From the Institute of Neuro- and Bioinformatics
of the University of Lübeck
Director: Prof. Dr. rer. nat. Thomas Martinetz

Machine Learning Methods for Genome-Wide Association Data

Dissertation
for Fulfillment of
Requirements
for the Doctoral Degree
of the University of Lübeck
from the Department of Computer Sciences/Engineering

Submitted by
Ingrid Brænne from Ålesund, Norway

Lübeck 2011

First referee: Prof. Dr. rer. nat. Amir Madany Mamlouk
Second referee: Prof. Dr. rer. nat. Jeanette Erdmann
Date of oral examination: 24.11.2011

Approved for printing. Lübeck, 24.11.2011

Niemand kennt den Lauf der Dinge,
trotzdem streben wir voran.
Was das Leben immer bringe,
packen wir es fröhlich an.
N.N

Acknowledgements

I have many people to thank for supporting me during the time of this work. A simple thank you or "takk" cannot at all do them justice.

As a graduate school student I have had the luck of having two supervisors. First of all, I thank Jeanette Erdmann and Amir Madany Mamlouk for always having an open office door when advice was needed, not only for scientific input but also for moral support. I also thank them for the high degree of freedom I was given during this work.

I want to thank Thomas Martinetz for giving me the opportunity to pursue my PhD in a great research environment. I thank Heribert Schunkert for providing me with data and for allowing me to join his research group.

I thank everyone at the Institute of Neuro- and Bioinformatics for contributing to the nice working atmosphere by being not only my colleagues but also my friends. Furthermore, I would like to thank my colleagues from the Medizinische Klinik II, especially Christina Willenborg for being the right person to go to when data problems had to be solved.

Special thanks go to Kai Labusch, with whom I worked together and who taught me so much of what I know about machine learning. Particular thanks go to Martin Böhme, who has been a hero in almost every possible way.

I also thank the undergraduate students whose theses I supervised, Carina Hauft and Ida Massow.

During my time as a PhD student, I was funded by the Graduate School for Computing in Medicine and Life Sciences funded by Germany's Excellence Initiative [DFG GSC 235/1].

A big "takk" goes to all my friends for always supporting me and for all the good times besides work. Finally, my sincerest thanks go to my family for always believing in me and encouraging me over all these years.

Takk!

Contents

Introduction

What makes us ill and what keeps us healthy? What can we do to live a long and healthy life? These questions have probably been asked all the way back to the first humans living on earth.

Is a disease always an outside interference or is it predetermined by our genes? And if it is written into our genes, what can we do at least to influence the course of the disease?

The quest to explain the development of a disease is a complex undertaking; it is a delicate search for clues; it is the work of a detective. All the tiny clues, all the small pieces need to be assembled into one final picture, like the pieces of a puzzle. Indeed, it is even more complicated than that. With an ordinary puzzle, we can only arrange the pieces in one particular way. Now imagine a "disease puzzle" that has more pieces than the number needed to complete the puzzle; that can be solved in several possible ways, all of which lead to the same final picture; and which, finally, and, perhaps, most puzzlingly, contains pieces that can be merged together to form new shapes. This analogy may give a hint of the complexity of the genetics that underlie a disease. I will later come back to this analogy.

Traditionally, people have searched for "visible" factors that cause or increase the risk of a disease: the environmental factors. These factors can be identified by observing the environment and the behaviour of affected individuals. Determining the environmental factors is not straightforward, however, because they are caused by outside influence they have been studied more intensively. But what about the less visible factors? What about the inherited risk?

Unravelling the heritability and genetic causes behind traits has kept scientists busy for centuries. Probably the first scientist who comes to mind in this context is Gregor Mendel, the father of modern genetics. But not only scientists have studied the inheritance of traits. People have long observed the degree of resemblance between parents and their offspring, long before Mendel performed his first experiments. When a child is born, one of the first things people notice is that it has "the eyes of her mother" or "the nose of his father". Hence, it is only logical to posit that not only looks but also diseases may be inherited.

Just as the heritability of eye colors, body height and skin color is complex, so too is the genetic component of most diseases. In fact, most of today's diseases are influenced by many genetic factors. In addition, a trait may be attributed not only to

genetic factors. Body height, for instance, is also influenced by environmental factors, such as nutrition. Altogether, unraveling the genetics behind a disease is a complex undertaking because every disease itself is complex.

It seems clear that we can only influence the course of a disease if we understand the mechanisms that underlie it. A large fraction of the environmental factors that influence diseases have been identified, but it is a different picture when it comes to the genetic factors. The reason for this imbalance is that it is more complicated by far to unravel the genetic mechanisms that influence a disease. We need to take an inside view, a look at the genome. For a long time, a systematic search was not possible and the discovery of new genetic factors progressed very slowly. However, during the past few years, genome-wide association (GWA) studies have enabled an efficient search across the whole genome. Several new genetic factors have been identified in this way and, thanks to GWA studies, we are now slowly beginning to understand the genetic causes of complex diseases. But we are not there yet. A great fraction of the genetic factors of complex diseases have not yet been identified. Some pieces of the puzzle are still missing.

At the beginning, I described the genetics of a disease as a very complex puzzle. The genetic and environmental factors are the basic puzzle pieces. But what did I mean when I described puzzle pieces that merge together to form new shapes? In my analogy, these pieces symbolize epistatic effects. A disease may not be fully explained by the sum of the individual causal factors. Indeed, certain factors, environmental or genetic, may only have an effect at all in combination with other factors. If we only search for individual factors, we cannot expect to fully understand the mechanisms of a disease nor influence its course.

The goal of this work is to propose risk prediction models for GWA data as well as methods for identifying interacting variants and disease structures which may broaden our knowledge of the genetics underlying a disease.

The tools used to achieve this goal will be taken from the area of machine learning. In the vocabulary of this discipline, risk prediction corresponds to classification or regression; the identification of interacting variants corresponds to feature selection. I will describe the development and introduction of new methods into a field in which machine learning applications had only been applied and explored to a limited degree, probably because their application is not straightforward. While this work was being prepared, other groups in parallel were also making progress on this front.

Introducing machine-learning techniques to the field of GWA studies is not only a question of implementing new ideas and writing numerous lines of code, of evaluating the performance on simulated data and testing the performance on real data sets. A compounding difficulty is that we do not know all of the characteristics of the data – how complex its structure is, what its quality level is, and what pitfalls it may harbour. As an example, hours of work are spent only on identifying quality issues that might

otherwise undermine any good approach. I can report, for instance, that excellent classification can be achieved solely by exploiting different missing rates between cases and controls.

In the first part of this thesis, I will explore to what degree risk prediction can be improved by classifying on multiple instead of single variables. One method I will use is the support vector machine, one of the most popular classification algorithms. Another is the well-known Adaboost algorithm, which I will modify for use on GWA data. Both of these classifiers show convincing results and predict disease risk more accurately than would be possible with individual SNPs. I will also evaluate how the prediction quality depends on the number of variants used and show how we can use the support vector machine to rank variants according to their relevance for the disease. The modified Adaboost algorithm is published in [1].

The second part of this thesis deals with feature selection. The most common way of selecting the most relevant variants is to rank them by p-value, a statistical measure of significance. However, this ranking assigns high values only to variants that have an individual effect; interactions between variants are ignored. I will show how interactions can be identified using the sparse coding algorithm, which can find patterns in multi-dimensional data. On simulated data that contains a known pattern of interacting SNPs, the sparse coding algorithm successfully identifies these SNPs. This result carries over to real data: On one of the real data sets, if the support vector machine is restricted to use only the relevant SNPs identified by sparse coding, it predicts disease risk better than if it is applied to all SNPs. Using sparse coding to identify SNP patterns is published in [2].

In the third and final part, I will show how a good classification can be achieved with only a very small number of variables. This result is important because classifiers based on large ensembles of SNPs are difficult to interpret. If the number of SNPs is small, we have a real chance of understanding *why* the classification is successful, and this, in turn, may give us new insights into the genetic causes of the disease.

I approach this problem by introducing a sparse linear regression algorithm; this algorithm performs a combination of feature selection and classification. I show that the algorithm can be applied to large data sets without any preselection step, unlike other algorithms, which often require a preselection of variables to ease the computational burden. On one data set, I was able to identify a truly disease-specific variable that could not be linked to the disease by its individual effect. The sparse linear regression algorithm for interpretive risk prediction is introduced in [3].

Much of the work that I have done over the last years has been as part of a group; for consistency, I will use the personal pronoun “we” throughout the thesis even if the results are solely my own.

Part I

Basics

1 On the trail of genome studies

Genetic factors play an important role in the pathogenesis of most of today's diseases. One can broadly categorize these diseases into two classes: Mendelian diseases and complex diseases. Mendelian diseases are caused by a mutation in a single gene and are typically rare in the general population. In contrast, complex diseases are expected to be largely multifactorial and are much more common in the population. Hence, they are often referred to as common complex diseases.

The genetic influence is probably most obvious when it comes to Mendelian diseases, where a mutation in a single gene is sufficient. The genetic influence on complex diseases, such as inflammatory bowel disease (IBD) and coronary artery disease (CAD) is on the other hand less clear. Twin studies are one possibility to estimate the heritability of a complex disease. The heritability is calculated by comparing the risk between monozygotic and dizygotic twins given that one twin is diseased. Whereas the monozygotic twins share nearly all of their genes, dizygotic twins share, on average, about 50% of their genes [4]. We can estimate the extent to which the genetic variation influence the disease if we compare the occurrence of a trait for monozygotic twins with the occurrence for dizygotic twins. The heritability is calculated as twice the difference between the correlations of the monozygotic and dizygotic twins [5]. For most common complex diseases the heritability is estimated to range from 30% to over 50% [6].

Whereas Mendelian diseases are caused by rare variants, complex diseases are expected to be largely influenced by common variants. The so-called common-disease-common-variant (CD/CV) hypothesis assumes that common genetic variants play a role in the development of common diseases [7]. The premise underlying the CD/CV hypothesis is that common variants are responsible for 99% of the variation in the population. Most susceptibility variants are expected to have only small effects, and hence several variants will influence common diseases [8]. For a long time, common variants were expected to play the major role in the development of common diseases. However, this view has recently changed and rare variants are now also thought to have a large influence [9].

Before the age of genome-wide association (GWA) studies, there were mainly two strategies to unravel the genetic cause of a disease: linkage analysis and the candidate gene approach. Linkage analyses genotyped families with numerous affected individuals. This dataset is searched for genetic loci that are inherited along with a disease.

The candidate gene approach concentrates on specified genes that are expected to have an influence on a disease. These candidate genes are chosen based on a-priori assumptions. The main disadvantage of the candidate gene approach is that it only accounts for the variants within the selected region and hence misses the possible causal variants at other locations in the genome [10]. The linkage approach does not concentrate only on specific genes; however, it may not have enough power to detect causal variants due to the limited number of individuals in the scanned families [10].

Linkage analyses have been very successful in identifying the genes that underlie Mendelian diseases. The success of linkage studies is mainly due to the fact that the disease specific gene can readily be traced back to affected individuals in the pedigree [10]. Common complex diseases, on the other hand, do not have a clear pattern of inheritance, since several factors, and no single gene mutation, cause the disease [11]. Hence it is not surprising that, unlike for Mendelian diseases, the linkage analysis has been disappointingly unsuccessful in unraveling the genetic causes of common complex diseases. Over the years, linkage studies have identified only about a dozen genetic variants (aside from the HLA locus) that they could reproducibly associate with a common complex disease. To put this number into relation: in the last decade, GWA studies have identified around 1000 variants [8].

GWA studies offer a solution to the drawbacks of both linkage and candidate gene studies [8]. First, GWA studies have greater power to detect common disease variants than linkage studies and second, GWA studies allow a broader coverage of the genome than candidate gene studies [10]. GWA studies have revolutionized our perspective on the function and information content of the human genome. GWA studies allow us to analyze the genome without any hypotheses or use of a-priori knowledge and hence allow an unbiased investigation of the biological and genetic causes of a disease. The breakthroughs that GWA studies have achieved have only been possible because of the new genotyping technologies [12]. With the ability to sequence the human genome with genotyping arrays, often referred to as SNP chips, it is now feasible to genotype large numbers of SNPs. Today, these SNP chips allow up to 2 million variants to be genotyped simultaneously [8]. In addition, imputation methods enable to predict the genotypes of SNPs that are not directly genotyped in the study. This increases the power and coverage of GWA studies [13]. By testing large number of variants distributed all across the genome, GWA studies have been able to identify new regions not considered so far, and are hence slowly beginning to unravel the genetic causes of complex diseases.

1.1 What have we learned from GWA studies so far?

Genome-wide association studies have been tremendously successful in identifying new loci associated with common complex diseases. On the basis of coronary artery disease (CAD) and myocardial infarction (MI) one can clearly see the progress over the last few years.

In the first wave of GWA studies, three independent studies were published that identify genetic loci for coronary artery disease [14–16]. Surprisingly, all three studies reported significant association for the same locus on chromosome 9p21.3. No prior studies had so far linked nor assumed this locus to be associated with CAD. The chromosome 9p21.3 locus clearly demonstrates the value of GWA studies. Only the systematic search for inherited components all across the genome can now easily identify loci that would be neglected in candidate gene analysis and would not be identified in linkage analysis due to lack of power.

The 9p21.3 locus lies in a non-coding region and has so far not been linked to any gene. The closest genes are CDKN2A, CDKN2B and ARF. These genes are however about 100 kb away, making it difficult to link the locus to a causal variant that influences CAD. However, it appears that the 9p21.3 locus harbors a non-coding RNA gene (ANRIL) that affects the regulation of other genes [17, 18]. Since the first studies reported the 9p21.3 locus, this locus has been validated in several subsequent studies [19].

In addition to the 9p21.3 locus, the first GWA studies identified several other disease loci that were also unexpected because they were not linked to human disease mechanisms. For instance, a locus on chromosome 3p22.3 was linked to the pathogenesis of MI [14]. The locus is found in the muscle RAS oncogene homologue gene (MRAS), which has a high expression rate in several tissues, especially in the heart. The MRAS locus was previously linked to inflammation [20], and the results indicate a role of MRAS in adhesion signaling, which is important in CAD [21].

Besides identifying unforeseen susceptibility loci, the studies also identified a number of loci that can be linked to traditional risk factors. A prominent example is the chromosome 1p13.3 locus [14]. This locus has, after it was first associated with CAD, been linked to LDL cholesterol (LDL-C) and a decrease in LDL-C levels [22, 23].

Subsequent GWA studies replicated some of the reported loci and identified new variants. After the first wave of GWA studies, about a dozen loci were associated with CAD [21]. The heritability of CAD is expected to account for 30 to 60% of the risk [24, 25]. However, the variants identified in the first wave explain only a small fraction of the expected heritability. Some of the missing heritability is expected to be found within the variants with small effects or small allele frequencies.

Since the first wave of GWA studies had insufficient power to identify such variants, the second wave of GWA studies aimed to increase the power by increasing the

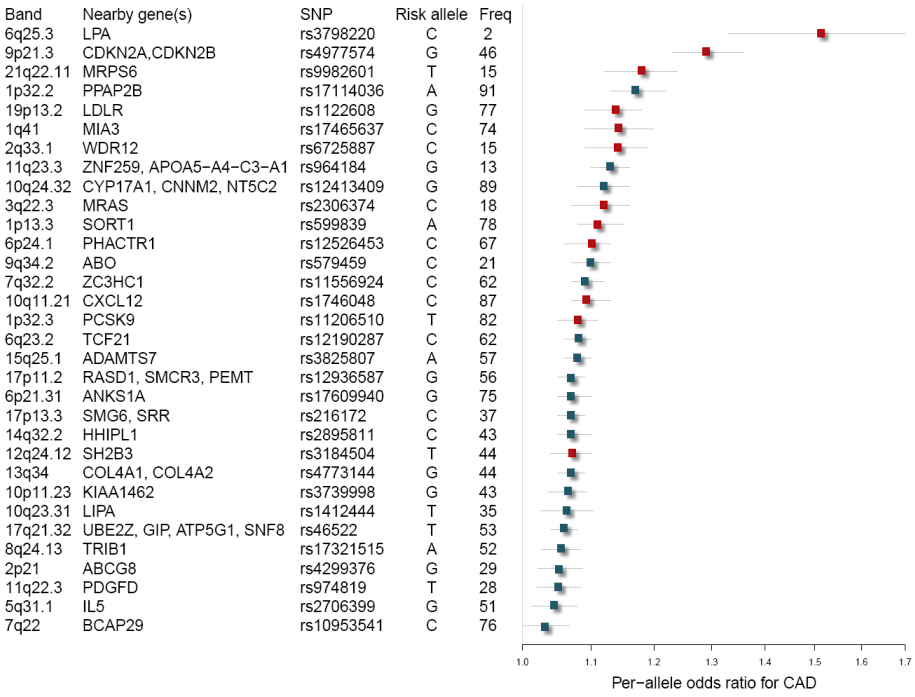


Figure 1.1: Illustration of the CAD- and MI- associated loci detected in the first and second wave of GWA studies. The red- and the blue squares represent the first and second wave loci respectively. Of the loci that were detected in the second wave, all but one show small effect size, hence previous GWA studies lacked the power to identify these loci. The one locus with stronger effect size has a small minor allele frequency which prevented this locus from being detected in the first wave. (Reproduced with permission of Jeanette Erdmann.)

sample size. The CARDIoGRAM consortium [26] assembled more than 22,000 cases and 64,000 controls of European ancestry from 14 GWA studies on CAD for a meta-analysis and additionally more than 56,000 individuals for a replication study. 10 of the 12 previously reported loci were validated by the meta-analysis and replication study, and an additional 13 new loci were associated with the risk of CAD. Of the 13 new loci, all but one show small effect size, hence previous GWA studies lacked the power to identify these loci, as shown in Figure 1.1. The one locus with stronger effect size has a small minor allele frequency, and hence the GWA studies of the first wave also lacked the power to detect this locus. The majority of the newly identified variants lie in gene regions that were not so far linked to the pathogenesis of CAD. Hence, the variants indicate that previously not identified pathways may play a role in the development of CAD.

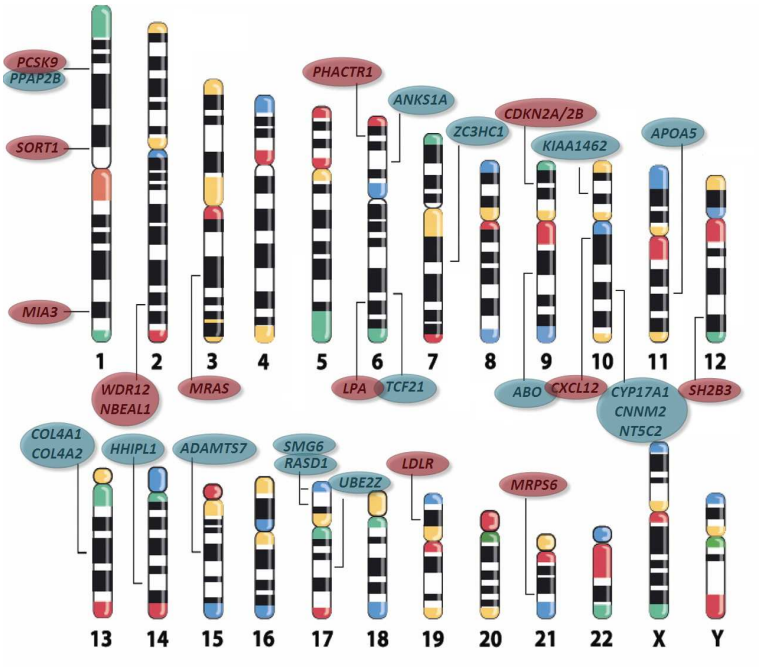


Figure 1.2: Schematic illustration of the genetic loci associated with coronary disease and myocardial infarction. The red- and the blue colour represent the first and second wave loci respectively. (Reproduced with permission of Jeanette Erdmann)

Despite the efforts of the second wave of GWA studies, the CAD-associated loci identified so far still explain only about 10% of the heritability of CAD [26]. The relatively small fraction of explained heritability is however consistent with the results of most complex diseases [27].

After the first years of GWA studies the question arises: How can we transfer the GWA study results to clinical medicine, risk prediction and understanding of human biology and pathophysiology? In order to use the new knowledge gained by GWA studies for treatment and prevention, efforts have to be made to understand the genetic cause of a disease by analysing the function of the genes affected by the variants as well as the mechanisms of the affected pathways. Linking the disease-associated variants to biological mechanisms is a key step in improving our understanding of a disease. Some of the identified variants have been linked to pathways whereas other variants are studied in animal models to better understand the biological process. However, only a small fraction of the identified variants can so far be linked to the causal mechanism, and the identification of disease-associated variants has not been completed.

1.2 The missing brick

Despite the success of GWA studies for complex diseases, the genetic heritability is far from being explained. The variants identified so far explain only a small fraction of the expected heritability. The rest of the heritability remains to be identified. Meta-analyses have more power to identify rare variants and variants with small effects, and further loci might hence be identified.

But how many more variants will need to be identified to fully explain the heritability? And will these variants be relatively few, but with a strong effect, or comparatively many, each with only a weak effect? The heritability of a disease is mainly calculated based on family studies, these calculations are generally based on the assumption that the effect of multiple variants present simultaneously is the sum of the individual effects [27]. In other words, the assumption is that multiple variants combine additively but that there is no interaction or epistasis between the variants. If there is significant interactions, then the number of variants required to fully explain the heritability might be smaller than that which we would expect if we assume that multiple variants combine purely additively [8]. It therefore seems important that studies of disease heritability should be able to account for gene-gene and gene-environment interactions. This is an area where GWA studies that focus on single-locus effects alone have a substantial weakness in that they may not identify the more complex genetic mechanisms underlying multifactorial traits [28–30]. Indeed certain variants may only have an effect at all in combination with other variants; such variants would be missed entirely by single-locus studies.

1.3 How do we get from disease association to risk prediction?

So far, GWA studies have enabled for a systematic search for genetic variants associated with a disease. Besides the aim of understanding the genetic cause of a disease, a further goal is to allow a risk prediction based on genetic variants. However, whether the variants that have been identified so far or that may be identified in the future will be sufficient to make useful risk prediction remains to be seen. Some efforts have been made to incorporate identified variants in a risk score [26, 31]. However, it seems that the identified variants explain too little in order to permit a suitable risk prediction.

This failure of risk prediction may however not only be due to the lack of variants explaining the heritability of a disease, but also due to the fact that interactions between variants are ignored. A risk score based on the additive combination of known disease variants does not cover interactions between the variants. In addition, as previously discussed, a disease effect might only come about through an interaction of variants. If we only account for single-effect loci, we will miss these variants. Interac-

tion analysis might hence not only be of value in to describe more of the heritability of a disease, but also to permit an improved risk prediction.

Epistatic effects are, however, not straightforward to identify. The main reason for this is the large number of combinations that have to be tested. For example, assume that a GWA study has genotyped only 1000 loci. To identify all single-locus effects, 1000 test have to be performed. If we wish to identify all potential interactions of two loci, we already need to perform a million tests; to identify all three-locus interactions, we need to perform a billion tests; and so on. This combinatorial explosion is even more severe for the hundreds of thousands of loci found in today's datasets. Hence, suitable feature selection and classification methods as well as a combination of both are needed to obtain satisfactory risk prediction and a better understanding of the mechanisms underlying a disease.

1.4 The goal

The goal of this work is to predict disease risk from GWA data as well as to identify interacting variants and thus better understand the cause of a disease. Several disease-associated loci have been identified through GWA studies. Initially, evaluations of GWA studies focused mainly on single-locus effects, but research groups are now starting to search for more complex structures. As we saw above, the large number of loci in today's GWA studies makes it impossible to test all possible combinations of variants for disease-associated interactions. Suitable feature selection approaches are thus needed to identify a subset of variants that should be studied further. A standard feature selection approach is to select certain variables based on their individual effects. The problem with this approach is that some variants may only have an effect in combination, but not individually. Since such variants will obviously not be found if we look only at individual effects, there is a clear need for alternative approaches. We will in this work suggest feature selection approaches that select combinations of variants largely independent of their individual effects.

Whether genetic risk factors are useful in a clinical setting depends strongly on their predictive value. A variant that only causes a small increase in risk does not on its own permit a useful risk prediction. A better risk prediction may be possible if several such variants are combined in a suitable way. So far, studies that have attempted to do this only on known risk factors have not been successful in making useful risk predictions. One reason for this may be that, in general, these methods only account for additive effects and ignore possible interactions, even though such interactions may have a strong effect on risk. Thus, there is a clear need for more complex methods that will account for these interactions. We will in this work introduce and adapt existing

classification approaches to the field of GWA studies and use them to perform risk prediction.

Most GWA studies typically genotype several thousands of SNPs all across the genome. The number of genotyped variants is expected by far to exceed the number of variants that will be associated with the disease. Classification methods that incorporate large numbers of variants hence also incorporate large numbers of variants that do not influence disease risk. Indeed, this noise may even make the classification worse than if it were based on only a few true risk variants. Moreover, a classifier based on a large number of SNPs is difficult to interpret if our goal is to understand the genetic mechanisms underlying a disease.

Classification is generally improved if the data include interactions between the variants. If we look for a small subset of SNPs that permits a close to optimal classification, this subset will probably include these interacting variants. Hence, we will in this work also build classifiers that are only based on a small but optimal subset of SNPs. With this method we aim to improve classification as well as to identify possible epistatic effects and broaden our understanding of the genetic structure underlying a disease.

2 Data

In this chapter, we will describe the various data sets that are used in this thesis in detail. These data sets encompass simulated and real GWA data and will be used to test the algorithms developed in later chapters.

2.1 Simulated Datasets

Simulated data sets are of great use for evaluating and comparing feature selection and classification algorithms because the ground truth is known and can be compared to the results of the algorithms and because the exact properties of the data set can be controlled. Real GWA data have known and unknown properties that can influence the performance of an algorithm. Since we know little about the complex structures and relationships between SNPs that may be contained in these data sets – after all, the very purpose of this work is to find these structures – it can be difficult to understand why an algorithm behaves the way it does on these data sets.

Simulated data sets have the advantage that we can control the effect size, frequencies of the alleles and the number of relevant variables. Moreover, we can incorporate patterns of a specific size into the data. Because we know which SNPs are disease-relevant, we know exactly which of these SNPs a particular algorithm has found or not found. This helps us to adjust and improve the algorithms. If we know the possibilities and limits of the algorithms, we may also better interpret the results on the real data sets.

2.1.1 Simulated data sets with disease-specific and disease-unspecific patterns

Besides individual effects, we expect that disease-specific structures and interactions between SNPs also influence the risk of a disease. Disease-specific structures may however not be specific for all affected individuals; they may be characteristic only for subgroups of the affected individuals. In addition, environmental factors play a large role in the development of most complex diseases. Hence, genetic risk factors need not even be the main cause of the disease in all of the affected individuals.

SNPs that lie only a few base pairs apart are generally inherited together. In addition, some combinations of SNPs, which need not necessarily be SNPs with neighbouring base pair positions, occur more or less often than we would expect if they were

statistically independent. Such an association between SNPs is referred to as linkage disequilibrium. The fact that linkage disequilibrium is widespread in the genome implies that we have disease-unspecific patterns in the data, and this can influence the performance of feature selection and classification algorithms.

To summarize, a particular disease-specific pattern may not occur in all affected individuals, some cases may not be distinguishable from the controls by their genotype, and there may be patterns that are disease-unspecific.

In the following we will describe how we simulate data sets with disease-specific and unspecific patterns. We generate the data in two steps: We first generate a basis set of genotypes with various realistic statistical properties; we then introduce the patterns. The first step, generating the basis set, is done in the following way. We generate genotypes by random sampling, with the constraint that the distribution of the genotypes is in accordance with the Hardy-Weinberg equilibrium (HWE) [32]. The MAF of the simulated SNPs lies between 0.4 and 0.5. To obtain a set of cases and controls, we simulated two data sets with 500 individuals each.

The next simulation step introduces SNP patterns. Since we want to obtain a data set where only the pattern and not the individual SNP are associated with the disease, we cannot alter the frequency of the SNPs. If we did alter the frequency of a SNP in one of the two groups, this would cause the individual SNP to be associated with the disease. To avoid this, we introduce patterns by re-ordering the genotypes for the SNPs involved in the pattern. More precisely, we choose a subgroup of individuals where we want to introduce a pattern and a set of SNPs that should contain the pattern. We want this set of SNPs to have the same genotype for each individual in the subgroup. For each individual, we check whether each SNP has the desired genotype; if not, we exchange it with an individual outside the subgroup that does have the desired genotype. The size of the subgroup is restricted because we also want some individuals outside the subgroup to have the same genotype. This is also the reason why we simulated common instead of rare variants. In the results shown here, we will use a subgroup size of 100 cases.

To simulate unspecific structures in the data, we additionally incorporate patterns that occur in subgroups of affected and unaffected individuals. The patterns are simulated in the same way as described above, but now for both classes. Each subgroup contains 100 cases and 100 controls.

The size of the disease-specific and unspecific patterns, i.e. the number of the SNPs in the pattern, and the total number of SNPs are adjustable parameters.

A closer look at the data

In the following we will take a closer look at the simulated data. An example of a simulated data set is shown in Figure 2.1. The data is plotted as a matrix where

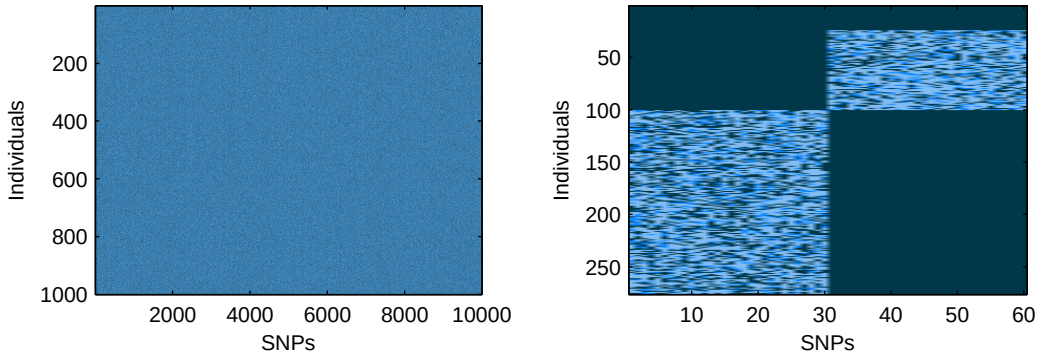


Figure 2.1: Image of the data set with disease-specific and disease-unspecific patterns. The data set contains 500 cases and 500 controls with 10,000 SNPs and is displayed as a matrix where the three possible genotypes for a SNP are encoded by light blue, medium blue, and dark blue. There are two patterns in the data, one disease-specific and one disease-unspecific. The individuals participating in the two patterns partly overlap. The SNPs of the patterns do not overlap. a) The genotypes of the complete data set. b) The genotypes of the individuals with a disease-unspecific or disease-specific pattern. Only the pattern-specific SNPs are shown.

different color shading (from light to dark) encodes the genotype. Figure 2.1a shows the complete data set, which consist of 1000 individuals, of which 500 are cases and 500 are controls. The total number of SNP is 10,000. We have incorporated two patterns into the data, one disease-specific and one disease-unspecific. As we see, the patterns cannot be identified by visual inspection of the complete data set. The reason for this is that the patterns are very small and the SNPs and individuals participating in the patterns are not neighbouring. Figure 2.1b plots only the individuals and SNPs participating in the two patterns. Now, we can clearly see that a subset of SNPs have identical genotype for a few individuals.

Next, let us take a closer look at the individual SNPs. We will evaluate the same data set as above. Figure 2.2a plots the p-values for all SNPs of the data set. The pattern-specific SNPs are marked with red circles. We see that the SNPs are not individually associated with the phenotype. Figure 2.2b plots the odds ratios (ORs) for all SNPs; most of the SNPs have an OR of around 1. The pattern-specific SNPs cannot be identified by their ORs. In summary, the simulated data set consists of disease-specific and disease-unspecific patterns where the individual SNPs are neither associated with the disease nor have large ORs.

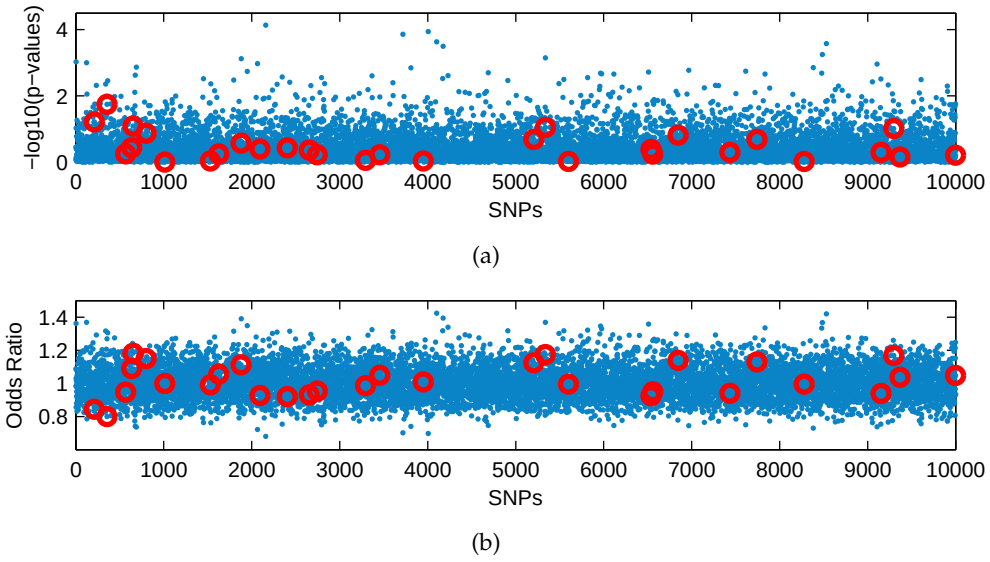


Figure 2.2: Data set with one disease-specific and one disease-unspecific pattern. The data set contains 500 cases and 500 controls with 10,000 SNPs. a) p-values of the SNPs, b) odds ratios of the SNPs.

2.2 Genome-wide association data

In this thesis, we will test our algorithms on two different GWA data sets: a coronary artery disease data set and an inflammatory bowel disease data set.

2.2.1 Coronary artery disease data set

The coronary artery disease (CAD) data set originates from the German MI family study II [33]. This data set was included in the large CARDIoGRAM meta-analysis [26] (see Chapter 1). It contains 2506 individuals, of which 1222 are cases and 1284 are controls, 1643 are males and 1284 are females. All individuals are Caucasians of European origin. All cases had a validated myocardial infarction prior to an age of 60 years. Approximately half of the cases have a positive family history for CAD. The control individuals are population based. The total number of SNPs is 478,040. The missing values are imputed [34].

We filtered the data with standard quality criteria: MAF $<1\%$ and a maximum deviation from the Hardy-Weinberg equilibrium (HWE) of 10^{-4}. There are no missing values in the data because these values were imputed. Hence, we did not need to apply a threshold to restrict the number of missing values. After quality filtering, the total number of SNPs is 432,189.

As we discussed in the previous section, SNPs that lie only a few base pairs apart are generally inherited together. This means that these SNPs carry the same information and are hence to a certain degree redundant. In addition, some combinations of SNPs, which need not necessarily lie close to each other, occur more often than we would expect by chance. Such associations between SNPs are referred to as linkage disequilibrium (LD). To obtain a more manageable data set, we reduced the number of SNPs by applying LD pruning. This removes much of the redundant information, since groups of LD SNPs that lie close together on the chromosome and hence have a similar inheritance pattern are now represented by only a few SNPs.

We LD pruned the data set using the PLINK software with the "indep" option. The LD pruning in PLINK is based on the variance inflation factor (VIF) and recursively removes SNPs within a sliding window [35]. The VIF is given by $\frac{1}{1-R^2}$, where R^2 is the measure of LD (see Chapter 3).

We used the standard settings: Window size of 50 SNPs, window shift of 5 SNPs in each step, and a VIF threshold of 2. After LD pruning, the remaining number of SNPs is 118,247. For cross-validation, we divided the data into five sets with equal numbers of cases and controls; each cross-validation fold consists of 488 individuals.

A closer look at the data

Figure 2.3 plots the p-values and OR for all SNPs, ordered by base pair position. Alternate chromosomes are shaded light and dark. 13 SNPs are genome-wide significant at a significance threshold for the p-values of 10^{-8}; this threshold is corrected for multiple testing. The MAF for these SNPs lies between around 1% and 15%, and the the ORs range from 0.1061 to 19.89; these are relatively large effect sizes. 4 of the significant SNPs have an MAF slightly larger than 5%, and the ORs for these SNPs range from 0.31 to 2.56. The SNPs that have been validated in the literature to be associated with CAD are not among the SNPs that are significant in this data set and have smaller effect sizes. This may indicate that the effect sizes for some of the SNPs found to be significant here are artefacts. Note also that the search for significant SNPs was performed on the LD pruned data set; this may explain why the significant SNPs did not include those previously reported to be associated with CAD.

2.2.2 Inflammatory bowel disease data set

The inflammatory bowel disease (IBD) data set was obtained from the public database dbGap run by the NCBI [36,37]. This data set was included in the large Crohn's disease meta-analysis published in [38]. The data set contains 1760 individuals, of which 813 are cases and 947 are controls. Approximately half of the data set are Crohn's disease (CD) patients and controls of non-Jewish European ancestry. The other half of the data

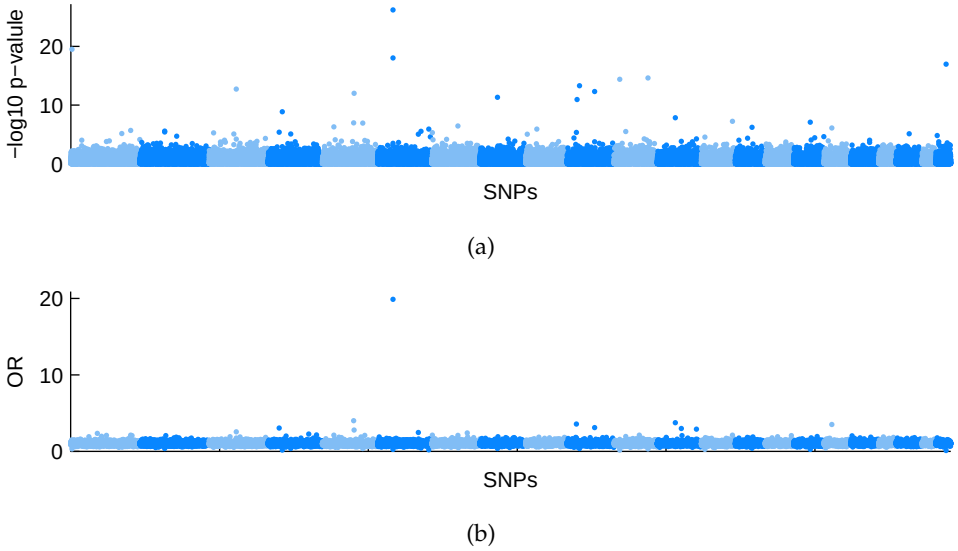


Figure 2.3: CAD data set. Alternate chromosomes are shaded light and dark. a) p-values, b) odds ratios.

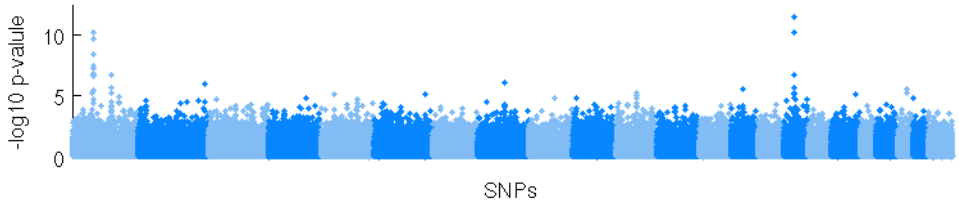
set contains individuals of Jewish ancestry. The total number of SNPs is 317,503. We applied the same quality criteria as for the CAD data set: MAF <1% and a maximum deviation from HWE of 10^{-4}. Unlike the CAD data set, the IBD data set contains missings that have not been replaced by imputed values. Therefore, we also applied a missing threshold of 2% per SNP. Since some of the individuals have a high missing rate, we additionally applied a threshold of 6% missings per individual.

The data set remaining after quality filtering contains 1719 individuals, of which 789 are cases and 930 are controls. The total number of SNPs is 292,162. No LD pruning was applied to this data set.

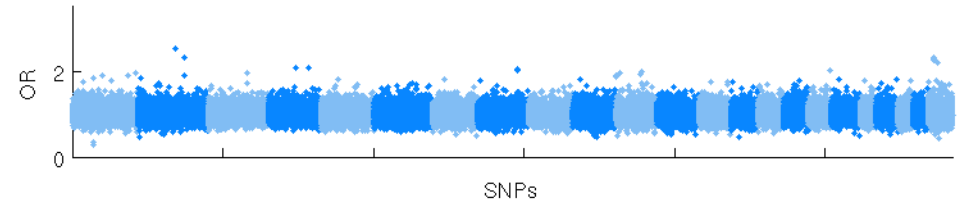
For cross-validation, we divided the data into five sets with equal numbers of cases and controls; each cross-validation fold consists of 314 individuals.

A closer look at the data

Figure 2.4 plots the p-values and OR for all SNPs, ordered by base pair position. Alternate chromosomes are shaded light and dark. 5 SNPs are genome-wide significant at a significance threshold for the p-values of 10^{-8}; this threshold is corrected for multiple testing. The ORs of these SNPs range from 0.29 to 1.69 with an MAF of 5% to 36%. The significant SNPs lie within two genes, the NOD2 and the IL23R gene. Both are known susceptibility genes for IBD [39].



(a)



(b)

Figure 2.4: IBD data set. Alternate chromosomes are shaded light and dark. a) p-values, b) odds ratios.

3 General methods

3.1 Sparse approximation methods

In many machine learning applications, a classification problem can be simplified by finding the most important features that best represent the given data. A simple way of doing this is to select the most informative dimensions in the data and discard the rest. A more sophisticated way is to specify a set of basis vectors (chosen so that they can represent the data well) and to represent each data point as a linear combination of these basis vectors. If we add the constraint that only a certain small number of basis vectors may be used per data point, we call the resulting representation a *sparse* representation. Finding such a sparse representation of the data is however not straightforward.

To put this in mathematical terms, let $\mathbf{x}$ be the data point to be represented, and let $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ be a matrix of basis vectors. We then seek a coefficient vector $\mathbf{w} = (w_1, \ldots, w_M)$ such that

$$\mathbf{x} = \sum_{m=1}^{M} \mathbf{c}_m \cdot w_m. \tag{3.1}$$

For a sparse estimation problem, we additionally have the constraint $\|\mathbf{w}\|_0 \leq k$[1]. Under this sparseness constraint, it is typically no longer possible to find an exact representation for $\mathbf{x}$, and instead we seek to find the best possible approximation by minimizing the objective function

$$g(\mathbf{w}) = \mathbf{x} - \sum_{m=1}^{M} \mathbf{c}_m \cdot w_m. \tag{3.2}$$

This is, in general, an NP-hard problem [40]; in other words, finding the best approximation would require all $\binom{n}{k}$ possible solutions to be tested. However, there are a number of approximation methods that yield close to optimal solutions [41].

In the problem statement given above, we have assumed that the basis vectors C are given and that $\mathbf{w}$ should be determined; this is the problem we will be dealing with in the rest of the section. In Section 3.2, we will then deal with the problem of how to find a suitable set of basis vectors C.

[1] the notation $\|\mathbf{w}\|_0$ denotes the number of non-zero entries in $\mathbf{w}$

3.1.1 Orthogonal Matching Pursuit

One popular approximation method is orthogonal matching pursuit (OMP) [42]. OMP is a simple algorithm that aims to identify the most important basis vectors in a greedy fashion. The algorithm keeps track of the residual $\boldsymbol{\epsilon}$, i.e. the remaining approximation error at a given point in the algorithm, which is computed as

$$\boldsymbol{\epsilon} = \mathbf{x} - C\mathbf{w}^{\mathrm{OMP}}, \tag{3.3}$$

where $\mathbf{w}^{\mathrm{OMP}}$ is the coefficient vector of the current iteration.

In each iteration, the algorithm identifies the basis vector that has the highest overlap (i.e. scalar product) with the residual. The index l_{win} of this "winning" column is obtained as

$$l_{\mathrm{win}} = \underset{l, l \notin U}{\arg\max} \left(\mathbf{c}_l^{\mathrm{T}} \boldsymbol{\epsilon}\right)^2, \tag{3.4}$$

where U denotes the set of indices of those basis vectors that have already been selected. Initially, $U = \emptyset$ and $\boldsymbol{\epsilon} = \mathbf{x}$.

After each iteration $\mathbf{w}^{\mathrm{OMP}}$ is estimated by solving the optimization problem

$$\mathbf{w}(U)^{\mathrm{OMP}} = \underset{\mathbf{w}(U)}{\arg\min} \|\mathbf{x} - C^U \mathbf{w}(U)\|_2^2, \tag{3.5}$$

where C^U denotes the columns of C that are indexed by U. Initially, all weights are set to zero, $\mathbf{w}^{\mathrm{OMP}} = 0$. Subsequently, the new residual is computed, and U is updated with $U := U \cup l_{\mathrm{win}}$.

The described steps are repeated for a fixed number of k iterations, yielding a set of k basis vectors with indexes U.

The columns of C are in general not pairwise orthogonal. Hence, the way the winning column $\mathbf{c}_{l_{\mathrm{win}}}$, is selected is not optimal in terms of minimizing the residual that is obtained after $\mathbf{c}_{l_{\mathrm{win}}}$ has been added [43]. An optimized version of OMP, the so-called optimized OMP (OOMP) [44], employs a selection criterion that is optimized with respect to this matter.

3.1.2 Optimized Orthogonal Matching Pursuit

Whereas the OMP algorithm selects the "winning" column by calculating the maximal overlap, the OOMP algorithm selects the column $\mathbf{c}_l$ that yields the smallest residual after it is included. This is done by solving the following minimization problem:

$$l_{\mathrm{win}} = \underset{l, l \notin U}{\arg\min} \min_{\mathbf{w}(U \cup l)} \|\mathbf{x} - C^{U \cup l} \mathbf{w}(U \cup l)\|_2^2. \tag{3.6}$$

As for the OMP method, U is subsequently updated with $U := U \cup l_{\text{win}}$, and again, $\mathbf{w}^{\text{OOMP}}$ is calculated by

$$\mathbf{w}(U)^{\text{OOMP}} = \underset{\mathbf{w}(U)}{\arg\min} \|\mathbf{x} - C^U \mathbf{w}(U)\|_2^2, \tag{3.7}$$

and the current residual is obtained by

$$\boldsymbol{\epsilon} = \mathbf{x} - C\mathbf{w}^{\text{OOMP}}. \tag{3.8}$$

The algorithm is repeated until k iterations have been performed.

In order to reduce the computational complexity, the OOMP algorithm can be implemented using a temporary matrix R, which contains a copy of the basis vectors C orthogonalized with respect to C^U, as described by Labusch et al. [43]: Initially $R = (\mathbf{r}_1, \ldots, \mathbf{r}_M) = C$. Conceptually, R is obtained by first projecting the columns of C onto the columns of C^U and then subtracting this projection from C; in practice, as we will see, we only need to orthogonalize with respect to the current winning basis vector. The residual $\boldsymbol{\epsilon}$ is correspondingly orthogonalized with respect to C^U. Initially, $\boldsymbol{\epsilon} = \mathbf{x}$.

By orthogonalizing R in this way, the winning basis vector can now easily be found by looking for the orthogonalized basis vector $\mathbf{r}_l$ that has maximal overlap with the residual $\boldsymbol{\epsilon}$:

$$l_{\text{win}} = \underset{l, l \notin U}{\arg\max} \left(\mathbf{r}_l^{\mathrm{T}} \boldsymbol{\epsilon}\right)^2. \tag{3.9}$$

After each iteration, the orthogonal projection onto $\mathbf{r}_{l_{\text{win}}}$ is removed from R and $\boldsymbol{\epsilon}$:

$$\mathbf{r}_l := \mathbf{r}_l - \left(\mathbf{r}_{l_{\text{win}}}^{\mathrm{T}} \mathbf{r}_l\right) \mathbf{r}_{l_{\text{win}}}, \tag{3.10}$$

$$\boldsymbol{\epsilon} := \boldsymbol{\epsilon} - \left(\mathbf{r}_{l_{\text{win}}}^{\mathrm{T}} \boldsymbol{\epsilon}\right) \mathbf{r}_{l_{\text{win}}}. \tag{3.11}$$

As previously described, U is updated after each iteration by $U := U \cup l_{\text{win}}$. (Note that, due to the orthogonalization, all of the $\mathbf{r}_l$ with $l \in U$ become zero.) The $\mathbf{r}_l$ with $l \notin U$ are normalized in each iteration step.

The algorithm terminates when the stopping criterion $|U| = k$ is reached. The final $\mathbf{w}$ is obtained by recursively determining the contribution of each basis vector in C used during the iteration in question with respect to the normalization of R in each iteration.

3.1.3 Bag of Pursuits

The bag of pursuits (BOP) method [45] is derived from the OOMP method and is a further optimization. In general terms, the OOMP method is extended such that not only the best but the P best coefficient vectors $\mathbf{w}^j$ $(j = 1, \ldots, P)$ are determined;

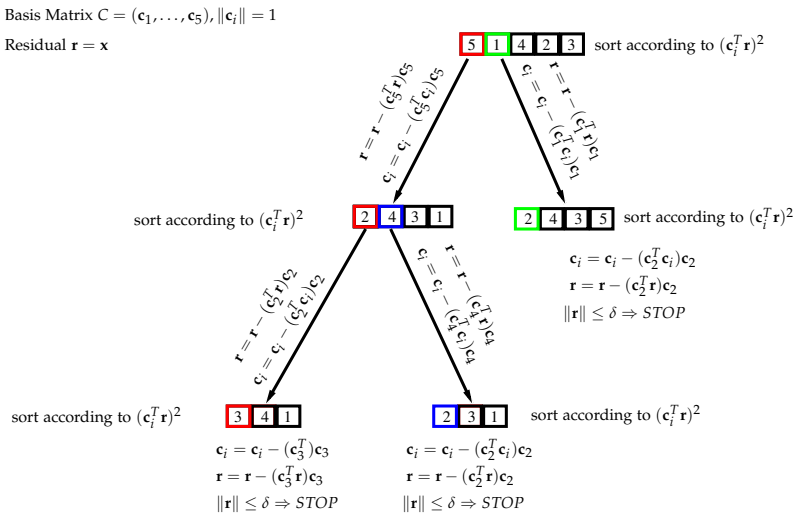


Figure 3.1: BOP algorithm. The BOP algorithm performs a tree-like search for a predefined number of approximations, or *pursuits*. In the first step, the basis vector that has the largest overlap with the residual is selected. The remaining basis vectors and the residual are orthogonalized with respect to to the winning basis vector. The pursuit is continued until a stopping criterion is reached. The second pursuit is obtained by find the maximum overlap that has not been used so far and continuing the pursuit from this point. This figure shows the computation of 3 pursuits; the columns are sorted according to the overlaps. The colour highlights the identified solutions. (Image courtesy of Kai Labusch, reproduced with permission) Note that this illustration uses a different stopping criterion ($\|\mathbf{r}\| \leq \delta$) than the one used in the text (number of basis vectors used is equal to k).

each of these solutions is called a *pursuit*. If $P = 1$, the single pursuit obtained is identical to the OOMP solution; more generally, the first pursuit is the OOMP solution. After the P pursuits have been determined, we choose the one that yields the lowest approximation error.

The general idea behind the BOP algorithm is as follows: The first pursuit is computed as for OOMP. To compute the second pursuit, the algorithm revisits all of the decisions where a winning basis vector was chosen and determines the overlap of each "runner up", i.e. the basis vector with the second largest overlap in that step. Among the runners up, the algorithm then chooses the one with the largest overlap and continues from this point, replacing the winning basis vector of that step with the runner up. This procedure is repeated for the remaining pursuits. Figure 3.1 illustrates the principle of the algorithm.

We will now explain the algorithm in more detail.

The selection of the basis vector proceeds as for the OOMP algorithm, i.e. the BOP algorithm selects the basis vector that has maximum overlap with the residual. How-

ever, we now need to parameterize all of the quantities by the pursuit index j as well as the iteration number n; this corresponds to the number of basis vectors that have been used up to that point.

To simplify the equations, we introduce the auxiliary quantity $\mathbf{y}(j,n)$, a vector containing the overlaps for iteration n of pursuit j:

$$\mathbf{y}(j,n) = \big((\mathbf{r}_1(j,n)^{\mathrm{T}}\boldsymbol{\epsilon}(j,n))^2, \ldots, (\mathbf{r}_l(j,n)^{\mathrm{T}}\boldsymbol{\epsilon}(j,n))^2, \ldots, (\mathbf{r}_M(j,n)^{\mathrm{T}}\boldsymbol{\epsilon}(j,n))^2\big). \tag{3.12}$$

The winning column l_{win} is selected by

$$l_{\mathrm{win}}^{j,n} = \underset{l, l \notin U^{j,n}}{\arg\max}\, y(j,n)_l. \tag{3.13}$$

As for the OOMP algorithm, the orthogonal projection of $\mathbf{r}_{l_{\mathrm{win}}^{j,n}}(j,n)$ is subtracted from $R^{j,n}$:

$$\mathbf{r}_l(j,n) := \mathbf{r}_l(j,n) - \big(\mathbf{r}_{l_{\mathrm{win}}^{j,n}}(j,n)\mathbf{r}_l(j,n)\big)\mathbf{r}_{l_{\mathrm{win}}^{j,n}}(j,n). \tag{3.14}$$

The $\mathbf{r}_l(j,n)$ are normalized in each iteration step.

So far, the BOP algorithm is identical to the OOMP. But whereas the OOMP algorithm stops at this step, the BOP algorithm proceeds by determining further pursuits (coefficient vectors) $\mathbf{w}^j$, with $j = 1, \ldots, P$. In each pursuit, all overlaps $\mathbf{y}(j,n)$ that are computed during the pursuit are stored. To begin the next pursuit, the algorithm looks for the basis vector with maximum overlap that has not been used so far. The basis vectors that have already been used are encoded by the following function (1 means "used", 0 means "not used"):

$$Q(l,j,n) = \begin{cases} 0: & \text{if there is no previous pursuit that is equal to the } j\text{-th pursuit up to the } n\text{-th iteration, where in that iteration basis vector } l \text{ has been selected} \\ 1: & \text{otherwise.} \end{cases} \tag{3.15}$$

For each pursuit j, we define s_j as the number of iterations performed in this pursuit. (For our stopping criterion, we always have $s_j = k$.) The overlap vectors for the individual iterations are

$$\mathbf{y}(j,0), \ldots, \mathbf{y}(j,n), \ldots, \mathbf{y}(j,s_j - 1). \tag{3.16}$$

If m pursuits have been performed so far, we can find the maximum overlap that has not been used as follows:

$$j_{\mathrm{target}} = \underset{j=1,\ldots,m}{\arg\max} \max_{n=0,\ldots,s_j} \max_{l, Q(l,j,n)=0} y(j,n)_l \tag{3.17}$$

$$n_{\mathrm{target}} = \underset{n=0,\cdots,s_{j_{\mathrm{target}}}}{\arg\max} \max_{l, Q(l,j_{\mathrm{target}},n)=0} y(j_{\mathrm{target}},n)_l \tag{3.18}$$

$$l_{\mathrm{target}} = \underset{l, Q(l,j_{\mathrm{target}},n_{\mathrm{target}})=0}{\arg\max} y(j_{\mathrm{target}},n_{\mathrm{target}})_l \tag{3.19}$$

To yield the new pursuit with index $j = m + 1$, the pursuit j_{target} is repeated up to iteration n_{target}. From this point, the algorithm continue with the "runner up" basis vector l_{target} instead of the winner that was chosen by pursuit j_{target} in this step. The pursuit is continued until the stopping criterion is reached. The described procedure is repeated until the desired number P of pursuits have been computed.

3.2 Dictionary learning

Previously in this chapter, we have described how we can find a sparse coefficient vector $\mathbf{w}$ that best approximates a data point for a given dictionary of basis vectors C.

We will now deal with the problem of how to find a suitable dictionary C; this can be done using different approaches. In the following, we will discuss vector quantization and principal component analysis.

3.2.1 Vector quantization

Vector quantization (VQ) is a family of algorithms used in the field of lossy data compression [46]. VQ algorithms aim to partition the data into groups where the differences between the data points within a group tend to be smaller than the differences between different groups. Each group is represented by a prototype vector, also referred to as the *codebook vector* $\mathbf{c}$. In general, VQ aims to find a codebook matrix $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ that minimizes the reconstruction error [47]

$$E = \frac{1}{p} \sum_{i=1}^{p} \sum_{j=1}^{k} r_{ij} \|\mathbf{x}_i - \mathbf{c}_j\|^2, \tag{3.20}$$

where r_{ij} indicates whether data point i belongs to group j:

$$r_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_{\mu=1,\ldots,k} \|\mathbf{x}_i - \mathbf{c}_\mu\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{3.21}$$

Several clustering algorithms can be used to perform vector quantization. In the following, we will discuss two hard-competitive learning algorithms and a soft-competitive version.

K-means

The k-means algorithm [48] is one of the most well-known clustering methods and is a key tool in the field of VQ [46]. K-means can be used to find a codebook C containing a predefined number k of codebook vectors that minimizes the reconstruction error according to Equation (3.20). The main advantage of the k-means algorithm is its simple

and fast computation. In the first step, the algorithm selects an initial codebook C that can, for example, be initialized by selecting a set of random data points. In each iteration, each data point is assigned to the closest codebook vector. We then update the codebook C by minimizing the objective function E in Equation (3.20) with respect to $c_1, \ldots, \mathbf{c}_k$ while keeping the group assignments r_{ij} fixed. This minimum can be found by setting the derivative with respect to the codebook vector $\mathbf{c}_j$ to zero, giving [47]

$$2 \sum_{i=1}^{p} r_{ij}(\mathbf{x}_i - \mathbf{c}_j) = 0, \tag{3.22}$$

which we can rearrange for $\mathbf{c}_j$ to obtain

$$\mathbf{c}_j = \frac{\sum_i r_{ij} \mathbf{x}_i}{\sum_i r_{ij}}. \tag{3.23}$$

Equation (3.23) is equivalent to calculating the mean of the data points assigned to the codebook vector $\mathbf{c}_j$. Hence, the update step simply corresponds to moving the codebook vectors to the mean of their assigned data points – hence the name k-means.

The algorithm is repeated until we reach a stopping criterion; a common stopping criterion is that the change of the codebook vectors from one iteration to the next lies below a given threshold. The algorithm as described is the batch version of k-means. This batch version is prone to getting stuck in local minima and is very sensitive to the initialization of C. A version that is more robust with respect to local minima is online k-means [47, 49], which we now describe.

Online k-means

Online k-means is a pattern-by-pattern algorithm. Whereas the batch k-means algorithm performs the update step after assigning all data points to the nearest codebook, the online version assigns one data point at a time to the closest or "winning" codebook vector and subsequently updates this codebook vector. The corresponding update rule is

$$\mathbf{c}_l^{\text{new}} = \mathbf{c}_l^{\text{old}} + \alpha_t(\mathbf{x}_i - \mathbf{c}_l^{\text{old}}), \tag{3.24}$$

where α_t is the learning rate. The learning rate is cooled off over the epochs t as follows:

$$\alpha_t = \alpha_0 \left(\frac{\alpha_{\text{final}}}{\alpha_0} \right)^{\frac{t}{t_{\max}}}. \tag{3.25}$$

α_0 and α_{final} denote the initial and final learning rate. If the learning rate is cooled off slowly, the algorithm performs a stochastic gradient descent on the objective function (3.20). This makes online k-means less prone to getting stuck in local minima than the batch version [49].

K-means is a hard-competitive learning algorithm; this means that, in each iteration, only the codebook vector closest to the presented data point is updated. This hard-competitive learning may lead to suboptimal solutions with respect to Equation (3.20) if local minima are present [50]. In addition, both versions depend strongly on the initialization of the codebook and are prone to slow convergence. We will therefore now discuss an extended soft-competitive version of the online k-means algorithm, the Neural Gas algorithm.

Neural Gas

The Neural Gas (NG) algorithm [50, 51] mitigates some of the drawbacks of the k-means algorithm. Its advantages are fast convergence, low reconstruction error with respect to Equation (3.20), and close to optimal quantization of the given datapoints by the codebook vectors.

NG is a soft-competitive learning algorithm. Whereas the k-means algorithm only updates the "winning" codebook vector, the NG algorithm updates all codebook vectors; the closer a codebook vector is to the given data point $\mathbf{x}_i$, the more it is changed. This is done by ranking the codebook vectors according to the reconstruction error $\|\mathbf{x}_i - \mathbf{c}_j\|$. The codebook vector closest to the data point receives rank 0, the second-closest receives rank 1, and so on. Mathematically, the rank $r_l = 0, \ldots, k-1$ of the codebook vector $\mathbf{c}_l$ is given by the number of codebook vectors $\mathbf{c}_j$ that are closer to the data point, i.e.

$$r_l = \sharp\{j = 1, \ldots, k, j \neq l \quad | \quad \|\mathbf{x}_i - \mathbf{c}_j\| < \|\mathbf{x}_i - \mathbf{c}_l\|\}. \tag{3.26}$$

After computing the ranking, the codebook vectors are updated using the neighbourhood function

$$h_t(r_j) = e^{r_j/\lambda_t}, \tag{3.27}$$

where λ_t is the so-called the neighbourhood radius. The codebook vector $\mathbf{c}_j$ is updated as follows:

$$\mathbf{c}_j := \mathbf{c}_j + \alpha_t h_t(r_j)(\mathbf{x}_i - \mathbf{c}_j), \tag{3.28}$$

and λ_t is cooled off over the epochs t with:

$$\lambda_t = \lambda_0 \left(\frac{\lambda_{\text{final}}}{\lambda_0}\right)^{\frac{t}{t_{\max}}}. \tag{3.29}$$

λ_0 and λ_{final} denote the initial and final neighborhood radius. The learning rate α_t is cooled off in the same way. If the parameters λ_t and α_t are cooled off slowly enough, it can be shown that the Neural Gas algorithm will quantize the dataset optimally [50, 51].

3.2.2 Principal component analysis

Principal component analysis (PCA) is one of the most well-known techniques for dimensionality reduction, lossy data compression, feature selection etc. [47]. PCA can be defined as an orthogonal projection of the data onto a lower-dimensional subspace under the constraint that as much of the variance as possible is preserved. This is done by calculating the directions of greatest variance, also referred to as the principal components.

PCA can also be defined equivalently as a projection to a lower-dimensional subspace that minimizes the reconstruction error

$$E = 1/N \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2, \tag{3.30}$$

where $\mathbf{x}_i$ and $\hat{\mathbf{x}}_i$ are the original and projected data point, respectively. Under this interpretation, we are attempting to represent a data point $\mathbf{x}_i$ as a linear combination of a dictionary $C \in \mathbb{R}^{N \times M}$ ($M \leq N$), with a coefficient vector $\mathbf{w}$:

$$\mathbf{x}_i = C\mathbf{w} + \epsilon, \tag{3.31}$$

where ϵ is the reconstruction error.

The direction of largest variance in a data set can be calculated by a single artificial neuron using Oja's rule [52]. The corresponding update rule for the vector $\mathbf{c}$ is

$$\Delta\mathbf{c} = \alpha_t e^{-k/\lambda_t} y \left(\mathbf{x} - y\mathbf{c}\right) \ , \tag{3.32}$$

where $\mathbf{x}$ is a randomly selected data point, α_t is the learning rate, and $y = \mathbf{c}^{\mathrm{T}}\mathbf{x}$. Oja's rule can only calculate a single direction of greatest variance. To calculate the second and subsequent directions of greatest variance, we can apply Oja's rule several times on input data that has been orthogonalized with respect to the previously determined directions of greatest variance.

An alternative to calculating the directions of greatest variance successively is the so-called Sanger's rule, which can calculate the first M directions of greatest variance simultaneously. Sanger's rule is an extension to Oja's rule that updates M directions $c_1, \ldots, c_M$ in each step, as follows:

$$\Delta\mathbf{c}_j = \alpha_t e^{-k/\lambda_t} y_j \left(\mathbf{x} - \sum_{l<j} y_l \mathbf{c}_l - y_j \mathbf{c}_j\right) \ , \tag{3.33}$$

where $y_l = \mathbf{c}_l^{\mathrm{T}}\mathbf{x}$.

3.3 The genotype score

The genotype score (GS) is a traditional risk prediction model [31, 53]. Typically, the GS is calculated from the number of risk alleles (those associated with the disease phenotype) that are carried by a given individual for a predefined set of SNPs. We will in the following consider a data set $X = \mathbf{x}_1, \ldots, \mathbf{x}_M$ with $\mathbf{x}_i \in \{0, 1, 2\}^D$. The homozygous genotype for the risk allele is coded with 2, heterozygous with 1 and homozygous for the non-risk allele with 0. Hence, the entry for x_{ij} corresponds to the number of risk alleles that are carried by the individual i for SNP j. The GS for an individual i can be calculated as

$$R_i = \sum_{j=1}^{D} x_{ij} \, . \tag{3.34}$$

3.4 Linkage disequilibrium

SNPs that lie only a few base pairs apart are generally inherited together. In addition, some combinations of SNPs (haplotypes), which need not necessarily be SNPs with neighbouring base pair positions, occur more or less often than we would expect if they were statistically independent. Such an association between SNPs is refered to as linkage disequilibrium (LD). In other words, LD is a measure that describes the difference between an expected and an observed haplotype frequency.

A commonly used measure for LD is called D [54]. Let us imagine we have a haplotype of 2 SNPs. We have the allele frequencies p_1 and p_2 for the first SNP and q_1 and q_2 for the other SNP. We require p_1 and q_1 to be the common alleles, i.e. $p_1 \geq p_2$ and $q_1 \geq q_2$. Under the assumption of statistical independence, the haplotype ij has the expected frequency $x_{ij} = p_i q_j$. D can now be calculated as

$$D = x_{11} - p_1 q_1 \, . \tag{3.35}$$

The R^2 statistic is another commonly used measure for LD [55]; this LD measure is more robust for finite populations. R^2 is calculated as

$$R^2 = \frac{D^2}{p_1 p_2 q_1 q_2} \, . \tag{3.36}$$

Part II

Learning from what we know

Predicting the risk of a disease is of great importance in clinical medicine. So far, the risk of a disease is mainly estimated for environmental factors. However, genetic factors also play an important role in the pathogenesis of most of today's diseases. If we identify the genetic factors that influence a disease, we may broaden our understanding of the mechanisms underlying the disease as well as potential therapeutic targets. Hence, genetic risk prediction opens up a new area of personalized medicine. The principle of personalized medicine is to adjust the clinical treatment and preventive action to the genetic pattern of each individual. Genetic testing is already in common use for Mendelian diseases, where the underlying genetic factor is known [8,28]. However, genetic tests are not yet available for complex diseases. This is because of their complex structures, where multiple loci work together to increase or decrease the risk of a disease.

GWA studies aim to identify the genetic risk factors that influence the risk of a disease. Most GWA studies have so far concentrated on individual SNPs. Each SNP can be seen as a weak classifier, which can be used to estimate the risk of the disease. In other words, each SNP in isolation contributes a small amount of information about the risk of the disease.

This knowledge is gained by looking at the distribution of the genotypes in known cases and comparing these to control individuals. Subsequently we can estimate the risk of a disease from what we have observed. In other words, we learn from the data at hand and use this knowledge to predict the condition of unseen subjects.

To be useful in clinical medicine, a genetic model for risk prediction needs to have a certain minimum predictive value beyond the prediction made using traditional risk factors. Most SNPs identified to date have only a small effect on risk. Hence, so far, a single SNP is not suitable for predicting an individual's genetic risk of disease. An application in personalized medicine is thus not possible.

If we predict risk on a combination of SNPs taken together, the prediction may improve; we call this a multilocus approach. Figure 3.2 illustrates the classification based on single SNPs versus the multilocus approach. Each axis represents a single SNP and the distribution of the genotypes for the two classes. If we only look at one SNP at a time, we are not able to distinguish between cases and controls since each genotype is equally frequent. However, if we combine the two SNPs, we can clearly see a disease-specific pattern.

Classification based on a combination of features is a general task in most machine learning applications. Machine learning algorithms aim to build a model based on existing data which can help us to predict the outcome, in our case the phenotype, on unseen data. Classification algorithms typically take the input, in our case the SNPs of a single individual, and assign it to a discrete class, the phenotype. The prediction is only based on the SNPs, and the relationship between these SNPs and the phenotype is modeled based on a dataset where we know the phenotype for each individual.

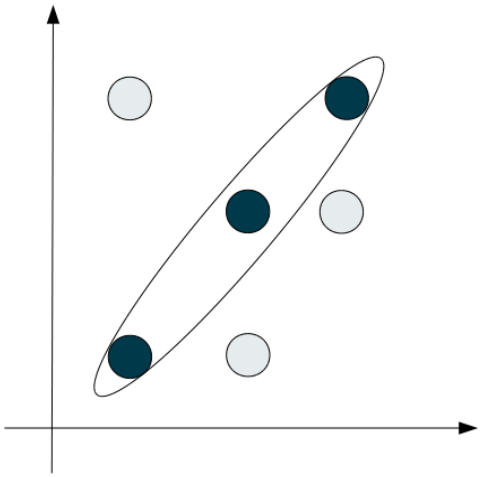


Figure 3.2: Schematic view of the classification problem. Both axes illustrate the distribution of the genotypes of a given SNP. A single SNP is not sufficient to separate the classes due to the equal frequencies of the genotypes. However, by looking at the two SNPs at the same time, we can differentiate between the cases and the controls.

Several studies have aimed to combine multiple genetic variants to assess the risk of a common complex disease [23, 53, 56, 57]. However, the consensus of these studies is rather disappointing. Many of these studies have combined the SNPs that have been validated so far to see whether risk prediction can be improved. One of the most used approaches is the risk prediction based on the genotype score (GS) [23, 53]. The GS is an additive model that counts the number of risk alleles to make a risk prediction.

However, each of the risk factors known so far has only low discriminatory and predictive value, and so far, even if they are combined, theses variants only explain a very small proportion of the genetic contribution. The predictive value of these studies is hence far away from being useful in a clinical setting.

The traditional risk prediction strategies (e.g. GS) have several weak points: The small number of SNPs on which classification is based, the selection strategy and the linear fashion of the classifiers. Let us discuss each of these weak points more closely.

First, most approaches classify based on the risk variants identified so far, and these are only small in number. Moreover, these SNPs generally all have marginal effects, making classification a difficult task. Second, selection of only the validated SNPs may not be the best choice since these SNPs explain too little about the genetic risk of a disease. Third, most of the studies only allow for linear main effects and ignore interactions and more complex relationships between variants [28]. Interactions between SNPs may however increase the predictive value since interactions are expected to have larger effect sizes than individual SNPs and their additive effects. In addition to

SNP interactions in the traditional genetic sense, there may also be other, more general types of disease-specific structures in the data, which the algorithm will be able to exploit to improve classification further.

The drawbacks of the standard feature selection approaches, as we discussed above, can however easily be avoided by applying well-known classification methods from the field of machine learning. Since we do not expect all susceptibility loci to be identified yet, we believe that there may be better strategies than selecting only the SNPs that have been validated so far. Instead, we aim to incorporate large numbers of SNPs into the model. Second, we expect there to be more complex relationships, such as interaction between the SNPs. Hence, we aim to identify these structures and hence improve classification.

We will apply two approaches that yield classifiers which are based on large ensembles of SNPs. The first of these approaches is the well-known support vector machine (SVM). (While this work was being prepared, other groups also started using the SVM on GWA data [28, 58, 59].) The advantage of the SVM is that it utilizes all features and is applicable to very large datasets. A drawback of the SVM is however that the number of SNPs used for classification can only be controlled by the number of features in the data set. If a classification based on a smaller number of SNPs is desired, the SNPs have to be selected beforehand. This is generally done by selecting a subset based on the individual significance values, the p-values [23, 28, 57].

Second, we will apply a boosting algorithm modified for use on GWA data. The main idea of boosting is to combine several weak classifiers, in our case SNPs, to one strong classifier. The advantage of the boosting algorithm is that it selects SNPs one after another in order to obtain a set of SNPs that successively improve the classification. Hence, the number of SNPs can be better controlled by the boosting algorithm than by the SVM since no preselection of SNPs is needed.

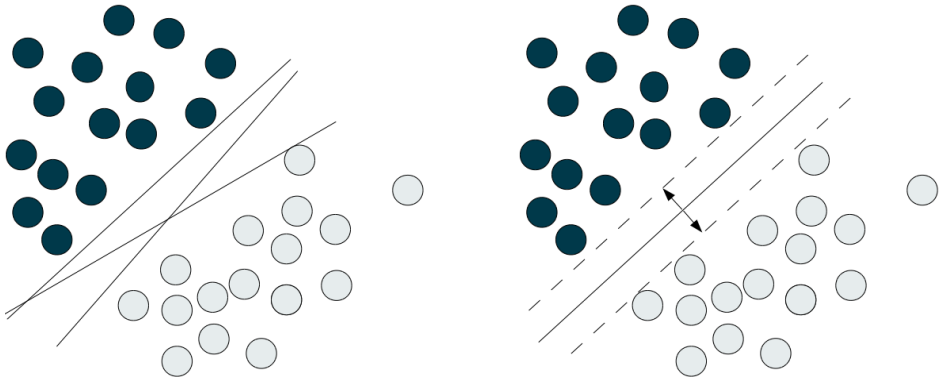
4 Classification of GWA data with the support vector machine

Risk prediction based on GWA data is a very difficult task. The number of SNPs which are typically genotyped is expected by far to exceed the number of truly disease-specific variables. The variants identified so far explain only a small fraction of the genetic risk of a disease. Hence, since the majority of the risk factors still remain to be identified, a good risk prediction does not seem possible based only on the variables validated so far. Consequently, the prediction may be improved if we ensure that none of the disease relevant variables are excluded from the model. Moreover, if we feed a large number or all of the genotyped SNPs into the classifier, it may be able to exploit previously unidentified relationships and structures in the data. Hence, if we build a classifier based on large ensembles of SNPs, then all disease-relevant variables, interactions between the variables and disease-specific structures are still present and classification may be improved.

Traditional risk prediction models, such as the genotype score, lack the capability of handling interaction effects, since they are generally based on the assumption that the effect of multiple variants present simultaneously is the sum of the individual effects. In other words, the assumption is that multiple variants combine additively, hence are not taken into account interactions or epistasis between the variants.

Unlike the traditional risk models, the support vector machine (SVM) [60,61] is capable of handling both individual effects as well as interactions. Moreover, the SVM can be applied to very large data sets, which makes it a suitable candidate for classification without first having to reduce the number of variables. Hence, the SVM algorithm is a promising candidate for improving risk prediction since it is able to take advantage of disease-specific structures, which are neither ignored by the algorithm nor excluded from the model in a preselection step. While this work was being carried out, the SVM was successfully applied to GWA data for type 1 and 2 diabetes [28, 58].

The support vector machine is a supervised learning algorithm. Broadly speaking, the SVM analyzes data with known class labels and determines a hyperplane that best separates the classes. Two classes that can be separated with a hyperplane without any misclassified data points, are refered to as linear separable. If two classes are linear separable, there typically exist an infinite number of hyperplanes that solve the classification problem, as illustrated in Figure 4.1a. However, not all hyperplanes are



(a) linear separation (b) maximum margin classification

Figure 4.1: The obtained separating planes for a linear separable data set. a) Arbitrary linear separation planes. b) Separating plane obtained by maximum margin classification.

optimal in the sense of generalization. This means that, if we have a slight change in the distribution of the data points, some hyperplanes may no longer classify correctly. The generalization error is a measure of how well the obtained classifier will perform on the true underlying data point distribution. The SVM aims to minimize this error by finding the optimal separating hyperplane, which is defined as the separating plane that maximizes the distance to the closest point of each class, as illustrated in Figure 4.1b. This distance is called the margin, and hence the SVM is also referred to as a maximum margin classifier.

4.1 Support vector machine

The SVM determine a hyperplane that separates the binary classification problem with maximum margin [46,47].

As input, the SVM takes data points $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, $\mathbf{x}_i \in \mathbb{R}^D$ and class labels $Y = (y_1, \ldots, y_N)$ with $y_i \in \{1, -1\}$. From this input, the SVM determine an optimal separating hyperplane, which is defined by a weight vector $\mathbf{w} \in \mathbb{R}^D$ and a threshold b. The hyperplane is then given by the set of points $\mathbf{x}$ that satisfy the plane equation

$$\mathbf{w}^T\mathbf{x} - b = 0\,. \tag{4.1}$$

The class assignment of a data point $\mathbf{x}_i$ is given by:

$$g(\mathbf{x}_i) = \text{sign}[\mathbf{w}^T\mathbf{x}_i - b]. \tag{4.2}$$

In order to find the optimal separating hyperplane, we want to maximize the minimal distance between the hyperplane and the data points of the two classes (the

margin). Hence, we want to maximize

$$\arg\max_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} M(\mathbf{w}, b) \ , \tag{4.3}$$

where $M(\mathbf{w}, b)$ is the margin, defined as

$$M(\mathbf{w}, b) = \frac{1}{\|\mathbf{w}\|} \min_{i=1,\dots,N} y_i(\mathbf{w}^T\mathbf{x}_i - b) \ . \tag{4.4}$$

($\frac{1}{\|\mathbf{w}\|}$ can be moved out of the minimization term since it does not depend on i.) Note that the maximum in Equation (4.3) is not unique. If a certain $\mathbf{w}$ and b maximize $M(\mathbf{w}, b)$, then any scalar multiples $\alpha\mathbf{w}$ and αb ($\alpha \neq 0$) yield the same maximum, i.e. $M(\mathbf{w}, b) = M(\alpha\mathbf{w}, \alpha b)$.

We therefore need to add an additional constraint on the scaling of $\mathbf{w}$ and b. Intuitively, we might demand that $\mathbf{w}$ should have a certain constant length, i.e. unit length. However, the maximization problem is hard to solve in this form. If, instead we choose the constraint

$$\min_{i=1,\dots,N} y_i(\mathbf{w}^T\mathbf{x}_i - b) = 1 \ , \tag{4.5}$$

then we see that the margin becomes simply

$$M(\mathbf{w}, b) = \frac{1}{\|\mathbf{w}\|} \ . \tag{4.6}$$

By minimizing $\|\mathbf{w}\|$ we maximize the margin M. Hence we can write the objective function as

$$\arg\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \mathbf{w}^T\mathbf{w} \tag{4.7}$$

with the constraint stated in Equation (4.5).

Equation (4.7) is a quadratic programming problem where we aim to minimize a quadratic function with linear constraints. This minimization problem can be solved by using the Lagrange multipliers [47]. An alternative approach to finding the optimal separating hyperplane for linearly separable data sets is the DoubleMinOver algorithm [62].

4.1.1 DoubleMinOver

The optimal separating hyperplane can also be found by using a perceptron-like learning rule. The so-called MinOver (MO) algorithm is a slight modification of the perceptron algorithm [63], and its learning rule can be derived from constrained gradient descent [64]. The DoubleMinOver (DMO) algorithm is an extension of the MO algorithm. Both the MO and the DMO algorithm provide the maximum margin solution for a linear separable classification problem; however the difference between the two

algorithms is that MO does not include a threshold or bias b. The DMO algorithm is a modification which incorporates the threshold.

Let us assume we have a linearly separable classification problem. We refer to the data points of the class $y = 1$ as X^+ and the data points of the class $y = -1$ as X^-:

$$\begin{aligned} X^+ &= \{\mathbf{x}_i | y_i = 1\} \\ X^- &= \{\mathbf{x}_i | y_i = -1\} \end{aligned} \tag{4.8}$$

The weight vector $\mathbf{w}$ is initialized with $\mathbf{w} = 0$ and the weights are updated according to the learning rule

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}^{\min +} - \mathbf{x}^{\min -} , \tag{4.9}$$

where

$$\begin{aligned} \mathbf{x}_t^{\min +} &= \arg\min_{\mathbf{x}_\mu \in X^+} (\mathbf{w}_t^T \mathbf{x}_i) , \\ \mathbf{x}_t^{\min -} &= \arg\min_{\mathbf{x}_\mu \in X^-} -(\mathbf{w}_t^T \mathbf{x}_i) , \end{aligned} \tag{4.10}$$

and t is the current iteration number. After a sufficient number τ of iterations have been performed to obtain convergence the threshold b can be obtained by

$$b = \frac{\mathbf{w}_\tau^T (\mathbf{x}^{\min +} + \mathbf{x}^{\min -})}{2}. \tag{4.11}$$

The final weight vector $\mathbf{w}_\tau$ which defines the separating hyperplane is given by

$$\mathbf{w}_\tau = \sum_{t=0}^{\tau-1} (\mathbf{x}_t^{\min +} - \mathbf{x}_t^{\min -}) . \tag{4.12}$$

We obtain the classification for a new data point $\mathbf{x}$ as

$$h(\mathbf{x}) = \mathbf{w}_\tau^T \mathbf{x} - b . \tag{4.13}$$

4.1.2 Kernel SVM

Not all classification problems can be solved by a linear classifier. However, it may be possible to make the points linear separable by transforming them into a higher dimensional feature space using a suitable transform ϕ. A disadvantage of this is that it requires more computation and memory for the classifier.

The so-called kernel trick is a way of implicitly transforming the data points into a higher dimensional feature space without ever having to manipulate the high dimensional points explicitly. Instead, we replace all occurrences of the scalar product

in high-dimensional space (i.e. $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$) by a so-called kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ that can be evaluated in the original data space.

In this work, we use a linear and a Gaussian kernel function. The *linear* kernel is the simplest kernel function,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j. \tag{4.14}$$

The *Gaussian* kernel is commonly used and takes the form

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)}. \tag{4.15}$$

σ is an adjustable parameter which controls the width of the Gaussian curve.

4.1.3 DMO with Kernel

The DMO algorithm can be expressed in a way that allows the kernel trick to be applied. To do this, we need to make sure that data points only occur in scalar products, which can then be replaced with the kernel function.

The separating hyperplane is defined by the weight vector $\mathbf{w}$, which is given in Equation (4.12). This equation can be rewritten as

$$\mathbf{w}_T = \sum_{i=1}^{N} y_i n_i(T)\mathbf{x}_i \ , \tag{4.16}$$

where $n_i(T)$ is the number of times $\mathbf{x}_i$ is used for updating the weights in T iterations. With Equation (4.16), the DMO algorithm simply consists of identifying $\mathbf{x}^{\min +}$ and $\mathbf{x}^{\min -}$ at each iteration t and incrementing the number of times these data points are selected to update w. The class assignment can be written in the so-called dual form:

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{N} y_i n_i \mathbf{x}_i^T\mathbf{x} \ . \tag{4.17}$$

If the data points are transformed into another feature space using a transform ϕ we obtain

$$h(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) = \sum_{i=1}^{N} y_i n_i \phi(\mathbf{x}_i)\phi(\mathbf{x}) \ . \tag{4.18}$$

Given the kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$, we can rewrite this class assignment to

$$h(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) = \sum_{i=1}^{N} y_i n_i K(\mathbf{x}_i, \mathbf{x}) \ . \tag{4.19}$$

As a consequence, $\mathbf{x}^{\min +}$ and $\mathbf{x}^{\min -}$ are determined as follows:

$$\mathbf{x}^{\min +} = \underset{\mathbf{x}_\mu \in x^+}{\arg\min}\, h(\mathbf{x}_\mu) \,, \tag{4.20}$$
$$\mathbf{x}^{\min -} = \underset{\mathbf{x}_\mu \in x^-}{\arg\min}\, -h(\mathbf{x}_\mu) \,,$$

and the training step simply consists of incrementing n_i for the identified $\mathbf{x}^{\min +}$ and $\mathbf{x}^{\min -}$:

$$n_{\min +}(t+1) = n_{\min +}(t) + 1 \,, \tag{4.21}$$
$$n_{\min -}(t+1) = n_{\min -}(t) + 1 \,.$$

The threshold is calculated by

$$b = \frac{h(\mathbf{x}^{\min +}) - h(\mathbf{x}^{\min -})}{2} \,. \tag{4.22}$$

4.1.4 DoubleMaxMinOver

The data points that define the weight vector w are called the support vectors. These support vectors are the in general the data point with the smallest distance to the separating hyperplane. Besides the support vectors, no other data points are required to determine the separating hyperplane.

A drawback of the DMO algorithm is that the linear combination for the weight vector $\mathbf{w}$ contains not only support vectors but (typically) also other data points. This means that not only the support vectors have non-zero entries in n_i. Especially at the beginning of the training, DMO selects data points that eventually will not turn out to be support vectors. These vectors are not required to define the final hyperplane and hence, need not be stored. Moreover, these vectors may even be unfavorable for convergence [64].

The DoubleMaxMinOver (DMMO) algorithm deal with these drawbacks. It is a further extension of the DMO algorithm and was introduced by Martinetz et al. [64]. DMMO includes a forgetting term construct in such a way that the hyperplane obtained at the end of the training is solely determined by support vectors.

Let $V_t = \{\mathbf{x}_i | \mathbf{x}_i \in X, n_i > 0\} \subseteq X$ be the set of data points for which the count n_i is not equal to zero at iteration t. At the end of the training, we aim to have $V_t = X_s$, where $X_s \subseteq X$ are the support vectors. At iteration t we now not only look for $\mathbf{x}^{\min +}$ and $\mathbf{x}^{\min -}$ but also for the data points that have the maximum distance to the hyperplane, $\mathbf{x}^{\max +}$ and $\mathbf{x}^{\max -}$:

$$\mathbf{x}^{max+} = \underset{\mathbf{x}_i \in V_t^+}{\arg\max}(\mathbf{w}^T \mathbf{x}_i) \,, \tag{4.23}$$
$$\mathbf{x}^{max-} = \underset{\mathbf{x}_i \in V_t^-}{\arg\max} -(\mathbf{w}^T \mathbf{x}_i)$$

with

$$V_t^+ = V_t \cap X^+ \,, \qquad (4.24)$$
$$V_t^- = V_t \cap X^- .$$

The update step is completed by not only adding $x^{\min +}$ and $x^{\min -}$ but additionally subtracting $x^{\max +}$ and $x^{\max -}$ from $\mathbf{w}$. Hence, we have the following update step:

$$\mathbf{w} = \mathbf{w} + 2(\mathbf{x}^{\min +} - \mathbf{x}^{\min -}) - (\mathbf{x}^{\max +} - \mathbf{x}^{\max -}). \qquad (4.25)$$

If we use the kernel trick we not only increase the coefficient entries n_{max+} and n_{max-} but also decrease n_{max+} and n_{max-}:

$$\begin{aligned} n(t+1)_{\min +} &= n(t)_{\min +} + 2 \,, \qquad (4.26)\\ n(t+1)_{\min -} &= n(t)_{\min -} + 2 \,, \\ n(t+1)_{\max +} &= n(t)_{\max +} + 1 \,, \\ n(t+1)_{\max -} &= n(t)_{\max -} + 1 \,, \end{aligned}$$

with

$$\begin{aligned} \mathbf{x}^{\min +} &= \arg\min_{\mathbf{x}_\mu \in x^+} yh(\mathbf{x}_\mu) \qquad (4.27)\\ \mathbf{x}^{\min -} &= \arg\min_{\mathbf{x}_\mu \in x^-} yh(\mathbf{x}_\mu) \\ \mathbf{x}^{\max +} &= \arg\max_{\mathbf{x}_\mu \in V_t^+} yh(\mathbf{x}_\mu) \\ \mathbf{x}^{\max -} &= \arg\max_{\mathbf{x}_\mu \in V_t^-} yh(\mathbf{x}_\mu) \end{aligned}$$

Decreasing n_i corresponds to removing data points from V_t and this has the effect, as can be shown, that at the end of training, V_t consists only of support vectors.

4.1.5 Soft-margin SVM

In practice, we will not be able to make every problem linearly separable by transforming it into a higher-dimensional feature space. Hence, we may need to allow some of the data points to be misclassified. A solution to this problem is the so called soft-margin SVM [65]. The soft-margin SVM incorporates slack variables ξ which govern the permissible error.

The objective function for the soft-margin SVM is

$$\arg\min_{\mathbf{w}\in\mathbb{R}, b\in\mathbb{R}, \xi\in\mathbb{R}^+, i=1,\ldots,N} \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \qquad (4.28)$$

under the constraint

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 - \xi_i \quad i = 1, \ldots, N \,, \tag{4.29}$$

with $\xi_i \geq 0$. The parameter C defines the softness of the SVM. The smaller C the softer the margin and the more error is allowed.

To compute the soft-margin solution, we use a suitable modified version of the DMMO algorithm [64].

$$g^*(\mathbf{x}_i) = y_i \sum_{j=1}^{N} y_j n_j \big(K(\mathbf{x}_j, \mathbf{x}_i) + \frac{\mathbf{x}_i}{C}\big) = g(\mathbf{x}_i) + \frac{n_i}{C} \tag{4.30}$$

4.2 Results and discussion

In the following, we will test whether we can use the SVM algorithm to improve classification by incorporating large ensembles of SNPs. We will apply the SVM on two different data sets: an inflammatory bowel disease (IBD) data set and a data set on coronary artery disease (CAD). The performance was evaluated according to on the area under the curve (AUC) metric. To explore the effect of random variations in the data, we performed a 5-fold cross-validation.

4.2.1 Performance on the IBD data set

The IBD dataset [36] is described in detail in Chapter 2. In brief, the data set contains 1719 individuals of which 789 are cases and 930 are controls. The total number of SNPs is 292,162 after quality filtering.

Linear kernel SVM

To begin with, we run a linear kernel SVM (LSVM) on all SNPs. The algorithm yielded a mean AUC of 0.68 over all cross-validation folds. To put this in relation, the best SNP according to the p-values only achieved an AUC of 0.58 on the complete data set. Hence, a large ensemble of SNPs can predict the phenotype substantially better than a single SNP alone.

As previously discussed, we can generally improve classification if there are interactions and structures in the data which we can exploit. Hence, an advantage of using all SNPs for classification is that we do not miss possible interactions between SNPs nor other disease-specific structures since all SNPs are still present in the input data. On the other hand, since we do not expect all SNPs to be associated with the disease, many of the SNPs we incorporate will simply contain noise and we cannot ensure that these variables will not interfere with the classifier. Moreover, it may be sufficient

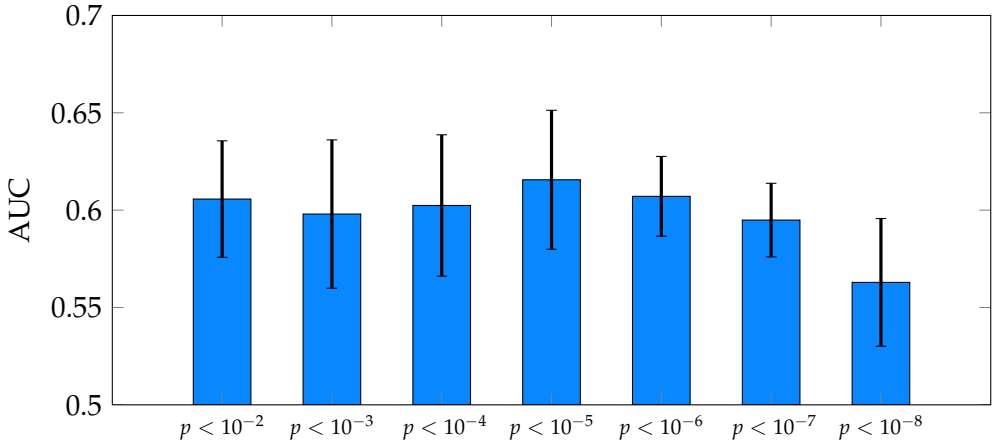


Figure 4.2: IBD data set. The performance of the linear SVM with different p-value cutoff thresholds. The standard deviation is shown as error bars.

to use a small number of SNPs for classification since a larger set of SNPs need not contain more information. Hence, to explore the effect of noise on the classification performance, we used different p-value cutoff thresholds to see whether better performance can be achieved with more stringent p-values. The cutoff thresholds ranged from $p < 10^{-8}$ to $p < 10^{-2}$. The results are shown in Figure 4.2.

The results clearly demonstrate that classification is significantly better on a large ensemble of SNPs. The performance is similar for all cutoff thresholds other than $p < 10^{-8}$. The best performance is reached with the cutoff threshold $p < 10^{-5}$, with an AUC of 0.62. This performance is however not as good as the classification results gained on all SNPs. Conversely, if we classify only on a small set of stringently selected SNPs ($p < 10^{-8}$), the LSVM only yields an AUC of 0.56 which is not significantly better than chance. The subset selected in this case consist of only 2–4 SNPs in each cross-validation fold.

A few discrete p-value cutoff thresholds do not give the complete picture of the trade-off between the subset size and the amount of noise since we only evaluate on a few discrete cases. Moreover, the size of the subset obtained for a given cutoff value may differ among the 5 cross-validation folds. Hence, rigid p-value cutoff thresholds do not show the detailed relationship between a specific subset size and the performance. To deal with this issue, we next evaluated the performance of the SVM with more continuously increasing subset sizes. To do so, we gradually increased the number of SNPs in the order of their p-values and trained an LSVM on each subset of SNPs. We did not run the SVM on all possible subset sizes, which would have been very computationally demanding. Instead, we first increased the size of the subset by

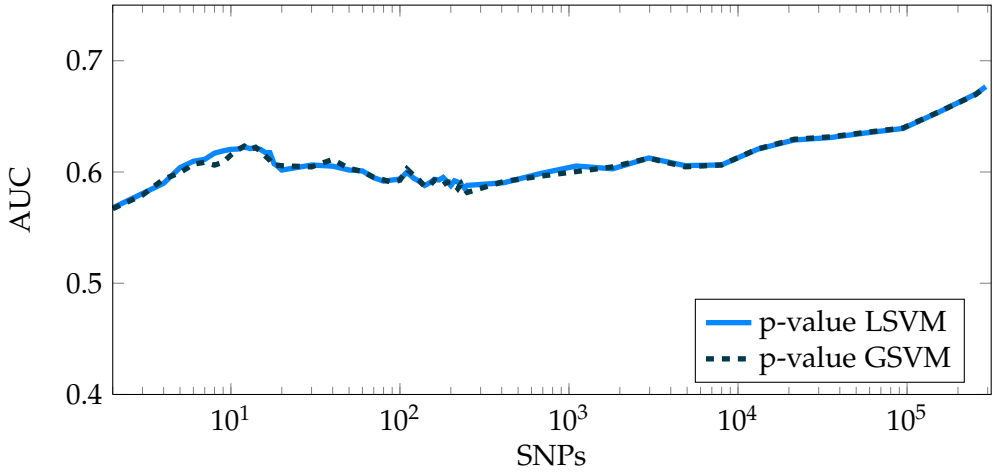


Figure 4.3: IBD data set. The performance of the linear and Gaussian kernel SVM based on p-value ranking. A logarithmic (base 10) scale is used for the X-axis.

one until we reached a subset size of 20. Then, we increased the subset size by 10 SNPs up to a subset size of 100 SNPs. Thereafter, we increased the size of the subset by a fixed factor. Hence, the resolution is larger for smaller numbers of SNPs. The results are shown in Figure 4.3.

In this more detailed picture of the performance characteristics of the LSVM we clearly see that we obtain the best result if we use all SNPs for classification. We see that initially, the performance of the LSVM gradually improves as we increase the size of the subset. Beyond around 10 SNPs we see a small decline. This decline is probably caused because the SNPs in question only add additional noise to the subsets. After the small decline, the performance reaches a plateau, where classification neither improves nor worsens with additional SNPs. The performance remain stable up to a subset size of around 10,000. Subsequently, more predictive variants are included again, which improves in the performance. At the end of the graph, we see that including all SNPs leads to the best classification results.

So far, we have trained a LSVM with a soft margin that is adjusted in a 10-fold cross validation. Linear classifiers have only limited power on a non-linear classification problem. The soft margin used in the results seen so far may be beneficial if the data contains outliers that should not influence the orientation of the hyperplane. In our case, however, we have used softness not to eliminate outliers but to enable the linear SVM to compute a solution at all on our non-linearly-separable data set. This works to a certain degree but is obviously not ideal. The preferable solution, as we discussed previously, is to project the data into a higher-dimensional space, where a linear separation is possible. As described in section 4.1.2, we can implicitly perform

this transformation by applying the kernel trick. Hence to explore whether we can improve classification in this way, we next trained a non-linear SVM with a gaussian kernel (GSVM).

Gaussian kernel SVM

The performance of the GSVM is shown in Figure 4.3. For smaller numbers of SNPs, we see that the GSVM and LSVM perform equally good. For both approaches, the best performance is obtained if all SNPs are included; in both cases, the AUC is about 0.68.

The fact that classification is best when all SNPs are used is surprising since we expect the number of noise SNPs by far to exceed the number of relevant SNPs. If the criterion used to order the SNPs (in this case, the p-value) was effective in selecting the disease-associated SNPs before the noise SNPs, what would we expect the shape of the curve to be? First, as we add more and more disease-specific SNPs, we would expect to see the AUC increase; then, as we add disease-specific SNPs that are redundant, i.e. that give the same information as already added SNPs, we would expect the AUC to reach a maximum and flatten out; and finally, when only noise SNPs are left to add, we would expect the AUC to decrease again. Our curve does not, however, match this expected shape. On the contrary, we first see an increase, then a plateau and finally again an increase throughout the remaining SNPs to a final maximum when all SNPs are included. This indicates that the relevant SNPs are not all included prior to the very end. The p-value is apparently not an effective measure for ranking SNPs according to their relevance and excluding noise SNPs. Can we improve the classification if we select the SNPs with a different strategy?

SVM ranking

An alternative strategy is to use the SVM itself to rank the SNPs. We can use the LSVM to obtain information about the importance of the SNPs by looking at the normal vector of the resulting separating hyperplane. The larger a particular entry of the normal vector is, the more the corresponding SNP influences the orientation. A ranking of the SNPs can hence be obtained through the absolute values of the entries of the normal vector. These entries can be seen as the weights of the SNPs, hence the name weight vector; we will also refer to these weights as the svmscore.

To obtain the svmscore ranking, we first trained the LSVM on the complete data set. Then, as in the previous experiments, we again ran both the LSVM and the GSVM on increasing subsets of SNPs (now ranked using the svmscore). The results are shown in Figure 4.4.

For larger subset sizes, we see that the LSVM and the GSVM with the svmscore ranking clearly outperform the same classifiers with the p-value ranking. For small

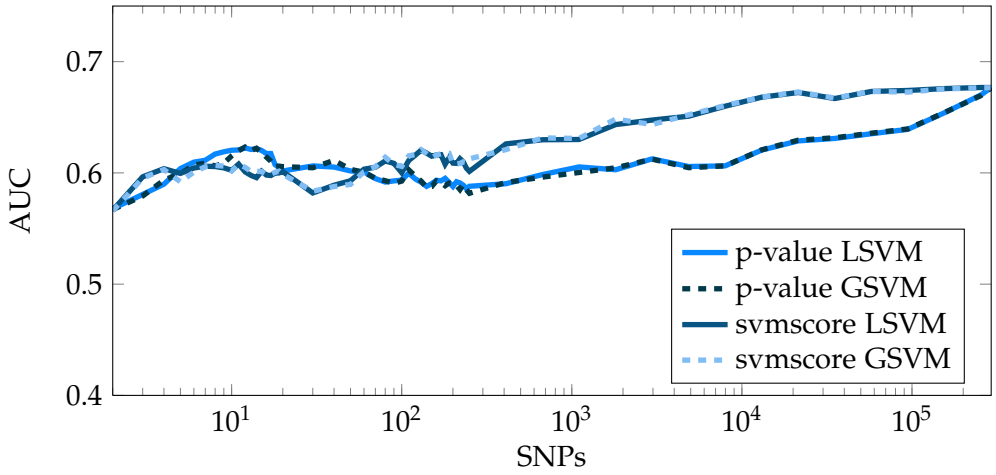


Figure 4.4: IBD data set. The performance of the linear and Gaussian kernel SVM based on p-value ranking and based on the SVM ranking. A logarithmic (base 10) scale is used for the horizontal axis.

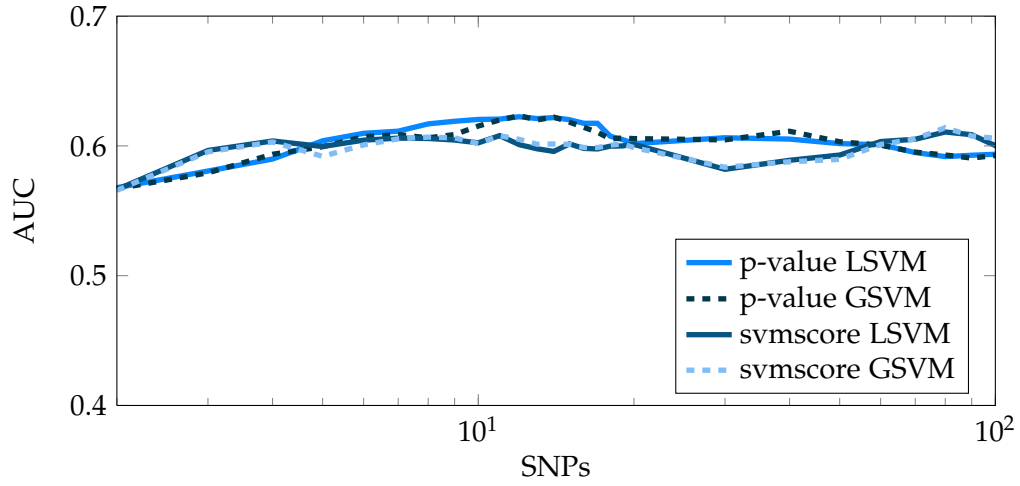


Figure 4.5: IBD data set. The performance of the linear and Gaussian kernel SVM based on p-value ranking and based on the SVM ranking up to a subset size of 100 SNPs. A logarithmic (base 10) scale is used for the horizontal axis.

numbers of SNPs, the GSVM and the LSVM on the p-value ranking and on the svmscore ranking perform similarly (see also Figure 4.5). The difference between the two ranking approaches becomes far more distinct for larger numbers of SNPs. At first, the performance of both ranking strategies improves as we increase the size of the subset. However, whereas the performance of the p-value ranking initially reaches a plateau for subsets of more than 15 SNPs, the performance of the svmscore ranking improves almost continually (apart from a small temporary decline around 30 SNPs) up to a peak AUC of 0.68 for the LSVM and GSVM until all SNPs are included.

Note that, in contrast to the p-value ranking, the svmscore ranking nearly reaches the best performance prior to including all SNPs. For both the LSVM and the GSVM the performance reaches a plateau, where classification neither significantly improves nor worsens with additional SNPs beyond 10,000 SNPs. Since we do not run the SVM on all possible subset sizes, the maximum performance may even be better for slightly larger or smaller subsets.

The reason why the performance curve, after reaching its maximum, remain stable throughout the remaining subset sizes is probably that only noise SNPs are added beyond this point. Indeed, the shape of the curve for the svmscore ranking is closer to the shape that we would expect if the SNPs were added successively according to their true influence on the disease. This indicates that the svmscore ranking is closer to this ideal than the p-value ranking and is therefore better suited for selecting a subset of SNPs for classification. The classification results themselves bear this out: The svmscore ranking yields the nearly the overall best AUC by far smaller numbers of SNPs.

A final property of the curves that is worth discussing is that both rankings reach the same AUC when all SNPs are included. This is not surprising, since in this case, the same set of SNPs is used for classification, only in a different order, and the SVM is not sensitive to the order in which the features are presented.

The SNPs of the svmscore ranking

As we have just seen, using the svmscore ranking to select subsets of SNPs improves the performance of the SVM noticeably compared to the p-value ranking. But how do the selected SNPs differ and to which extent? To better understand the difference between the two ranking strategies we determined the p-values of the SNPs with the highest svmscores. Figure 4.6 plots the p-values for all SNPs, ordered by base pair position. Alternate chromosomes are shaded light and dark. The SNPs that rank highest on the svmscore are marked in red; a SNP is included in this set if it is among the 100 highest ranked SNPs in at least one cross-validation fold. We can clearly see that the SNPs that have a high rank on the svmscore overlap to some extent with the SNPs ranked highest according to the p-values. However, this does not apply to

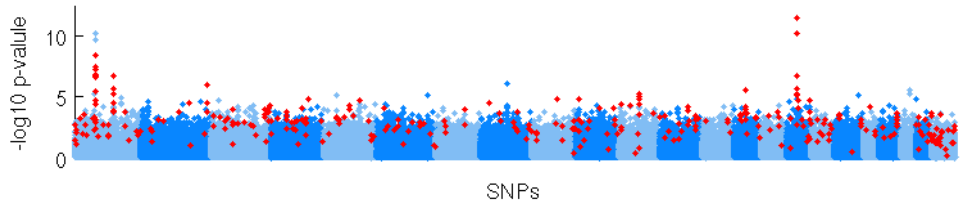


Figure 4.6: IBD data set. The p-values for all SNPs, ordered by base pair position. Alternate chromosomes are shaded light and dark. The SNPs that rank highest on the svmscore are marked in red; a SNP is included in this set if it is among the 100 highest ranked SNPs in at least one cross-validation fold.

the majority of the SNPs with high svmscore, indeed most of them have rather low p-values. Hence, we can conclude that the SNPs selected using the svmscore clearly have other features which qualify them for classification. The svmscore rank is based on the influence of each SNP on the weight vector for the classifier trained on all SNPs. In contrast to the p-values, which only assign a high value to those SNPs that have an effect individually, the svmscore also assigns a high value to combinations of SNPs that only have an appreciable effect when they occur together. The svmscore can do this because these SNPs jointly influence the orientation of the separating plane and thus have relatively high values in the weight vector. Hence, interacting SNPs and SNPs forming disease-specific patterns will have high svmscores. Consequently, the subsets that are formed based on these ranks include a large fraction of the disease-specific patterns, whereas we would lose most of these patterns if we used the p-value ranking. In other words, it is no longer necessary to include as many SNPs as possible in the classification for fear of losing important information; rather, we can exclude non-relevant SNPs, which effectively constitute noise and would only reduce the performance of the classifier.

4.2.2 Performance on the CAD data set

Having evaluated the performance of the SVM on the IBD data set, we next analyze the performance on data with a different phenotype, coronary artery disease (CAD) [33]. The data set contains 2506 individuals, of which 1222 are cases and 1284 are controls. The total number of SNPs is 118,247 after quality filtering and LD pruning. This data set is described in detail in Chapter 2.

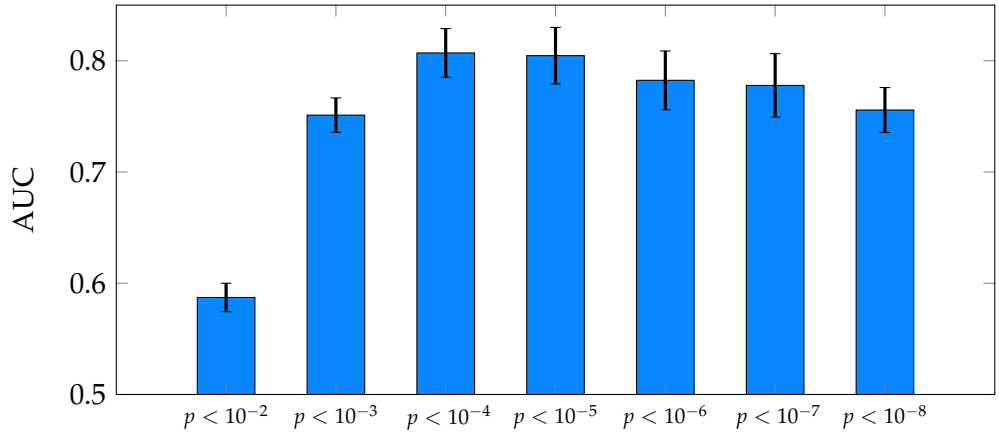


Figure 4.7: CAD data set. The performance of the linear SVM with different p-value cutoff thresholds. The standard deviation is shown as error bars.

Linear SVM

First, we run the LSVM on all SNPs. Surprisingly, the SVM yields a mean AUC of only around 0.58 over all cross-validation folds. To put this into relation, classification on all SNPs of the IBD data set yielded a mean AUC of 0.68. The poor CAD classification result may be due to several factors: a large amount of noise, small effect size per SNP, or weak or missing disease-specific structures in the data. If the latter two reasons apply, it will be barely possible to improve classification. However, if the first reason, a large amount of noise, is the cause for this poor classification, there is still hope.

To explore whether classification can be improved by reducing the noise, we generate a few subsets with different p-value cutoff thresholds. We use the same cutoff values as also used in the previous section on the IBD data, ranging from $p < 10^{-8}$ to $p < 10^{-2}$. The results are shown in Figure 4.7.

Indeed, the performance can clearly be improved by applying more stringent p-value cutoff thresholds. The best AUC, with a value of 0.81, seems to be achieved if a p-value cutoff between $< 10^{-4}$ and $< 10^{-5}$ is used. The corresponding subsets contain 21–25 and 47–53 SNPs, respectively, in the 5 cross-validation folds. Classification with a p-value cutoff $< 10^{-2}$ yields an AUC of less than 0.59.

As discussed in the previous section, where we evaluated the SVM performance on IBD data, a few discrete p-value thresholds do not give a detailed picture of the relationship between the subset size and the performance. Hence, in order to explore this relationship in detail, we next evaluate the classification on more continuously increasing subsets.

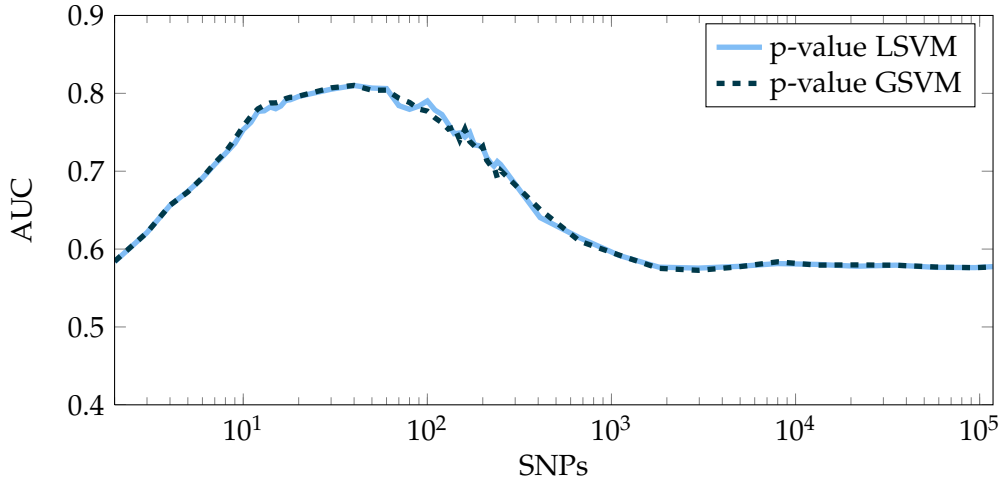


Figure 4.8: CAD data set. The performance of the linear and Gaussian kernel SVM based on p-value ranking. A logarithmic (base 10) scale is used for the X-axis.

We run the LSVM on gradually increasing subsets as also done for the IBD data set. The results are shown in Figure 4.8. Compared to the IBD data set, we see two clear differences: the curve shape and the large difference in performance if we classify on large ensembles of SNPs. Initially we observe a similar behavior of the SVM on the two data sets. The performance on both the CAD and the IBD data improves with increasing subset size and reaches a peak at around 10–20 SNPs. The performance on the CAD data is however clearly superior with an AUC of around 0.81 versus an AUC of around 0.62 on the IBD data. Whereas the performance on the IBD data subsequently increases throughout the remaining subset sizes, the performance on the CAD data declines. In other words, we see exactly opposite behaviour on the two data sets: While performance improves with subset size on one data set, it declines on the other data set.

Also, classification on all SNPs yields the best performance (AUC of 0.68) on one data set but a clearly inferior result (AUC of 0.58, compared to a maximum of 0.81) on the other data set. An explanation for this may be that the CAD data set contains more noise, and therefore we need to restrict ourselves a small subset of SNPs to eliminate the noise SNPs.

As for the IBD data set, we applied the SVM with a gaussian kernel to explore whether we can improve classification in this way. The performance of the GSVM is shown in Figure 4.8. We clearly see that the LSVM and the GSVM perform almost equal. For both approaches, the best performance is obtained with around 30 SNPs and an AUC of 0.81.

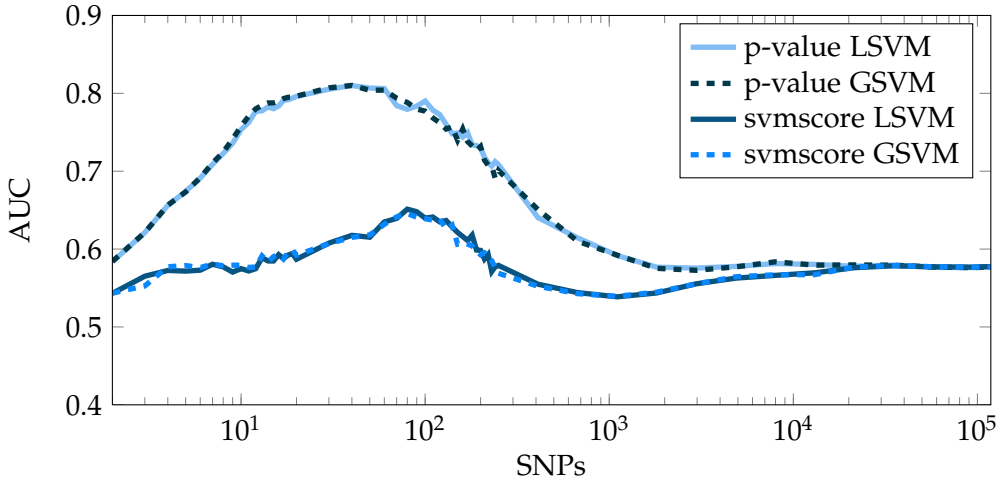


Figure 4.9: CAD data set. The performance of the linear and Gaussian kernel SVM based on p-value ranking and based on the SVM ranking. A logarithmic (base 10) scale is used for the horizontal axis.

SVM ranking

In the previous section, where we trained the SVM on the IBD data, we obtained a better result when using the svmscore to rank SNPs instead of the p-values. We will now examine whether this also holds true for the CAD data set. We again train an SVM on increasing subsets of SNPs, this time selected using the svmscore. The results are shown in Figure 4.9.

In contrast to the performance on the IBD data, we see that on this data set the p-value ranking yields clearly better results than the svmscore ranking. The GSVM and the LSVM on the svmscore ranking yield equal performance.

Despite the poor results of the svmscore ranking, we clearly see that the performance is better for subsets based on the svmscore than if we include all SNPs. This indicates that the svmscore ranking is not entirely ineffective, but on this data set, the p-value ranking is clearly more suitable.

If we take a closer look at the power of the classifier on which the svmscore is based, these results are however not surprising. The svmscore is calculated from the normal vector of the separating hyperplane for the SVM trained on all SNPs. As shown previously, the performance on all SNPs is poor for this dataset. Hence, we rank the SNPs based on a poor classifier. The separating plane is far from optimal, and consequently the weights of the SNPs do not correspond to suitable disease-specific features. In other words, the quality of the svmscore depends on the performance of the classifier on which it is based. This is why we obtain such different results on the two data

sets: On the IBD data set, the all-SNP classifier performs well, and svmscore therefore produces a good ranking; on the CAD data set, the all-SNP classifier performs badly, and the svmscore ranking is correspondingly poor.

4.2.3 Influence of the minor allele frequency on classification

As we have seen, there are large qualitative differences in the classification results on the two diseases we have studied: The best performance on the IBD data set is obtained by including all SNPs, whereas the best performance on the CAD data set is obtained for a relatively small subset of SNPs. Hence the questions arises: Which characteristics of a given data set influence the classification performance of the SVM?

In the course of this work, we have observed that the classification of GWA data is strongly influenced by small changes in the data. An area where such a small change can have a relatively large effect on the result are the quality criteria used in GWA studies.

The minor allele frequency (MAF) is a common criterion for quality filtering in GWA studies. SNPs with a MAF below a certain threshold, i.e. rare variants, are excluded because they are expected to be more prone to false positive results due to genotyping issues. Marginal variations when assigning the genotypes can lead to a large percentage increase or decrease in the frequencies of the rare variants, and so small uncertainties about the genotype can have a large impact on the association results for these SNPs. Moreover, technical limitations prevent very rare variants from being found at all on standard SNP chips. A further issue is the size and hence the power of the study: Most GWA studies lack the power to detect rare variants. Whereas some studies use a MAF threshold of 1%, other groups use a threshold of 5%. In this work we have so far used a threshold of 1% MAF.

Our reasoning in using a threshold of 1% MAF has been that, in contrast to studies that examine p-values for individual SNPs, we are not as concerned about the increased risk of false positives for rare variants because we do not test SNPs individually but construct a classifier on many SNPs simultaneously, thus limiting the effects that a random bias in an individual SNP can have. Moreover, our interest lies not such much in identifying individual disease-associated SNPs but in constructing a classifier that can predict disease risk.

A strong reason for including rare variants is that they often have stronger effects and are hence better predictors than the common variants. In other words, the rare variants may be of great value to enable or improve classification on GWA data. Hence, in the following, we will explore the effect of incorporating rare variants versus excluding these by varying the MAF threshold. We will first perform this evaluation on the CAD data set.

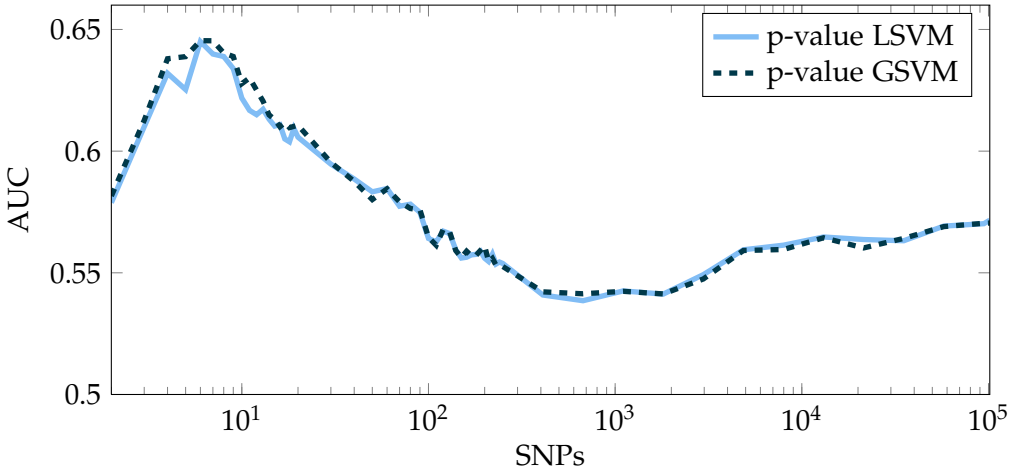


Figure 4.10: CAD data set. The performance of the linear and Gaussian kernel SVM on data with an MAF of 5% . A logarithmic (base 10) scale is used for the x-axis.

The result of the SVM on data with an MAF of 5% is shown in Figure 4.10. The LSVM and the GSVM yield similar results. As for the results of the SVM on SNPs with an MAF of 1%, we see that the performance is better if we classify on a smaller number of SNPs.

Figure 4.11 plots the performance curves obtained with the SVM for both MAF thresholds; the p-value ranking was used. If we use all SNPs for classification, we see that the performance is similar for the two MAF thresholds. However, if we compare the maximum AUCs obtained for each condition, we see that the better result is obtained if we include rare variables, i.e. for an MAF of 1%. With an MAF of 5%, the maximum AUC is below 0.65 versus a maximum AUC of around 0.81 for 1% MAF. The results for the svmscore ranking are similar: again, 1% MAF achieves better results than 5% MAF.

The results suggest that rare variants are of great importance to enable a suitable risk prediction. Hence, the rare variants may be more important for classification than has been assumed so far. These variants may be more disease-specific and may play a greater role in the development of a disease. Hence, focusing on these variants may be of interest to better understand the genetics underlying a disease.

Next, we will evaluate the influence of the MAF threshold on the IBD data. Different phenotypes may not be equally influenced by rare variants. The common variants associated with IBD have larger effect size than the common variants identified for CAD. Hence, on the IBD data, the rare variants with stronger effect may not be as crucial for good disease prediction. The results on the IBD data are shown in Figure 4.11b.

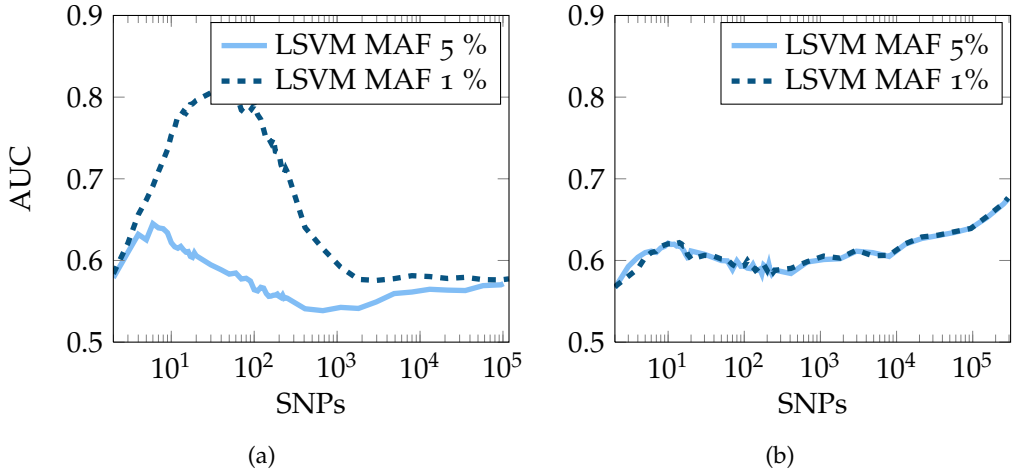


Figure 4.11: Performance of the LSVM on data sets with an MAF threshold of 1% vs. 5%. The p-value ranking was used. A logarithmic (base 10) scale is used for the x-axis. a) CAD data, b) IBD data

In contrast to the results on the CAD data, the performance of the SVM on the IBD data is not significantly altered if we exclude the rare variants. Hence, the rare variants are probably not important for classifying the data and are hence not needed by the algorithm. This may be due to the relatively large effects of the common variants. An alternative reason that the SVM does not use the rare variants may be that the rare variants do not provide additional discriminatory information over the common variants.

Figure 4.12 provides further insight into the reasons for the different behaviours observed on the two data sets. The top two plots plot the p-values for all SNPs, ordered by base pair position, on the two data sets. Alternate chromosomes are shaded light and dark. The rare variants, i.e. SNPs that have an MAF between 1% and 5%, are marked in red. On the CAD data set (Figure 4.12a), we can clearly see that some of the rare variants correspond to the SNPs ranked highest according to the p-values. The SNPs with the best p-values are almost all rare variants. Hence, if we select the SNPs according to the ranking of the p-values, we get a clearly different subset for the two MAF cutoff thresholds.

Conversely, on the IBD data (Figure 4.12b), the rare variants do not overlap with the SNPs ranked highest according to the p-values. Apart from one SNP, all rare variants have rather low-ranked p-values. Hence, in contrast to the CAD data set, if we select the SNPs by the rank of their p-values, we select almost the same subset for the two MAF cutoff thresholds.

Figure 4.12c,d shows similar plots but with OR instead of p-values. For both data sets, the rare variants have relatively large effect sizes compared to the common variants. However, the effect size differs widely. The rare variants of the CAD data set have ORs up to 20, whereas the maximum OR of the IBD data set lies around 2.5. In addition, for the CAD data, the rare variants with large effect sizes correspond to the SNPs ranked high according to the p-values. This effect is not seen on the IBD data set; indeed, none of the rare variants have particularly high p-values anyway. In summary, if we select SNPs by the ranks according to the p-values, then on the CAD data, we select rare SNPs that have large effect size. For the IBD data, the rare variants are not among the highest ranked and have smaller effect size. Hence, these variants do not influence the classification as much as the rare variants in the CAD data.

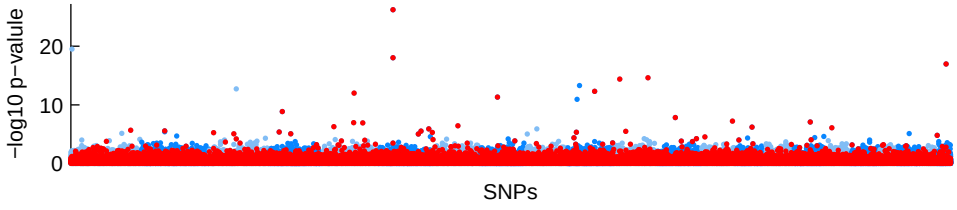
To sum up, the rare variants improve classification performance in some cases while leaving it unchanged in others, but in any case they do not appear to hurt performance. For this reason, a low MAF threshold appears warranted in general, since it may add important discriminatory information for at least some data sets.

4.3 Conclusion SVM

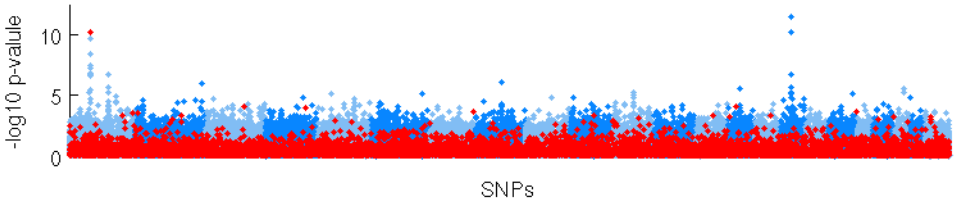
In this chapter, we have evaluated whether we can build a good classifier based on large numbers of SNPs. We have explored the effect of rigid p-value cutoff thresholds as well as a detailed evaluation of the classification performance for SNP subsets of increasing size, ranked by p-values. Moreover, we have compared this p-value ranking with a ranking based on SVM weight vector entries. We have also evaluated the influence of rare variants on classification performance.

We tested the suitability of the SVM for risk prediction on two different GWA datasets, a CAD data set and a IBD data set. Overall, the SVM algorithm achieved convincing performance on both datasets. If we compare the performance of the two algorithms we see that the SVM obtain better results on the CAD data than on the IBD data, a maximum AUC of 0.81 versus 0.68. Whereas the classification with the SVM on the IBD data yields the best results if all SNPs are used (suggesting that the classification is unaffected by noise), this is not true on the CAD data set, where the classification on all SNPs yields poor results. If we however apply a p-value threshold to the CAD data, we obtain good classification performance. In other words, if we select smaller subsets of SNPs according to the ranking of the p-values, and in this way exclude noise SNPs, we can noticeably improve classification.

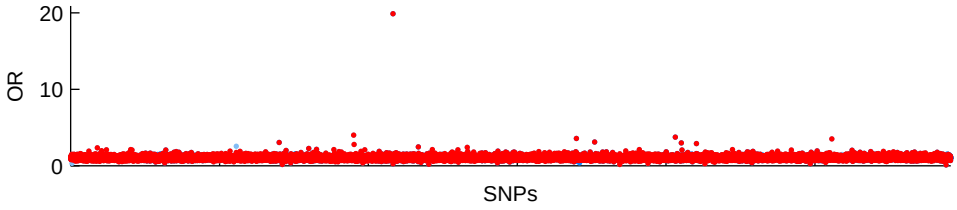
We have in this chapter also evaluated the performance of a ranking based on a score calculated using the SVM classifier. The score for each SNP is obtained from the absolute value of its entry in the normal vector. We observed that the quality of this svmscore is highly influenced by the underlying classifier. A subset that is selected



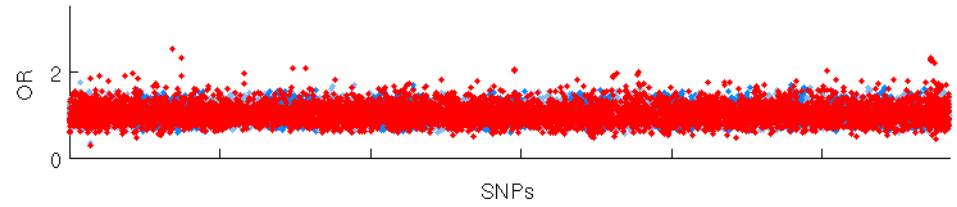
(a) CAD



(b) IBD



(c) CAD



(d) IBD

Figure 4.12: p-values and OR for all SNPs, ordered by base pair position. Alternate chromosomes are shaded light and dark. The SNPs that have an MAF between 1% and 5% are marked in red. a) p-values CAD data b) p-values IBD data c) OR CAD data d) OR IBD data

based on a well-performing classifier improves classification over the p-value ranking, as we saw for the IBD data. On the other hand, an svmscore that is based on a poor classifier leads to poor performance. Because of this, the svmscore ranking performs worse than the p-value ranking for the CAD data.

Finally, we explored the effect of rare variants in the data set. The influence of rare variants on classification performance is different for the two data sets, IBD and CAD. Whereas the performance is worsened if we exclude the rare variables in the CAD data, the performance is unchanged for the IBD data. The reason for this can be explained if we take a closer look at the OR and p-values of the rare variants. For the CAD data set, the rare variants score high in the ranking of the p-values. In addition, we observe that these SNPs have large effect sizes. Hence, the SNPs have a large influence on the classification performance especially if we select SNP subsets according to the p-values. On the IBD data set, the rare SNPs have rather low p-values and are hence not primarily selected. In addition, these SNPs have by far smaller effect sizes and are hence not as predictive as the rare CAD SNPs. Hence, rare variants may improve classification since they may have larger effect size than common variants. If the variants are unpredictive, the SVM will ignore these and classification is not worsened. However, if we include rare variants, we must bear in mind that these are more prone to produce false positive results. Hence, the generalization error may be increased. Nonetheless, we would suggest to keep the rare variants in the data set since, in our experience, they may improve performance significantly and, at worst, do not change it.

5 Classification of GWA data with the SNPboost algorithm

The premise of the previous Chapter (Chapter 4) was that classifying on large ensembles of SNPs improves performance over single-SNP classification because all disease-relevant variables, interactions between the variables and disease-specific structures present in the data can be taken into account. Indeed, as we were able to show, an SVM based on large ensembles of SNPs achieves convincing performance. Classification on large numbers of SNPs clearly performed better than classification on only a few SNPs.

However, the results suggest that classification on all SNPs may be influenced by noise and, as we showed, a better classification can be achieved if we apply a p-value threshold. Whereas the classification with the SVM on the IBD data yields the best results if all SNPs are used, the classification with the SVM on the CAD data yields poor results. However, if we apply a p-value threshold to the CAD data, we obtain good classification results. But is the p-value the most effective measure for ranking SNPs according to their relevance and excluding noise?

The p-value ranking assigns high values only to those SNPs that have an effect individually. To overcome this limitation, we also used the SVM to rank the SNPs. However, we observed that the quality of this ranking is highly influenced by the underlying classifier. A subset that is selected according to the SVM ranking on a well-performing classifier improves classification over the p-value ranking. This is not true if the underlying classifier is poor. Thus, each of the two methods that we have tried so far works well in specific cases, but we would ideally like to have a method that always works well. What could such a method be?

Given a large amount of only marginally predictive variants, a common strategy to improve classification is known under the name "boosting" [46]. The main idea is to use the marginally predictive variants to construct a set of so-called "weak" classifiers, each of which performs only slightly better than chance. Then, several weak classifiers are selected and combined into one strong classifier, which performs by far better than each of the individual weak classifiers. Such boosting algorithms are simple yet some of the most powerful learning strategies.

5.1 Adaboost

Adaboost, introduced by Freund and Schapire [66, 67], is one of the most popular boosting algorithms. The use of Adaboost was popularized by Viola and Jones, who used a cascade of Adaboost classifiers for face and object detection [68]. The premise underlying its selection strategy is that each variable describes different characteristics. In the first step, the best weak classifier (i.e. most predictive variable) is selected. Then, the second classifier is selected to improve the classification. For this purpose, it is not as important that the data points that are correctly classified by the first classifier are also correctly classified by the second. On the contrary, it is more crucial to improve the classification of the incorrectly classified data points.

To focus on these data points, the algorithm applies weights. Initially, all data points have the same weight. Then, after the first classifier has been selected, the weights are increased for the incorrectly classified data points and decreased for the correctly classified ones. With these weights, the selection of the second classifier is influenced more by the incorrectly than by the correctly classified data points.

The remaining classifiers are selected in the same way. The algorithm increases and decreases the weights after each selection step. This means that the data points that are often incorrectly classified obtain ever-increasing influence. In this way, the final classifier aims to account for as much of the variability in the data as possible. Adaboost may hence be a suitable algorithm for classification on GWA data, where each SNP is only marginally predictive.

5.1.1 The Adaboost algorithm

As input the Adaboost algorithm takes data points $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, $\mathbf{x}_i \in \mathbb{R}^D$, and class labels $Y = (y_1, \ldots, y_N)$, $y_i \in \{1, -1\}$. In addition, we have a set H of weak classifiers. Each weak classifier $h \in H$ produces a binary output $h(\mathbf{x}) \in \{1, -1\}$ [46].

As described above, the main idea of boosting is to combine several weak classifiers into a stronger one. The Adaboost algorithm trains the weak classification algorithm several times (in the formalism introduced above, this corresponds to selecting a weak classifier from the set H); each time, the weak classifier sees a slightly different version of the data because, as we have seen above, weights $w_i, i = 1, \ldots, N$ are applied to the data points. The weight of each data point is initialized with $w_i = \frac{1}{N}$.

In each step of the algorithm we determine the weak classifier with the minimal training error $\epsilon(h)$, which is defined as follows:

$$\epsilon(h) = \sum_{i=1}^{N} |h(\mathbf{x}_i) - y_i| \quad w_i. \tag{5.1}$$

We use h_m to denote the weak classifier that is selected in the m'th iteration of the algorithm. For each classifier h_m, we calculate a weight α_m, which will be used to weight the contribution of this weak classifier to the strong classifier. The more accurate a weak classifier is, the larger is its weight. α_m is calculated as follows:

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - \epsilon(h_m)}{\epsilon(h_m)} \right). \tag{5.2}$$

Having determined h_m and calculated the weight α_m of the classifier, the algorithm next updates the weights of the data points. The weights of the correctly classified data points are decreased, and the weights of the incorrectly classified points are increased. The corresponding update rule is

$$w_i^{\text{new}} = \frac{w_i^{\text{old}}}{Z^{\text{old}}} \cdot \begin{cases} e^{-\alpha_m} & \text{if } h_m(\mathbf{x}_i) = y_i \\ e^{\alpha_m} & \text{otherwise,} \end{cases} \tag{5.3}$$

where Z is a normalization factor chosen such that $\|(w_1, \ldots, w_N)\|_2 = 1$.

The algorithm is repeated until a stopping criterion is fulfilled, e.g. a defined number of iterations is completed. At this point, the algorithm has obtained a series of weak classifiers $h_1, \ldots, h_M$. These weak classifiers are then combined to produce a joint output where each classifier is weighted by the weight α_m:

$$P(x) = \text{sign} \left(\sum_{m=1}^{M} \alpha_m \cdot h_m(x) \right). \tag{5.4}$$

5.1.2 Adaboost on GWA data: SNPboost

To apply Adaboost to GWA data, we first have to define the set of weak classifiers H [1]. In this work, each weak classifier predicts the phenotype of an individual based on the genotype of a specific SNP. For instance, a weak classifier could assign all individuals as affected that are homozygous for the first allele and classify as unaffected all individuals with the two other possible genotypes, heterozygous and homozygous for the second allele.

In the general case, there are 8 possible binary outputs (see table 5.1). However, a classifier with only one possible output, i.e. affected or unaffected regardless of the genotype, obviously has no predictive relevance and is of no use for classification. Consequently, we ignore these and obtain a total of 6 weak classifiers per SNP. The total number of weak classifiers for D SNPs is thus $D \cdot 6$. The remaining algorithm proceeds as described above.

input	output							
	1	2	3	4	5	6	7	8
AA	1	-1	-1	-1	1	1	1	-1
AB	-1	1	-1	1	-1	1	1	-1
BB	-1	-1	1	1	1	-1	1	-1

Table 5.1: There are 8 possible binary outputs if we classify based on the genotype of a single SNP. The classifiers with only one possible output, i.e. affected or non-affected regardless of the genotype, are excluded because they are not predictive. A and B are the alleles, with the genotypes AB (heterozygous), AA and BB (homozygous for the first and second allele respectively).

5.2 Results and discussion

To evaluate the SNPboost algorithm, we again tested the performance on the inflammatory bowel disease (IBD) data set and the coronary artery disease (CAD) data set. The performance was evaluated using the area under the curve (AUC). To explore the effect of random variations in the data, we performed a 5-fold cross-validation.

We compared the results with the performance of the SVM, which can be seen as a standard benchmark method for classification. The SNPboost algorithm selects one SNP at a time to successively improve classification. We hence trained the SVM on subsets with a size corresponding to the number of SNPs used so far by the SNPboost algorithm.

5.2.1 Results IBD data

The IBD data set [36] is described in detail in Chapter 2. In brief, the data set contains 1719 individuals, of which 789 are cases and 930 are controls. The total number of SNPs is 292,162 after quality filtering.

First, we evaluated the performance of the SNPboost algorithm and compared it with the performance of the LSVM. For the LSVM, we used the p-value ranking to select SNP subsets. The results are shown in Figure 5.1a.

The SNPboost algorithm clearly performs worse than the LSVM on the p-value ranking. Both algorithms reach a performance peak at around 10 SNPs. SNPboost reaches an AUC of 0.59, and the LSVM on the p-value ranking reaches an AUC of 0.62. The performance of both algorithms declines after this. This decline is however much steeper for the SNPboost algorithm.

SNPboost and the LSVM are both linear classifiers. The main difference between these two algorithms is in how the weights are determined and how the SNPs are selected. A drawback of the SNPboost algorithm is that the weights are determined

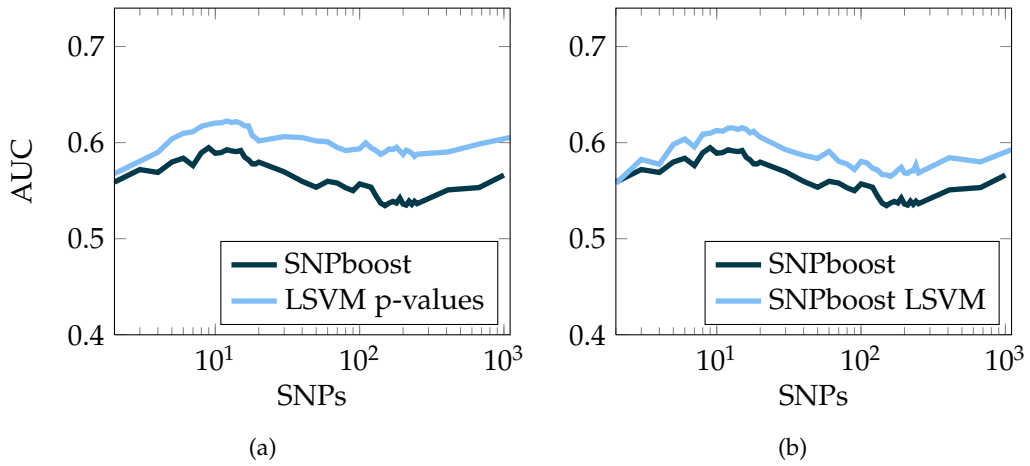


Figure 5.1: IBD data a) SNPboost versus p-value LSVM. b) SNPboost vs. SNPboost LSVM

individually based on the accuracy of the corresponding weak classifiers. This means that the SNPboost algorithm is, in a sense, unaware of interactions between SNPs and cannot fully exploit the structures in the data. The LSVM, on the other hand, adjusts all weights simultaneously to account for the structures and interactions in the data. Because of this, the LSVM may be a better classifier than the SNPboost algorithm.

The second difference between the algorithms is the way in which the SNPs are selected. Does the SNPboost algorithm select more favorable SNPs than the ranking by the p-values? This question inspired us to evaluate whether we can obtain better results with the SVM based on the SNPboost selected SNPs instead on the p-value ranking, i.e. to use the SNPboost algorithm as a feature selection strategy.

SNPboost for feature selection

We first compared the performance of the LSVM on SNPboost selected subsets with the SNPboost algorithm by itself. The results are shown in Figure 5.1b. We clearly see that the performance is improved if we combine the SNPboost feature selection with the LSVM instead of using the SNPboost algorithm alone. For smaller number of SNPs, we see a large difference in the performance. The best AUC is reached with around 10 SNPs for both algorithms: 0.59 for the SNPboost algorithm and 0.62 for the LSVM with SNPboost feature selection.

Hence, as we suspected above, the SNPboost is not the most best-performing classifier probably because it only weights and adds up the effects of the individual SNPs and misses the interactions as well as disease-specific structures. The SVM has a more sophisticated way of determining the weights that can account for these additional structures and in this way improve the classification performance. This may be the rea-

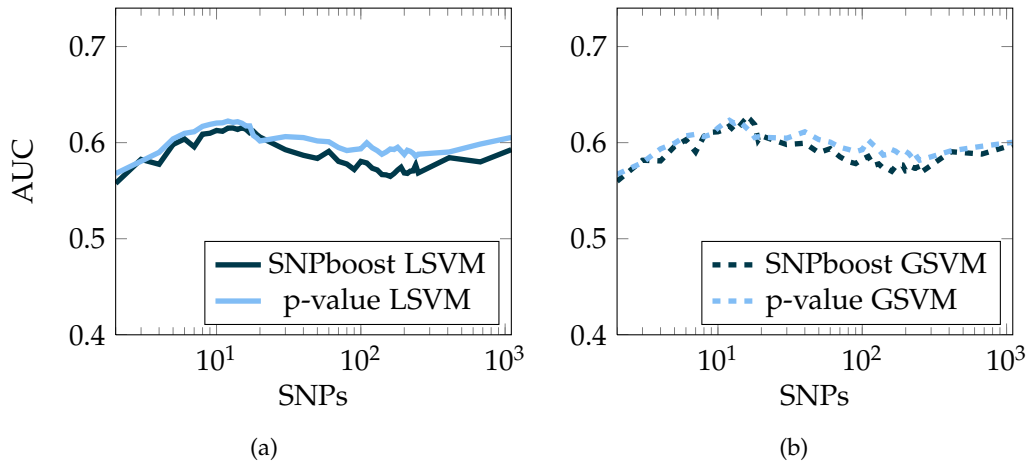


Figure 5.2: IBD data a) SNPboost LSVM versus p-value LSVM. b) SNPboost LSVM versus p-value GSVM

son that the SVM clearly classifies better also on the subsets selected by the SNPboost algorithm.

As discussed above, the SNPboost algorithm aims to select weak classifiers that represent as much of the variability in the data as possible. Hence, the selected SNPs are as little redundant as possible since they are selected successively to improve classification. This selection strategy may thus be more appropriate than the p-value ranking, since it is only based on the individual effects. Hence, we compared the performance of SNPboost feature selection with the p-value ranking. To begin with, we compared the performance using the LSVM. The results are shown in Figure 5.2a.

The performance of the LSVM is nearly equal for both feature selection strategies. SNPboost selection, however, performs better for smaller number of SNPs. For larger subset sizes, p-value selection is slightly superior. The overall best performance lies around 0.62 for both approaches. Note that we only tested the algorithm up to a subset size of 1000 SNPs.

A Gaussian kernel SVM can further improve classification, as discussed in Chapter 4. Hence, we next compared the performance of the two selection strategies using a Gaussian kernel. The results are shown in Figure 5.2b. We see that there is no significant difference between the p-value ranking and the SNPboost selection in the classification performance of the GSVM.

For the SNPboost selection, the overall best performance is marginally increased compared with the LSVM: 0.623 (GSVM) versus 0.615 (LSVM) on the SNPboost subsets. For p-value selection, both the LSVM and GSVM yield the best performance with an AUC of 0.623. The two feature selection approaches are hence comparable. But how

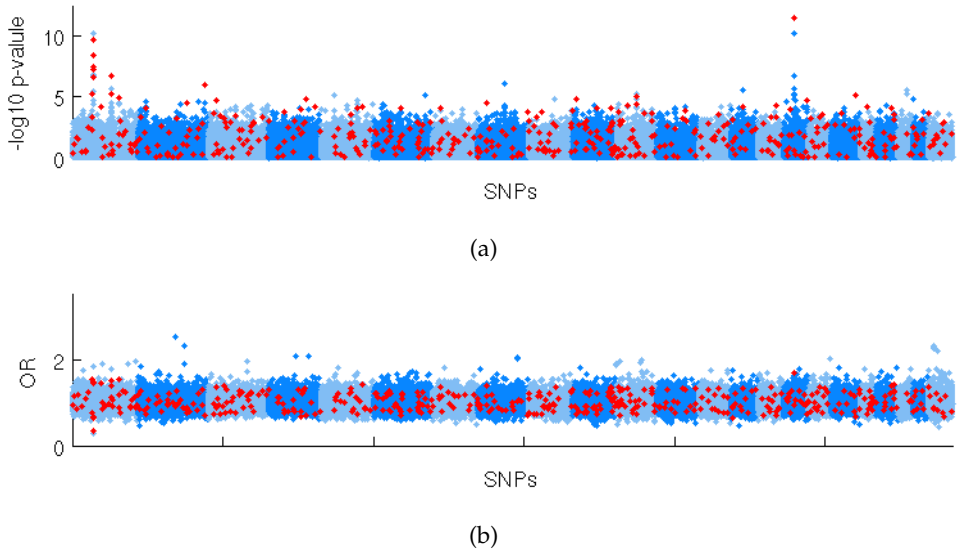


Figure 5.3: SNPs per chromosome according to base pair positions: a) p-value b) odds ratio. The red points represent the SNPboost selected SNPs

do the SNPs that are selected by the SNPboost algorithm and by the p-value ranking differ? This is the question we will investigate in the next section.

The SNPs of the SNPboost algorithm

As we have seen so far, SNPboost seems to perform a suitable feature selection. The performance of the SVM is nearly equal for SNPboost feature selection and the p-value ranking. To better understand the difference between these two selection strategies, we determined the p-values of the SNPs that are selected by the SNPboost algorithm. Figure 4.6 plots the p-values for all SNPs, ordered by base pair position. Alternate chromosomes are shaded light and dark. The SNPs that are selected by the SNPboost algorithm are marked in red; a SNP is included in this set if it is among the 100 first selected SNPs in at least one cross-validation fold. We can see that some of the SNPs that are selected by the SNPboost algorithm overlap with the SNPs ranked highest according to the p-values. However, this does not apply to the majority of the selected SNPs, indeed most of them have rather low p-values.

If we compare the selected SNPs with the effect size (odds ratio) of the individual SNPs, as shown in Figure 5.3b, we again cannot find any correlation. Hence, we can conclude that the SNPs selected using the SNPboost algorithm clearly have other properties which qualify them for classification. We will therefore now take a closer look at the SNPboost selected SNPs.

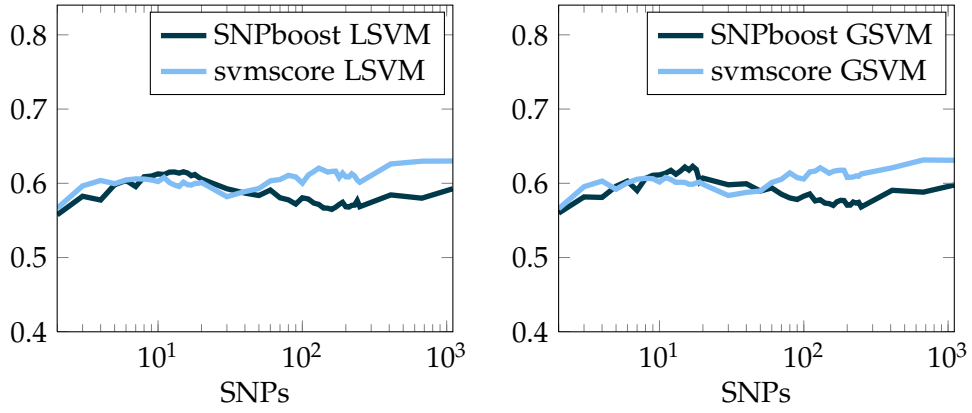


Figure 5.4: IBD. SNPboost versus p-value and svmscore for feature selection. a) LSVM. b) GSVM

To reduce the effect of random variations in the data, we will only consider SNPs selected in multiple cross-validation folds. Among the 20 first selected SNPs, there are two SNPs that occur in each fold. We identify SNPs by their reference SNP ID number ("rs#"), as assigned by the dbSNP database of the NCBI [69]. The first of these is rs2076756 on chromosome 16 in the NOD gene, and the second is rs1942239 on chromosome 18 and does not lie within a gene. rs2076756 is genome-wide significant and is a known susceptibility gene. rs1942239 is not a known susceptibility gene and is also not genome-wide significant. However, since it is selected in each fold, it seems that this SNP is important for classification and hence an interesting candidate for further evaluation.

The pairwise interaction between the two SNPs rs2076756 and rs1942239, as evaluated using logistic regression, is not significant. This is however not surprising since the SNPboost algorithm selects SNPs that represent the variability in the data. The selection is hence not primarily focused on interactions between SNPs.

SNPboost versus SVM feature selection

In the previous section (Chapter 4), we evaluated the classification performance of the SVM on the IBD data. Besides selecting SNPs according to the p-value ranking, we also used the svmscore ranking. On large subsets, the svmscore ranking obtained clearly better results than the p-value ranking. Hence, we next compared the SNPboost feature selection approach with svmscore selection.

First, we compared the performance of the LSVM on the two selection strategies. The results are shown in Figure 5.4a. For subsets between 10 and 40 SNPs, we see that

SNPboost selection improves classification over the svmscore ranking. However, for subsets with more than 40 SNPs, svmscore becomes superior.

The results of the GSVM for the two selection strategies are shown in Figure 5.4b. On SNPboost selected SNPs, the GSVM improves performance over the LSVM. On the p-value selection, the LSVM and GSVM perform equally well. As for the LSVM, we see that the svmscore is clearly better for subsets with more than 40 SNPs. However, for subsets between 10 and 40 SNPs, the SNPboost is superior. Compared to the LSVM, this improvement is widened.

Overall, SNPboost feature selection is less suitable for classification than the svmscore, except for small subset sizes. The superior performance of the SVM with svmscore selection may be because the same basic algorithm is used for selection and classification, i.e. the svmscore selects SNPs that are already known to be important for an SVM classifier. Hence, the selected SNPs may be better suited for the SVM than SNPs selected with other approaches.

On the other hand, the svmscore depends largely on the performance of the classifier that incorporates all SNPs. Hence, for data sets where this all-SNP SVM classification fails, SNPboost selection may be the better choice.

Conclusion IBD data

Overall, the results obtained with the IBD data suggest that the combination of the SVM and SNPboost feature selection yields by far better performance than the SNPboost algorithm alone. Both of these algorithms, SNPboost and the LSVM, are linear classifiers. The main difference is in how the weights are determined. In the SNPboost algorithm, the weights are based on the performance of each individual classifier, causing it to miss the structures and the interactions between the SNPs. Conversely, for the LSVM, the weights are adjusted simultaneously and hence can account for structures and interactions between the SNPs.

To evaluate the feature selection of the SNPboost algorithm in isolation, we ran the LSVM and the GSVM on subsets selected with the SNPboost algorithm. The difference between the LSVM and the GSVM on the SNPboost SNPs is only marginal for small subsets. However, for larger subsets, the GSVM performs better.

SNPboost feature selection is comparable with the selection according to the p-value ranking. The overall best performance lie around 0.62 for both approaches.

Finally, we also compared the results of SNPboost feature selection with the svmscore ranking. For small numbers of SNPs, SNPboost feature selection seems to be better than svmscore selection. However, overall, the performance of svmscore selection is clearly superior.

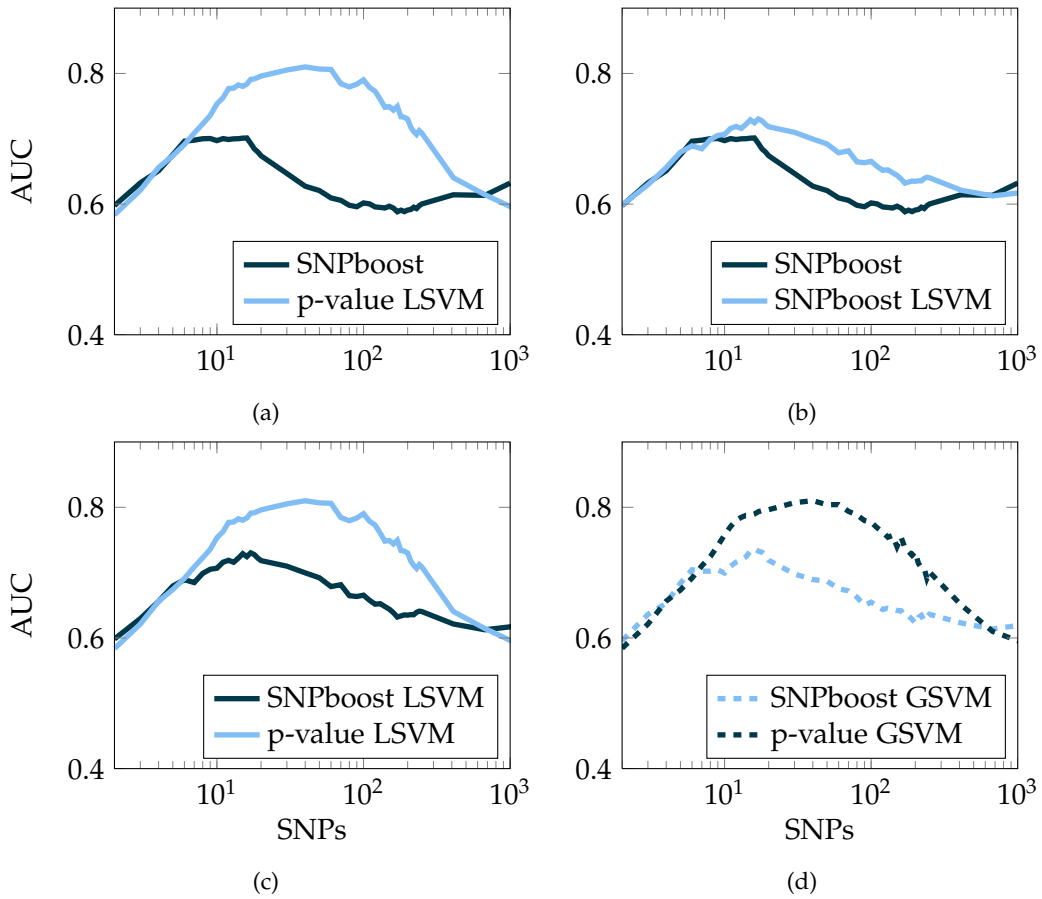


Figure 5.5: GWA2 data. a) SNPboost versus p-value LSVM, b) SNPboost vs. SNPboost LSVM, c) SNPboost vs. p-value LSVM, d) SNPboost vs. SNPboost LSVM.

5.2.2 Results CAD data

Having evaluated the performance of the SNPboost algorithm on the IBD data set, we next tested the performance on the coronary artery disease (CAD) data set [33]. The data is described in Chapter 2. In brief, the data consist of a total of 2506 individuals, with 1222 cases and 1284 controls, and a total of 118247 SNPs after quality filtering and LD pruning.

As in the previous section, we again first compared the performance of the SNPboost algorithm with the LSVM using p-value ranking. The results are shown in Figure 5.5a. In contrast to the performance on the IBD data, we see that the LSVM using p-value ranking performs substantially better than the SNPboost algorithm on larger subsets. For subsets with less than 10 SNPs, we get equal performance.

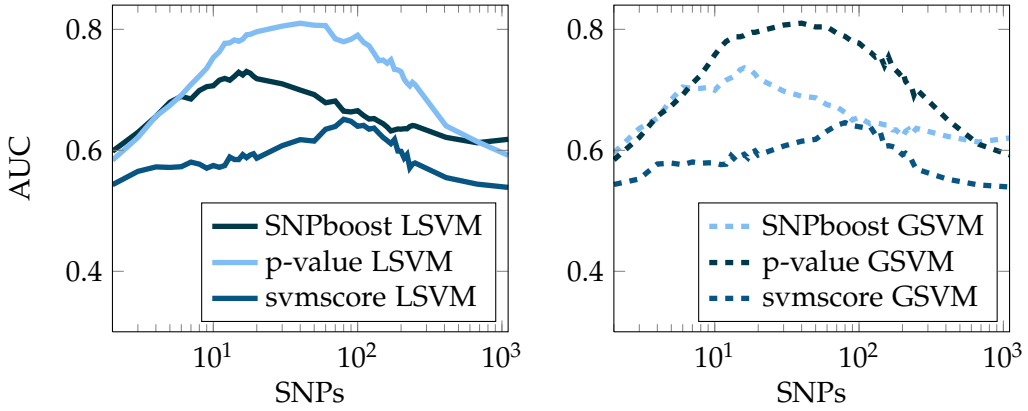


Figure 5.6: GWA2 data. SNPboost versus p-value and svmscore for feature selection. a) LSVM. b) GSVM

However, as we have seen in the previous section, the performance can be improved if we train the LSVM on the SNPs selected by the SNPboost algorithm. We therefore tested whether this also applies to the CAD data set by comparing the performance of SNPboost with the LSVM using SNPboost feature selection. The results are shown in Figure 5.5b. We can clearly see that LSVM using SNPboost feature selection by far outperforms the SNPboost algorithm by itself. This does however not apply for subsets up to a size of 10 SNPs.

If we compare the results of the LSVM using SNPboost feature selection with the LSVM using p-value selection, we see that initially both selection strategies perform equally well, as shown in Figure 5.5c. However, p-value selection is still superior for larger subsets. The results of the GSVM with the two selection strategies are shown in Figure 5.5d. Similar to the LSVM, we again see that p-value selection performs better than SNPboost selection, especially for larger subset sizes.

In the previous section, which dealt with the IBD data set, we also compared SNPboost feature selection to svmscore selection and saw that svmscore is clearly superior for larger numbers of SNPs. We also performed this analysis on the CAD data set, even though we had already seen in Chapter 4 that the performance of the SVM worsens on this data set if we use svmscore selection instead of p-value selection.

As expected, SNPboost selection performs far better than svmscore selection, as shown in Figure 5.6. Hence, even if p-value selection clearly performs best on this data set, SNPboost selection also yields good results and may hence be a suitable alternative approach.

Conclusion CAD

On the CAD data, overall, the SNPboost algorithm clearly performs worse than the SVM with p-value selection; a similar result was seen on the IBD data set. For very small subsets, we get equal performance between the two algorithms. If we train a SVM on the SNPs selected by the SNPboost algorithm, we see that the classification performance is clearly improved over the SNPboost algorithm by itself on larger subsets. However, this SNPboost feature selection, can only keep up with the performance of the p-value selection on subsets with small numbers of SNPs.

Whereas svmscore selection performs poorly on this data set (see Chapter 4), we see that SNPboost selection still yields good classification results, though it does not reach the performance level of p-value selection.

5.2.3 Influence of the minor allele frequency on classification

In Chapter 4.2.3 we explored the effect of rare variants on classification. We applied two different MAF cutoff thresholds: 1% and 5%. In the 5% case, which excluded the rare variants, we saw that the performance was clearly different: The classification based only on relatively frequent SNPs clearly performed worse than the case where the rare variants were included in the data set.

In testing the SNPboost algorithm, we have so far used an MAF cutoff threshold of 1%. To see whether the classification performance of the SNPboost algorithm is influenced by rare variants, we ran it again but this time with an MAF cutoff threshold of 5%.

The results for the CAD data set are shown in Figure 5.7a. Consistent with the results of the SVM on the CAD data (see Chapter 4), we see that the SNPboost algorithm performs better if rare variants are included. Whereas the AUC barely reaches 0.62 with a cutoff threshold of 5%, the maximum AUC for an MAF of 1% lies around 0.7.

Next, we applied the higher MAF of 5% to an LSVM with p-value selection and SNPboost selection. Figure 5.7b shows that p-value selection and SNPboost selection yield nearly equally good classification performance, although classification performance is worse for both approaches than with an MAF of 1%. The performance of the GSVM (see Figure 5.7c) is similar to the LSVM for both selection strategies. Hence, whereas p-value selection is clearly superior with a MAF of 1%, these two approaches now yield similar results with a MAF of 5% .

We now turn to the IBD data set. The performance of the SNPboost algorithm on this data set is not significantly altered if we exclude the rare variants (see Figure 5.7d). This result is consistent with the behaviour we have already observed with the SVM (see Chapter 4). Hence, the rare variables are not as crucial for classification on the IBD data as on the CAD data.

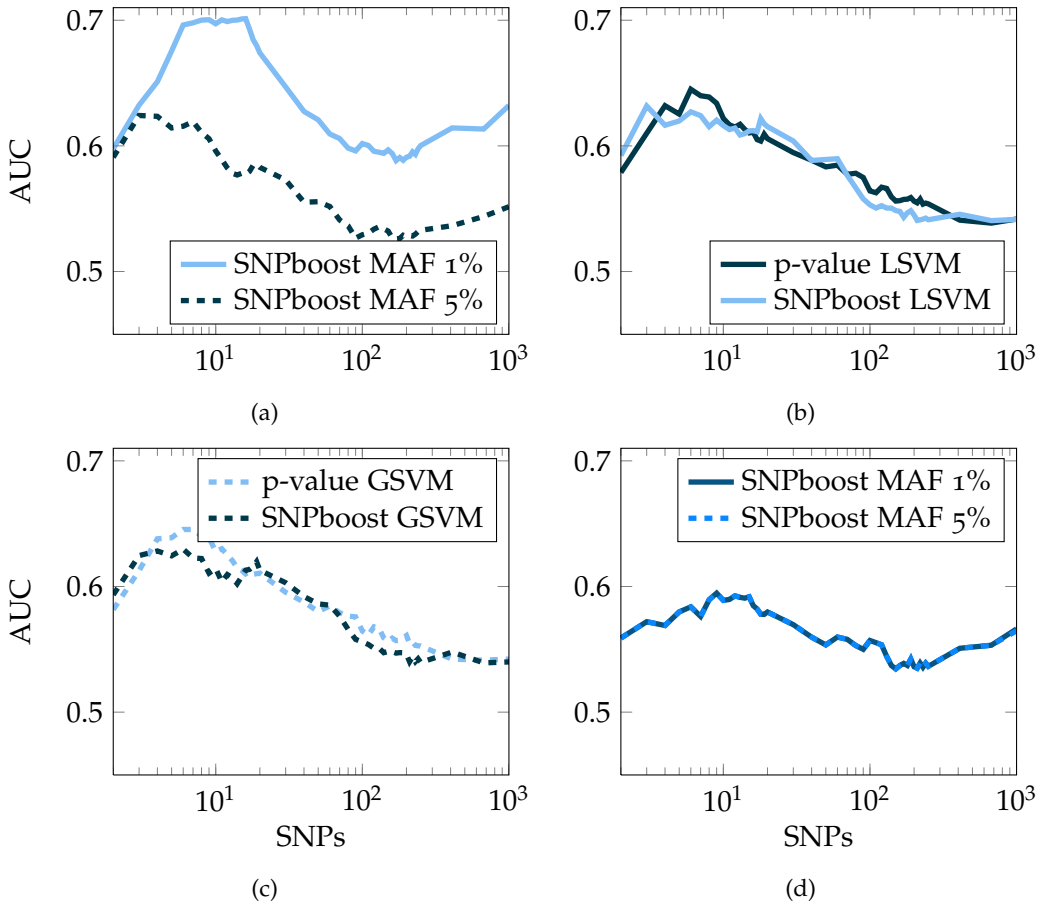


Figure 5.7: Performance of SNPboost as a classifier and as a feature selection approach on data sets with an MAF threshold of 1% vs. 5%. a) SNPboost CAD, b) SNPboost and p-value LSVM CAD, c) SNPboost and p-value GSVM CAD, d) SNPboost IBD.

Conclusion: MAF threshold

On the CAD data, when the MAF cutoff threshold is made more stringent, the performance of the SNPboost algorithm also declines, similar to the result we have already seen for the SVM. On the IBD data, the performance of the SNPboost algorithm on this data set is not significantly altered.

We also tested the performance of the SNPboost algorithm for feature selection. On the CAD data, we saw that the difference between the performance of the SVM using the SNPboost and the p-value selection strategy is by far more distinct with an MAF threshold of 1% than 5%.

5.3 Conclusion SNPboost

In this section, we evaluated whether the SNPboost algorithm is suitable for classification on GWA data. We compared the performance of SNPboost with the performance of the SVM for corresponding numbers of SNPs; the SVM can be seen as a standard benchmark classification approach.

We also evaluated the SNPboost algorithm as a feature selection approach for SVM classification. We compared SNPboost feature selection with p-value selection, a frequently used selection strategy, and with svmscore selection. Finally, we explored whether classification is influenced by inclusion versus exclusion of rare variants.

We performed these evaluations on two data sets, the IBD data set and the CAD data set. Overall, classification based solely on the SNPboost algorithm performs worse than if the algorithm is used as a feature selection approach for the SVM. This is probably due to the fact that the SVM determines weights simultaneously for all SNPs whereas the SNPboost algorithm weights each weak classifier individually, which prevents it from fully exploiting interactions and disease specific structures. The SNP selection strategy of the SNPboost algorithm aims to find a set of SNPs that together cover the variability between subgroups of individuals and hence account for subgroup-specific SNPs. The p-value selection approach, on the other hand, only evaluates how disease-specific a SNP is and does not specifically select SNPs that are characteristic for small subgroups.

SNPboost can also be used as a feature selection approach. Whereas on the IBD data set, SNPboost feature selection is comparable to p-value selection, we see opposite results on the CAD data, where p-value selection performs substantially better than SNPboost selection except for very small subset sizes. This result suggests that the CAD data set is less influenced by subgroup-specific structures, making p-value selection the better choice.

We also compared SNPboost feature selection to svmscore feature selection, which ranks SNPs by their weights in the normal vector of the separating plane (see Chap-

ter 4). The svmscore is highly dependent on the classifier it originates from. For the IBD data, the svmscore seem to be a promising feature selection approach: Its performance clearly exceeds that of p-value selection. On this data set, svmscore selection also performs better overall than SNPboost selection; however, the SNPboost selection is slightly better on small subsets. Conversely, for the CAD data, svmscore selection performs worse than p-value selection. Here, SNPboost feature selection yields by far better performance than svmscore selection and may hence be a promising alternative to p-value selection.

Finally, we evaluated the influence of rare variants on classification performance. A more stringent MAF cutoff threshold (which excludes the rare variants) leads to worsened performance on the CAD data. This is consistent with the result for the SVM (see Chapter 4). On the IBD data, the performance remains nearly equal. Hence, the results again suggest that the rare variants are not as crucial for good classification on the IBD data as they are for the CAD data.

Conclusion part II

We have evaluated different classification approaches to test whether we can build a classifier for risk prediction on GWA data. The variables identified to date explain only a small proportion of the risk of a disease. Hence, if we used only these variables for classification, we would miss the major fraction of the informative variants in the model. To achieve a risk prediction that is as good as possible, we have explored whether classification on large numbers of variables yields better results than using only a few susceptibility loci.

We tested different approaches for classification on two data sets: the coronary artery disease (CAD) data set and the inflammatory bowel disease (IBD) data set.

First, we evaluated the performance of the support vector machine (SVM) (see Chapter 4). Its major advantage is that, unlike the traditional risk models, it is capable of handling both individual effects and interactions. Moreover, the SVM can be applied to very large data sets, which makes it a suitable candidate for classification without first having to reduce the number of variables.

To explore the effect of the number of SNPs used in the prediction model, we formed subsets of increasing size, ranking the SNPs by their p-values. Overall, the SVM achieved satisfactory classification performance for both the CAD and the IBD data set. Classification performance improves clearly if we use more than one SNP for classification. Between the two data sets, there is a clear difference in classification performance. Whereas on the IBD data, classification based on all SNPs yields the best performance, we observed that, on the CAD data, a more optimal subset is needed to obtain a good classification. These results suggest that classification is more strongly influenced by noise on the CAD data than on the IBD data and that we therefore need to restrict ourselves to a small subset of SNPs to eliminate the noise SNPs.

The number of SNPs genotyped in most GWA studies is expected by far to exceed the number of variables that actually influence the risk of a disease. This implies that we have a large amount of noise in the data. As we observed on the CAD data, if we reduce the amount of noise, classification can be improved. Ranking the SNPs by p-values may however not be the best approach, since this ranking only selects SNPs that have an individual effect. In other words, we miss SNPs that only have an appreciable effect in combination with other SNPs.

For this reason, we explored different feature selection approaches for the SVM. First, we ranked the SNPs according to the entries of the weight vector for an SVM

trained on all SNPs. Not surprisingly, this ranking, which we call the svmscore, is highly dependent on the quality of the all-SNP classifier. This is why we obtain opposite results with the svmscore ranking on the two data sets. Whereas the classification on all SNPs yields the best performance on the IBD data, it yields nearly the worst performance on the CAD data. On the IBD data, the svmscore ranking clearly improves classification over the p-value ranking. Conversely, on the CAD data, the svmscore ranking performed worse than the p-value ranking. Hence, the svmscore is clearly not suitable for subset selection on the CAD data set.

Given a large amount of only marginally predictive variants, a common strategy to improve classification is is known under the name "boosting". Hence, in the second chapter of this part, we evaluated the performance of a boosting algorithm for classification as well as feature selection. Boosting algorithms aim to improve classification by combining several weak classifiers into one strong classifier. We established the so-called SNPboost algorithm for use on GWA data. The SNPboost algorithm was, like the SVM, tested on the CAD and IBD data sets.

On both the CAD and IBD data, the LSVM using SNPboost feature selection by far outperforms the SNPboost algorithm alone. The decision functions of the LSVM and SNPboost are quite similar; both are linearly weighted sums. We therefore assume that the LSVM performs better because it assigns better weights. In the SNPboost algorithm, the weights are based on the performance of each individual weak classifier; this prevents SNPboost from fully exploiting interactions and disease-specific structures. In contrast, the LSVM adjusts the weights simultaneously and hence can account for structures and interactions between SNPs.

But SNPboost is still attractive as a feature selection approach because it aims to find a set of SNPs that together cover as much of the variability between subgroups of individuals as possible. The p-value selection, on the other hand, only ranks each SNP separately and does not specifically select SNPs that are characteristic for small subgroups.

On the IBD data, SNPboost feature selection and p-value selection yield comparable SVM classification results. On the CAD data, the p-value ranking is clearly superior. This is not because the SNPboost feature selection performs particularly badly but because p-value ranking performs particularly well here. On this data set, the SNPs ranked highest according to the p-values are also the SNPs with the highest effect size (odds ratio), and these are likely to be a good choice for classification. Moreover, the results suggest that the CAD data set is less influenced by subgroup-specific structures, again making p-value selection the better choice. If we compare all three feature selection strategies on the CAD data set, we observe that the p-value ranking performs best, followed by SNPboost feature selection, with the svmscore ranking performing worst.

We also explored how rare variants influence classification performance by using an MAF of 5% instead of the value of 1% used previously. Whereas classification on the CAD is noticeably worsened if we exclude the rare variables, the performance on the IBD data is not influenced at all. This result is consistent for both classification approaches, the SVM and the SNPboost algorithm, and also for the different feature selection approaches, p-value, svmscore and SNPboost feature selection.

To sum up, we have seen that we can achieve better classification using multiple SNPs than if we classify only on a few variants. On the CAD data set, we have seen that the optimum number of SNPs is smaller than on the IBD data set: On the CAD data, the SVM yields its best performance on a subset of around 40 SNPs selected according to the p-values, and on the IBD data, the best performance is achieved when all SNPs are included.

Part III

Using patterns to rank SNPs

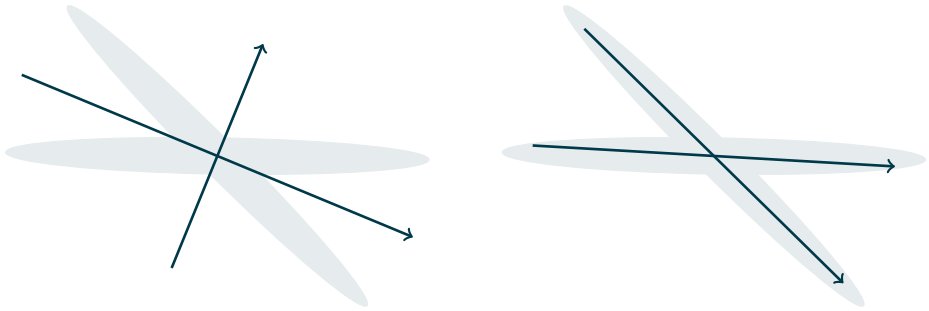
6 Sparse Coding for feature selection

GWA studies have revolutionized the search for genetic variants that influence the risk of common complex diseases. Whereas previous studies were limited by data generation, in the years of GWA studies one now faces a different problem: analyzing the sheer amount of data. Finding the most relevant information among large amounts of noise is a major challenge.

Intuitively, the simplest approach is to select the features according to individual relevance, i.e. according to the p-values. However, as discussed previously, SNPs might interact positively or negatively to increase or decrease the effect of the individual factors. In short, a disease effect might only come about through the interaction of multiple variants. Hence, the selection of SNPs according to the single-locus effect alone is not likely to reveal the more complex genetic mechanisms underlying a disease [28–30].

In the previous chapters, we explored different feature selection strategies for classification using the SVM. SNPboost selection (see Chapter 5) aims to cover as much of the variability in the data as possible. Its advantage is that it accounts for different subgroups in the data. However, a drawback is that it, like p-value selection, selects the SNPs individually. Hence, it may not identify patterns that consist of SNPs without individual effects. The svmscore ranks SNPs according to the entries of the weight vector for an SVM trained on all SNPs (see Chapter 4); its advantage is that it accounts for structures and interactions in the data since it ranks SNPs according to their influence on the classification. However, this means that the svmscore is highly dependent on the quality of the all-SNP classifier. In addition, as we will see later in this chapter, the svmscore is not always able to identify weak SNP patterns in the data.

In summary, two of the feature selection approaches introduced so far (p-value selection and SNPboost selection) do not account for patterns at all, and while the third (svmscore) can account for patterns in principle, we observe that it has significant weaknesses in practice. Identifying SNPs that form disease-specific patterns is an important goal, however, for two reasons: First, they may broaden our knowledge about the genetic mechanisms underlying a disease; and second, if we use these patterns for feature selection, we may improve classification on GWA data. In this chapter, therefore, we will discuss a method that is specifically geared to identifying SNP patterns.



(a) PCA (b) Sparse Coding

Figure 6.1: PCA versus sparse coding. The arrows mark the directions of largest variance obtained by the algorithms. The PCA fails to identify the true effects in the data due to its orthogonal constraint whereas sparse coding identifies the effects correctly.

Patterns alter the covariances in the data. Hence, one way to identify disease-specific structures and interactions is to determine the dimensions that explain as much of the variance in the data as possible.

The classic approach that embodies this idea is the Principal Component Analysis (PCA) [47]; it is one of the most used approaches for dimensionality reduction and feature extraction. The PCA determines the directions of greatest variance in the data. These directions, which are restricted to be orthogonal, are referred to as the principal components (PC) of the data. To use the PCA for feature selection, features are ranked by their influence on the principal components [46].

In this chapter we will explore sparse coding (SC) as an alternative approach for feature selection; to our knowledge, this is the first time this has been done on GWA data. Like the PCA, sparse coding aims to determine the directions of greatest variance. However, in contrast to the PCA, sparse coding does not require its basis elements to be orthogonal. Let us illustrate the difference between the two approaches by an example.

Assume we have a data distribution as shown in Figure 6.1; there are two principal effects in the data along axes that are not orthogonal to each other. We attempt to find these effects using the PCA and sparse coding. Whereas the PCA fails to identify the true effects in the data due to its orthogonal constraint (Figure 6.1a), sparse coding identifies the effects correctly (Figure 6.1b). Hence, sparse coding can identify non-orthogonal structures which are missed by the PCA algorithm.

6.1 Principal component analysis

As input the PCA algorithm [47] takes data samples $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, $\mathbf{x}_i \in \mathbb{R}^D$. The PCA algorithm aims to identify the directions of greatest variance in the data, called the principal components (PC).

In this work we calculate a number M of principal components with a perceptron-like learning rule. The principal components are identified iteratively. In each step of the algorithm, the next principal component is identified under the constraint that it should be orthogonal to the previously identified principal components. The details are described in Chapter 3.2.2.

PCA can be used for feature selection by ranking the features by their influence on the principal components [46]. To use the PCA algorithm for ranking SNPs, we apply it separately to the cases and controls. We refer to the principal components of the cases (class 1) by $V^1 = (\mathbf{v}_1^1, \ldots, \mathbf{v}_M^1)$ and to the principal components of the controls (class -1) by $V^{-1} = (\mathbf{v}_1^{-1}, \ldots, \mathbf{v}_M^{-1})$.

The larger the absolute entry of a principal component, the larger the contribution of the corresponding SNP to this direction of variance. However, SNPs with large contributions are not necessarily useful for classification if they contribute equally for cases and controls. We are particularly interested in SNPs whose contribution differs between cases and controls. We use this idea to construct the following score:

$$s_j = \left| \max_i |(\mathbf{v}_i^1)_j| - \max_i |(\mathbf{v}_i^{-1})_j| \right|. \tag{6.1}$$

The SNPs are then sorted according to the score s_j in descending order. We will in the following refer to this score as the pcascore.

6.2 Sparse Coding

Sparse coding is an algorithm that, like the PCA, aims to identify the directions of greatest variance in the data. However, in contrast to the PCA, the directions of greatest variance are not constrained to be orthogonal.

Each data point can be represented as a linear combination of vectors from a dictionary $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$, with a coefficient vector $\mathbf{a} \in \mathbb{R}^M$. Additionally, we impose the constraint that only a maximum of k entries from the dictionary may be used. The coefficient vector $\mathbf{a}$ is hence a sparse vector, which gives rise to the name sparse coding.

Given a set of data points $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, $\mathbf{x}_i \in \mathbb{R}^D$, we find the dictionary C and coefficient vector $\mathbf{a}_i$ for each data point by minimizing the following objective

function [43]:

$$\min_{\mathbf{c}_1,\ldots,\mathbf{c}_M,\mathbf{a}_1,\ldots,\mathbf{a}_N} \sum_{i=1}^{N} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \quad \text{with} \quad |\mathbf{a}_i|_0 \leq k \quad \text{and} \quad \|\mathbf{c}_l\|_2^2 = 1 \,, \tag{6.2}$$

where $|\mathbf{a}_i|_0$ denotes the number of non-zero entries in the coefficient vector $\mathbf{a}_i$.

We will describe an algorithm for computing this minimum by Labusch et al. [43]; before presenting the algorithm in the general case, we will first consider two easier special cases.

Special case 1: $|C| = 1$

We will first consider the case where $|C| = 1$, i.e. the dictionary contains only one vector. This implies that, instead of coefficient vectors $\mathbf{a}_i$, we have scalar coefficients a_i and hence, the number k of non-zero entries in the coefficient vector is necessarily 1. The optimization problem simplifies to the following:

$$\min_{\mathbf{c},a_1,\ldots,a_N} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{c}a_i\|_2^2 = \min_{\mathbf{c},a_1,\ldots,a_N} \sum_{i=1}^{N} \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i - 2a_i\mathbf{c}^{\mathrm{T}}\mathbf{x}_i + a_i^2 \text{ subject to } \|\mathbf{c}\|_2^2 = 1 \,. \tag{6.3}$$

If we fix $\mathbf{x}_i$ and $\mathbf{c}$, Equation (6.3) is minimal with $a_i = \mathbf{c}^{\mathrm{T}}\mathbf{x}_i$. If we substitute this into Equation (6.3), we obtain:

$$\min_{\mathbf{c}} \sum_{i=1}^{N} \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i - 2(\mathbf{c}^{\mathrm{T}}\mathbf{x}_i)^2 + (\mathbf{c}^{\mathrm{T}}\mathbf{x}_i)^2 \,, \tag{6.4}$$

which is equal to

$$\min_{\mathbf{c}} \sum_{i=1}^{N} \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_i - (\mathbf{c}^{\mathrm{T}}\mathbf{x}_i)^2 \,. \tag{6.5}$$

Since $\mathbf{x}_i$ is fixed, we obtain the final optimization problem:

$$\max_{\mathbf{c}} \sum_{i=1}^{N} (\mathbf{c}^{\mathrm{T}}\mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{c}\|_2^2 = 1 \,. \tag{6.6}$$

This expression corresponds to the variance of the data along the direction of $\mathbf{c}$; hence, Equation (6.3) is minimized if we identify the basis vector $\mathbf{c}$ that corresponds to the direction of greatest variance. This direction of greatest variance can be found using Oja's learning rule [52], as discussed in Chapter 3.2.2.

Special case 2: $k = 1$

So far, we have discussed the simple case $|C| = 1$. We will now see how we can determine a dictionary C with more than one element but will, for the time being, still

impose the constraint $k = 1$, i.e. that the coefficient vector may have only one non-zero entry.

We can solve this variant of the optimization problem using a combination of Oja's rule and the Neural Gas (NG) algorithm [43]. The Neural Gas algorithm is discussed in Chapter 3.2.1.

The algorithm iteratively updates the dictionary and the coefficient vectors until convergence is achieved.

In each iteration, we choose a random data point $\mathbf{x}$. We rank the dictionary vectors by their scalar product with the data point $\mathbf{x}$, in decreasing order:

$$\left(\mathbf{c}_{l_0}^{\mathrm{T}}\mathbf{x}\right)^2 \geq \cdots \geq \left(\mathbf{c}_{l_j}^{\mathrm{T}}\mathbf{x}\right)^2 \geq \cdots \geq \left(\mathbf{c}_{l_{M-1}}^{\mathrm{T}}\mathbf{x}\right)^2 . \tag{6.7}$$

We update each dictionary vector by moving it a certain distance towards the current data point according to their rank:

$$\Delta\mathbf{c}_{l_j} = \alpha_t e^{-j/\lambda_t} y \left(\mathbf{x} - y\mathbf{c}_{l_j}\right) , \tag{6.8}$$

where $y = \mathbf{c}_{l_j}^{\mathrm{T}}\mathbf{x}$, α_t is the learning rate, and λ_t is the neighbourhood size. Both α_t and λ_t are exponentially decreased in each iteration t as follows:

$$\alpha_t = \alpha_0 \left(\alpha_{\text{final}}/\alpha_0\right)^{t/t_{\max}} , \tag{6.9}$$

$$\lambda_t = \lambda_0 \left(\lambda_{\text{final}}/\lambda_0\right)^{t/t_{\max}} . \tag{6.10}$$

The basis vectors are normalized in each learning step.

The coefficient vectors $\mathbf{a}_i$ are determined as described in Chapter 3.1.

General case

So far, we have only allowed $\mathbf{x}_i$ to be represented by an arbitrary single element of the dictionary C that is scaled by the coefficient vector $\mathbf{a}_i$, with $\|\mathbf{a}_i\|_0 \leq 1$. The general version of the sparse coding algorithm uses multiple (at most k) elements of the dictionary C to represent the data point $\mathbf{x}_i$. This corresponds to the general case of the objective function (Equation 6.2) without any further constraints.

The algorithm for solving this general problem is known as Generalized Sparse Coding Neural Gas [43]. Again, the algorithm iteratively updates the dictionary and the coefficient vectors until convergence is achieved. In each iteration, we again choose a random data point $\mathbf{x}$. We use the optimized orthogonal matching pursuit (OOMP) algorithm (see Chapter 3.1) to compute an approximation to $\mathbf{x}$ using k of the current dictionary vectors; in every step of the OOMP algorithm, we update the dictionary vectors using a Neural Gas learning rule as before. An additional twist is that, in

each step of the OOMP algorithm, we remove the projection of the winning dictionary vector from the dictionary and the current data point.

This general idea is implemented as follows: Let $R = (\mathbf{r}_1, \ldots, \mathbf{r}_M) = C$ denote the temporary copy of the dictionary that is made at the beginning of the OOMP algorithm; let $\boldsymbol{\epsilon} = \mathbf{x}$ denote the residual; let $U = \emptyset$ denote the indexes of the dictionary vectors selected by OOMP.

In each iteration of the OOMP algorithm we update C and R according to a combination of Oja's rule and Neural Gas, as before. However, the learning rule is modified to minimize only the distance to the current residual. As before, we rank the dictionary vectors $\mathbf{r}_i$ by their scalar product with the residual $\boldsymbol{\epsilon}$.

$$\left(\mathbf{r}_{l_0}^{\mathrm{T}}\boldsymbol{\epsilon}\right)^2 \geq \cdots \geq \left(\mathbf{r}_{l_j}^{\mathrm{T}}\boldsymbol{\epsilon}\right)^2 \geq \cdots \geq \left(\mathbf{r}_{l_{M-|U|-1}}^{\mathrm{T}}\boldsymbol{\epsilon}\right)^2 \quad , \; l_0, \ldots, l_{M-|U|-1} \notin U \; . \tag{6.11}$$

Note that the ranking is now based on the subspace that is orthogonal to C^U, where C^U are the dictionary elements that have been used so far; also, we consider only those dictionary vectors that have not yet been selected by OOMP. The learning rule is now

$$\begin{aligned} \mathbf{c}_{l_j} &= \mathbf{c}_{l_j} + \Delta_{l_j} \; , \\ \mathbf{r}_{l_j} &= \mathbf{r}_{l_j} + \Delta_{l_j} \end{aligned} \tag{6.12}$$

with

$$\Delta_{l_j} = \alpha_t e^{-j/\lambda_t} y \left(\boldsymbol{\epsilon} - y\mathbf{r}_{l_j}\right) \; , \tag{6.13}$$

where $y = \mathbf{r}_{l_j}^{\mathrm{T}}\boldsymbol{\epsilon}$, and α_t and λ_t are the learning rate and neighbourhood size, as before.

Using Sparse Coding to rank SNPs

As for the PCA, we learn two separate dictionaries, one for the cases and one for the controls [2]. We refer to the dictionary of the cases (class 1) by $C^1 = (\mathbf{c}_1^1, \ldots, \mathbf{c}_M^1)$ and to the dictionary of the controls (class -1) by $C^{-1} = (\mathbf{c}_1^{-1}, \ldots, \mathbf{c}_M^{-1})$.

The larger the absolute value of an entry of a dictionary vector $\mathbf{c}_i$, the larger the contribution of the corresponding SNP to the orientation of this dictionary vector. Hence, we again rank the SNPs in the same way as for the PCA:

$$s_j = \left| \max_i |(\mathbf{c}_i^1)_j| - \max_i |(\mathbf{c}_i^{-1})_j| \right| \; . \tag{6.14}$$

The SNPs are then sorted according to the score s_j in descending order. We will in the following refer to this score as the scscore.

In contrast to PCA, Sparse Coding has two parameters that have to be chosen by the user: M, the number of dictionary vectors, and k, the number of non-zero entries in the coefficient vectors $\mathbf{a}$.

6.3 Results and discussion

To evaluate sparse coding for feature selection, we tested the performance on simulated as well as on real data sets and compared it with state-of-the-art algorithms: selection according to the p-values, the svmscore (see Chapter 4) and the pcascore.

For the PCA and sparse coding algorithms, we have to define the number M of principal components (for PCA) or dictionary vectors (for sparse coding). We tested different choices for M and finally used $M = 5$ for both algorithms. For the sparse coding algorithm, we also have to choose the number k of non-zero entries in the coefficient vectors. As for M we tested different choices for k and finally used $k = 4$.

6.3.1 Performance on simulated data set

Simulated data sets are of great use for evaluating algorithms because the ground truth is known and can be compared to the result of the algorithm, and because the exact properties of the data set can be controlled. In our case, this means that we can incorporate patterns of SNPs in the data set and can easily evaluate how well the algorithms identify these SNPs.

Comparing the selection strategies

We evaluated the performance of the four different feature selection approaches on data sets simulated as described in Chapter 2.1. The first data set we will use contains 1000 individuals, of which 500 are cases and 500 controls. The total number of SNPs is 10,000.

In addition to incorporating disease-specific patterns, we also incorporate disease-unspecific patterns to account for structures induced by linkage disequilibrium (see Chapter 2.1 and 3.4). In total, we incorporated 6 patterns in the data, of which one is disease-specific and 5 are disease-unspecific. Each disease-unspecific pattern consists of 100 individuals of each class, whereas the disease-specific pattern consists of 100 cases only. The number of SNPs in each pattern is 30. The patterns do not overlap.

To assess the consistency of the results, several simulated data sets with the same parameters were created, with similar results in each run. The results on one representative data set are shown in Figure 6.2. Each subfigure plots the SNP scores obtained by one algorithm. The red circles mark the SNPs of the disease-specific pattern.

Figure 6.2a shows the results for the p-value ranking. We see that the highest ranked p-values do not overlap with the SNPs of the disease-specific pattern. In other words, if we select SNPs according to the p-value ranking, we do not identify the relevant SNPs. The reason for this is that the p-value ranking only assigns high values to SNPs with an individual effect. In this data set, however, none of the SNPs that form the disease-specific pattern are associated with the disease.

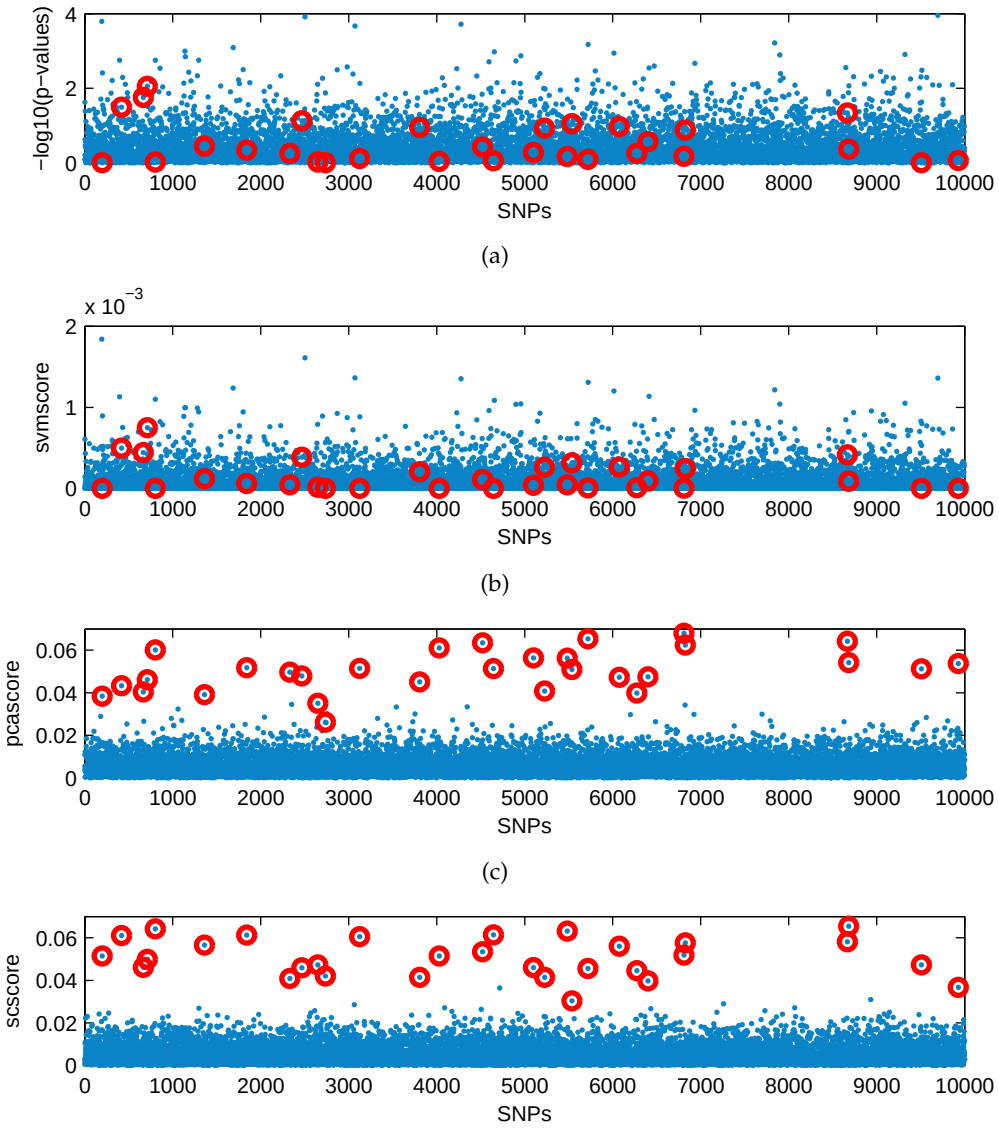


Figure 6.2: Evaluation of four feature selection approaches (p-values, svmscore, pcascore and scscore) on a 10,000 dimensional data set with one disease-specific pattern and 5 disease-unspecific patterns. The SNPs in the disease-specific pattern are marked with red circles. For the p-value and svmscore ranking, the SNPs with the highest ranks do not overlap with the SNPs of the disease-specific pattern. In contrast, for the pcascore and scscore ranking, the SNPs of the disease-specific pattern stand out. The number of principal components for the pcascore was $M = 5$. For the scscore, the number of dictionary vectors was also $M = 5$, and the number of non-zero entries k in the coefficient vectors $\mathbf{a}_i$ was 4.

	total number of SNPs			
	15000	20000	25000	30000
pattern size (1 disease-specific pattern, 1 unspecific pattern)	10	10	10	10
	20	20	20	20
	30	30	30	30
	40	40	40	40
	50	50	50	50
pattern size (1 disease-specific pattern, 5 unspecific patterns)	10	10	10	10
	20	20	20	20
	30	30	30	30

Table 6.1: Simulated data sets with 1 disease-specific pattern and 1 or 5 disease-unspecific patterns.

Figure 6.2b shows the results for the svmscore ranking. Again, the SNPs ranked highest according to the svmscore do not overlap with the SNPs of the disease-specific pattern. Hence, the svmscore also misses the disease-relevant SNPs. This happens despite the fact that the SVM should be able to take advantage of disease-specific patterns. However, in our simulated data set, the disease-specific pattern is very small: The pattern is only specific for $\frac{1}{5}$ of the cases, while $\frac{4}{5}$ of the cases cannot be distinguished from the controls. Without any strong disease-specific patterns or SNPs that enable a classification, the SVM classification plane is mainly based on random structures and the pattern-specific SNPs may be missed. For this reason, the svmscore is not useful for selecting SNPs that are specific only for a small subgroup of individuals when the other SNPs contain only noise.

The results of the selection according to the pcascore and scscore are shown in Figures 6.2c,d. We see that, in contrast to the p-values and svmscore, the pcascore and scscore identify the disease-specific SNPs. Hence, both of these scores are clearly suitable selection strategies on this data set. The reason that PCA and sparse coding can identify the patterns is that they detect the changes in the covariances they cause.

6.3.2 Sparse Coding versus PCA on high-dimensional data

In the previous section, we saw that only the PCA and sparse coding algorithms identify the SNPs of the disease-specific patterns. Hence, in the following, we will compare the performance of only these two approaches in more detail. To do this, we simulated

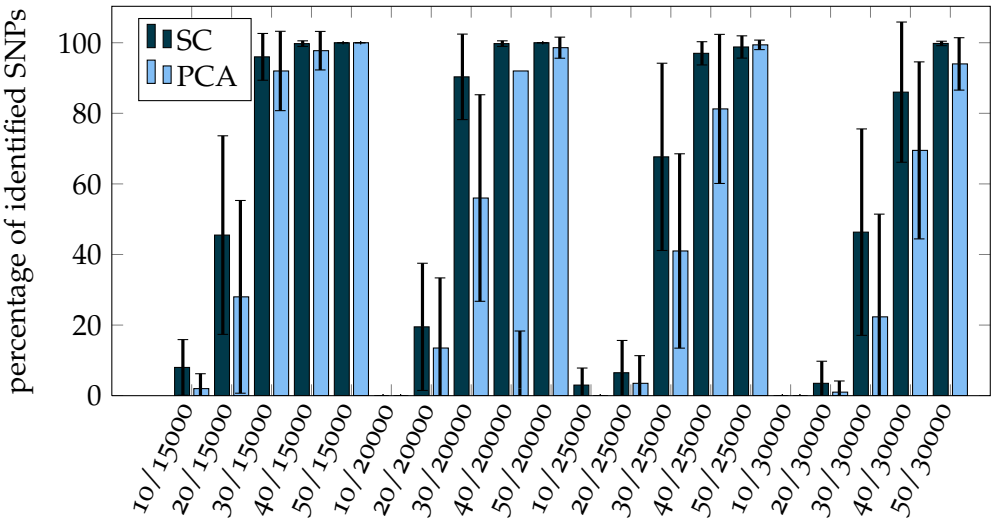


Figure 6.3: Percentage of pattern-specific SNPs among the 100 SNPs ranked highest by the PCA and sparse coding algorithms. Error bars show standard deviations. The data sets contain one disease-specific and one unspecific pattern.

different data sets with a varying total number of SNPs from 15,000–30,000. In addition, we also varied the pattern size from 10 to 50 SNPs in steps of 10. The data sets contained one disease-specific pattern and either one or five disease-unspecific patterns. The parameters of the simulated data sets are shown in Table 6.1.

To explore the effect of random variations in the data, a total of 10 random data sets were generated. We assessed the quality of the results by counting the number of disease-specific SNPs identified by the respective algorithm among the 100 highest-ranked SNPs.

Figure 6.3 shows the results on the data sets containing one disease-unspecific pattern.

Overall, the performance of the two algorithms declines as we increase the total number of SNPs or decrease the size of the disease-specific pattern. The reason for this is probably that we have more noise, making it harder for the algorithms to identify the true patterns among the noise. Sparse coding performs better than the PCA for harder data sets, suggesting that the sparse coding algorithm is less influenced by noise than the PCA.

In addition, we see that if the pattern size is too small (10 SNPs), both algorithms fail to identify the relevant SNPs. The performance on this pattern size is nearly equal for all data sets. We conclude that, to obtain good performance, the pattern needs to

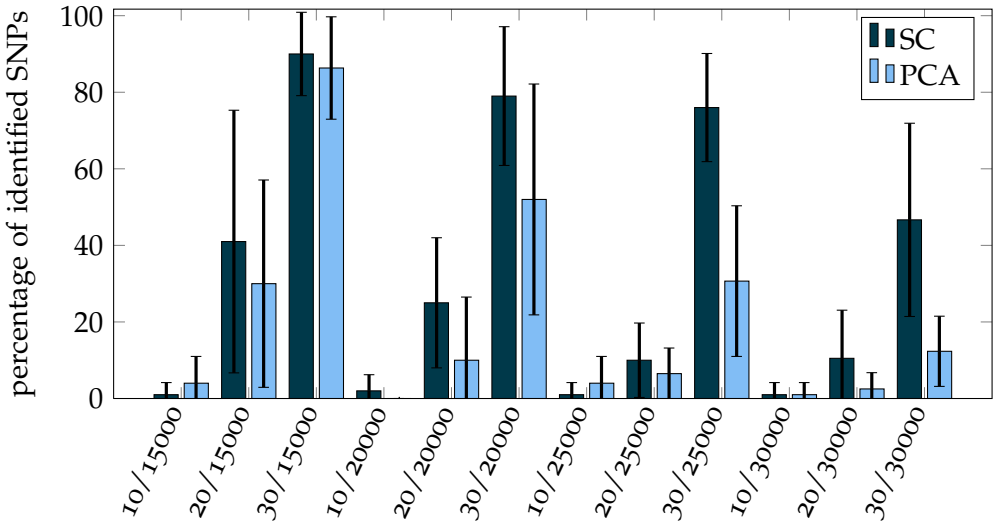


Figure 6.4: Percentage of pattern-specific SNPs among the 100 SNPs ranked highest by the PCA and sparse coding algorithms. Error bars show standard deviations. The data sets contain one disease-specific and five unspecific patterns.

exceed a critical size. On the other hand, we see that for large patterns, both algorithms identify nearly all SNPs on all data sets.

We now turn to the data sets with five disease-unspecific patterns. Here, we only used patterns size from 10 to 30 SNPs to evaluate the algorithms at their limits.

The results are shown in Figure 6.4. Overall, the mean performance of the pcascore ranking is worse than on data sets with only one disease-unspecific pattern. The performance of the scscore ranking is not affected as much. This means that the advantage of the scscore over the pcascore is more pronounced than on the previous data sets.

The reason for the superior performance of the sparse coding algorithm is probably that, unlike the PCA, it does not force the dictionary vectors to be orthogonal. Hence, if we have several non-orthogonal structures in the data, sparse coding may identify all of them whereas the PCA may fail to identify even one. (For an illustration of this, see Figure 6.1.)

6.3.3 Conclusion simulated data sets

We have evaluated the performance of four different feature selection approaches: p-values, svmscore, pcascore and scscore. To evaluate the algorithms, we simulated data sets with disease-specific and disease-unspecific patterns.

Except for the disease-specific pattern, there are no other variables that discriminate between the cases and the controls. In addition, only $\frac{1}{5}$ of the cases are associated with the pattern and can hence be identified at all.

The p-value and svmscore rankings do not give high rank to the disease-specific SNPs. For the p-values, the reason is obvious: It only assigns high rank to SNPs with individual effects. For the svmscore, the reason is more subtle: The SVM aims to find an optimal separating hyperplane between the two classes. However, in this data set, $\frac{4}{5}$ of the cases cannot be distinguished from the controls, and so the SVM apparently overfits to random structures in the data.

PCA and sparse coding clearly outperform the other two approaches. PCA as a feature selection algorithm yields good results for large pattern sizes with only one disease-unspecific pattern. If we introduce five disease-unspecific patterns, the performance of the PCA deteriorates compared to the results with one unspecific pattern.

Sparse coding performed consistently better than the PCA, particularly on high-dimensional datasets with multiple unspecific patterns. This suggests that sparse coding is a promising feature selection approach for identifying disease-specific SNP patterns that only occur for a small number of individuals.

6.3.4 Performance on GWA data

Evaluating the performance on simulated data can help us to better understand the advantages, disadvantages and limits of the tested algorithms. But it is not sufficient to evaluate an algorithm only on simulated data sets. Real data sets have several properties that go beyond what can be simulated, since we have only limited knowledge about the complex genetic relationships between the SNPs and we know neither the size nor the number of disease-specific and disease-unspecific patterns.

In the previous section, we evaluated the performance of different feature selection approaches by counting the number of identified disease-specific SNPs. In these tests, the SNPs formed disease-specific patterns, which are clearly difficult to identify using standard approaches, such as a selection according to the p-values. However, the results on the simulated data sets suggest that the scscore can identify these SNPs. But does the scscore also improve classification on real data sets?

On the simulated data set, we evaluated the performance by counting the number of identified disease-specific SNPs. Since we do not know these SNPs in real GWA data, we instead used the SVM classification performance as our evaluation measure.

We applied the algorithm to the two data sets used previously: the inflammatory bowel disease (IBD) data set [36] and the coronary artery disease (CAD) data set [33]. The data sets are described in detail in Chapter 2.2. To explore random variations in the data, we performed a 5-fold cross-validation.

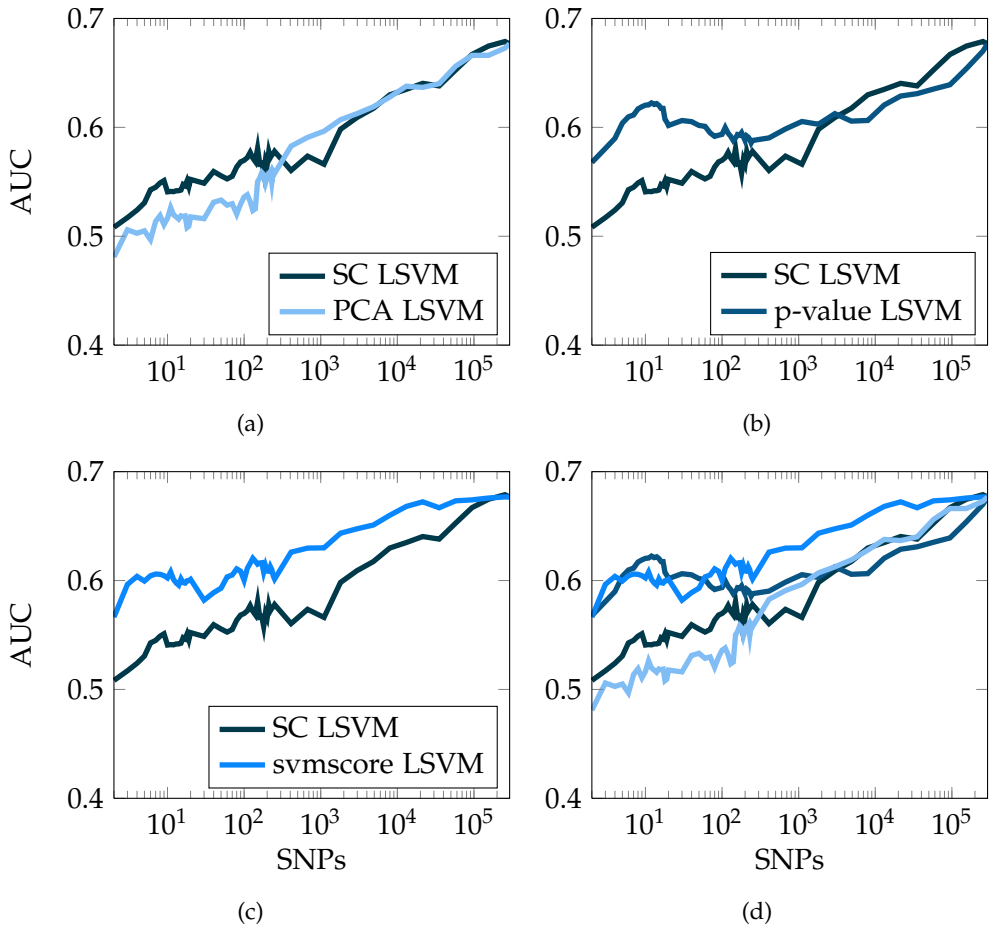


Figure 6.5: IBD data set: Performance (AUC) of a linear SVM on the scscore ranking compared with: a) pcascore ranking, b) p-value ranking, c) svmscore ranking, d) all algorithms. A logarithmic (base 10) scale is used for the x-axis.

6.3.5 Performance on the IBD data set

The IBD data set [36] is described in Chapter 2.2.2. In brief, the data set contains 1719 individuals, of which 789 are cases and 930 are controls. The total number of SNPs is 292,162 after quality filtering.

We first compared the performance of the linear kernel SVM (LSVM) on pcascore and scscore ranking (see Figure 6.5a). The LSVM on the scscore ranking reaches the best performance prior to including all SNPs, with 253,869 SNPs and an AUC of 0.679. The LSVM on the pcascore ranking reaches the best performance if all SNPs are included (AUC of 0.677). In other words, the pcascore cannot identify a subset that improves the classification over all SNPs. In contrast, the scscore can identify a subset

of SNPs that enables better classification. Even if the performance is only marginally improved this is still a notable result since a substantially smaller number of SNPs is required.

If we evaluate the performance of the two selection approaches in more detail, we see that initially, the performance of the LSVM gradually improves with increasing subset size. For subsets up to around 400 SNPs, the scscore ranking clearly outperforms the pcascore ranking. Then, the pcascore selection yields the better results, up to a subset size of over 1800 SNPs, after which both selection strategies yield comparable results throughout the remaining subsets.

Next, we compared the performance of the scscore with the p-value ranking (see Figure 6.5b). The p-value ranking is superior for smaller subset sizes, whereas, for subsets with more than 4800 SNPs, the scscore yields the better results. The p-value selection cannot identify a subset that performs better than the LSVM on all SNPs.

Finally, we compared the scscore with the svmscore (see Figure 6.5c). Overall, the svmscore is clearly superior. However, like the p-value ranking and pcascore, the svmscore cannot identify a subset that performs better than the classification on all SNPs. This means that the best performance overall is reached when using the scscore.

As discussed in the previous chapters (see Chapter 4), a Gaussian instead of a linear kernel may improve the classification performance. Hence, we repeated the tests with a GSVM. The results are shown in Figure 6.6.

As above, we will first discuss the performance of the pca- and scscore ranking (see Figure 6.6a). Overall, the shape of the performance curve is similar for the LSVM and GSVM. However, for small numbers of SNPs (up to around 10), the GSVM performs better than the LSVM. Beyond this point, the GSVM on the scscore continues to perform better than the corresponding LSVM up to more than 13,000 SNPs. This is not, however, true for the pcascore. Hence, for subsets up to 400 SNPs, the difference between the two feature selection approaches is even larger on the GSVM than on the LSVM. For subset sizes between 400 and 1800, the pcascore and scscore GSVM perform equally well, in contrast to the result for the LSVM.

The best performance is equal to the results of the LSVM with a maximum AUC of 0.677 and 0.679 for the pcascore and scscore, respectively. For the scscore, the best performance is reached prior to including all SNPs, with 154,860 SNPs. This is again not the case for the pcascore.

Next, we compared the performance of the GSVM on the scscore and the p-value ranking. For subsets up to 240 SNPs, p-value ranking is clearly superior on the GSVM. The p-value and scscore ranking perform equally good up to subsets with 4800 SNPs. After this point, the scscore performs better than the p-value ranking throughout the remaining subset sizes. The best performance is reached by the scscore.

Finally, we compared the performance of the scscore with the svmscore ranking. The LSVM and GSVM yield equal results on subsets ranked with the svmscore, whereas

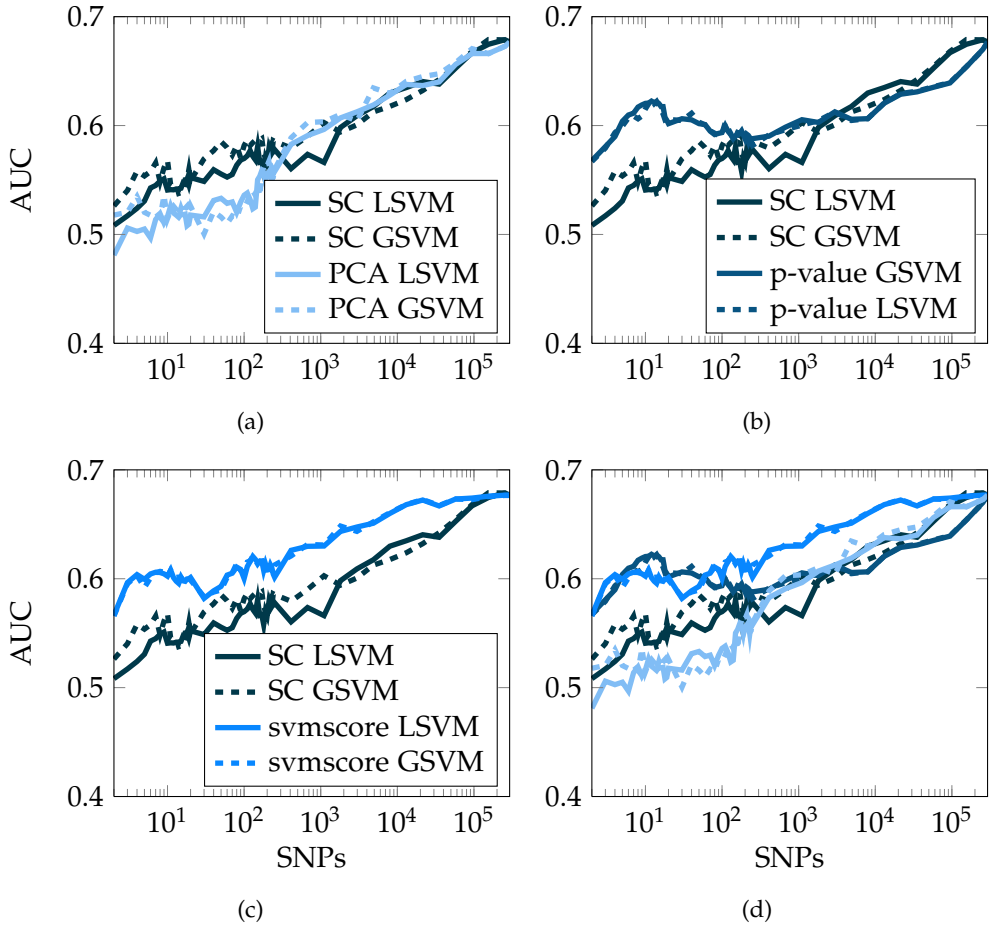


Figure 6.6: IBD data set: Performance of the GSVM (dashed line) and LSVM (solid line) on the scscore ranking compared with: a) pcascore ranking, b) p-value ranking, c) svmscore ranking, d) all algorithms. A logarithmic (base 10) scale is used for the x-axis.

the GSVM performs better than the LSVM on the scscore subsets. This means that the difference between the two selection algorithms is narrowed.

To sum up, the pcascore, svmscore and p-value ranking yield the best performance if all SNPs are included, with an AUC of about 0.677. The scscore reaches the best performance prior to including all SNPs, with an AUC of 0.679, and this is also better than the best result on the LSVM. Hence, the best performance overall is obtained by the GSVM on a subset of SNPs selected using the scscore.

The selected SNPs

As we have just seen, using the scscore ranking to select subsets of SNPs improves the performance of the SVM noticeably compared to the p-value ranking on large subset sizes. But how do the selected SNPs differ and to which extent?

To better understand the difference between the two ranking strategies, we determined the p-values of the SNPs with the highest scscores. Figure 6.7a plots the p-values for all SNPs (vertical axis), ordered by base pair position (horizontal axis). Alternate chromosomes are shaded light and dark. The SNPs that rank highest on the scscore are marked in red; a SNP is included in this set if it is among the 3000 highest-ranked SNPs in at least one cross-validation fold.

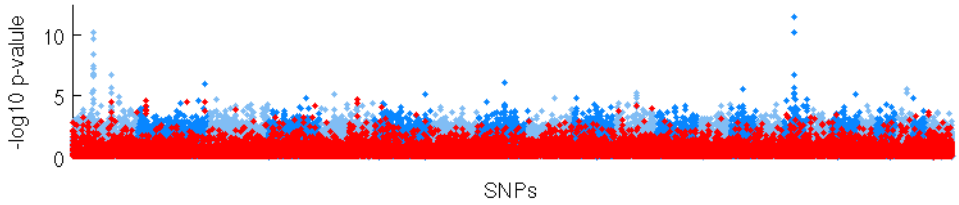
We observe that the SNPs that have a high scscore rank do not overlap with the SNPs ranked highest according to the p-values. Indeed, the SNPs selected using the scscore have rather low p-values. Hence, we can conclude that these SNPs have other features which qualify them for classification.

In contrast to the p-values, which only assign a high value to those SNPs that have an effect individually, the scscore is based on the patterns in the data that differ between the cases and the controls. In other words, the scscore assigns high values to SNPs of disease-specific structures, and we can conjecture that it is these structures that improve the classification performance.

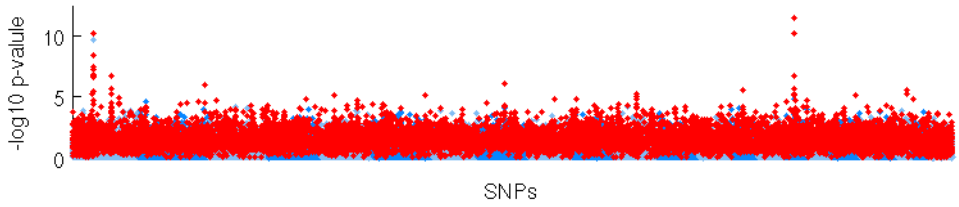
Figure 6.7b shows the same type of evaluation but for the svmscore, which, as we have seen, yields even better overall performance than the scscore. The most noticeable difference between the two selection strategies is that many of the SNPs in the svmscore subset have high p-values, whereas the scscore subset consists mainly of SNPs with very low p-value ranks.

Note that, in contrast to the similar comparison in Chapter 4.2.1, we now use a much larger subset: A SNP is now included in the subset if it is among the 3000 highest ranked SNPs in at least one cross-validation fold. Due to the larger subset size, we now see a large overlap between the SNPs with high p-values and high svmscores.

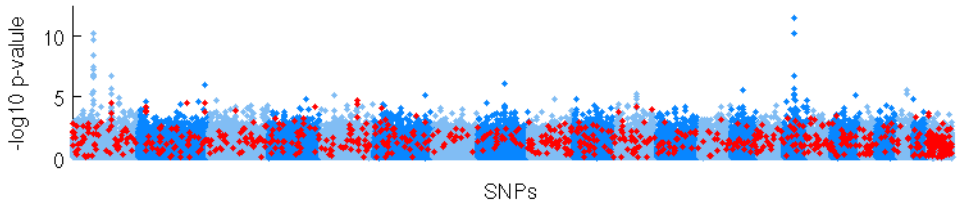
Finally, Figure 6.7c shows the same type of evaluation but includes only those SNPs that are selected by both selection strategies. What is most noticeable is that the overlap between the scscore SNPs and svmscore SNPs is quite small. The overlap is less than



(a) SNPs selected by scscore



(b) SNPs selected by svmscore



(c) SNPs selected by both scscore and svmscore

Figure 6.7: IBD data set, comparison of feature selection strategies. The graph plots p-values for all SNPs (vertical axis), ordered by base pair position (horizontal axis). Alternate chromosomes are shaded light and dark. The SNPs that are selected by the given feature selection strategy are marked in red; a SNP is included in this set if it is among the 3000 highest-ranked SNPs in at least one cross-validation fold. a) scscore feature selection; b) svmscore feature selection; c) combined scscore and svmscore feature selection; only SNPs that were selected by both feature selection strategies are marked red.

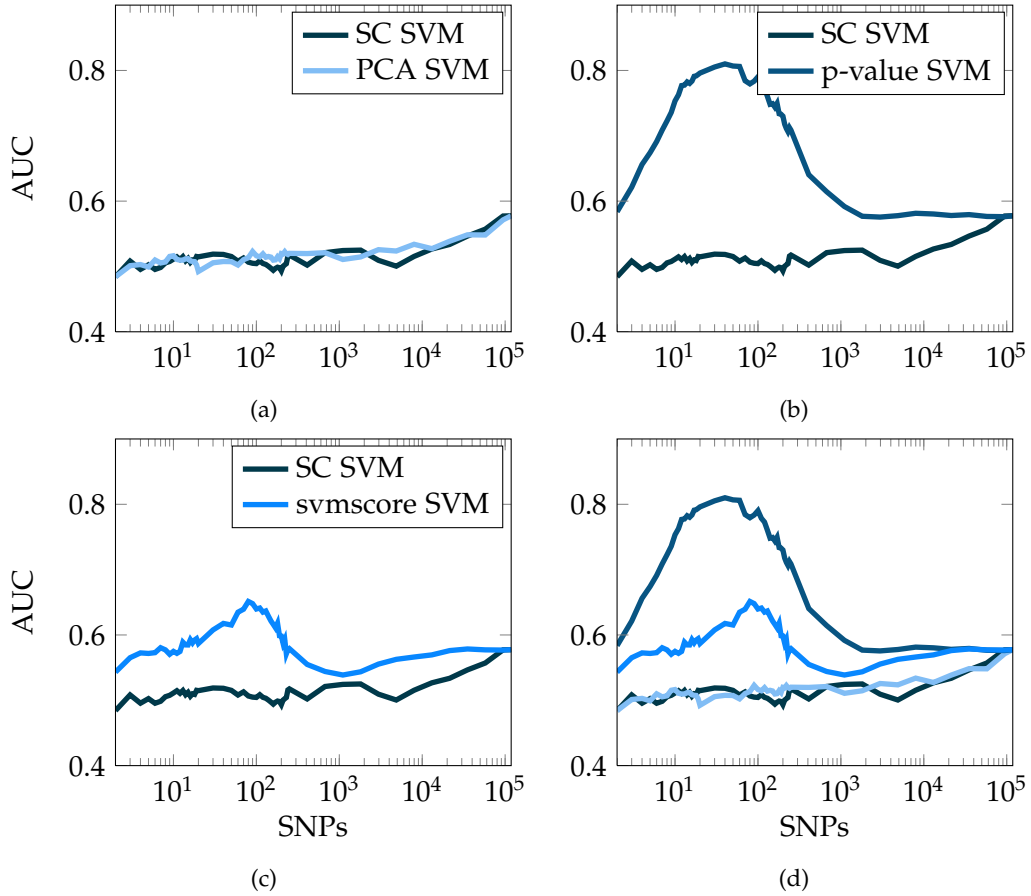


Figure 6.8: CAD data set: Performance of the linear SVM on the scscore ranking compared with: a) pcascore ranking, b) p-value ranking, c) svmscore ranking, d) all algorithms. A logarithmic (base 10) scale is used for the x-axis.

80 SNPs for each fold. The SNPs that do overlap correspond to scscore SNPs with higher p-values. Because, as we have seen, the svmscore yields better results than the scscore and tends to select SNPs with higher p-values, it seems that it is important to include these SNPs to achieve good classification.

6.3.6 Performance on the CAD data set

Having evaluated the sparse coding algorithm on the IBD data set, we will next evaluate it on the coronary artery disease (CAD) data set [33]. This data set is described in Chapter 2.2.1. Briefly, the data set contains 2506 individuals, of which 1222 are cases and 1284 are controls. The total number of SNPs is 118,247 after quality filtering and

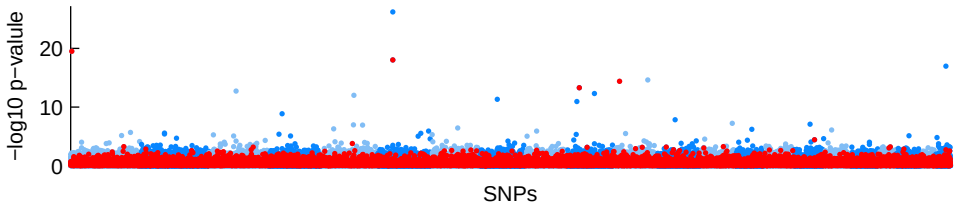


Figure 6.9: CAD data set. The graph plots p-values for all SNPs (vertical axis), ordered by base pair position (horizontal axis). Alternate chromosomes are shaded light and dark. The SNPs that rank highest on the scscore are marked in red; a SNP is included in this set if it is among the 3000 highest ranked SNPs in at least one cross-validation fold.

LD pruning. As for the IBD data, we performed a 5-fold cross-validation to account for random variations in the data, The results are shown in Figure 6.8.

We first evaluated the performance on the LSVM. Figure 6.8a compares the performance of the scscore and pcascore rankings. We see that both selection strategies yield poor performance, with an AUC of around 0.5, comparable to chance. The best performance is reached if all SNPs are included (AUC of 0.58). Hence, the selection strategies do not identify a subset which improves classification and do not provide a real benefit.

Next, we compared the performance of the scscore and p-value rankings (see Figure 6.8b). In contrast to the scscore, the p-value ranking succeeds in identifying a subset that improves classification, with an AUC of around 0.81.

The svmscore ranking also clearly outperforms scscore ranking (see Figure 6.8c). Hence, we can conclude that the failure of the scscore and pcascore is not because feature selection is generally impossible on this data set.

There may be several reasons why the scscore and pcasore perform badly. First, the algorithms may overfit in this case, i.e. they may identify patterns that are only specific for the training data set; we will explore this issue more closely in Section 6.3.7. Second, the number of unspecific patterns may be large compared to the disease-specific ones. Because the algorithms begin by determining patterns in the two groups independently, they cannot differentiate between disease-specific and unspecific patterns at this point. Hence, if unspecific patterns dominate the data set, the algorithms will concentrate on them and hence not identify the truly discriminative patterns.

The selected SNPs

On the CAD data set, as we have just seen, the scscore ranking fails to select subsets that enable meaningful classification, whereas the p-value ranking delivers good results. What, then, is the difference between the scscore and the p-value ranking?

To explore this question, we determined the p-values of the SNPs with the highest scscores.

Figure 6.9 plots the p-values of all SNPs (vertical axis) ordered by base pair position (horizontal axis). The SNPs that rank highest on the scscore are marked in red; a SNP is included in this set if it is among the 3000 highest-ranked SNPs in at least one cross-validation fold.

We observe that the SNPs that have high scscore rank overlap to some extent with the SNPs ranked highest according to the p-values. However, the majority of the scscore SNPs have rather low p-values. As in the the equivalent figure for IBD data set, we again see that the majority of the selected SNPs have very low-ranked p-values. In other words, based on the p-values, we cannot identify an obvious difference between the behaviour of the scscore on the two data sets.

6.3.7 Does the sparse coding algorithm overfit?

We have already speculated that the bad performance of the scscore on the CAD data set may be due to overfitting. To explore this further, we will look at the relative fraction of SNPs that is identical over all cross-validation folds; we call this the overlap fraction. If overfitting is indeed taking place, we would expect this metric to be low, since the SNPs are essentially being selected at random. If truly disease-specific SNPs are selected, we would expect the same SNPs to be selected in each fold, causing the overlap fraction to be higher.

The overlap fraction is computed as follows. Let c be the number of cross-validation folds, and let s be the number of SNPs per fold. We now determine the number u of unique SNPs across all cross-validation folds; from this, we can calculate the number of duplicates $d = c \cdot s - u$. The maximum number of duplicates is reached if all cross-validation folds yield the same result, i.e. for 100% overlap; in this case, we have $u = s$ unique SNPs and hence $(c-1) \cdot s$ duplicates. We therefore compute the overlap fraction o as

$$o = \frac{d}{(c-1) \cdot s}. \tag{6.15}$$

Figure 6.10 plots the overlap fraction for subset sizes from 10 to 10,000 SNPs. Overall, there is noticeable overlap for both data sets on all subset sizes and definitely more than would be expected by chance (see the dashed line). For the most part, the overlap between the cross-validation folds lies between 20% and 30%.

On the CAD data, for subsets with more than 300 SNPs, the overlap fraction increases as the subset size increases. A similar effect occurs on the IBD data, but it only sets in after around 2000 SNPs.

The larger the subset size, the larger the overlap simply because a greater fraction of the SNPs are selected. However, this effect does not dominate the curve since we

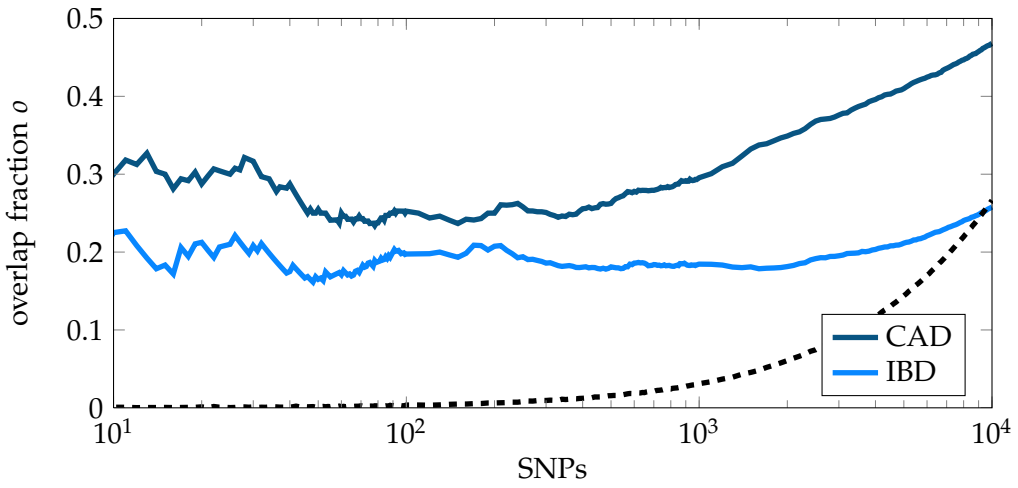


Figure 6.10: The relative fraction o of SNPs that overlap between cross-validation folds as a function of subset size. The dashed line shows the random overlap fraction for a dataset with 100,000 SNPs. The IBD data set consist of 292,162 SNPs and the CAD of 118,247 SNPs

only use subsets of up to 10,000 SNPs and the total number of SNPs is in the hundreds of thousands. Hence, we assume that the overlap fraction is a valid measure for the consistency of the sparse coding algorithm.

Given the poor performance of sparse coding feature selection on the CAD data set, one could intuitively expect that the subsets are overfitted to the fold on which they are selected; this would result in a small overlap fraction. It is therefore surprising that the overlap fraction for the CAD data is noticeably higher than for the IBD data. Even if the maximum overlap on the CAD data of 30% is relatively small, one would still expect reasonable classification to be possible, since the overlapping fraction should describe patterns that are specific for the phenotype. If instead we select a random subset of SNPs, we obtain an AUC of around 0.5. On the IBD data, where the overlap is smaller than on the CAD data, the scscore ranking yields convincing performance. This implies that the poor performance on the CAD data is not primarily due to overfitting. Instead, the results suggest that the scscore algorithm does identify real patterns in the CAD data, but not ones that are disease-specific.

An explanation for this could be the number of disease-unspecific patterns. If there are many of these patterns, they may overshadow the disease-specific ones. In this case, one solution to improve the selection would be to increase the number of dictionary vectors learned by the algorithm. If we use too few dictionary vectors, we may not be able to include all patterns in the data; in particular, the dictionary may miss the disease-specific patterns.

6.3.8 Conclusion GWA data

As we have seen above, the performance of the scscore ranking is not consistent over the two data sets on which we tested it. Whereas on the IBD data, the scscore achieves the best results of all feature selection methods, it fails completely on the CAD data. In an attempt to explain this difference, we determined the degree of overlap between the SNPs of the different cross-validation folds. On the CAD data, the overlap fraction was actually higher than on the IBD data set. In other words, we cannot explain the failure on the CAD data by overfitting. Instead, the results suggest that the sparse coding algorithm identifies the same patterns for both classes on the CAD data set and thus does not select discriminatory SNPs.

On the IBD data, the pcascore, svmscore and p-value ranking yield the best performance if all SNPs are included, whereas the scscore reaches its best performance prior to including all SNPs. In other words, the scscore ranking is the only approach that yields a subset on which the SVM classifies better than on all SNPs.

6.4 Conclusion sparse coding

In the first part of this thesis we explored different classification approaches for risk prediction on large numbers of SNPs (see Chapters 4 and 5). The number of SNPs in most GWA data sets is expected by far to exceed the number of SNPs truly associated with the disease. Hence, classification may be improved if we exclude irrelevant SNPs to reduce the amount of noise. There are several strategies for excluding disease-unspecific SNPs. One strategy is to select SNPs that are individually associated with the disease. However, we do not expect all disease-specific SNPs to have an individual effect; SNPs that only have an appreciable effect when they occur together may also be important for classification.

Disease-specific patterns are however not straightforward to identify. In this chapter, we have evaluated whether principal component analysis (PCA) and sparse coding are suitable tools for this task. Both PCA and sparse coding are approaches that identify structures in data on the premise that structures correspond to the directions of greatest variance. We use the PCA and sparse coding to score SNPs; the larger the influence of a SNP is on a direction of variance identified by the algorithm, the larger is the so-called pcascore or scscore.

To test whether SNP patterns can be identified using the pcascore and scscore, we first evaluated the algorithms on simulated data sets. We also evaluated the p-value ranking and svmscore on these data sets. We simulated data sets with a pattern specific for the disease phenotype and tested whether the algorithms identify these SNPs and to which extent. Whereas the p-value ranking and svmscore clearly fail to identify the SNPs, the relevant SNPs scored high on both the pcascore and scscore. However, the

performance for these two latter algorithms does decrease with increasing numbers of unspecific patterns in the data. The size of the patterns also strongly influences the performance: If the size of the pattern is too small, the algorithm fails to identify it. The scscore performs consistently better than the pcascore and is thus the most suitable feature selection approach on the simulated data.

We then proceeded to evaluate the four feature selection strategies on real data. Whereas on simulated data sets, we can assess the quality of the results by counting the number of disease-specific SNPs identified by the algorithms, on real GWA data, we do not know these SNPs. Hence, we evaluated the algorithms by the performance of the SVM on the selected subsets. We obtain different results for the IBD and CAD data. On the IBD data, the scscore ranking yields a subset that improves the classification over that on all SNPs. On the CAD data, however, the scscore fails to select a suitable subset; except on very large numbers of SNPs, its performance is equal to that obtained by chance.

As discussed above, there are probably many reasons why scscore selection fails to identify disease-specific patterns on the CAD data set. As we experienced on the simulated data set, the algorithm clearly struggles to identify patterns that consist only of a small number of SNPs. Hence, if the patterns are too small, the algorithm cannot identify them.

In the simulated data set, we incorporated a defined number of disease-unspecific patterns to test how they influence the performance of the algorithms. However, we do not know how many such patterns exist in real data sets and how large they are. In the simulated data set, we saw that more disease-unspecific patterns decrease the performance. The more unspecific patterns there are in the data, the more the algorithm struggles to identify the disease-specific ones. We conclude that one possible reason why the scscore fails on the CAD data may be that the number of unspecific patterns is large. In such cases, the dictionary that is learned by the sparse coding algorithm may not be large enough to contain all structures and may hence miss the disease-specific ones. One possible solution is to increase the number of basis vectors in the learned dictionary. A drawback of this solution is, however, that this further increases the computational effort of an already computationally demanding algorithm.

A further weakness of the sparse coding selection strategy is that we apply the algorithm to the two groups (cases and controls) separately. If we have strong unspecific patterns in the data, the sparse coding algorithm will select these in both groups. If we then determine the difference between the two groups, we only find small random differences and the scscore is hence of no use.

A major difference between the IBD and CAD data is that the CAD data is LD-pruned (see Chapter 2.2.1). LD-pruning aims to eliminate redundant SNPs that carry the same information as neighbouring SNPs. One possible explanation why sparse coding selection fails on the CAD data could be that by eliminating these SNPs, we

also miss some disease-specific patterns. Hence, it would be interesting to see whether the performance of sparse coding selection improves if we omit the LD-pruning.

To sum up, the proposed sparse coding algorithm can be used to identify a subset of SNPs that improves classification over the set of all SNPs, as shown on the IBD data set. However, as we have seen on the CAD data, the algorithm may also fail. This underlines the need to explore the performance on further data sets.

Part IV

Interpretive classification

7 Interpretive risk prediction with sparse linear regression

In this thesis, we have so far mainly aimed to classify well. This goal has been approached by classification on large ensembles of SNPs and by selecting more optimal subsets using various feature selection approaches.

Classification on large numbers of SNPs has the advantage that it will not miss disease-specific variants. Moreover, all disease-specific structures and interactions between SNPs are still present in the data. However, we do not expect all SNPs to be relevant for classification. This means that if we incorporate all SNPs in the classifier we also include noise. Indeed, of the hundreds of thousands of SNPs, there may only be a small subset that actually influence the risk.

It is therefore desirable to find such a subset. However, as we have seen, selecting a subset of SNPs that incorporates interactions as well as other disease-specific structures is not straightforward. A selection strategy based on p-values may miss more complex relationships between the SNPs. The svmscore ranking (see Chapter 4) can account for such structures. However, the quality of this ranking is highly influenced by the underlying classifier.

In Chapter 6 we selected SNPs based on structures in the data that differed between the cases and controls; sparse coding was used to identify these structures. However, the approach did not work well for all data sets and, in addition, requires large amounts of memory and computation time. A weakness of the sparse coding algorithm is that it identifies patterns separately for the cases and controls and compares these. If the sparse coding algorithm identifies the same patterns in both groups, the score cannot be used to distinguish between the groups.

In this chapter, our primary aim is not to classify well but to identify SNPs that allow us to better understand the genetics underlying a disease. We will still, however, be using classifiers since the main idea of this chapter is that a small subset of SNPs that enable good classification consists of SNPs that to some extent describe the difference between the the cases and the controls. Moreover, a subset with disease-specific structures may enable a better classification than a subset consisting only of SNPs with individual effects. Such structures may be of great interest because they reveal more complex relationships between SNPs.

A classifier based on large ensembles of SNPs is not suitable for the purpose we have just outlined since may be difficult to interpret. Hence, we will in this chapter aim to identify a small subset of SNPs that enable satisfactory classification.

Linear regression algorithms model the relationship between input variables and an output variable, i.e. in our case between the genotypes and the phenotypes. A linear model has the advantage of being simple and easy to interpret. Linear regression is linear in the sense that the inputs $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$, $\mathbf{x}_i \in \mathbb{R}^D$ and outputs $Y = (y_1, \ldots, y_N)$ are related by a linear mapping of the form

$$y_i = \beta_0 + \sum_{j=1}^{D} \beta_j \cdot x_{ij}. \tag{7.1}$$

Fitting a linear regression model to very large numbers of variables, e.g. hundreds of thousands of SNPs, is not feasible. Not only would this require great computational effort, but prediction accuracy may decrease for large numbers of variables due to overfitting [46]; this danger is particularly great in our case since we do not even expect all variables, i.e. SNPs, to influence the risk. At any rate, our intention is to include only a small number of SNPs in the model so that it can be interpreted easily. Thus, we want to demand that the model should use at most k variables and aim to find the most suitable SNPs under this constraint.

There are different approaches to solving this problem: best subset selection and shrinkage methods. Best subset selection directly selects a subset of size k that minimizes the reconstruction error. Shrinkage methods control the size of the subset indirectly by imposing a penalty that shrinks the coefficients toward zero. The amount of shrinkage is controlled through the shrinkage constant λ; the larger the value, the greater the amount of shrinkage.

The LASSO algorithm (see e.g. [46]) is a popular example of a shrinkage algorithm. In this chapter, we propose to use an alternative approach for linear-regression-based subset selection that we call sparse linear regression (SLR) [3]. SLR is an example of a best subset selection algorithm and is efficient in terms of both computation time and memory use. As for the LASSO algorithm, we restrict the number of variables used for regression. However, whereas LASSO indirectly controls the number of variables via a shrinkage constant λ, the proposed sparse linear regression algorithm directly specifies the permissible number k of non-zero coefficients, i.e. the size of the SNP subset. We did not compare the performance of SLR with the LASSO algorithm because of the resources this would have required; the LASSO algorithm consumes a lot of computation time and memory.

7.1 Sparse linear regression

Sparse linear regression (SLR) allows the number k of variables selected by the algorithm to be specified directly. The entries of the coefficient vector $\boldsymbol{\beta}$ can be seen as the weights of the variables. For consistency with the other chapters we will hence in the following refer to the coefficients as a weight vector, denominated by $\mathbf{w}$.

In general, we seek to find a weight vector $\mathbf{w}$ that approximates the output vector $\mathbf{y}$ as a linear combination of the data X, where the number of non-zero entries of the weight vector is equal to k[1]. SLR estimates the weights by minimizing the following objective function:

$$f(\mathbf{w}) = \sum_{i=1}^{N} \left| y_i - \sum_{j=1}^{D} w_j \cdot x_{ij} \right| \quad \text{with } \|\mathbf{w}\|_0 = k. \tag{7.2}$$

Equation (7.2) is an NP-hard combinatorial problem [40]. Several approximation methods have been proposed to solve it, of which a well-known one is Orthogonal Matching Pursuit (OMP) [44]. In this work, we apply two optimized versions of the OMP method, optimized OMP (OOMP) [44] and the bag of pursuits method (BOP) [45]; Chapter 3 gives a more detailed description of these methods.

7.2 Results and discussion

To evaluate the SLR algorithm, we tested its performance on two distinct data sets: the GWA study on coronary artery disease (CAD) [33] and the GWA study on inflammatory bowel disease (IBD) [36]. The data sets are described in detail in Chapter 2.

We compared the algorithm with state-of-the-art algorithms: the linear SVM (see Chapter 4) and the standard genotype score (GS) (see Chapter 3).

The performance was evaluated for varying subset sizes k using the area under the curve (AUC) metric. For the SLR, we can specify the subset size directly. For the GS and SVM, we have to use a feature selection algorithm to select the SNPs. We have already used many different feature selection methods in this work; here, we will select SNPs according to their individual p-values, since this is one of the most common strategies and since, for small subset sizes, it yields comparable results as shown in the previous chapters (see Chapter 4 and Chapter 6). To explore the effect of random variations in the data, we performed a 5-fold cross-validation.

[1] We will denote the number of non-zero entries in $\mathbf{w}$ by $\|\mathbf{w}\|_0$.

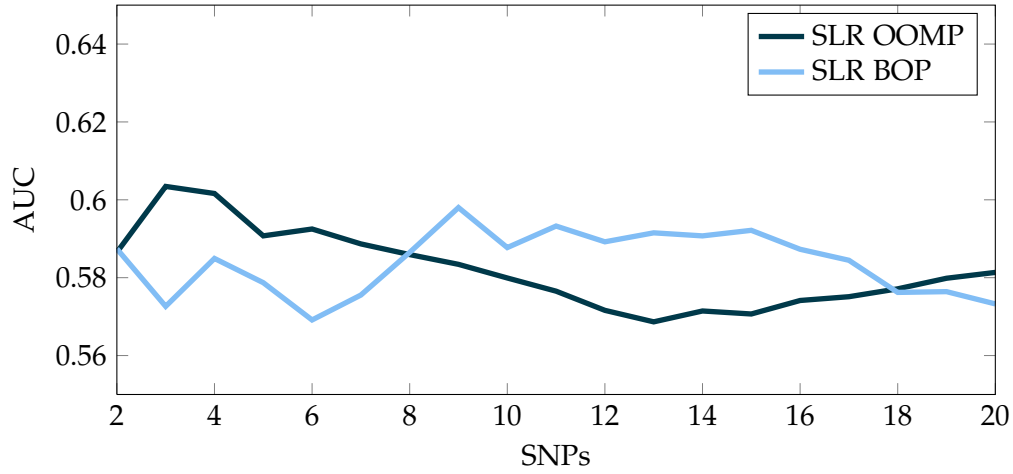


Figure 7.1: Performance of SLR based on OOMP approximation versus BOP approximation. The number of BOP solutions calculated is 100.

7.2.1 Performance on the IBD data set

The IBD data set is described in detail in Chapter 2.2.2. In brief, the data set contains 1719 individuals, of which 789 are cases and 930 are controls. The total number of SNPs is 292,162 after quality filtering.

We varied k, the number of SNPs on which we classify, from 1 to 20. To explore the effect of random variations in the data, we performed a 5-fold cross-validation.

Comparing the optimization approaches

In Section 7.1, we introduced two algorithms for computing the SLR solution: OOMP and BOP. First, we will compare the results obtained using OOMP and BOP.

Figure 7.1 shows the performance of the SLR algorithm using the OOMP and BOP approximation methods. The results are based on the mean performance of a 5-fold cross-validation, and the number of BOP solutions calculated is 100.

For subsets of less than 8 SNPs, OOMP yields better results than BOP. Then, BOP is superior up to subset size of 18 SNPs. After this point, the performance of the two approaches is approximately equal.

The time taken by the BOP method to compute the results is proportional to the number of BOP solutions calculated. For 100 solutions, BOP takes approximately 100 times longer than OOMP to compute the result. This means that, on large GWA data sets, BOP requires noticeably more time. In addition, the performance of BOP is only superior for larger subset sizes. All in all, there is no clear advantage in using the

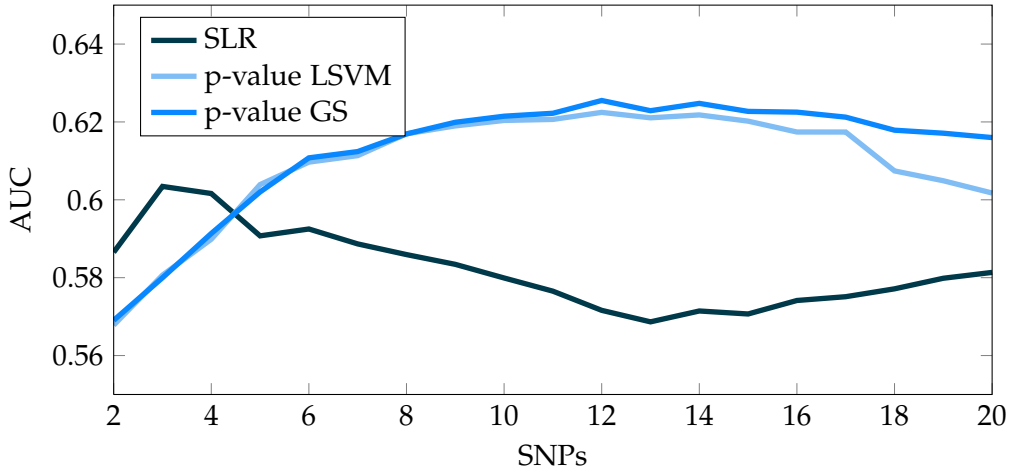


Figure 7.2: Performance of the SLR, the linear SVM and the GS. The SNP subsets for the LSVM and GS are selected according to the p-value rank.

computationally more demanding BOP method. Hence, we will in the following use the OOMP method to compute the SLR results.

Comparison with LSVM and GS

We will now compare the performance of the SLR with the LSVM and the GS (see Figure 7.2). The SLR approach obtains the best results for subsets with up to 4 SNPs. For larger subset sizes, the LSVM and the GS perform better than SLR. Surprisingly, the GS and the LSVM yield similar performance even though the GS weights all SNPs equally and does not incorporate interaction effects. This result suggests that the p-value selected subset may not include interacting SNPs nor other disease-specific structures. In the absence of more complex disease-specific structures, the difference between the LSVM and the GS is only that the LSVM weights the individual SNPs. However, for for the small subset sizes that were tested, this does not seem to be a clear advantage. On the contrary, the GS performs slightly better for subsets with more than 17 SNPs.

Both the LSVM and SLR are based on a linearly weighted sum of the individual SNPs. The main difference between the two algorithms is in how the weights are determined and how the SNPs are selected. Whereas the LSVM determines only the weights and requires a separate feature selection method, the SLR performs both of these tasks at the same time. We would like to evaluate the feature selection performed by SLR independently of the weight determination; hence we evaluated the SLR as a feature selection method for the LSVM and compared this to the SLR by itself. The results are shown in Figure 7.3.

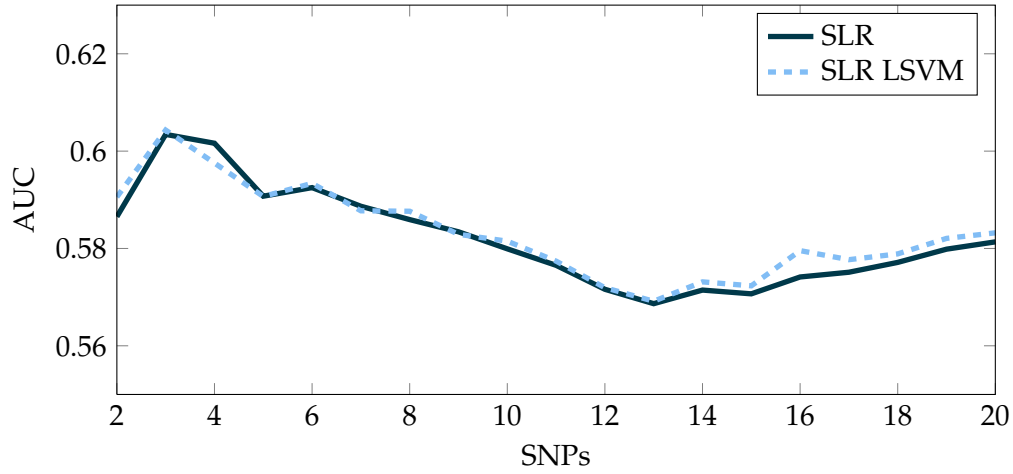


Figure 7.3: IBD data. Performance of SLR compared with the LSVM trained on SLR selected SNPs

We see that the performance of the LSVM on the SLR selected SNPs is essentially identical to that of the SLR approach by itself. This result suggests that the differences we observed above between the LSVM and SLR are due to the feature selection, not the weighting. As the selection strategy therefore seems to be the key difference that influences classification performance, we will in the following compare the performance of different feature selection approaches.

Comparing the selection strategies

We will now compare the performance of the SLR as a feature selection strategy with other feature selection approaches: p-value selection and SNP selection based on the random jungle (RJ) algorithm [70], a fast implementation of random forest. Random forest [46] is a classification method based on a collection of decision trees and can also be used to rank features according to an importance score. We did not compute the RF score ourself but selected the SNPs as ranked in Schwarz et al. [70]. The feature selection performed in this work was based on half of the data set, and no cross-validation was carried out. This means that we will have to use the same RF selected SNPs for all cross validation folds, and in addition, the selection will have been based partly on SNPs from the test data set. The effect of this is that the RF selection has an advantage over the other two feature selection strategies, SLR and p-value ranking, which select SNPs on a training set and test the performance on a completely disjoint test set.

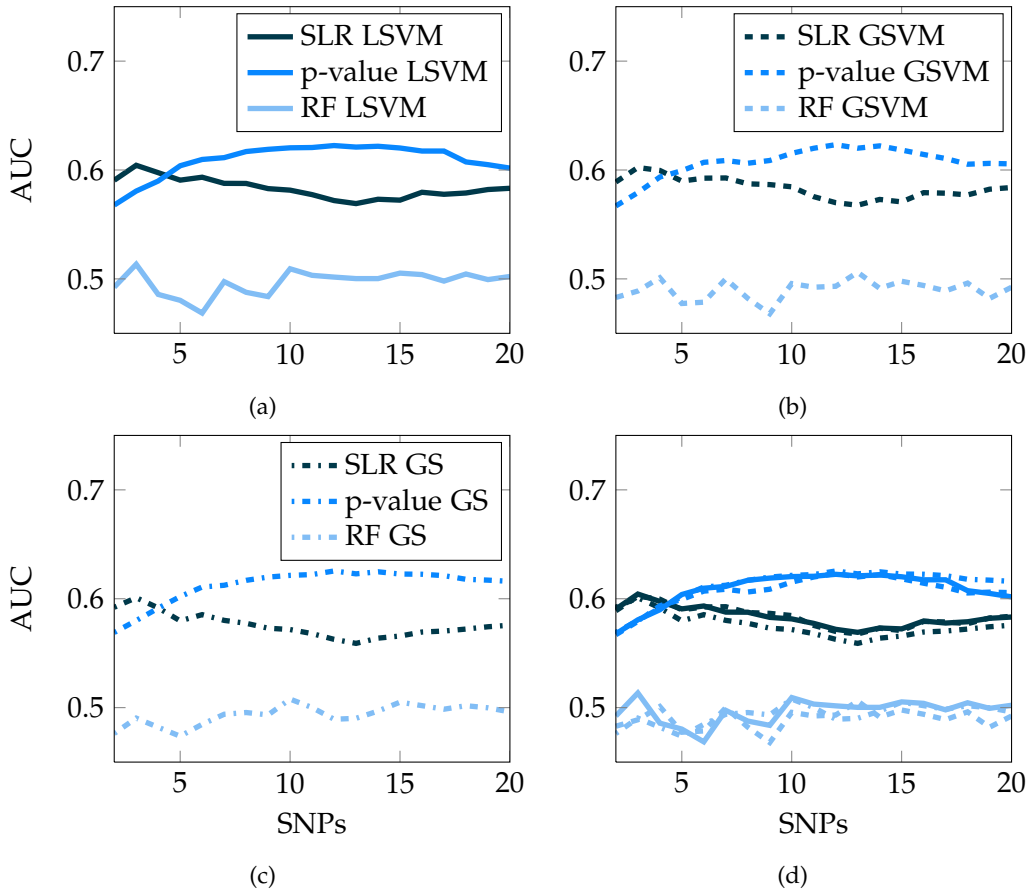


Figure 7.4: IBD data: Classification performance for the three different selection strategies: Sparse Linear Regression (SLR), p-values, Random Forest (RF). a) linear kernel SVM (LSVM), b) Gaussian kernel SVM (GSVM), c) genotype score (GS), d) results of all classifiers.

We will first compare the results of the LSVM for these selection strategies (see Figure 7.4a). Surprisingly, the LSVM based on RF rank yields an AUC of only around 0.5. SLR and p-value selection yield substantially better results. Overall, p-value selection performs best; however, for small numbers of SNPs, SLR selection is superior.

A Gaussian kernel SVM can further improve classification, as discussed in Chapter 4. Hence, we will next compare the performance of the selection strategies with a Gaussian SVM.

The results are shown in Figure 7.4c. Similar to the results with the LSVM, the RF selection yields an AUC of around 0.5. The performance of SLR selection and p-value selection is also not improved by using the GSVM instead of the LSVM. This means that p-value selection again performs better overall than SLR selection and, again, SLR selection is still superior for small subset sizes.

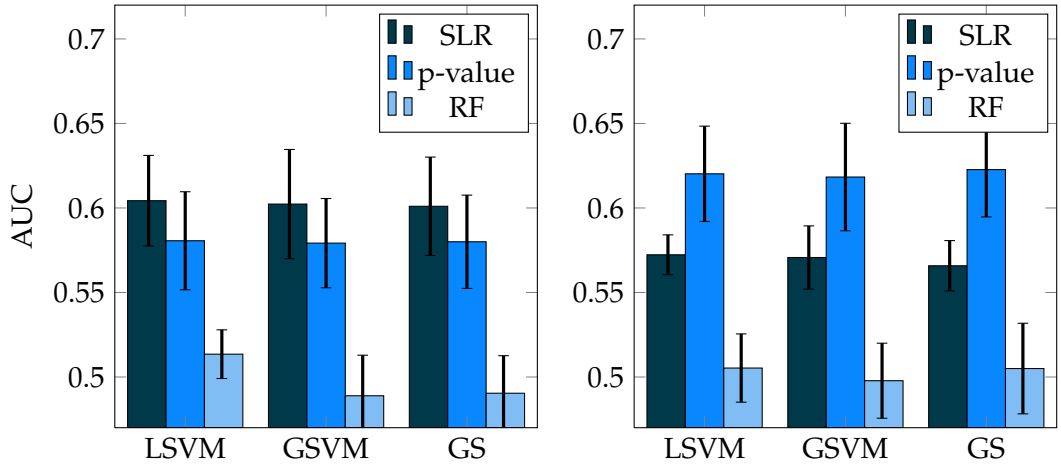
In addition to the SVM classifiers, we also evaluated the selection strategies with the GS (see Figure 7.4c). As for the two SVMs, the RF selection strategy yields the worst performance. As this result is consistent over multiple classification approaches, the RF algorithm does not seem to select subsets that enable a good classification.

The performance of p-value and SLR selection is again clearly better than that of RF selection. As for the LSVM and GSVM, we again see that p-value selection yield the best overall performance and that SLR again is the best-performing method on small subsets. If we compare the performance of the LSVM, GSVM and GS on the SLR selection (see Figure 7.4d), we see that the GS yields the worst results; the LSVM and GSVM perform similarly. On the p-value selection, we see that all three approaches perform equally well.

As discussed above, the GS, in contrast to the LSVM and GSVM, does not apply weights to the SNPs. This means that all SNPs have the same influence on classification. In addition, the GS cannot incorporate interaction effects. The fact that the GS performs worse on SLR selection but not on p-value selection suggests that SLR selection may include more complex relationships between the SNPs than p-value selection.

Overall, the different classification approaches yield the best results when trained on SNPs selected by p-values, but the best performance on small subset sizes is obtained using SLR selection. SLR yields a maximum AUC of 0.60 for a subset of 3 SNPs. The best performance for p-value selection is obtained for 11 SNPs with an AUC of 0.62. However, it is important to note that the SLR approach yields its best performance on a much smaller subset size.

Figure 7.5 compares the performance of the three classifiers and selection strategies for two subset sizes, $k = 3$ SNPs and $k = 15$ SNPs, in more detail. For $k = 3$, SLR feature selection performs substantially better than the p-value selection and RF selection. The standard deviation over all 5 folds is equal for the p-value selection and SLR selection. Hence, the performance of the SLR selection is a robust result and not just a chance



(a) k =3 (b) k =15

Figure 7.5: IBD data: Classification performance of SLR, p-value and RF selection with the linear kernel SVM (LSVM), Gaussian kernel SVM (GSVM) and genotype score (GS). The subset size was a) k =3 SNPs and b) k =15 SNPs. Black error bars show the standard deviation across the cross-validation folds.

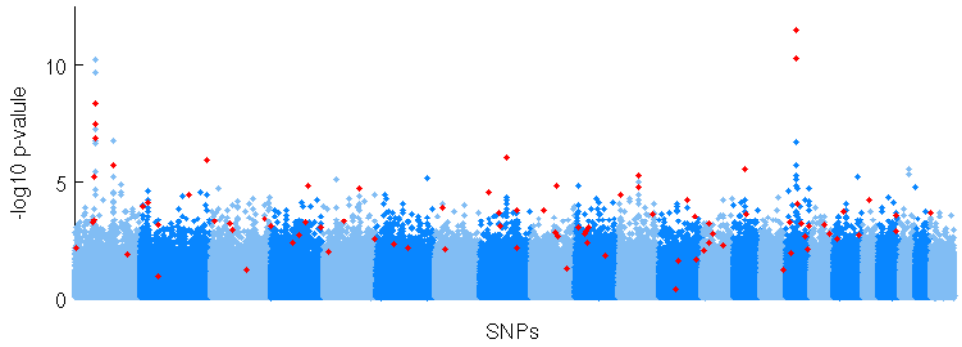
occurrence. For each selection strategy, the three classification approaches yield similar results.

We next evaluate the performance of the feature selection approaches for k =15. Overall, we see that the p-value selection is clearly superior. The standard deviation of the classifiers based on the p-value selection is however wider than with the SLR selection. Hence, the results suggest that the SLR selection yields more robust results than the p-value selection.

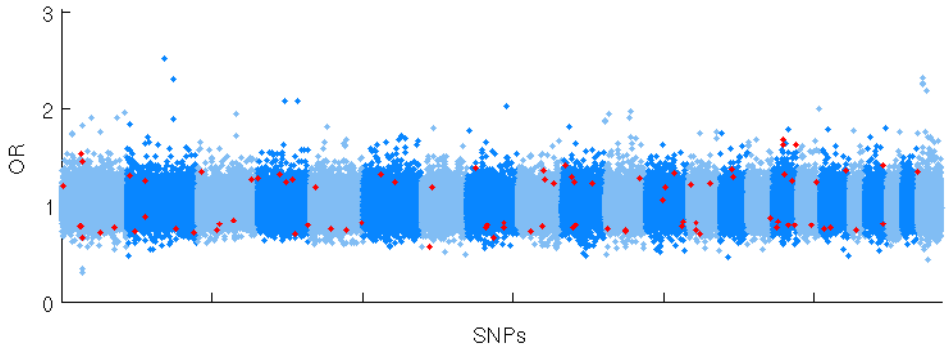
The selected SNPs

As we have seen so far, SLR seems to perform a suitable feature selection and, for small subsets, yields better performance than p-value selection. To better understand the difference between these two selection strategies, we determined the p-values of the SNPs that are selected by the SLR algorithm.

Figure 7.6a plots the p-values for all SNPs, ordered by base pair position. Alternate chromosomes are shaded light and dark. The SNPs that are selected by the SLR algorithm are marked in red; a SNP is included in this set if it is selected in at least one cross-validation fold.



(a)



(b)

Figure 7.6: IBD data: SNPs per chromosome ordered by base pair positions: a) p-value, b) odds ratio. The red points represent the SLR selected SNPs.

We can see that the SLR selected SNPs to some extent overlap with the SNPs ranked highest according to the p-values; however, the majority of the SLR selected SNPs have rather low p-values.

If we compare the SLR selected SNPs with the effect size (odds ratio) of the individual SNPs (see Figure 7.6b), we again cannot find any connection. Hence, we can conclude that the SNPs selected using the SLR algorithm clearly have other properties which qualify them for classification.

In this chapter, our primary aim was not to classify well but to use classification as a tool for identifying SNPs that allow us to better understand the genetics underlying a disease. The SLR algorithm aims to identify a subset of SNPs that enables the best regression performance. Whereas p-value selection only assigns high values to SNPs that have an individual effect, the SLR algorithm selects combinations of SNPs that together yield a low regression error. The selected subset may consist of SNPs that only have an appreciable effect in combination. Hence, the SLR selected SNPs may include interacting SNPs and SNPs forming disease-specific patterns and may therefore be of great interest for identifying epistatic effects. We will therefore now take a closer look at the SLR selected SNPs.

To reduce the effect of random variations in the data, we will only consider SNPs selected in multiple cross-validation folds. For $k = 3$, there are three SNPs that occur in 3 of 5 folds: rs7517847, rs2241880, rs2076756.

All three SNPs lie within genes. The first SNP lies in the IL23R gene on chromosome 1, the second SNP in the ATG16L1 gene on chromosome 2, and the third SNP in the NOD2 gene on chromosome 16. All three genes are connected to the innate immune system [39].

Both the NOD2 and IL23R gene are known candidate genes, and the corresponding SNPs identified above are genome-wide significant also in this data set. The rs2241880 SNP on the ATG16L1 gene is not genome-wide significant in this GWA study. The gene is however also a known susceptibility gene, and the SNP has been associated with the disease in several other studies [38, 71]. An interaction has been reported between the NOD2 and the ATG16L1 genes [72]. The interaction between the SNPs is however not significant.

Summing up, the SLR approach has allowed us to identify a SNP that is not genome-wide significant on this data set but has been associated with the disease on other data sets. This means that the SLR approach can identify SNPs that influence the disease based on disease-specific patterns in the data.

Conclusion IBD data set

So far, we have evaluated the performance of the SLR approach on the IBD data set.

We first compared the performance of two approximation methods for computing the SLR solution: OOMP and BOP. OOMP performs better for small subset sizes, whereas BOP performs better for large subset sizes. We selected the OOMP for all further tests since we could not see a clear advantage in using the computationally much more expensive BOP.

We then compared the performance of the SLR used by itself for classification with the LSVM and GS; the latter two algorithms used p-value feature selection. We saw that for small numbers of SNPs, SLR is superior, and that the other approaches perform better for larger subsets.

We then evaluated the performance of the SLR as a feature selection approach for the LSVM, GSVM and GS and compared it to p-value feature selection and random forest (RF) feature selection. For small subset sizes, all classification approaches yield the best results on SLR feature selection. For larger subset sizes, p-value selection performs best. RF selection does not perform well, yielding an AUC of around 0.5 for all subset sizes.

When we compared the classifiers against each other, we saw that all classifiers perform equally well on p-value selection, but on SLR selection, GS performed worse than the other two. This is probably because the GS does not apply weights to the SNPs and SNP interactions cannot be exploited by this algorithm.

Our primary aim in this chapter is to identify SNPs that give insights into disease mechanisms. We focused on the SNPs that were consistently selected in multiple cross-validation folds and found three SNPs that occurred in 3 out of 5 folds. Two of these SNPs are genome-wide significant in this data set. The third is not, but it is associated with the disease on other IBD data sets. This third SNP lies within a gene that interacts with the gene containing one of the other two SNPs. This is probably the reason why it is detected by the SLR algorithm. The interaction between the SNPs themselves is not significant but may induce a pattern that is recognised by the algorithm. We can conclude that the SLR approach identifies SNPs that may not be detected with standard statistical approaches but may be important to better understand the genetics underlying the disease.

7.2.2 Performance on the CAD data set

Having evaluated the SLR algorithm on the IBD data set, we will next test it on the coronary artery disease (CAD) data set [33]. Again, we will be using the OOMP method to compute the SLR throughout.

We first test SLR selection and p-value selection with the LSVM. The results are shown in Figure 7.7a. Overall, the performance increases with increasing subset size. In contrast to the results on the IBD data, we see that p-value selection performs better than SLR selection. The difference is larger for small subsets and decreases as the

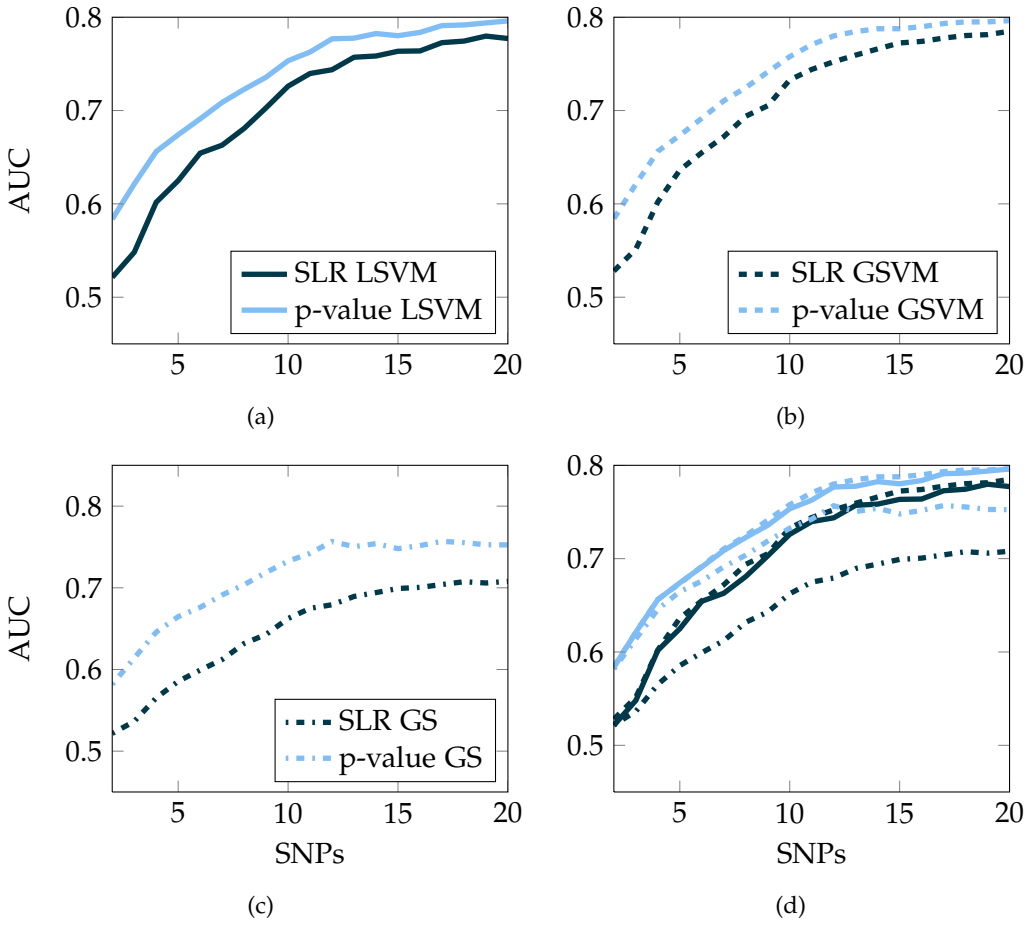


Figure 7.7: CAD data: Classification performance for the two different selection strategies: Sparse Linear Regression (SLR) and p-values. a) linear kernel SVM (LSVM), b) Gaussian kernel SVM (GSVM), c) genotype score (GS), d) results of all classifiers.

subset size increases. For 20 SNPs, p-value selection is only slightly better than SLR selection.

We will next evaluate the performance of the GSVM for both selection strategies (see Figure 7.7b). For both selection strategies, the GSVM performs slightly better than the LSVM. If we compare SLR selection to p-value selection, we see that the difference is slightly smaller than for the LSVM. However, overall, p-value selection still performs better than SLR selection.

Finally, we will evaluate the performance of the two selection strategies with the GS (see Figure 7.7c). On the p-value selection, the GS performs only marginally worse than the LSVM and GSVM. In contrast, on the SLR SNPs, the GS is clearly outperformed by the LSVM and GSVM.

Figure 7.7d compares all classifiers. For p-value selection, the LSVM and GSVM perform slightly better than the GS. Compare this to the IBD data set, where all three classifiers performed equally well on p-value selection.

For SLR selection, GS performs noticeably worse than the LSVM and GSVM. A similar effect was observed on the IBD data set, but the difference there was only small; here, the difference is much larger.

These results suggest that the SLR SNPs probably hold more information about complex relationships than the p-value SNPs. The GS cannot exploit such relationships and thus performs worse than the LSVM and GSVM when they are present.

Overall, on the CAD data set, the SNPs selected by the SLR strategy are not better suited for classification than the p-value SNPs, but they may include SNPs that form disease-specific structures and may hence be of great value to better understand the mechanisms underlying CAD.

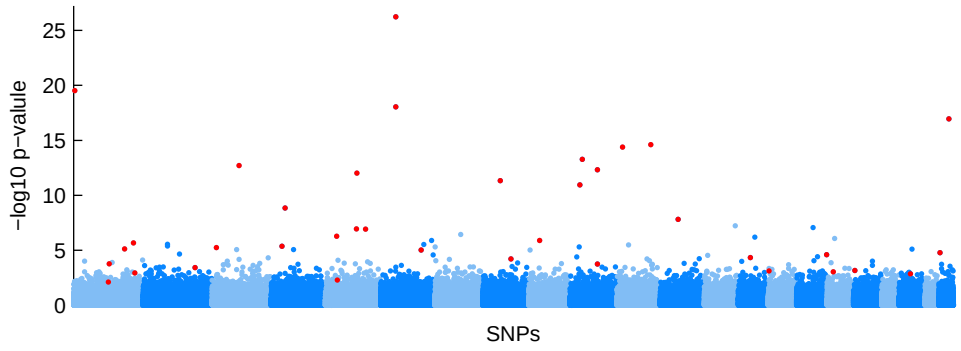
The selected SNPs

To further interpret the results of the SLR approach, we again take a closer look at the SNPs that are selected by the algorithm.

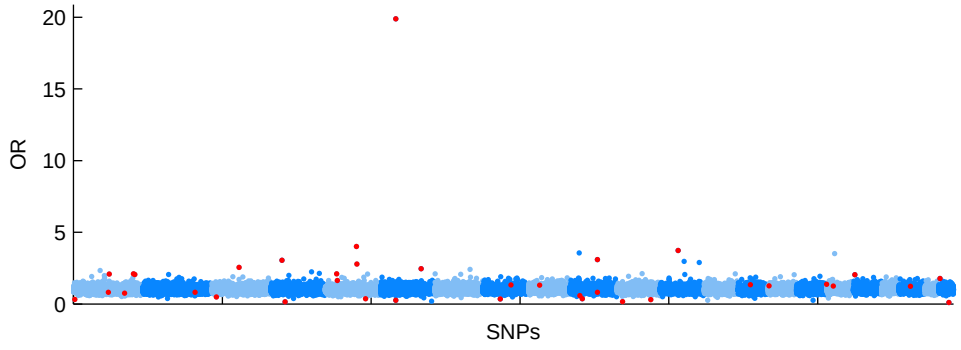
Figure 7.8a plots the p-values for all SNPs, ordered by their base pair positions. Alternate chromosomes are shaded light and dark. The SNPs that are selected by the SLR algorithm are marked in red; a SNP is included in this set if it is selected in at least one cross-validation fold.

We can see that the SLR selected SNPs largely overlap with the SNPs ranked highest according to the p-values. Of the 15 best ranked p-value SNPs, 14 are also selected by the SLR algorithm in at least one of the folds. However, some SNPs selected by SLR differ from the best ranked p-value SNPs. Furthermore, the order in which the SNPs are selected differs for the two selection approaches.

Figure 7.8b plots the ORs for all SNPs. As for the p-values, we see that most, but not all, SLR selected SNPs have large OR.



(a)



(b)

Figure 7.8: CAD data: SNPs per chromosome ordered by base pair positions: a) p-values, b) odds ratio. The red points represent the SLR selected SNPs.

These results are in contrast to those observed on the IBD data, where we only saw a small overlap between the SLR selected SNPs and p-value selected SNPs. The reason we have a large overlap on the CAD data is probably that the p-value SNPs in this data set have strong effects and are good classifiers on an individual level. Because these strong individual SNPs are present in the data set, the SLR does not appear to focus as much on patterns consisting of multiple SNPs.

If we take a closer look at the SNPs selected by the SLR in each cross-validation fold, we see that 12 SNPs are consistently selected in all five folds. 11 of these SNPs correspond to the 11 best ranked p-value SNPs. The last SNP has rank 17 according to the p-values. Hence, the SNPs selected consistently correspond to those with high p-value rank. The pairwise interaction between these SNPs, as computed using a logistic regression, was not found to be significant.

Conclusion CAD data set

On the CAD data, the p-value selection yielded better results on all subset sizes than the selection by the SLR algorithm. In contrast, on the IBD data set, SLR selection is favourable for small subsets.

For each of these two selection strategies, we compared the performance of the different classifiers. On the p-value selection, the GS performs only slightly worse than the LSVM and GSVM. On the SLR selection, the GS performs noticeably worse than the other two classifiers. This suggests that the SLR selection contains more interactions, which the GS cannot exploit.

If compare the SNPs selected by the SLR algorithm in the five cross-validation folds, we find that 12 SNPs are consistently selected over all folds. Of these SNPs, all but one are genome-wide significant, and all are among the 20 SNPs ranked highest by p-values; however, they are selected in a different order. In the individual folds, however, not all SLR selected SNPs also have high p-values. If this were the case, we would not see an appreciable difference in performance between the two methods.

7.3 Conclusion SLR

In GWA studies, the number of SNPs that are genotyped are generally expected by far to exceed the number of SNPs that have an influence on the disease. In this chapter, we have aimed to identify a small subset of SNPs that enable good classification. The motivation for this was that only a classifier that uses a small number of SNPs can easily be interpreted.

Identifying such a small subset is, however, a major challenge. SNPs selected according to their p-value rank are only selected based on individual effects. However,

a disease effect may come about only through the combination of multiple SNPs, and these SNPs will not be found using p-value selection.

The premise underlying this chapter is that SNPs that form disease-specific patterns may improve classification over SNPs with only individual effects. Hence, if we search for the optimal subset for classification with a given small number of SNPs, we may identify disease-specific SNPs that cannot be identified using individual effects. These SNPs may broaden our knowledge about the genetic mechanisms underlying a disease.

We therefore evaluated the performance of sparse linear regression (SLR) as a classifier and feature selection approach; SLR is a linear regression algorithm that allows the number of variables used to be controlled precisely. In addition, it is fast and can be applied to very large data sets. An advantage of SLR is that it is free of any a-priori assumptions on which SNPs are relevant. In other words, the algorithm searches for the best subset among all SNPs and not only on a predefined subset. Since disease-specific structures and interactions between SNPs can be exploited by SLR to minimize the approximation error, SLR selection may favor such SNPs.

We evaluated the SLR approach on the inflammatory bowel disease (IBD) data set and the coronary artery disease (CAD) data set. The overall performance on the CAD data (AUC 0.78) is better than the performance on the IBD data (AUC 0.60).

On the IBD data, SLR selection yields better performance on small subsets than p-value selection. For larger subsets, the performance of p-value selection is superior. On the CAD data, p-value selection is superior on all subset sizes.

We compared the performance of the GS and the LSVM and GSVM on SLR selected SNPs. The GS classifies worse than the SVMs on both data sets, but on the CAD data, this difference is by far more distinct. There are two main reasons why the GS performs worse than the LSVM and GSVM. First, it cannot account for disease-specific structures; and second, it does not apply different weights to the individual SNPs. This means that all SNPs equally influence the classification result.

On the IBD data set, an evaluation of the SNPs selected by the SLR approach yields an interesting result. Three SNPs were consistently selected in three of five cross-validation folds. Two of these SNPs are genome-wide significant in this data set, however, the third is not. This third SNP is however associated with the disease on other IBD data sets. This SNP lies within a gene which interacts with the gene containing one of the other consistently selected SNPs. This interaction probably induces a pattern that is identified by the SLR algorithm. In other words, the SLR approach can identify SNPs that cannot be found by their individual effects; it may therefore be useful for detecting more complex relationships between SNPs and thereby identifying new SNPs that allow us to better understand the genetics underlying a disease.

To sum up, instead of cherry-picking a few SNPs according to the p-values, selecting SNPs based on structures in the data can improve classification on small sub-

sets. Moreover, the SLR SNPs may be of great value for identifying new susceptibility SNPs, interactions, and other disease-specific structures. Investigating these SNPs may broaden our understanding of the complex mechanisms underlying a disease.

Conclusion

During the past few years, genome-wide association (GWA) studies have enabled an efficient search for genetic risk factors all across the genome. Several new genetic factors have been identified, and we are now slowly beginning to unravel the genetic causes of complex diseases.

We can broadly differentiate between two goals for GWA studies: explaining the cause of a disease and predicting risk. In the first years of GWA studies, the focus was on the first of these goals: explaining causes. The aim was to identify genes and pathways that can be used as clinical targets for prevention, drug design, and other therapeutic measures. This goal was approached by identifying individual SNPs that are associated with the disease. The reason for the focus on individual SNPs is the computational aspect, single SNP analysis is easier to carry out.

Once the first variants associated with the disease had been identified, the first attempts to use the variants to predict disease risk soon followed. However, the conclusion from these first attempts was that the identified variants explain too little of the genetic risks for a clinically useful risk prediction to be possible.

But why should we restrict ourselves only to variants with individual effects? There is no obvious reason why a risk prediction model should only use such variants, since some variants may only have an effect at all in combination with others. Hence, a risk prediction model that also accounts for these interacting variants may be substantially more accurate than a model based only on the individual effects.

If we argue that risk prediction may improve if we incorporate epistatic effects, the logical consequence is that they are also of importance for explaining the cause of the disease. Indeed, the variants identified so far explain only a small fraction of the expected heritability of a disease; this implies that the genetic causes are far from being understood. Epistatic effects may be responsible for some of the so-called missing heritability. Furthermore, because epistatic effects by definition imply a connection between multiple loci, they immediately provide hypotheses about interactions between different parts of the genome.

In spite of this, epistatic effects were overlooked and disregarded for a long time, probably because their identification is not straightforward. This work has focused on finding methods for overcoming these difficulties. The tools have been taken from the field of machine learning, and they were used to approach both of the main goals

of GWA studies: enabling a good risk prediction on GWA data and identifying new variants that explain the cause of the disease.

We explored to what degree risk prediction, the first goal, can be improved by classifying on large numbers of variables instead of only a few, and which types of classifiers are most appropriate for this task. To generate the large ensembles of SNPs used by the classifiers, we explored the use of various feature selection methods.

The second goal, explaining disease causes, was approached by searching for small subsets of SNPs on which good classification was still possible. The underlying idea was that such a subset may include SNPs that only have an effect in combination with the other SNPs and that, in this way, we can find epistatic effects and genetic interactions that cannot be identified by their individual effects.

To test the different approaches of this work, we used two established GWA data sets: a coronary artery disease (CAD) data set and an inflammatory bowel disease data set (IBD).

The support vector machine (SVM) is one of the most popular classification methods; it is a powerful algorithm which can exploit individual effects as well as interactions. In this work, we used the SVM for both classification and feature selection, and we find it in every chapter. In the first part of this thesis, we tested the SVM in its primary role as a classifier for predicting risk on GWA data (Chapter 4). We applied the SVM to subsets of increasing size selected by the p-values of the individual SNPs.

The SVM yields convincing classification performance for both data sets and predicts disease risk more accurately than would be possible with individual SNPs. A difference between the performance on the IBD and CAD data is the size of the SNP subset that yields the best classification. Whereas on the IBD data the best classification is reached by including all SNPs, on the CAD data, the best performance is obtained on a smaller subset.

In addition to classifying with the SVM, we also tested the performance of a boosting algorithm (Chapter 5). To do so, we modified the well-known Adaboost algorithm for use on GWA data. The resulting SNPboost algorithm, as we call it, performs a combination of feature selection and classification. It selects SNPs successively and aims to find a set of SNPs that together cover as much of the variability in the data as possible.

We compared the performance of SNPboost with the SVM on the same subsets of SNPs. As feature selection is part of the SNPboost algorithm, those SNP subsets that had been chosen by SNPboost were then also used for the SVM. Overall, the SVM performed better than SNPboost on both data sets, suggesting that the SVM is the better classifier. However, as a feature selector, SNPboost appeared to work well.

In addition to comparing different classification algorithms, we also compared different feature selection methods. We have already discussed feature selection by p-values and by the SNPboost algorithm. Since both of these approaches select SNPs

individually, they may not identify patterns that consist of SNPs without individual effects. We therefore evaluated other methods that select subsets based on patterns in the data, which we call the svmscore, scscore, and pcascore. The svmscore is based on the SVM and ranks SNPs according to the entries of the weight vector for an SVM trained on all SNPs (Chapter 4). The scscore is, in contrast to the svmscore, primarily focused on patterns in the data. It is based on sparse coding and ranks the SNPs based on the directions of greatest variance in the data (Chapter 6). The pcascore is based on principal component analysis, a classic statistical method that embodies a similar idea as sparse coding to identify patterns in the data. We will in the following summarize the results of all feature selection approaches as a preprocessing step for the SVM.

On the IBD data set, we observe that all feature selection approaches yield ever-increasing performance as we increase the subset size. Overall, the svmscore performs best for most subset sizes; for very large numbers of SNPs, the scscore performs slightly better than the svmscore. On the CAD data set, we obtain very different results. All of the feature selection approaches achieve their respective maximum performance for small numbers of SNPs (around 40) and fall off thereafter. The best-performing method on this data set is p-value selection; the svmscore, which was the best performer on the IBD data set, only delivers mediocre results on CAD. Surprisingly, scscore and pcascore failed to deliver any useable results at all; their performance was not better than chance.

We also evaluated the influence of rare variants on classification performance. To do so, we varied the MAF threshold and tested the effect of this on the performance of both the SVM and SNPboost. On the CAD data, a more stringent MAF threshold of 5% instead of 1% reduces the performance of both classification approaches. Conversely, on the IBD data, the performance does not change noticeable if we vary the MAF threshold. These results imply that rare variants may be of great importance to enable a good classification, but not on every data set.

So far we have summarized how we approached the first goal, to predict risk well. The final part of this thesis focused on the second goal, explaining disease causes. As we have already discussed, we approached this goal by searching for small subsets of SNPs on which good classification was still possible.

To do this, we used a sparse linear regression (SLR) approach, which performs a combination of feature selection and classification. The advantage of SLR is that it can be applied to large data sets without any preselection step, unlike other algorithms, which often require a preselection of variables to ease the computational burden.

To identify new disease-relevant SNPs, we determined the SNPs that were consistently selected by SLR over several cross-validation folds. To validate that the SLR selects SNPs that are important for explaining the disease we looked at the classification performance; the SLR performed well both as a feature selection method for the SVM and as a classification method by itself.

On the IBD data set we identified a truly disease-specific SNP (rs2241880) that could not be linked to the disease by its individual effect. The gene (ATG16L1) in which it lies interacts with a gene (NOD2) containing one of the other consistently selected SNPs. This interaction probably induces a pattern that is identified by the SLR algorithm. Because the SLR approach can identify SNPs based on patterns, it may be useful for detecting more complex relationships between SNPs that allow us to better understand the genetics underlying a disease.

As we have remarked above, almost all of the methods used in this thesis exhibited very different behaviour on the two data sets. On the IBD data, the best choice is to use large ensembles of SNPs and sophisticated feature selection approaches. On the CAD data, it is best to use only a few SNPs with the simple p-value feature selection method.

Having obtained such different results on these two data sets, what can we expect on other data sets? Is the difference in behaviour due to quality issues, does it depend on the particular disease studied or is it perhaps influenced by the population from which the data is drawn?

To determine the crucial factors that cause these differences, we will need to test the algorithms on more and more diverse data sets. Based on the results that were obtained in this work, we do not expect that there will be one approach that works well on all data sets. Each phenotype probably has its own characteristics. Performance may vary with the genotype frequencies, the effect size, disease incidence, and other factors. An approach that performs well on one phenotype may not be suitable at all on the next.

Another question is whether we have already reached the limit of what is possible on GWA data sets. And is the quality of today's GWA data even good enough to identify complex structures? Can we expect multifactorial methods to detect much more than single-SNP analyses? And can we even expect to fully explain the cause of a disease using GWA data?

The premise underlying GWA studies has been that common variants play the major role in the development of common diseases. However, this view has recently changed, and rare variants are now also thought to have a large influence [9]. Some even argue that the rare variants could be the primary cause and that common diseases may even be more similar to Mendelian diseases than was assumed so far [73]. GWA studies do not have enough power to detect rare variants and, at any rate, GWA studies samples the genome too coarsely to be able to detect rare variants reliably. In addition, rare variants may be specific to families or even individuals [74]. Hence, we cannot expect to find these variants in case control data.

The area of genome studies is now moving towards sequencing the complete genome. Up to recently, this was not possible, but new machines can now sequence the complete genome in a reasonable length of time and at an affordable price [73]. By sequencing

the whole genome, we may be able to identify the real disease-causing variants, which many suspect have not been covered by GWA studies so far [73]. The initial goal of this so-called next generation sequencing is to identify rare variants with a large effect size. One strategy for identifying such rare variants is to select families that have multiple affected individuals. I look forward to working in this area in the coming years – the genome is certainly one of the greatest puzzles mankind has ever tried to solve.

Bibliography

[1] Ingrid Brænne, Jeanette Erdmann, and Amir Madany Mamlouk. SNPboost: Interaction Analysis and Risk Prediction on GWA Data. In *Artificial Neural Networks - ICANN 2011, 21th International Conference, Espoo, Finland, June 14-17th, 2011, Proceedings*, volume 6792 of *Lecture Notes in Computer Science*, pages 111–118. Springer, 2011.

[2] Ingrid Brænne, Kai Labusch, and Amir Madany Mamlouk. Sparse Coding for Feature Selection on Genome-wide Association Data. In *Artificial Neural Networks - ICANN 2010, 20th International Conference, Thessaloniki,Greece, September 15-18, 2010, Proceedings*, volume 6352 of *Lecture Notes in Computer Science*, pages 337–346. Springer, 2010.

[3] Ingrid Brænne, Kai Labusch, Thomas Martinetz, and Amir Madany Mamlouk. Interpretive Risk Assessment on GWA Data with Sparse Linear Regression. *Machine Learning Reports*, pages 61–68, 2010.

[4] Alun Evans, G Caroline M Van Baal, Peter McCarron, Marlies DeLange, Thorkild I Soerensen, Eco J C De Geus, Kirsten Kyvik, Nancy L Pedersen, Tim D Spector, Toby Andrew, Christopher Patterson, John B Whitfield, Gu Zhu, Nicholas G Martin, Jaakko Kaprio, and Dorret I Boomsma. The genetics of coronary heart disease: the contribution of twin studies. *Twin Research: The Official Journal of the International Society for Twin Studies*, 6(5):432–441, October 2003.

[5] Dorret Boomsma, Andreas Busjahn, and Leena Peltonen. Classical twin studies and beyond. *Nature Reviews Genetics*, 3(11):872–882, November 2002.

[6] Joel N. Hirschhorn and Zofia K. Z. Gajdos. Genome-Wide association studies: Results from the first few years and potential implications for clinical medicine. *Annual Review of Medicine*, 62(1):11–24, 2011.

[7] D E Reich and E S Lander. On the allelic spectrum of human disease. *Trends in Genetics: TIG*, 17(9):502–510, September 2001.

[8] Eric S. Lander. Initial impact of the sequencing of the human genome. *Nature*, 470(7333):187–197, February 2011.

[9] Andrew Singleton and John Hardy. A generalizable hypothesis for the genetic architecture of disease: pleomorphic risk loci. *Human Molecular Genetics*, 2011.

[10] Joel N. Hirschhorn and Mark J. Daly. Genome-wide association studies for common diseases and complex traits. *Nature Reviews Genetics*, 6(2):95–108, February 2005.

[11] Janine Altmüller, Lyle J. Palmer, Guido Fischer, Hagen Scherb, and Matthias Wjst. Genomewide scans of complex human diseases: True linkage is hard to find. *The American Journal of Human Genetics*, 69(5):936–950, November 2001.

[12] John S. Witte. Genome-Wide association studies and beyond. *Annual Review of Public Health*, 31(1):9–20, 2010.

[13] Jonathan Marchini and Bryan Howie. Genotype imputation for genome-wide association studies. *Nature Reviews Genetics*, 11(7):499–511, June 2010.

[14] Nilesh J Samani, Jeanette Erdmann, Alistair S Hall, Christian Hengstenberg, Massimo Mangino, Bjoern Mayer, Richard J Dixon, Thomas Meitinger, Peter Braund, H-Erich Wichmann, Jennifer H Barrett, Inke R König, Suzanne E Stevens, Silke Szymczak, David-Alexandre Tregouet, Mark M Iles, Friedrich Pahlke, Helen Pollard, Wolfgang Lieb, Francois Cambien, Marcus Fischer, Willem Ouwehand, Stefan Blankenberg, Anthony J Balmforth, Andrea Baessler, Stephen G Ball, Tim M Strom, Ingrid Brænne, Christian Gieger, Panos Deloukas, Martin D Tobin, Andreas Ziegler, John R Thompson, and Heribert Schunkert. Genomewide association analysis of coronary artery disease. *The New England Journal of Medicine*, 357(5):443–453, August 2007.

[15] Ruth McPherson, Alexander Pertsemlidis, Nihan Kavaslar, Alexandre Stewart, Robert Roberts, David R. Cox, David A. Hinds, Len A. Pennacchio, Anne Tybjaerg-Hansen, Aaron R. Folsom, Eric Boerwinkle, Helen H. Hobbs, and Jonathan C. Cohen. A common allele on chromosome 9 associated with coronary heart disease. *Science*, 316(5830):1488 –1491, June 2007.

[16] Anna Helgadottir, Gudmar Thorleifsson, Andrei Manolescu, Solveig Gretarsdottir, Thorarinn Blondal, Aslaug Jonasdottir, Adalbjorg Jonasdottir, Asgeir Sigurdsson, Adam Baker, Arnar Palsson, Gisli Masson, Daniel F. Gudbjartsson, Kristinn P. Magnusson, Karl Andersen, Allan I. Levey, Valgerdur M. Backman, Sigurborg Matthiasdottir, Thorbjorg Jonsdottir, Stefan Palsson, Helga Einarsdottir, Steinunn Gunnarsdottir, Arnaldur Gylfason, Viola Vaccarino, W. Craig Hooper, Muredach P. Reilly, Christopher B. Granger, Harland Austin, Daniel J. Rader, Svati H. Shah, Arshed A. Quyyumi, Jeffrey R. Gulcher, Gudmundur Thorgeirsson, Unnur Thorsteinsdottir, Augustine Kong, and Kari Stefansson. A common

variant on chromosome 9p21 affects the risk of myocardial infarction. *Science*, 316(5830):1491 –1493, June 2007.

[17] Yan Liu, Hanna K. Sanoff, Hyunsoon Cho, Christin E. Burd, Chad Torrice, Karen L. Mohlke, Joseph G. Ibrahim, Nancy E. Thomas, and Norman E. Sharpless. INK4/ARF transcript expression is associated with chromosome 9p21 variants linked to atherosclerosis. *PLoS ONE*, 4(4):e5027, April 2009.

[18] Olga Jarinova, Alexandre F.R. Stewart, Robert Roberts, George Wells, Paulina Lau, Thet Naing, Christine Buerki, Bradley W. McLean, Richard C. Cook, Joel S. Parker, and Ruth McPherson. Functional analysis of the chromosome 9p21.3 coronary artery disease risk locus. *Arterioscler Thromb Vasc Biol*, 29(10):1671–1677, 2009.

[19] Heribert Schunkert, Anika Götz, Peter Braund, Ralph McGinnis, David-Alexandre Tregouet, Massimo Mangino, Patrick Linsel-Nitschke, Francois Cambien, Christian Hengstenberg, Klaus Stark, Stefan Blankenberg, Laurence Tiret, Pierre Ducimetiere, Andrew Keniry, Mohammed J R Ghori, Stefan Schreiber, Nour Eddine El Mokhtari, Alistair S Hall, Richard J Dixon, Alison H Goodall, Henrike Liptau, Helen Pollard, Daniel F Schwarz, Ludwig A Hothorn, H-Erich Wichmann, Inke R König, Marcus Fischer, Christa Meisinger, Willem Ouwehand, Panos Deloukas, John R Thompson, Jeanette Erdmann, Andreas Ziegler, and Nilesh J Samani. Repeated replication and a prospective meta-analysis of the association between chromosome 9p21.3 and coronary artery disease. *Circulation*, 117(13):1675–1684, April 2008.

[20] Yoko Yoshikawa, Takaya Satoh, Takashi Tamura, Ping Wei, Shymaa E. Bilasy, Hironori Edamatsu, Atsu Aiba, Koko Katagiri, Tatsuo Kinashi, Kazuki Nakao, and Tohru Kataoka. The M-Ras-RA-GEF-2-Rap1 pathway mediates tumor necrosis factor-α –dependent regulation of integrin activation in splenocytes. *Molecular Biology of the Cell*, 18(8):2949–2959, August 2007.

[21] Heribert Schunkert, Jeanette Erdmann, and Nilesh J Samani. Genetics of myocardial infarction: a progress report. *European Heart Journal*, 31(8):918–925, April 2010.

[22] Nilesh J Samani, Peter S Braund, Jeanette Erdmann, Anika Götz, Maciej Tomaszewski, Patrick Linsel-Nitschke, Cother Hajat, Massimo Mangino, Christian Hengstenberg, Klaus Stark, Andreas Ziegler, Mark Caulfield, Paul R Burton, Heribert Schunkert, and Martin D Tobin. The novel genetic variant predisposing to coronary artery disease in the region of the PSRC1 and CELSR2 genes on chromosome 1 associates with serum cholesterol. *Journal of Molecular Medicine (Berlin, Germany)*, 86(11):1233–1241, November 2008.

[23] Sekar Kathiresan, Olle Melander, Dragi Anevski, Candace Guiducci, Noël P Burtt, Charlotta Roos, Joel N Hirschhorn, Göran Berglund, Bo Hedblad, Leif Groop, David M Altshuler, Christopher Newton-Cheh, and Marju Orho-Melander. Polymorphisms associated with cholesterol and risk of cardiovascular events. *The New England Journal of Medicine*, 358(12):1240–1249, March 2008.

[24] JJ Nora, RH Lortscher, RD Spangler, AH Nora, and WJ Kimberling. Genetic–epidemiologic study of early-onset ischemic heart disease. *Circulation*, 61(3):503–508, March 1980.

[25] Marjorie E. Marenberg, Neil Risch, Lisa F. Berkman, Birgitta Floderus, and Ulf de Faire. Genetic susceptibility to death from coronary heart disease in a study of twins. *New England Journal of Medicine*, 330(15):1041–1046, 1994.

[26] Heribert Schunkert, Inke R König, Sekar Kathiresan, Muredach P Reilly, Themistocles L Assimes, Hilma Holm, Michael Preuss, Alexandre F R Stewart, Maja Barbalic, Christian Gieger, Devin Absher, Zouhair Aherrahrou, Hooman Allayee, David Altshuler, Sonia S Anand, Karl Andersen, Jeffrey L Anderson, Diego Ardissino, Stephen G Ball, Anthony J Balmforth, Timothy A Barnes, Diane M Becker, Lewis C Becker, Klaus Berger, Joshua C Bis, S Matthijs Boekholdt, Eric Boerwinkle, Peter S Braund, Morris J Brown, Mary Susan Burnett, Ian Buysschaert, John F Carlquist, Li Chen, Sven Cichon, Veryan Codd, Robert W Davies, George Dedoussis, Abbas Dehghan, Serkalem Demissie, Joseph M Devaney, Patrick Diemert, Ron Do, Angela Doering, Sandra Eifert, Nour Eddine El Mokhtari, Stephen G Ellis, Roberto Elosua, James C Engert, Stephen E Epstein, Ulf de Faire, Marcus Fischer, Aaron R Folsom, Jennifer Freyer, Bruna Gigante, Domenico Girelli, Solveig Gretarsdottir, Vilmundur Gudnason, Jeffrey R Gulcher, Eran Halperin, Naomi Hammond, Stanley L Hazen, Albert Hofman, Benjamin D Horne, Thomas Illig, Carlos Iribarren, Gregory T Jones, J Wouter Jukema, Michael A Kaiser, Lee M Kaplan, John J P Kastelein, Kay-Tee Khaw, Joshua W Knowles, Genovefa Kolovou, Augustine Kong, Reijo Laaksonen, Diether Lambrechts, Karin Leander, Guillaume Lettre, Mingyao Li, Wolfgang Lieb, Christina Loley, Andrew J Lotery, Pier M Mannucci, Seraya Maouche, Nicola Martinelli, Pascal P McKeown, Christa Meisinger, Thomas Meitinger, Olle Melander, Pier Angelica Merlini, Vincent Mooser, Thomas Morgan, Thomas W Mühleisen, Joseph B Muhlestein, Thomas Münzel, Kiran Musunuru, Janja Nahrstaedt, Christopher P Nelson, Markus M Nöthen, Oliviero Olivieri, Riyaz S Patel, Chris C Patterson, Annette Peters, Flora Peyvandi, Liming Qu, Arshed A Quyyumi, Daniel J Rader, Loukianos S Rallidis, Catherine Rice, Frits R Rosendaal, Diana Rubin, Veikko Salomaa, M Lourdes Sampietro, Manj S Sandhu, Eric Schadt, Arne Schäfer, Arne Schillert, Stefan Schreiber, Jürgen Schrezenmeir, Stephen M

Schwartz, David S Siscovick, Mohan Sivananthan, Suthesh Sivapalaratnam, Albert Smith, Tamara B Smith, Jaapjan D Snoep, Nicole Soranzo, John A Spertus, Klaus Stark, Kathy Stirrups, Monika Stoll, W H Wilson Tang, Stephanie Tennstedt, Gudmundur Thorgeirsson, Gudmar Thorleifsson, Maciej Tomaszewski, Andre G Uitterlinden, Andre M van Rij, Benjamin F Voight, Nick J Wareham, George A Wells, H-Erich Wichmann, Philipp S Wild, Christina Willenborg, Jaqueline C M Witteman, Benjamin J Wright, Shu Ye, Tanja Zeller, Andreas Ziegler, Francois Cambien, Alison H Goodall, L Adrienne Cupples, Thomas Quertermous, Winfried März, Christian Hengstenberg, Stefan Blankenberg, Willem H Ouwehand, Alistair S Hall, Panos Deloukas, John R Thompson, Kari Stefansson, Robert Roberts, Unnur Thorsteinsdottir, Christopher J O'Donnell, Ruth McPherson, Jeanette Erdmann, and Nilesh J Samani. Large-scale association analysis identifies 13 new susceptibility loci for coronary artery disease. *Nature Genetics*, 43(4):333–338, 2011.

[27] Teri A. Manolio, Francis S. Collins, Nancy J. Cox, David B. Goldstein, Lucia A. Hindorff, David J. Hunter, Mark I. McCarthy, Erin M. Ramos, Lon R. Cardon, Aravinda Chakravarti, Judy H. Cho, Alan E. Guttmacher, Augustine Kong, Leonid Kruglyak, Elaine Mardis, Charles N. Rotimi, Montgomery Slatkin, David Valle, Alice S. Whittemore, Michael Boehnke, Andrew G. Clark, Evan E. Eichler, Greg Gibson, Jonathan L. Haines, Trudy F. C. Mackay, Steven A. McCarroll, and Peter M. Visscher. Finding the missing heritability of complex diseases. *Nature*, 461(7265):747–753, October 2009.

[28] Zhi Wei, Kai Wang, Hui-Qi Q. Qu, Haitao Zhang, Jonathan Bradfield, Cecilia Kim, Edward Frackleton, et al. From disease association to risk assessment: an optimistic view from genome-wide association studies on type 1 diabetes. *PLoS genetics*, 5(10):e1000678+, October 2009.

[29] Naomi R Wray, Michael E Goddard, and Peter M Visscher. Prediction of individual genetic risk of complex disease. *Current Opinion in Genetics and Development*, 18(73):257–263, 2008.

[30] Jason H. Moore. The ubiquitous nature of epistasis in determining susceptibility to common human diseases. *Human Heredity*, 56(1-3):73–82, 2003.

[31] Sekar Kathiresan, Olle Melander, Dragi Anevski, Candace Guiducci, Noel P. Burtt, Charlotta Roos, Joel N. Hirschhorn, Goran Berglund, Bo Hedblad, Leif Groop, David M. Altshuler, Christopher Newton-Cheh, and Marju Orho-Melander. Polymorphisms Associated with Cholesterol and Risk of Cardiovascular Events. *The New England Journal of Medicine*, 358(12):1240–1249, 2008.

[32] James F. Crow. Hardy, Weinberg and Language Impediments. *Genetics*, 152(3):821–825, 1999.

[33] Jeanette Erdmann, Anika Großhennig, Peter S. Braund, Inke R. König, Christian Hengstenberg, Alistair S. Hall, Patrick Linsel-Nitschke, Sekar Kathiresan, Ben Wright, David-Alexandre Tregouet, Francois Cambien, Petra Bruse, Zouhair Aherrahrou, Arnika K. Wagner, Klaus Stark, Stephen M. Schwartz, Veikko Salomaa, Roberto Elosua, Olle Melander, Benjamin F. Voight, Christopher J. O'Donnell, Leena Peltonen, David S. Siscovick, David Altshuler, Piera Angelica Merlini, Flora Peyvandi, Luisa Bernardinelli, Diego Ardissino, Arne Schillert, Stefan Blankenberg, Tanja Zeller, Philipp Wild, Daniel F. Schwarz, Laurence Tiret, Claire Perret, Stefan Schreiber, Nour Eddine El Mokhtari, Arne Schafer, Winfried Marz, Wilfried Renner, Peter Bugert, Harald Kluter, Jurgen Schrezenmeir, Diana Rubin, Stephen G. Ball, Anthony J. Balmforth, H-Erich Wichmann, Thomas Meitinger, Marcus Fischer, Christa Meisinger, Jens Baumert, Annette Peters, Willem H. Ouwehand, Panos Deloukas, John R. Thompson, Andreas Ziegler, Nilesh J. Samani, and Heribert Schunkert. New susceptibility locus for coronary artery disease on chromosome 3q22.3. *Nature Genetics*, 41(3):280–282, Mar 2009.

[34] Yun Li, Cristen J Willer, Jun Ding, Paul Scheet, and Gonccalo R Abecasis. MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genetic Epidemiology*, 34(8):816–834, December 2010.

[35] Shaun Purcell, Benjamin Neale, Kathe Todd-Brown, Lori Thomas, Manuel A R Ferreira, David Bender, Julian Maller, Pamela Sklar, Paul I W de Bakker, Mark J Daly, and Pak C Sham. PLINK: a tool set for whole-genome association and population-based linkage analyses. *American Journal of Human Genetics*, 81(3):559–575, September 2007.

[36] Richard H Duerr, Kent D Taylor, Steven R Brant, John D Rioux, Mark S Silverberg, Mark J Daly, A Hillary Steinhart, Clara Abraham, Miguel Regueiro, Anne Griffiths, Themistocles Dassopoulos, Alain Bitton, Huiying Yang, Stephan Targan, Lisa Wu Datta, Emily O Kistner, L Philip Schumm, Annette T Lee, Peter K Gregersen, M Michael Barmada, Jerome I Rotter, Dan L Nicolae, and Judy H Cho. A genome-wide association study identifies IL23R as an inflammatory bowel disease gene. *Science (New York, N.Y.)*, 314(5804):1461–1463, December 2006.

[37] Matthew D Mailman, Michael Feolo, Yumi Jin, Masato Kimura, Kimberly Tryka, Rinat Bagoutdinov, Luning Hao, Anne Kiang, Justin Paschall, Lon Phan, Natalia Popova, Stephanie Pretel, Lora Ziyabari, Moira Lee, Yu Shao, Zhen Y Wang, Karl Sirotkin, Minghong Ward, Michael Kholodov, Kerry Zbicz, Jeffrey Beck, Michael Kimelman, Sergey Shevelev, Don Preuss, Eugene Yaschenko, Alan Graeff, James

Ostell, and Stephen T Sherry. The NCBI dbGaP database of genotypes and phenotypes. *Nature Genetics*, 39(10):1181–1186, October 2007.

[38] Andre Franke, Dermot P B McGovern, Jeffrey C Barrett, Kai Wang, Graham L Radford-Smith, Tariq Ahmad, Charlie W Lees, Tobias Balschun, James Lee, Rebecca Roberts, Carl A Anderson, Joshua C Bis, Suzanne Bumpstead, David Ellinghaus, Eleonora M Festen, Michel Georges, Todd Green, Talin Haritunians, Luke Jostins, Anna Latiano, Christopher G Mathew, Grant W Montgomery, Natalie J Prescott, Soumya Raychaudhuri, Jerome I Rotter, Philip Schumm, Yashoda Sharma, Lisa A Simms, Kent D Taylor, David Whiteman, Cisca Wijmenga, Robert N Baldassano, Murray Barclay, Theodore M Bayless, Stephan Brand, Carsten Buning, Albert Cohen, Jean-Frederick Colombel, Mario Cottone, Laura Stronati, Ted Denson, Martine De Vos, Renata D'Inca, Marla Dubinsky, Cathryn Edwards, Tim Florin, Denis Franchimont, Richard Gearry, Jurgen Glas, Andre Van Gossum, Stephen L Guthery, Jonas Halfvarson, Hein W Verspaget, Jean-Pierre Hugot, Amir Karban, Debby Laukens, Ian Lawrance, Marc Lemann, Arie Levine, Cecile Libioulle, Edouard Louis, Craig Mowat, William Newman, Julian Panes, Anne Phillips, Deborah D Proctor, Miguel Regueiro, Richard Russell, Paul Rutgeerts, Jeremy Sanderson, Miquel Sans, Frank Seibold, A Hillary Steinhart, Pieter C F Stokkers, Leif Torkvist, Gerd Kullak-Ublick, David Wilson, Thomas Walters, Stephan R Targan, Steven R Brant, John D Rioux, Mauro D'Amato, Rinse K Weersma, Subra Kugathasan, Anne M Griffiths, John C Mansfield, Severine Vermeire, Richard H Duerr, Mark S Silverberg, Jack Satsangi, Stefan Schreiber, Judy H Cho, Vito Annese, Hakon Hakonarson, Mark J Daly, and Miles Parkes. Genome-wide meta-analysis increases to 71 the number of confirmed crohn's disease susceptibility loci. *Nature Genetics*, 42(12):1118–1125, December 2010.

[39] R. J. Xavier and D. K. Podolsky. Unravelling the pathogenesis of inflammatory bowel disease. *Nature*, 448(7152):427–434, July 2007.

[40] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13(1):57–98, 1997.

[41] J. A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[42] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *Proceedings of the Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, 1:40–44, November 1993.

[43] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas: Learning of Overcomplete Data Representations. *Neurocomputing*, 72(7-9):1547–1555, 2009.

[44] L. Rebollo-Neira and D. Lowe. Optimized orthogonal matching pursuit approach. *IEEE Signal Processing Letters*, 9(4):137–140, 2002.

[45] Kai Labusch and Thomas Martinetz. Learning Sparse Codes for Image Reconstruction. In Michel Verleysen, editor, *Proceedings of the 18th European Symposium on Artificial Neural Networks*, pages 241–246. d-side, 2010.

[46] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2nd ed. 2009. corr. 3rd printing 5th printing. edition, February 2009.

[47] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007.

[48] J B MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 13:266–267, 1967.

[49] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. Academic Press, 4 edition, November 2008.

[50] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. ;Neural-gas' network for vector quantization and its application to time-series prediction. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 4(4):558–569, 1993.

[51] T. Martinetz and K. Schulten. A "Neural-Gas" Network Learns Topologies. *Artificial Neural Networks*, I:397–402, 1991.

[52] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, 1982.

[53] James B. Meigs, Peter Shrader, Lisa M. Sullivan, Jarred B. McAteer, Caroline S. Fox, Josee Dupuis, Alisa K. Manning, Jose C. Florez, Peter W.F. Wilson, Sr. D'Agostino, Ralph B., and L. Adrienne Cupples. Genotype Score in Addition to Common Risk Factors for Prediction of Type 2 Diabetes. *The New England Journal of Medicine*, 359(21):2208–2219, 2008.

[54] R. C. Lewontin and Ken ichi Kojima. The evolutionary dynamics of complex polymorphisms. *Evolution*, 14(4):458–472, December 1960.

[55] W. G. Hill and Alan Robertson. Linkage disequilibrium in finite populations. *Theoretical and Applied Genetics*, 38(6):226–231, 1968.

[56] Peter Kraft and David J Hunter. Genetic risk prediction–are we there yet? *The New England Journal of Medicine*, 360(17):1701–1703, April 2009.

[57] Johanna Jakobsdottir, Michael B Gorin, Yvette P Conley, Robert E Ferrell, and Daniel E Weeks. Interpretation of genetic association studies: markers with replicated highly significant odds ratios may be poor classifiers. *PLoS Genetics*, 5(2):e1000337, 2009.

[58] H. J. Ban, J. Y. Heo, K. S. Oh, and K.J. Park. Identification of type 2 diabetes-associated combination of SNPs using support vector machine. *BMC Genetics*, 11(1):26, 2010.

[59] Yeomin Yoon, Junghan Song, Seung Ho Hong, and Jin Q Kim. Analysis of multiple single nucleotide polymorphisms of candidate genes related to coronary heart disease susceptibility by using support vector machines. *Clinical Chemistry and Laboratory Medicine: CCLM / FESCC*, 41(4):529–534, April 2003.

[60] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[61] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[62] Thomas Martinetz, Kai Labusch, and Daniel Schneegaß. SoftDoubleMinOver: A Simple Procedure for Maximum Margin Classification. In Wlodzislaw Duch, Janusz Kacprzyk, Erkki Oja, and Slawomir Zadrozny, editors, *Artificial Neural Networks: Biological Inspirations. ICANN 2005: 15th International Conference. Proceedings, Part II*, Lecture Notes in Computer Science, pages 301–306, 2005.

[63] W Krauth and M Mezard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A: Mathematical and General*, 20(11):L745–L752, 1987.

[64] Thomas Martinetz, Kai Labusch, and Daniel Schneegaß. SoftDoubleMaxMinOver: Perceptron-like Training of Support Vector Machines. *IEEE Transactions on Neural Networks*, 20(7):1061–1072, 2009.

[65] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2nd edition, November 1999.

[66] Yoav Freund and Robert E Schapire. A short introduction to boosting. *Japonese Society for Artificial Intelligence*, (5):771–780, 1999.

[67] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[68] Paul Viola and Michael J. Jones. Robust Real-Time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[69] S T Sherry, M H Ward, M Kholodov, J Baker, L Phan, E M Smigielski, and K Sirotkin. dbSNP: the NCBI database of genetic variation. *Nucleic Acids Research*, 29(1):308–311, January 2001.

[70] Daniel F. Schwarz, Inke R. König, and Andreas Ziegler. On safari to random jungle: a fast implementation of random forests for high-dimensional data. *Bioinformatics*, 26(14):1752 –1758, July 2010.

[71] Jochen Hampe, Andre Franke, Philip Rosenstiel, Andreas Till, Markus Teuber, Klaus Huse, Mario Albrecht, Gabriele Mayr, Francisco M De La Vega, Jason Briggs, Simone Gunther, Natalie J Prescott, Clive M Onnie, Robert Hasler, Bence Sipos, Ulrich R Folsch, Thomas Lengauer, Matthias Platzer, Christopher G Mathew, Michael Krawczak, and Stefan Schreiber. A genome-wide association scan of nonsynonymous SNPs identifies a susceptibility variant for crohn disease in ATG16L1. *Nature Genetics*, 39(2):207–211, February 2007.

[72] Leonardo H Travassos, Leticia A M Carneiro, Mahendrasingh Ramjeet, Seamus Hussey, Yun-Gi Kim, Joao G Magalhaes, Linda Yuan, Fraser Soares, Evelyn Chea, Lionel Le Bourhis, Ivo G Boneca, Abdelmounaaim Allaoui, Nicola L Jones, Gabriel Nunez, Stephen E Girardin, and Dana J Philpott. Nod1 and nod2 direct autophagy by recruiting ATG16L1 to the plasma membrane at the site of bacterial entry. *Nat Immunol*, 11(1):55–62, January 2010.

[73] 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, June 2010.

[74] Gregory M. Cooper and Jay Shendure. Needles in stacks of needles: finding disease-causal variants in a wealth of genomic data. *Nature Reviews Genetics*, 12(9):628–640, 2011.

Own publications

1. **Ingrid Brænne**, Jeanette Erdmann, and Amir Madany Mamlouk. SNPboost: Interaction Analysis and Risk Prediction on GWA Data. In *Artificial Neural Networks - ICANN 2011, 21th International Conference, Espoo, Finland, June 14-17th, 2011, Proceedings*, volume 6792 of *Lecture Notes in Computer Science*, pages 111–118. Springer, 2011.

2. **Ingrid Brænne**, Kai Labusch, and Amir Madany Mamlouk. Sparse Coding for Feature Selection on Genome-wide Association Data. In *Artificial Neural Networks - ICANN 2010, 20th International Conference, Thessaloniki,Greece, September 15-18, 2010, Proceedings*, volume 6352 of *Lecture Notes in Computer Science*, pages 337–346. Springer, 2010.

3. **Ingrid Brænne**, Kai Labusch, Thomas Martinetz, and Amir Madany Mamlouk. Interpretive Risk Assessment on GWA Data with Sparse Linear Regression. *Machine Learning Reports*, pages 61–68, 2010.

4. Patrick Linsel-Nitschke, Anika Goetz, Jeanette Erdmann, **Ingrid Brænne**, Peter Braund, Christian Hengstenberg, Klaus Stark, Marcus Fischer, Stefan Schreiber, Nour Eddine El Mokhtari, Arne Schaefer, Juergen Schrezenmeier, Diana Rubin, Anke Hinney, Thomas Reinehr, Christian Roth, Jan Ortlepp, Peter Hanrath, Alistair S Hall, Massimo Mangino, Wolfgang Lieb, Claudia Lamina, Iris M Heid, Angela Doering, Christian Gieger, Annette Peters, Thomas Meitinger, H-Erich Wichmann, Inke R Koenig, Andreas Ziegler, Florian Kronenberg, Nilesh J Samani, Heribert Schunkert. Lifelong reduction of LDL-cholesterol related to a common variant in the LDL-receptor gene decreases the risk of coronary artery disease–a Mendelian R,omisation study, *PloS One* 3(8):e2986, 2008

5. Nilesh J Samani, Jeanette Erdmann, Alistair S Hall, Christian Hengstenberg, Massimo Mangino, Bjoern Mayer, Richard J Dixon, Thomas Meitinger, Peter Braund, H-Erich Wichmann, Jennifer H Barrett, Inke R König, Suzanne E Stevens, Silke Szymczak, David-Alexandre Tregouet, Mark M Iles, Friedrich Pahlke, Helen Pollard, Wolfgang Lieb, Francois Cambien, Marcus Fischer, Willem Ouwehand, Stefan Blankenberg, Anthony J Balmforth, Andrea Baessler, Stephen G Ball, Tim M Strom, **Ingrid Brænne**, Christian Gieger, Panos Deloukas, Martin D Tobin, Andreas Ziegler, John R Thompson, and Heribert Schunkert. Genomewide associ-

ation analysis of coronary artery disease. *The New England Journal of Medicine*, 357(5):443–453, August 2007.