, which is not the focus of this thesis, will not be discussed further.

Rare mutations have previously been overlooked due to their demand of large sample sizes and the expense in gathering such vast data sets. Namely, the rarity of certain variants requires large volumes of individuals to be genotyped before a sufficient number of samples with the given rare genotype can be found and before meaningful analysis can take place. However, recently the cost of whole genome sequencing has decreased rapidly. This means that genome-wide sequencing data will be generated at an increased rate and should become more available to cater for the needs of true genome wide association studies. The wealth of data can be expected to lead to the identification of formerly unobserved rare causal variants in the genome.

At the same time, the arrival of cheap sequencing technology can be expected to increase the number of analyzed causal variants from a few million towards the three billion nucleotides present in the whole human genome (Bush and Moore, 2012). Consequently data analysis in these higher dimensional spaces will be difficult and the high dimensional data poses significant challenges to modern day feature analysis techniques. Thus the following subsections first briefly describe the data and tools available for GWAS before defining the environment in which current analysis methods operate. This chapter ends in the proposal of a novel feature selection wrapper and a heuristic designed to help scale it to high dimensional feature spaces.

3.1 Data in GWAS

The data generated in genetic studies comes generally in either of two forms: microarrays or whole genome sequences. Microarrays contain measurements from certain positions of an organisms genome, whereas genome-wide sequencing generally captures the base-pairs of nearly the entire genome of an organism. Formerly, much of the performed genotyping has concentrated on the mapping of common variants, which act largely as proxies for the true underlying functional variants. This type of genotyping has been possible largely due to the considerable correlation existing between causal SNPs and studied common variants. However, these indirect mapping methods are not optimal for studying rare variant associations since they typically show low correlations with the commonly surveyed proxies. Therefore, the rare causal variants must be identified and mapped directly to the relevant phenotypes. (Liu and Leal, 2010)

With the availability of complex, high dimensional data comes the need for novel analysis methods. In fact, experiments have indicated that conventional methods that work well in the analysis of common variants are inefficient for analyzing rare variants (Liu and Leal, 2010). From a low level perspective, the data available to researchers in GWAS suffers from two issues: the presence of missing values and the need for conversion between different formats. These are issues that may impede a non-domain-expert from performing data analysis in the first place or that needlessly complicate matters even for the seasoned analyst.

3.1.1 Missing Values

It is not uncommon to observe multiple missing values in a given GWAS data set. Furthermore, the practice of labeling these missing values is not always consistent between data sets and the values may be indicated with any of following symbols: 'NA', 0, -9 or 'N'. Moreover, missing values may be encountered both in genotype and phenotype files. In the case of missing phenotypes, the respective samples will simply need to be dismissed from the given experiments. However, missing values in genotype can possibly be inferred using imputation tools. These tools generally require a reference file containing the common haplotypes for the given genetic sequences and they infer the missing values based on these reference panel values as well as the other values present in the analyzed data.

Some popular tools for imputation include IMPUTE[1], PLINK[2] and MACH[3]. However, the use of these tools may be challenging for researchers who are unfamiliar with the many formats of genetic data on the offer. In addition, the data may require significant preprocessing before the use of these imputation tools. The possible preprocessing tasks include the exchange of the missing value symbol to an alternative and the conversion of the data to one of the few data formats accepted by the given imputation tool. This gets us to the other prominent nuisance — the conversion issues. These and their solutions are discussed in Appendix A. The following section goes over two state-of-the-art analysis programs that work with the PLINK data format and provide accurate association results.

3.2 Analysis Tools in GWAS

There are several tools that are publicly available for studying genome wide associations between single nucleotide polymorphisms and phenotypes of interest. This thesis will look at two recently developed tools that have shown good performance in modeling complex SNP interactions. The first of these tools is the GEMMA[4] software package whose Bayesian sparse linear mixed model (BSLMM) was discussed in Section 2.4.3.2. The second tool is LDAK[5] that, above others, offers the means of utilizing a MultiBLUP classifier and has challenged BSLMM in efficiency.

3.2.1 GEMMA

GEMMA is a tool that fits Bayesian sparse linear mixed models and expects its inputs in either PLINK or BIMBAM formats. The strong suit of the BSLMM model is its adaptability over various settings with a hybrid of a linear mixed model and a sparse model. More importantly, BSLMM outperforms its two constituents (Zhou et al., 2013) and should therefore be

[1]`https://mathgen.stats.ox.ac.uk/impute/impute_v2.html`
[2]`http://pngu.mgh.harvard.edu/~purcell/plink/`
[3]`http://csg.sph.umich.edu//abecasis/MaCH/tour/imputation.html`
[4]`http://www.xzlab.org/software.html`
[5]`http://dougspeed.com/ldak/`

ideal in cases where one would traditionally apply either linear mixed models or sparse models and cannot determine which of these would be more appropriate.

3.2.2 MultiBLUP

MultiBLUP accepts data in PLINK, CHIAMO and SP formats. MultiBLUP is an extension of perhaps the most prevalent phenotype prediction method, the Best Linear Unbiased Prediction (BLUP), which fits a linear mixed model with a single random effect term and with the assumption of identically distributed effect sizes. MultiBLUP improves the model by extending it to include multiple random effects drawn from potentially distinct distributions. (Speed and Balding, 2014)

The efficiency of BLUP based methods have led geneticists to adopt them in place of previously popular sparse models (Speed and Balding, 2014). According to the authors, their MultiBLUP method outperforms GEMMA's BSLMM method both in terms of time and memory expenditure with it requiring only 10 % of the time and 5 % of the memory required by BSLMM. Being a competing method to GEMMA's BSLMM, it is an interesting inclusion to the panel of analysis tools used to analyze the proposed feature selection method. At this point, the chapter has reviewed the data and tools used in GWAS. The following section will now move on to discuss the challenges of feature selection in these studies.

3.3 Feature Selection in GWAS

Complex algorithms are prone to suffer from the curse of dimensionality, which refers to the condition where the increase in the dimensionality of data leads to an even steeper increase in the computational time and memory requirements of algorithms that use the data. The exhaustive analysis of high dimensional data may in some cases be impractical if not practically impossible, and the increased computational costs are grounds for developing new heuristics that either reduce the number of analyzed features or — in the case of search algorithms — take shortcuts when exploring the feature space. As already pointed out, many previously used analysis techniques are becoming outdated with the growing prevalence of high dimensional sequencing data. Also, beyond the computational issues, model and feature selection tasks are becoming more challenging and the classification accuracy of obtained models is bound to deteriorate. There is therefore a real need for either computationally more efficient learning algorithms or new feature selection methods that can reduce the high dimensional data sets to a size suitable for existing learning algorithms.

However, instead of designing a new modeling technique for complex genetic interactions, this thesis concentrates on the development of a new feature selection method. Besides helping other algorithms cope with sequencing data by first reducing it to a more manageable size, feature selection also facilitates the coordination of the international research effort. The identification of important SNPs allows subsequent research to concentrate in the analysis of a smaller set of potentially relevant SNPs. In fact, the development of robust and stable feature selection techniques is only a very recent endeavor and is still surprisingly seldom discussed in publications (Abeel et al., 2010). Thus, this thesis presents a new method to meet this need of appropriate feature selection techniques. The following chapter proceeds to present the proposed feature selection wrapper and a speedup heuristic for improved scalability.

Chapter 4

Methods

The structure of complex genetic interactions varies among different diseases. Without knowledge of that exact structure, it is difficult to choose which learning algorithm to use to identify relevant genetic variants. For this reason, the use of specific embedded feature selection techniques may only be appropriate once the nature of interaction is confirmed.

In contrast, wrapper methods make no assumption regarding the underlying structure of SNP interactions and accept any classifier for use in feature evaluation. Thus, different classifiers can be used to test the nature of genetic interactions in each particular case. Additionally, wrapper methods are deemed more trustworthy than filter methods as they conduct the feature selection using advanced learning algorithms and optimize the resulting subset for that a specific classifier. More importantly, their performance tends to surpass that of filter techniques. For this reason, they are the preferred way of performing feature selection.

The downside of wrapper approaches is that they are computationally heavy and scale poorly to higher dimensions. That is, as the dimensionality of a data set increases, the number of distinct feature subsets in the data grows rapidly. Hence, such feature selection techniques must find ways to reuse as much of the calculations performed in different iterations, subsets or cross-validation folds as possible to improve their scalability. (Okser, 2015) The solution seems to be a heuristic that mimics the behavior of embedded techniques in using intermediate results from previous computations to drastically reduce the number of subset evaluations in each consecutive iteration. Another option would be to base the candidate subset dismissals on some specifically computed metrics. However, the choice of appropriate metrics would need to be made on a case-by-case basis since a specific metric cannot be expected to work with all different classifiers. For this reason, this thesis presents a heuristic that is based on the reuse of feature rankings from prior iterations. These feature importance rankings are then useful in limiting future considerations only to features that have performed consistently well in previous iterations.

The first objective of this research is to develop a wrapper type feature selection method that can be used with available analysis tools to produce more robust feature subsets. Robustness in this context refers to the selection method's ability to avoid local optima that prevent it from reaching the globally optimal solution subset. The other objective is to design a speedup heuristic for use with the proposed wrapper method.

Forward directed selection is the chosen approach for the proposed wrapper. The justification for this is, as presented in Chapter 3, the small number of variants expected to be the cause of complex disease phenotypes. It is thus reasonable to start small and add features as they are tested for relevance. This wrapper method is presented in the following section. The speedup heuristic, on the other hand, is introduced in Section 4.2 and it is based on the notion of prior importance. Features available for selection are hence considered on the basis of their former performance with the previous feature subsets. The proposed method is expected to outperform the regular Forward Selection method due to its more extensive search into the feature space. The heuristic, on the other hand, is expected to improve the method's scalability without deteriorating the quality of its results.

4.1 The Parallel Forward Selection Method

Section 2.2.2 presented four different wrapper techniques. The proposed wrapper method is based on the first of these — forward selection. In fact, the method introduced here, named *Parallel Forward Selection* or *PFS*, is a container for several concurrently performed forward selection procedures. The relevant characteristics of the parallel part are then the initialization of the independent forward selection routines and the merging of the converged forward search paths. Figure 4.1 below illustrates the container concept.

As mentioned in the beginning of this chapter, the use of forward selection, as opposed to backward elimination, is justified for the small expected number of relevant features. On the other hand, forward-backward search is known for its slower convergence which in the case of high dimensional data translates to a potentially large increase in computational workload. Consequently, any intermediate backtracking steps cannot be recommended for the new method.

The proposed method borrows its parallelism from genetic algorithms. Like genetic algorithms, the method uses a pool of solutions to search for

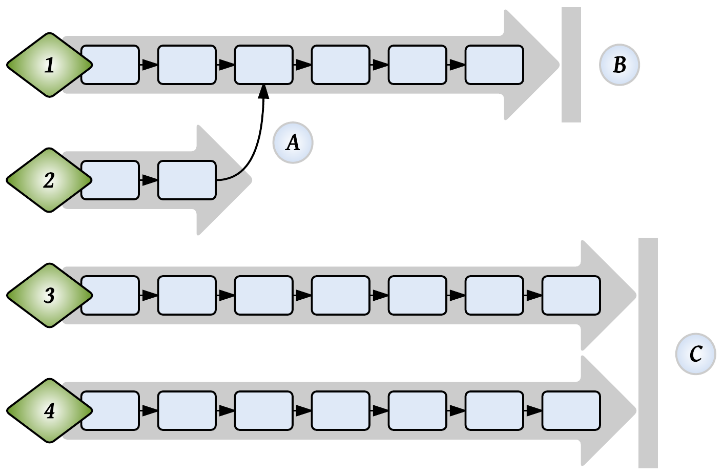

Figure 4.1: The parallel forward selection routine consists of several forward selection paths that are run in parallel. This diagram shows three labeled events in the process. **A:** path number 2 is merged with path number 1 after the second iteration as the sets have become identical. **B:** path number 1 is terminated after the sixth iteration due to descending validation score. **C:** all remaining processes are terminated as their score starts to decrease after the seventh iteration.

the global optimum. This way the resulting solution has a better chance of being closer to the global optimum than if it had explored just one path. However, genetic algorithms do not provide any guarantee that the resulting feature selection is optimal (Pudil, Novovičová, and Kittler, 1994), and for this reason neither they nor other purely random methods are recommendable in the identification of relevant SNPs in high dimensional data. Thus, a parallel, greedy forward selection approach is more justified when requiring the optimality of results.

The independent forward selection routines within the PFS container are called *paths*. *Core i* consists of the present state of all active paths of size i. The PFS method can proceed for a predefined number of iterations or until all of the paths have terminated. The advantages of the PFS method include (1) the ability to escape the pull of initial local optima with the proper initialization of the feature sets, (2) a more thorough forward search with several paths, and (3) its ability to perform at least as well as a regular forward selection routine, given it has been initialized to include that path.

The initialization can be realized using different techniques. A parameter

t determines the number of features in the sets of the initialized core. Thus, the method's first iteration will be tasked with determining the next core, *Core t+1*. On the other hand, the number of paths is controlled with a parameter k.

Moreover, the PFS method is relatively resource friendly since its parallel paths are run only as long as better validation scores can be achieved. That is, if a path reaches a point where no additional feature improves the validation score, that path is immediately terminated. Similarly, when two paths merge, the amount of remaining computational load is reduced significantly as one path is left in place of multiple paths.

However, the method comes with its own downsides. Firstly, its performance is largely determined by the underlying forward selection method's aptitude in the application area. Secondly, the ideal procedure used to initialize the parallel paths is case specific and may be difficult to determine. Yet, the main challenge with the parallel forward selection method is its time complexity which is a multiple of that of a standard forward search in the number of paths. Since wrapper techniques tend to be infeasible on large data sets, the higher complexity caused by the addition of new search paths calls for insightful heuristics that are capable of reducing the workload through intelligent feature management, especially as pertaining to the ranking of considered features. The following section will hence present one such heuristic, the *Nested Indexing Heuristic*.

4.2 The Nested Indexing Heuristic

This section presents the Nested Indexing Heuristic, or *NIH*, which is an attempt to facilitate the use of forward directed, wrapper type feature selection techniques in the large GWAS scale. As noted before in Section 3.3, wrapper techniques do not scale well due to their tendency to completely retrain the classifier in the evaluation of every feature subset. One way to improve the efficiency of wrappers is to reduce the number of candidate feature subsets that are evaluated with the classifier.

The proposed heuristic is illustrated in Figure 4.2. It first normally ranks the available features by their measured relevance with the classifier and selects the highest ranking feature as the first addition to the initial set. However, before proceeding to select the next feature, it reduces the list of considered features to the top performing fraction R of the remaining ranked features. The process of selecting a feature and reducing the subset of considered features is continued until the number of remaining considered features has fallen below a predefined threshold L. At this point the features stored at the first level of indexing are restored to allow the procedure to once again consider a larger portion of the unselected features, which in the case of the first index is fraction R of the full, original feature set. Then the same process of selections and reductions is repeated until the size of the subset is once again smaller than the threshold L. Here another list of indexes is loaded — this time at a level one higher than the previous time and the same process is repeated. Once the process reaches the limit L the last time and there is no index level to fall back on, the method enters the *exhaustion phase*. In this phase features are selected from the highest and most refined index until the selection of an additional feature no longer improves the validation score. Were there so many relevant features left that the entire topmost level of indexes were exhausted, the remaining features would be selected from the level below that. The pseudocode for this heuristic is presented in Algorithm 1 below as applied to a regular sequential forward selection process.

The proposed heuristic is based on four assumptions. The first is that the classifier should give higher scores to more relevant features. Secondly, the features that get selected in the first iterations strongly influence the remaining selections. Thirdly, features that do not interact well with previously selected features are less likely to perform well in the following iterations. Finally, the fourth assumption is that, as more features are selected, the effect of the newly added feature on the validation score diminishes.

The first of these assumptions is used to justify the permanent elimination of $1 - R$ of the original features at the end of the first iteration. The heuristic assumes that relevant features cannot be among the worst performing features and permanently dismisses them. This assumption is reasonable, given an appropriate choice of the R-parameter, because the objective is to look for a small number of strong features. Relevant features should therefore show strong signs of relevance with their validation scores.

The second assumption states that there is valuable information in the feature rankings computed at every iteration and that they should be used beyond the selection of one feature. That is, as the algorithm is selecting features, the feature rankings become more informative and accurate in indicating the region of the feature space that contains the remaining relevant features. It is therefore justified to reduce the subset of considered features to reflect the most recent feature rankings.

The third assumption is that conflicting candidate features will remain in conflict with the set of selected features throughout the selection process. This is supported by the fact that the method proceeds only in one direction and thus any conflicts with the selected features will remain in effect. On the other hand, any new selections involve features that are most compatible with the already selected features. However, at the same time, they may be

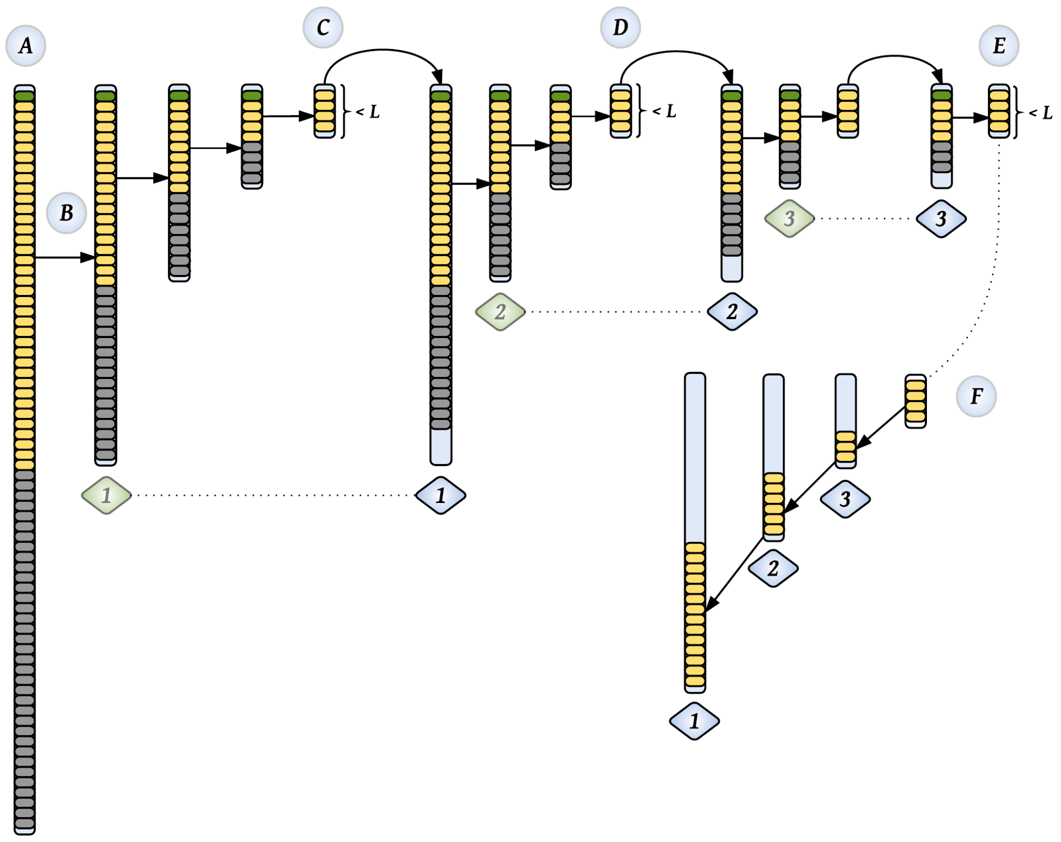

Figure 4.2: The Nested Indexing Heuristic. **A:** first all features are evaluated with the classifier and ranked in descending order of importance. The best feature is included into the base set. **B:** A fraction R of the highest ranking features are selected to become the candidate features in the next iteration and this subset is indexed as *level 1*. **C:** The process of selecting the top ranking feature and eliminating the poorest *1-R* fraction of features from the subset of considered candidate features is repeated until its size falls below some predefined threshold value L. The algorithm then restores the features present in the first index of level 1. **D:** The previous process is continued until, again, the threshold L is surpassed and a subset from the higher indexed *level 2* is loaded from memory. **E:** finally all indexing levels (1-3) have been reloaded and there is no level to fall back on. **F:** the levels are exhausted one at a time until the score starts to decrease.

Algorithm 1 The Nested Indexing Heuristic. The heuristic is shown here as applied to a regular forward selection routine.

```
Require: L, R, data
 1: $Selection \leftarrow$ [ ]
 2: $Considered \leftarrow$ all features in $data$
 3: $a \leftarrow 0$, $b \leftarrow 1$, $exhausting \leftarrow$ False
 4: for N iterations
 5:     if not $exhausting$ and length($Considered$) $<$ L
 6:         if $a > b$
 7:             $Considered \leftarrow$ LoadIndex($b$)
 8:             $a \leftarrow b$
 9:             $b \leftarrow b + 1$
10:         else
11:             $exhausting \leftarrow$ True
12:     if $exhausting$ and length($Considered$) $= 0$
13:         if $a > 1$
14:             $a \leftarrow a - 1$
15:             $Considered \leftarrow$ LoadIndex($a$)
16:         else
17:             break
18:     DispatchEvaluations($Selection, Considered$)
19:     $results \leftarrow$ WaitThenCollectResults()
20:     if not $exhausting$
21:         $ranked \leftarrow$ RankFeaturesByDescendingScore($results$)
22:         if score did not improve
23:             break
24:         else
25:             add $ranked$[0] to $Selection$
26:             if $a = b$
27:                 StoreIndex($b$)
28:             $Considered \leftarrow ranked$[1:R·length($Considered$)]
29:             $a \leftarrow a + 1$
30:     else
31:         $best$ = FindBestFeature($results$)
32:         if score did not improve
33:             break
34:         else
35:             add $best$ to $Selection$
36: return $Selection$
```

among the features showing most incompatibility with the given conflicting candidate features. For this reason the heuristic uses nested indexing where the worst performing features are dropped out in stages. Still, previous indexes are restored several times to ensure that features that were previously overlooked due to poor performance get another chance to be selected in case their contribution has changed along the growth of the selection set.

The final, fourth assumption is related to the complexity of the trained model. When a classifier is trained on a larger number of features, the influence of any single feature and that of the most recently added feature in the trained model can be expected to decrease. As a result, the classifier can be expected to provide similar predictions for different candidate features and the differences in their validation scores will decrease. Consequently, the confidence in these scores diminishes and thus more emphasis should be placed on the prior performance of these candidate features when making further selections. This is what the heuristic does upon reaching the exhaustion stage. In that stage, features are selected from a small subset of consistently eminent features.

The following chapter presents the results obtained with the proposed method on simulated and real GWAS data. Before that, it describes the stages in the implementation of the parallel forward selection container and the proposed heuristic.

Chapter 5

Results and Evaluation

This chapter presents empirical results suggesting that the proposed method is a promising feature selection method scalable to the GWAS scale. The results will show it outperform its rivals in the analysis of larger data sets. The speedup heuristic is shown to lead to faster convergence with no discernible negative effects on performance.

First Section 5.1 summarizes the implementation of the PFS method and its heuristic. Then Section 5.2 presents experiments conducted on simulated data. The objective of these experiments is to compare the proposed method to its rivals and to assess how well they identify relevant features in data. These rival techniques are popular filter, wrapper and embedded techniques. Finally, Section 5.3 presents the results of experiments performed on real data and analyzes the results from different perspectives. Namely, Section 5.3.1 tests the performance of several regression models with and without the proposed feature selection algorithm. Section 5.3.2 evaluates the feature selections obtained in the previous subsection in terms of their usefulness to distinct, external classifiers. Then, finally, the fourth section, Section 5.3.3, compares the proposed feature selection method against the rival selection techniques.

5.1 Implementation

This section explains how the proposed parallel forward selection algorithm was implemented in the form of a prototype and the final parallelized Linux cluster version. The first subsection is devoted to the general PFS wrapper container, while the second subsection discusses the implemental details of the speedup heuristic.

5.1.1 The Parallel Forward Selection Method

The PFS method was initially prototyped in MATLAB with a linear regression model. This is also when a regular forward selection method was implemented in MATLAB as a baseline for comparing the advantages of using the proposed method and its heuristic. The method was implemented in two main functions: the first took care of initializing the parallel paths and managing the iterations, while the other iterated over each path evaluating their candidate features and selected the next feature for each path.

The different tested initialization techniques included (1) the initialization of k paths to sets of size one using the k highest ranked features indicated by the p-values of the t-test statistic, (2) the initialization to a solution obtained in a previous experiment, and (3) a k-means++ (Arthur and Vassilvitskii, 2007) inspired initialization. The second technique allowed one to continue selecting more features to a solution obtained in a previous execution of the method by initializing a path to that solution and rerunning the selection process. The last technique, on the other hand, selected M features with the highest p-values to initialize the k paths. Here the first path's initial set of t features was sampled at random from the set of M features and the other paths' sets were initialized so as to maximize their Hamming distance to all previously chosen sets. The idea was to avoid the quick convergence of the paths and to enable the exploration of different initial subsets of the most promising features. However, preliminary tests showed that the k-means++ inspired initialization technique provided sub-optimal results for all paths and hence only the first two initialization methods were implemented in the final cluster version.

The other used classifier tested with the prototype was the BSLMM model provided by the GEMMA program. While the regular linear model was runnable on a single computer using all four available cores, the runtime of the PFS algorithm with BSLMM was too long to perform feature selection locally on a single computer — even in the case of small sample and feature sizes and with maximal usage of MATLAB's parallel pools. Consequently the PFS method was redesigned in Python for a SLURM managed Linux computer cluster. There were several stages to this task and these are outlined in Appendix B. Finally, the PFS method was able to operate in the cluster using several different classifiers. All of the classifiers besides the BSLMM were obtained from the *scikit-learn* machine learning library for Python. While the heuristic was actually integrated in the PFS method, its implementation is described separately below.

5.1.2 The Nested Indexing Heuristic

The heuristic was actually bolted into the PFS method for a faster implementation. As mentioned in the previous subsection, there were two parts to the main program. In terms of the heuristic, the first module was responsible for managing the heuristic's state and loading the feature indexes from memory as needed. The second module that performed the subset evaluations for each path was tasked with storing the indexes of required subsets and reducing the set of considered features in the indexing phase.

More specifically, the first module that prepared the paths and overlooked the iterations was amended with special parameters that indicated the current state of the heuristic. These were expressed with parameters a, b and *exhausting*. Parameter a indicated the current level of indexing, b the indexing level to store and revert to when the limit L had been reached, and *exhausting* expressed whether the exhaustion mode had been entered. The a parameter was initially set to zero and the b to one. The a variable was then increased every iteration and whenever it equaled the value of b, the second module was instructed to store the current subsets of considered features for each path. Whenever the size of the subset of considered features fell below the limit L, the subset for indexing level b was loaded in the first module and a was set to that level while b was set to the level above that.

At some point in the heuristic, the parameters a and b would be equal while at the same time the number of considered features would fall below threshold L. This would signal the start of the exhaustion mode. At that point, *exhausting* would be set to *True* and the index to exhaust would be singled out by parameter a. The second module would then know to only select features from the current index and not continue reducing it. Were the current indexing level pointed to by a completely exhausted, the parameter value would be set to one level lower and the latest subset for that indexing level would be restored from memory and the process would continue until the score would no longer improve. The pseudocode for these two modules is presented in Appendix B.2.

With the disclosure of these implemental details, it is time to examine the results of the various experiments conducted on the implemented methods. The following section describes experiments performed on simulated data.

5.2 Experiments on Simulated Data

This section discusses experiments conducted on simulated data. The simulated data consists of 1000 samples with 200 features from four (4) equalsized groups. Their phenotypes are affected by common variants and non-overlapping group specific causal variants as well as their interactions. The common features are features that affect the phenotypes in each group with relatively strong constant weights. The group specific features, on the other hand, are non-shared features that are important to one group only and their absolute weights are relatively smaller than those of the common features.

Initially 1000 samples with 500 features were sampled from a uniform distribution. These samples were then randomly assigned into the four groups with each receiving 250 samples. This data was analyzed for correlation among its features to select the 20 common, 20 group specific plus 160 irrelevant features. Each feature's levels of correlation with the other features were computed and its most strongly correlated counterparts were given points. Finally those features that had obtained most points from other features were selected as the common features while those that came after were selected as the general features. These group specific features were then assigned to the four groups so that each group had 5 group specific features. The irrelevant features were then randomly selected from among the remaining unassigned features. The purpose of this process was to select those features as the common features that had some definite structure in them.

The phenotypes y were computed for each group at a time according to the following formula

$$y = X_c B_c + X_s B_s + Z_c b_c + Z_{cs} b_{cs} + e \tag{5.1}$$

$$B_c \sim 0.5\,\mathcal{N}_{20}(-10 \cdot \mathbf{1}, 5^2\,\mathbf{I}_{20}) + 0.5\,\mathcal{N}_{20}(10 \cdot \mathbf{1}, 5^2\,\mathbf{I}_{20}) \tag{5.2}$$

$$B_s \sim \mathcal{N}_5(\mathbf{0}, 5^2\,\mathbf{I}_5) \tag{5.3}$$

$$b_c \sim \mathcal{N}_{190}(\mathbf{0}, 5^2\,\mathbf{I}_{190}) \tag{5.4}$$

$$b_{cs} \sim \mathcal{N}_{100}(\mathbf{0}, 5^2\,\mathbf{I}_{100}) \tag{5.5}$$

$$e \sim \mathcal{N}_{250}(\mathbf{0}, 0.1^2\,\mathbf{I}_{250})\ . \tag{5.6}$$

where the first two terms are the (250×20) matrix of common features X_c and (250×5) matrix of group specific features X_s multiplied by their (20×1) and (5×1) coefficient vectors B_c and B_s, Z_c is a (250×190) matrix of all possible two feature combinations among the 20 common features, Z_{cs} is a (250×100) matrix of all combinations between the 20 common and 5 group specific features, b_c and b_{cs} are their coefficient vectors of sizes (190×1) and (100×1) and e is a (250×1) vector of random errors. The weights of the common features B_c are obtained from a multivariate bimodal distribution that is effectively two multivariate normal distributions with mean vectors $\boldsymbol{\mu}_c = \pm 10 \cdot \mathbf{1}_{20}$ and a covariance matrix $\boldsymbol{\Sigma}_c = 5^2\,\mathbf{I}_{20}$, whereas the weights of the group specific features B_s and the interaction terms b_c and b_{cs} are drawn from multivariate normal distributions with zero mean vectors $\boldsymbol{\mu}_s = \mathbf{0}_5$, $\boldsymbol{\mu}_i = \mathbf{0}_{190}$ and $\boldsymbol{\mu}_j = \mathbf{0}_{100}$ and diagonal covariance matrices $\boldsymbol{\Sigma}_s = 5^2\ \mathbf{I}_5$, $\boldsymbol{\Sigma}_i = 5^2\ \mathbf{I}_{190}$ and $\boldsymbol{\Sigma}_j = 5^2\ \mathbf{I}_{100}$. The random error terms e are drawn from a multivariate normal random distribution with mean $\boldsymbol{\mu}_e = \mathbf{0}_{250}$ and diagonal covariance matrix $\boldsymbol{\Sigma}_e = 0.1^2\ \mathbf{I}_{250}$. Also, unlike the other coefficients and error values, the coefficients B_c of the common features were shared by all groups.

The matrix of interaction terms among common features was obtained by taking an element-wise product of each distinct vector pair of common features and storing them in a column in Z_c. As stated, there were 190 combinations of the 20 common features, resulting in the (250×190) matrix of interaction terms. Thus, the elements along each column are products of two common features. The interactions between the common and specific features were obtained similarly. This time there were 100 combinations of the 5 group specific features with the 20 common features, resulting in a (250×100) matrix of interaction terms.

The motivation for generating the phenotypes using both common and group specific causal factors is to make the model more complex and the prediction tasks performed on the data more challenging. As classifiers will be fitted using three of the four groups and the acquired model tested on the fourth unseen group, the group specific features will only be relevant to the training samples. A successful feature selection method should thus select features from the set of common features rather than those group specific features. The structure of the data will therefore help distinguish between selection algorithms that over-fit to the training data and ones that are better able to identify the generally important common features, which are more useful in understanding samples from all groups.

In addition, since genetic diseases commonly involve the interaction of several variants, such interaction terms were also added to the phenotype modeling function. Hence, the Z_c terms consider the effects of interaction between the common features. However, interaction terms between the common and group specific features were also introduced to mitigate the prominence of the common features. These terms received the coefficients b_c and b_{cs} drawn from a normal distribution with a zero mean and deviation $\sigma = 5$. The errors e were similarly sampled from a normal distribution with zero mean and standard deviation of 0.1.

Finally, after computing the phenotype values, structured noise was added to the original genotype data according to

$$\tilde{X} = X + \varepsilon \tag{5.7}$$

$$\varepsilon \sim \mathcal{N}_{200}(\mathbf{0}, Cov(Z)) \tag{5.8}$$

$$Z \sim \mathcal{U}(0,1)\ . \tag{5.9}$$

This structured noise was sampled from a multivariate normal distribution. The mean vector was a 200-by-1 vector of zeros. The covariance matrix was a positive semi-definite matrix built from a 100-by-200 matrix Z of uniformly distributed random data. The data in Z was zero centered along the first axis and the sample covariance matrix of Z was then used as the covariance matrix of the multivariate normal distribution. The obtained 200-by-1000 matrix of structured error terms was then added to the genotype data consisting of 200 features and 1000 samples to obtain the final genotype matrix $\tilde{X}$.

The data was then divided into training and test sets by taking group one as the test set and using groups two, three and four as the training set. Since the groups had equal numbers of samples, there were 750 training samples and 250 test samples. In the case of simulated data the feature relevance is known and no predictions need to be made on a test group to assess the performance. Rather, the evaluation can be done by measuring the portion of relevant features identified by the different selection methods. However, the prediction scores still reflect how well the selected features were able to guide a chosen classifier in the prediction task. Therefore, the assessed feature selection methods will be judged against both metrics.

The experiments on this simulated data tested the performance of the proposed PFS method against four rival methods: LASSO and Random Forest based embedded methods, the regular Forward Selection method, and a p-value filter method. The PFS method with NIH was run with the following settings: the threshold $L = 20$, reduction rate $R = 0.7$, eight paths ($k = 8$), and initialization to the first core (t=1). The L and R parameters were selected so as to ensure several loops of the indexing phase in the heuristic. Additionally, all feature subset evaluations for the FS and PFS methods and hyperparameter selections for the LASSO model were carried out using six-fold cross validation. The proposed PFS method terminated at a subset of 37 features. Consequently, the regular Forward Selection method (FS) was set to terminate after 40 iterations to ensure comparable solutions. LASSO, on the other hand, identified 58 features and the other two provided feature rankings for all features.

Ideally the methods should select a large portion of the common features among the first 20 selected features as these are features that were actually relevant to all groups. On the other hand, the group specific features were only relevant to one group and should not be selected as frequently. Thus, the group specific features should be selected after the common features if at all. Table 5.1 below presents the content of the first 20 feature selected by the different feature selection methods. The LASSO based method cannot be included in this first analysis as it returned a list of 58 features with non-zero coefficients without any relative ranking of the features. Table 5.1

Table 5.1: Types of features selected within the first 20 features by rival feature selection methods.

	Embedded		Wrapper		Filter
Features	LASSO	Rand. Forest	FS	PFS	P-value
Common	-	25 %	40 %	40 %	40 %
Specific	-	20 %	10 %	10 %	20 %

shows that the regular Forward Selection (FS) method, the Parallel Forward Selection (PFS) wrapper, and the P-value filter all selected 40 % of the common features among the first 20 features. However, FS and PFS methods did not identify as many group specific features as the filter method. The Random Forest importance list gave the worst concentration of common features within its 20 highest ranked features.

Table 5.2 below shows the content of the first 40 features identified by the feature selection methods or the content of the entire solution of 58 and 37 features of the LASSO and PFS methods respectively. It also lists the prediction accuracy of a linear regression model that was trained on the solution set features. The accuracy was measured as the correlation of the predicted and true phenotype values of the test set samples. The results in Tables 5.1 and 5.2 suggest that the two wrappers and the LASSO were more adapted to finding the common relevant features from the training data. In contrast, the Random Forest and P-value filter methods were seemingly less able to distinguish between the common features and group specific features and overall selected more irrelevant features than the other methods. Here the training set's group specific features were irrelevant to the test group and should therefore also be counted as irrelevant features.

The prediction scores presented in Table 5.2 support the above conclusion. LASSO, FS and PFS outperformed the Random Forest and P-value filter methods. Though, these higher scores reflect the fact that the FS and PFS methods used a linear classifier when evaluating the features and making their selections and LASSO is related to the regular linear model used to score the solutions. Out of the three top performing classifiers, PFS received a noticeably worse score than the other two. This suggests that its advantages over the regular forward selection method may not manifest on data sets with

Table 5.2: Types of features selected by the rival feature selection routines during their runtime or within the 40 highest ranked features. The exceptions were the LASSO and PFS methods that selected 58 and 37 features, respectively, as their solutions. Again the comparison is not exactly fair to the other methods as LASSO's result involves roughly 50 % more features.

	Embedded		**Wrapper**		**Filter**
Features	LASSO	Rand. Forest	FS	PFS	P-value
Common	60 %	35 %	50 %	50 %	40 %
Specific	30 %	30 %	25 %	15 %	35 %
Score	0.3733	0.3009	0.3799	0.3641	0.3388

high concentrations of relevant features. Namely, considering the common features as the sole relevant features, this data set had a 10 % concentration of relevant features. Meanwhile, in a real GWAS scale data sets, the portion of relevant features is likely to be significantly smaller. The following section tests these feature selection methods on real data.

5.3 Experiments on Real Data

The real data used in the experiments presented in this section have come from the GEMMA software distribution package version number 0.94 and originate from the Wellcome Trust Centre for Human Genetics[1]. The data contains mouse genotype and phenotype measurements described in Valdar et al. (2006), in both BIMBAM and PLINK binary PED formats. The data in the GEMMA package contains 1904 samples from 85 families of mice and readings for a total of 12226 SNPs. There are two phenotypes: percentage of CD8+ cells and mean corpuscular hemoglobin (MCH) measure that have readings for 1410 and 1580 samples, respectively. The original data was corrected for sex, age, and several other effects and the authors of the GEMMA tool then further quantile normalized it. In addition, missing genotype values were replaced with family means. (Zhou, 2014)

The experiments in this section studied associations just for the MCH counts. The feature subsets were scored by the correlation between test samples' true target phenotype values and the response values predicted with

[1]http://mus.well.ox.ac.uk/mouse/HS/

the used learning algorithm on the contained features. Unless otherwise stated, all cross validations were performed using six-fold cross validation. In the case of nested cross validation, the inner routine was conducted in the form of three-fold cross validation. Furthermore, all experiments on the PFS and heuristic used the minimum subset size $L = 100$, reduction ratio $R = 0.4$ and eight parallel paths ($k = 8$). Where the algorithm was executed a second time, the reduction ratio was dropped to $R = 0.3$ to ensure faster termination.

The reduction ratio value $R = 0.4$ was observed to suit well to the used data in preliminary tests. On the other hand, six-fold cross validation was deemed sufficient to bring about the advantages of cross validation without increasing the computational burden unreasonably high. Lastly, in the following experiments, PFS refers to the PFS method with the Nested Indexing Heuristic incorporated to it. Also, all of the reference classifiers are provided by the *scikit-learn* machine learning library and use their default values unless otherwise specified.

5.3.1 Evaluation Against Same Classifier

The results in this section demonstrate the usefulness of the PFS method as a data preprocessing procedure. Prediction results on four classifiers with and without PFS are listed below in Table 5.3 and the number of identified features in each solution is listed within parentheses. The first experiment initialized the paths to one feature while the second experiment reapplied the method on the best solution of the first experiment.

LASSO and SVM were cross validated using nested cross validation. While *scikit-learn*'s *LassoCV* handled the selection of the appropriate lambda regularization parameter, the *SVR* support vector regressor was optimized for hyperparameters with grid searches in the values $\{0.01, 1, 100\}$ for parameter C and $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ for γ, akin to the recommendations in Mittag et al. (2012). That is, lower values of C were favored for model simplicity and medium to larger values of γ for a less linear decision boundary. The Random Forest model, on the other hand, required no cross validation as it readily prepared the out-of-bag validation scores. The number of trees used in the model was 100 while the remaining settings were left to their default values.

The linear regression model benefited from the use of the novel feature selection method in the first run. The second application of the method on the previous trial's solution, however, lead to a selection with worse performance. LASSO, on the other hand, did not improve on its baseline score. This was however an attestation on LASSO's internal feature selection capa-

Table 5.3: Experiments assessing PFS's suitability as a preprocessing tool. The highest regressor specific score has been highlighted and the number of selected features is stated in parentheses. The PFS method improved the performance of the *linear regression* and *SVM* models but greatly deteriorated the performance of the *LASSO* and *Random Forest*.

		With PFS	
Regressor	**Baseline**	1st	2nd
Linear	0.4627	**0.5761** (40)	0.5589 (67)
LASSO	**0.6260** †	0.5682 (48)	–
Random Forest	**0.5525**	0.3189 (13)	0.3289 (15)
SVM	0.4523	**0.5148** (9) ⋆	–

† the objective function did not fully convergence
⋆ terminated early due to too many failed evaluations

bilities. Similarly, Random Forests performed better on their own and their performance deteriorated when the said wrapper was used to reduce the dimensionality of the training data. In contrast, SVM's performance improved upon feature selection.

However, the SVM method did not seem to benefit from multiple paths. The method seemed insensitive to initialization and selected the same features for each of the eight tested paths in the observed iterations. Hence, neither did the paths converge as they differed by their first, initial feature, which increased the selection method's computational load. Therefore, to avoid exploring redundant paths, the PFS method was run with only one path in the SVM case.

Despite this relief, the feature evaluations on the SVM model quickly consumed the time resources allocated to the cluster jobs and lead to the premature termination of the experiment. Namely, the runtime of fitting the SVM model grew quickly with the dimensionality of the training data and this was identified as the cause of the problem. This issue with cluster resources is discussed more in the next chapter.

Nevertheless, both the linear regression and SVM models worked well with the proposed feature selection method. Yet, it is unclear which of these classifiers should be used when PFS is compared to its rival feature selection methods. The following section attempts to answer this question by testing the subsets identified by these models on external, reference learning algorithms.

5.3.2 Solution Assessment with External Classifiers

In contrast to the results presented in the previous section, the following results measure the quality of discovered feature subsets independently of the classifier that was used to select them. This should give a more general estimate on the quality of discovered features by mitigating the effects of model specific factors and the fittingness to a specific model. The initial idea was to form the reference panel from the leading GWAS prediction tools GEMMA and LDAK and their BSLMM and MultiBLUP models to obtain a score that reflected the quality of the identified feature subsets. However, due to the lacking and incoherent documentation of the LDAK software and its inability to recover from certain missing values in the used data's BIM file, its use had to be waived.

Consequently, those regressors that themselves did not adapt well to PFS, as seen in the previous section, were used to reinforce the reference panel. Hence, the reference panel for the second trial consisted of the BSLMM, LASSO, and Random Forest regressors. The experiments then tested the solutions obtained with the linear regression and SVM models and the outcome of these experiments is laid out in Table 5.4 below.

Table 5.4: Benchmarking results for PFS's solutions as judged by the reference panel classifiers. The table lists the baselines obtained without the PFS algorithm, the solutions obtained using PFS and the prediction score obtained on the reference algorithms using the solution subsets identified by PFS. The baselines of the reference methods are shown to provide a context in which to assess the quality of the solutions.

			Reference Panel		
Regressor	**Baseline**	**PFS**	LASSO	BSLMM	RANFOR
Linear	0.4627	0.5761 (40)	0.5826	0.5827	0.5448
		0.4784 (9) †	0.4784	0.4785	0.3925
SVM	0.4523	0.5148 (9) ⋆	0.4679	0.4677	0.3861
		Baseline	0.6260	0.6633	0.5525

† using the first nine features of the full 40-feature subset
⋆ terminated early due to too many failed evaluations

Table 5.4 presents the baseline and PFS scores for the linear regression and SVM models as they were presented in Table 5.3 of the previous section.

The reference panel scores show that the solution set containing 40 features obtained with the linear regression algorithm outperformed the solution of nine features obtained using SVM. To make the comparison more valid, the first nine features of the linear regression solution were also identified. Still the linear regression solution scored higher on all reference classifiers.

In the case of the Random Forest model, the solution identified with the wrapped linear regression model nearly reached the same standard as the highly popular method on its own. While the score on LASSO was relatively close to its baseline, the differences between the reference scores on LASSO and BSLMM and their baselines were expected as these models add their own strong emphasis on their features. Still, the solution obtained with the linear regression model was a generally good solution that served a broader collection of classifiers than the solution received with the SVM model.

In summary, the results of this section suggest that the linear regression model is the most appropriate candidate for use with the proposed new wrapper technique. Hence, the last experiment will compare the results obtained on PFS with that learning algorithm against other known, alternative feature selection techniques.

5.3.3 Rival Feature Selection Techniques

Section 5.3.1 presented results obtained by running a classifier on the original data and then on a subset of that data identified with PFS and the heuristic. The results showed how much PFS and the heuristic improved the performance of the tested classifiers. Section 5.3.2 then took the two classifiers that improved their performance with the PFS method and cross evaluated their solutions using three different classifiers. There the linear regression model was identified as the better choice for use with the proposed selection method. Now, this section puts the previous results into context through a comparison with several alternative feature selection techniques. These include selection techniques of all three types: filter, wrapper and embedded.

Had there been different data sets available to these experiments, this section would have tested the performance of the different feature selection methods with those different data sets. However, since no other data sets were accessible, the relative importance of the found solutions is tested with different classifiers. This way the solutions by the different feature selection techniques can be assessed more thoroughly. The feature set solutions are evaluated using the regular linear model, BSLMM, and SVM with a radial basis function kernel.

The benchmarking process compared the implemented feature selection method against a p-value filter, a regular forward selection wrapper, and two embedded methods. The first of these embedded techniques selects as its feature subset the features with nonzero coefficients in a cross validated LASSO model, whereas the second one allows one to select a desired number of the most important features as identified by the Random Forest model.

Because the PFS method stopped at 40 features, the top 40 features identified by the p-value filter, FS wrapper and Random Forest model were used as the competing solutions in the comparison. In fact, the FS wrapper did not terminate within the 99 iterations given to it and its score had already started to deteriorate by the time of its forced termination. Therefore, in all fairness, the solution of the first 40 features was used instead in the comparison.

The LASSO-based selection technique, on the other hand, identified a much larger set of relevant features. The model was fitted to the training data using 10-fold cross validation to determine the optimal value of its λ regularization parameter and the resulting optimal model had nonzero coefficients on 419 of the 12226 features. The solution of the LASSO-based feature selection routine was therefore not fit for comparison with the other solutions whose feature sets were less than 10% of its size and consequently LASSO was excluded from the comparison. The large number of 419 features selected by it would have given too much room for the scoring classifiers to perform their own optimizations, which could have skewed the results in LASSO's favor.

Table 5.5: Benchmarking results on distinct feature selection methods. The FS and PFS wrappers used the regular linear regression model as their classifier. All scores were obtained using the best 40 features identified by the distinct selection techniques.

	Embedded	**Wrapper**		**Filter**
Scoring	Rand. Forest	FS [†]	PFS	P-value
Linear	0.4238	0.5746	**0.5761**	0.3141
BSLMM	0.4126	0.5791	**0.5827**	0.3013
SVM	0.4306	0.5871	**0.5937**	0.2881

[†] terminated at limit of 99 features with score 0.5541

As Table 5.5 suggests, the PFS method outperformed the other feature selection methods though its margin to the runner-up FS method was extremely small. The advantages of PFS over FS observed here could be the result of either the wider search provided by the additional paths or the speedup heuristic which diverted the selection process to concentrate on the consistently well performing features.

Considering the results presented in Section 5.2 for simulated data, the superior performance of PFS and FS methods is plausible. LASSO, on the other hand, seemed to suffer from an inability to detect a small number of relevant features. Moreover, this issue seemed to only worsen when switching to real data. Here, the potentially complex genetic interactions and a much smaller concentration of relevant features seemed to emphasize LASSO's weakness.

Overall, in summary, the novel PFS method with its speedup heuristic outperformed its rivals. The Random Forest and P-value filter methods were no match to the PFS and FS methods. The LASSO-based method, on the other hand, was an invalid comparison, because the size of its solution set was overly large compared to the other methods. Thus, PFS's closest competitor was the FS method but it lost to the PFS not just in accuracy but also in speed.

Chapter 6

Discussion

The experiment of Section 5.3.1 suggested that, while the parallel aspect of PFS method was not compatible with SVM, SVM's performance with the heuristic was good. Compared to the linear classifier, SVM seemed better at modeling the phenotype with fewer selected variables. However, the failures in the computations on the computer cluster prevented the closer examination of SVM's suitability to the proposed feature selection method.

Running the heuristic with SVM on the cluster proved problematic. The *SVR* module's runtime grew noticeably with the increasing dimensionality of the training subset and the resources requested from SLURM soon became insufficient. As the evaluation tasks were handled in groups to reduce the overhead of job launches and to stay under the cluster's limit on concurrent jobs, the otherwise small increments to the runtime of one evaluation task multiplied with the group size to result in a much larger increase in the job's runtime. It was therefore difficult to anticipate the changing resource needs.

Another issue was the waiting time implemented in the method to recover from failed jobs. The waiting time needed to be adjusted to the growing runtime of the jobs. For instance, the first experiment of the SVM regressor in the experiment of Section 5.3.1 terminated early because the waiting time limit became too small to allow the longer lasting computations to finish. Consequently several jobs were considered having failed and the sheer number of these failures lead to the termination of the experiment. To fix this issue, the requested time resources should change dynamically with the increasing size of the selection subset but this is difficult to realize. The option of reserving excessive resources to accommodate the growing resource needs would not work either since SLURM allocates resources to users based on the total amount of requested resources. Therefore inflated requests would soon deny further calculations the resources they would rightly need.

In short, the used cluster was an unreliable platform. Any attempt to account for the potential silent failures of jobs in the cluster were met with only more trouble in the form of new parameters demanding adjustment. However, more productive discussion could be had on the proposed feature selection method itself.

More specifically, other initialization techniques should be studied before giving a verdict on the PFS method. The initialization of the PFS was based on the p-value which may not be the optimal choice as it favors features with strong correlation with the phenotype but at the same time ones that do not necessarily interact well with any other features. With the objective being the discovery of variants with complex interactions, there should be less emphasis on simplistic correlation.

Furthermore, the experiments presented in this work suffered from the obvious lack of data. The results of this work would be more meaningful and credible had they been obtained using more than one set of real data. While considerable time was used to search the Internet for applicable data, none could be found freely available but the set contained in the GEMMA package. The first application to one data distributor was declined as they do not grant access to students. The following application made by the supervisor of this thesis did not bear fruit for lack of response and the request is still apparently pending. Hence, the results presented in this thesis serve only as a very coarse evaluation of the proposed method more experimentation is needed to make any strong conclusions on the usefulness of the proposed method.

Similarly it would be useful to test the proposed method on a much larger data set. For instance, the authors of MultiBLUP claim that the parallel nature of their program allows for the analysis of entire genomes as "there is essentially no limit to the number of predictors that MultiBLUP can analyze" (Speed and Balding, 2014). One interesting trial would be to run MultiBLUP on a huge genome wide data set and to see how well the proposed method could fare against it both in speed and the quality of results. Also, the data set used in this project contained just 12226 features which is far from genome wide scale. While the proposed method is completely parallelizable, it would still prove a point to test it on a truly large data set.

Additionally, the various parameters of the PFS and its heuristic such as the number of paths k, the depth of the initialization t, the reduction rate R, and the minimum subset size L were only slightly tested at the prototyping stage. More study should be conducted especially on the L and R parameters that control the number of nested indexing passes in the heuristic. These values greatly affect the extent to which the indexes are refined and updated until a potentially remarkable portion of the selected features are picked from a small feature subset in the exhaustion phase.

Equally, the choice of the initialization technique and the depth at which initialization is carried out is directly reflected in the quality of the obtained results. Early experimentation with the different initialization approaches showed that initialization beyond the first feature was a risky endeavor and should only be considered with an initialization routine that meticulously analyzed the available features and deliberately selected the most justified features into the initial sets. The utter failure of the k-means++ type initialization showed that even an assignment of the most promising features to the initial sets apparently lead to significant internal conflict. Also, the heterogeneity of the initial sets did not seem to be a useful objective in initialization as this prevented the paths from converging.

Besides experimenting on the above parameters and initialization techniques, one more interesting future experiment would be to treat the PFS method as an ensemble and to apply different set operations on the solutions of the multiple paths to obtain a more robust final solution. Especially if initialized appropriately, the different paths of the PFS method could be assumed to capture groups of features with strong interactions. Therefore, a collection of such results would be more robust in cases where there were complex interactions present in the data and the variants would need to be identified in groups.

Chapter 7

Conclusions

This thesis proposed a novel wrapper technique to serve in the identification of important variants in genome wide association studies. The results showed that the proposed feature selection algorithm worked better than the tested rival feature selection techniques in the analysis of genetic data. Additionally, it was able to improve the performance of the linear regression and Support Vector Machine models.

Most notably, the proposed method outperformed the Random Forest model, which is an extremely popular method in bioinformatics. Similarly, the LASSO regressor was unsuccessful in distinguishing relevant features and ended up giving non-zero coefficients to a large number of features. Furthermore, either due to the additional search paths or the intermediate elimination of poorly performing features, the PFS method consistently outperformed the regular FS method. At the same time, the experiments on simulated data suggested that the advantages of PFS over its closest rival, the regular forward selection method, surface when there is a small concentration of relevant features. Fortunately for the proposed method, this should be the case with real genetic data.

The proposed method answers to concerns of robustness with its parallel search paths that explore multiple solution subsets. Empirical results show that the wider search lead to the identification more optimal feature sets. Secondly, the speedup heuristic proved effective in reducing the runtime of a forward selection path to a fraction of that required by a traditional, exhaustive sequential forward selection process without any apparent loss in accuracy. Although the results looked promising for the novel method, there is still need for additional experimentation to be made on its parameters and initialization options. The method's performance could vary significantly with different choices of these values.

Bibliography

Thomas Abeel, Thibault Helleputte, Yves Van de Peer, Pierre Dupont, and Yvan Saeys. Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics*, 26(3):392–398, February 01 2010.

David Arthur and Sergei Vassilvitskii. k-means : The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

R. Berwick. An idiot's guide to support vector machines (svms), 2008.

William S. Bush and Jason H. Moore. Chapter 11: Genome-wide association studies. *PLoS Comput Biol*, 8(12):e1002822, 12/27 2012. URL `http://dx.doi.org/10.1371%2Fjournal.pcbi.1002822`.

Elizabeth T. Cirulli and David B. Goldstein. Uncovering the roles of rare variants in common disease through whole-genome sequencing. *Nature reviews. Genetics*, 11(6):415–425, print 2010. URL `http://dx.doi.org/10.1038/nrg2779`. 10.1038/nrg2779.

Daniel C. Koboldt, Karyn Meltz Steinberg, David E. Larson, Richard K. Wilson, and Elaine R. Mardis. The next-generation sequencing revolution and its impact on genomics. *Cell*, 155(1):27–38, 2013.

Leonid Kruglyak and Deborah A. Nickerson. Variation is the spice of life. *Nature genetics*, 27(3):234–236, print 2001. URL `http://dx.doi.org/10.1038/85776`. 10.1038/85776.

Dajiang J. Liu and Suzanne M. Leal. A novel adaptive method for the analysis of next-generation sequencing data to detect complex trait associations with rare variants due to gene main effects and interactions. *PLoS Genet*, 6(10):e1001156, 10 2010. URL `http://dx.doi.org/10.1371%2Fjournal.pgen.1001156`.

Florian Mittag, Finja Büchel, Mohamad Saad, Andreas Jahn, Claudia Schulte, Zoltan Bochdanovits, Javier Simón-Sánchez, Mike A. Nalls, Margaux Keller, Dena G. Hernandez, J. Raphael Gibbs, Suzanne Lesage, Alexis Brice, Peter Heutink, Maria Martinez, Nicholas W. Wood, John Hardy, Andrew B. Singleton, Andreas Zell, Thomas Gasser, and Manu Sharma. Use of support vector machines for disease risk prediction in genome-wide association studies: Concerns and opportunities. *Human mutation*, 33(12):1708–1718, 2012. URL `http://dx.doi.org/10.1002/humu.22161`.

Sebastian Okser. Scalable feature selection applications for genome-wide association studies of complex diseases, 2015.

P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 11 1994.

Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, October 01 2007.

W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, 10(5):335–347, 11 1989.

Doug Speed and David J. Balding. Multiblup: improved snp-based prediction for complex traits. *Genome research*, June 24 2014.

Jacob A. Tennessen, Abigail W. Bigham, Timothy O'Connor D., Wenqing Fu, Eimear E. Kenny, Simon Gravel, Sean McGee, Ron Do, Xiaoming Liu, Goo Jun, Hyun Min Kang, Daniel Jordan, Suzanne M. Leal, Stacey Gabriel, Mark J. Rieder, Goncalo Abecasis, David Altshuler, Deborah A. Nickerson, Eric Boerwinkle, Shamil Sunyaev, Carlos D. Bustamante, Michael J. Bamshad, and Joshua M. Akey. Evolution and functional impact of rare coding variation from deep sequencing of human exomes. *Science (New York, N.Y.)*, 337(6090):64–69, 05/17 2012. URL `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3708544/`. J1: Science.

Wouter G. Touw, Jumamurat R. Bayjanov, Lex Overmars, Lennart Backus, Jos Boekhorst, Michiel Wels, and Sacha A. F. T. van Hijum. Data mining in the life sciences with random forest: a walk in the park or lost in the jungle? *Briefings in Bioinformatics*, July 10 2012.

William Valdar, Leah C. Solberg, Dominique Gauguier, Stephanie Burnett, Paul Klenerman, William O. Cookson, Martin S. Taylor, J. N.

Rawlins, Richard Mott, and Jonathan Flint. Genome-wide genetic association of complex traits in heterogeneous stock mice. *Nature genetics*, 38(8):879–887, print 2006. URL `http://dx.doi.org/10.1038/ng1840`. 10.1038/ng1840.

Xiang Zhou. Gemma user manual, 2014.

Xiang Zhou, Peter Carbonetto, and Matthew Stephens. Polygenic modeling with bayesian sparse linear mixed models. *PLoS Genet*, 9(2):e1003264, February 7 2013.

Appendix A

Computing Environment

A.1 Data Formats and Conversion Tools

Firstly, the data in complex disease association studies comes in many different formats. Some formats are more common and tend to be accepted by the mainstream analysis and imputation tools. However, others have been designed for a particular purpose or a specific tool and not compatible with the mainstream tools.

One of the most common data formats is the native format of the genome wide association analysis toolset PLINK which comes in two files: PED and MAP. The PED files contain six columns for the IDs of each sample's family, self, father and mother, followed by the sex and phenotype fields. The remaining columns contain the genotype readings for the measured SNPs. The MAP file comprehends information on these SNPs with four items which are the chromosome number, SNP ID, genetic distance and base-pair position.

Furthermore, PLINK has given rise to three other formats: binary PED, dosage and raw file formats. However, many other formats exist, such as those accepted by BIMBAM[1], a Bayesian association analysis tool with imputation functions, and the sheer imputation tools of MACH, BEAGLE[2] and IMPUTE, plus the association analysis tools of SNPTEST[3], HAPLOVIEW[4], GenABEL[5] and the like. Since many GWAS tools come with their own formats, it is easy to see how data management between these analysis and

[1]`http://www.haplotype.org/bimbam.html`

[2]`https://faculty.washington.edu/browning/beagle/beagle.html`

[3]`https://mathgen.stats.ox.ac.uk/genetics_software/snptest/snptest.html`

[4]`https://www.broadinstitute.org/scientific-community/science/programs/medical-and-population-genetics/haploview/haploview`

[5]`http://www.genabel.org/`

imputation tools may become rather difficult.

In fact, the data being used with these tools may well have to be converted once or twice to reach a format supported by that particular tool. Also, with the various data formats comes the difficulty in interpreting the meaning of the obtained output files and comparing the results correctly.

Fortunately, several format conversion tools have been developed to facilitate data conversion and preprocessing. Some of the more referenced tools include GTOOL[6] and fcGENE[7]. The first of these is a tool developed for the specific needs of a family of genetic analysis tools whereas the second is a general, multi-purpose conversion tool designed to support multiple file formats and provide an interface for several third party imputation and analysis tools.

While these tools are generally a great aid in the data conversion task, there may still be a need for custom preprocessing scripts to fix corrupted data or data that does not quite abide by the ambiguous format it claims to represent. One such example is the incompatibility resulting from the data set and conversion tool's use of a different symbol for missing values.

Another observed shortage is the inability of certain tools to convert data with missing values in certain peripheral fields, such as the genetic distance or physical position, which may not even be relevant to the experiment in question but are required by the conversion tool in a certain specific form. This issue was witnessed when attempting to convert data to a format supported by an imputation tool so that missing genotype values could be imputed, but the process halted at the conversion step due to missing physical SNP position data.

Lastly, the conversion of data from one format to another results in loss of information if the target format does not have fields for the specific data. On the other hand, conversion in the other direction should not be possible if the source target does not possess the necessary fields.

[6]`http://www.well.ox.ac.uk/~cfreeman/software/gwas/gtool.html`
[7]`https://sourceforge.net/projects/fcgene/`

Appendix B

Implementation

B.1 Implemental Details for the PFS Method

The following subsections elaborate on the development stages of the SLURM managed cluster version of the *Parallel Forward Selection* method. The cluster version was developed to answer to the problems faced in running the method locally on one machine while analyzing large data sets with complex models.

B.1.1 Parallel Pool Using the Multiprocessing Module

The original approach to parallelizing the Python implementation was to use the parallel pools of workers offered by the standard Python module called *Multiprocessing*. The idea was to tap into the numerous cores available on the Linux cluster. This implementation offered an asynchronous way of evaluating the candidate features by dispatching the evaluations to available computing resources and collecting the results later.

The parallel pool consists of workers that are given a function to evaluate with a parameter that specifies the particular instance that that worker needs to evaluate out of a collection of instances. The pool function is useful for performing the same operation on multiple data as it the case with the evaluations of candidate features together with the same base set of selected features.

The downside to the Multiprocessing module is that it does not work directly with the SLURM clustering resource manager that is the manager of the cluster that was used. The Multiprocessing module's functions only see the resources of the computing node on which the said process was launched and unfortunately cannot communicate with SLURM to divide its jobs across multiple nodes. Consequently, the program was able to benefit only from 12 cores which was the number of cores on one node. This led to the switch to another parallelization module, the *Subprocess* module.

B.1.2 Separate Cluster Jobs as Subprocesses

The PFS method was redesigned to start subprocesses with a separate shell script that was invoked for each candidate feature and that launched a new evaluation batch job on the SLURM managed cluster. The shell script contained the required *SBATCH* parameters and invoked another python module that dealt with the cross validation of a particular candidate feature with the base set of previously selected features.

At this point, the main program would wait for the emergence of the expected cross validation score files of the evaluated candidate features. Once the results were ready, they were collected and processed to identify the best feature.

However, the cluster proved out to be an unreliable platform in that independent jobs could either stall, fail, or keep existing after termination. The first case seemed to stem from the cluster manager's attempt to either enforce user specific quotas by running other users' queued jobs in between the evaluations or possible traffic in the network. Nevertheless, this repeatedly caused several child batch jobs to be killed for exceeding the requested time limit. Then, together with the second case, the parent job would be left waiting for results on these failed evaluation jobs until it itself would time out before reaching the intended end of the experiment.

In addition to the limit on the amount of wall time allowed for each job, there are also limits on the number of simultaneously active jobs. The third issue meant that otherwise finished jobs would fail to terminate in time and would be counted towards the number of active jobs. New batches of jobs would then fail because they would exceed this upper limit. All these issues made it necessary to handle failed jobs separately and these optimizations are the topic of the following section.

B.1.3 SLURM: Issues and Optimization

The issues mentioned in the previous section were handled by adding timing features to the functions waiting for results. The time limit ensured that any job that had failed due to a rare error or had stalled for some unknown reason would not lead to the failure of the entire program. Instead the timed out evaluations would be logged as terminated and their features excluded from the rest of the experiment. However, if the number of unfinished jobs was larger than a predefined upper limit, the given path would be terminated with the results deemed unreliable.

The third issue of the previous subsection could not be directly addressed in the main program itself without functions that would query the number of active jobs from SLURM to avoid surpassing the quota. This would have required an advanced job submitting function while it was easier to dispatch jobs in batches while leaving sufficiently wide margins on the number of dispatched jobs. This latter approach was also justified considering that the problem of unterminated jobs was rare and particular to the SLURM manager and not the program itself.

While the fixes made the method more robust to external cluster computing and SLURM related uncertainties, another problem emerged. The encountered new problem was the wall-time provided to each user in the SLURM quota system. The dispatch of thousands of short jobs soon lead to the exhaustion of user specific wall time.

This problem was mitigated by organizing the feature evaluation tasks into groups of tens or hundreds of evaluations, depending on the classifier and its computational load. This lead to a more efficient use of resources as the amount of overhead related to the launch of new jobs was minimized.

Another crucial factor in cluster computing is the smart use of the different available memory partitions. The cluster nodes' local drives are the preferred storage location for temporary files and the output from certain classifiers was changed from the remote, high-traffic network storage to these local drives on the computer nodes. This lead to considerable speed up especially with the BSLMM classifier that writes out multiple files upon each execution.

B.2 Pseudocode for the Two Modules

The pseudocode for the PFS method with NIH is presented by Algorithms 2 and 3 below. The logic in the division is that Algorithm 2 is assigned with managing the iterations and determining when to load indexes from memory. Algorithm 3 on the other hand performs feature evaluations for all paths, updates the feature selections, and stores indexes at the required points. The final solution is then obtained by observing the validation scores stored for each core at the end of Algorithm 2 and selecting the set of the highest scoring path.

Algorithm 2 Parallel Forward Selection Using Nested Indexing. The ParallelForwardSelection procedure summarized here calls the GetNextCore function whose pseudocode is listed in Algorithm 3 and is assigned with managing the iterations and loading indexes from memory at the required stages.

```
1: procedure ParallelForwardSelection($t, k, I, R, L$)
2:     $F \leftarrow$ LoadData()                          ▷ $F$ = full set of N features
3:     $S \leftarrow$ InitializePaths($F, t, k$)                  ▷ $k$ sets of $t$ features
4:     $a \leftarrow 0$                                     ▷ current indexing level
5:     $b \leftarrow 1$                                     ▷ index to store/restore
6:     exhausting $\leftarrow$ False
7:     $C \leftarrow$ SetDifferences($F$,$S$)      ▷ $k$ sets of $(N - t)$ considered features
8:     for $I$ iterations
9:         if all paths terminated
10:            break
11:        else
12:            if not exhausting and length(C[$\cdot$]) $< L$
13:                if $a > b$
14:                    C $\leftarrow$ SetDifferences(LoadIndex($b$), S)
15:                    $a \leftarrow b$
16:                    $b \leftarrow b + 1$
17:                else
18:                    exhausting $\leftarrow$ True
19:            if exhausting and length(C[$\cdot$]) $= 0$
20:                if $a > 1$
21:                    $a \leftarrow a - 1$
22:                    C $\leftarrow$ SetDifferences(LoadIndex($a$), S)
23:                else
24:                    all indexes exhausted: break
25:            $[S, C, a]$ = GetNextCore($S, C, a, b, exhausting, R, L$)
26:            S $\leftarrow$ MergeConvergedPaths(S)
27:     solution = DetermineBestSetAmongCores()
28:     return solution
```

Algorithm 3 Parallel Forward Selection using Nested Indexing. This module is responsible for dispatching feature set evaluations for each path, updating the selections to include the highest scoring candidate feature of each path or terminating those paths whose score does not improve, and storing indexes at the required stages. The updated sets are returned to the calling ParallelForwardSelection procedure presented in Algorithm 2.

```
    procedure GetNextCore($S, C, a, b, exhausting, R, L$)
 2:     if not $exhausting$ and $a = b$                        ▷ index will be stored
            index ← [ ]                                        ▷ declare empty list
 4:     DispatchEvaluations($S, C$)                  ▷ scores all candidate expansions
        for each unterminated path $p$
 6:         $results$ ← WaitThenCollectResults($p$)
            if not $exhausting$
 8:             $ranked$ ← RankFeaturesByDescendingScore($results$)
                if score for $p$ did not improve
10:                 terminate path $p$
                else
12:                 add $ranked[0]$ to $p$'s current set in S[$p$]
                    if $a = b$
14:                     index[$p$] ← C[$p$]
                    C[$p$] ← $ranked$[1:R·$length$(C[$p$])]
16:         else
                $best$ ← FindBestFeature($results$)
18:             if score did not improve
                    terminate path $p$
20:             else
                    add $best$ to $p$'s set in S[$p$]
22:         SaveCoreAndScores(S)
            if not $exhausting$
24:             if $a = b$
                    StoreIndex($b, index$)             ▷ stores index to level $b$ index
26:             $a \leftarrow a + 1$
        return $S$, $C$, $a$
```