DEPARTMENT OF STATISTICS
University of Wisconsin
1300 University Ave.
Madison, WI 53706

TECHNICAL REPORT NO. 1147
December 7, 2008

LASSO-Patternsearch Algorithm

Weiliang Shi[1]
Department of Statistics, University of Wisconsin, Madison WI

[1]Research supported in part by NIH Grant EY09946, NSF Grant DMS 0604572 and ONR Grant N0014-06-0095.

LASSO-Patternsearch Algorithm

By

Weiliang Shi

A dissertation submitted in partial fulfillment of the

requirements for the degree of

doctor of philosophy

(statistics)

at the

UNIVERSITY OF WISCONSIN – MADISON

2008

Abstract

The LASSO-Patternsearch Algorithm and its variant the Grouped LASSO-Patternsearch Algorithm are proposed to efficiently identify patterns of multiple dichotomous risk factors for outcomes of interest in demographic and genomic studies. The patterns considered are those that arise naturally from the log linear expansion of the multivariate Bernoulli density. Both methods are designed for the case where there is a possibly very large number of candidate patterns but it is believed that only a relatively small number are important.

In the LASSO-Patternsearch Algorithm, a LASSO is used to greatly reduce the number of candidate patterns, using a novel computational algorithm that can handle an extremely large number of unknowns simultaneously. The patterns surviving the LASSO are further pruned in the framework of (parametric) generalized linear models. A novel tuning procedure based on the GACV for Bernoulli outcomes, modified to act as a model selector, is used at both steps. We applied the method to myopia data from the population-based Beaver Dam Eye Study, exposing physiologically interesting interacting risk factors. We then applied the the method to data from a generative model of Rheumatoid Arthritis based on Problem 3 from the Genetic Analysis Workshop 15, successfully demonstrating its potential to efficiently recover higher order patterns from attribute vectors of length typical of genomic studies.

The Grouped LASSO-Patternsearch Algorithm is used when the number of variables is too big for the LASSO-Patternsearch Algorithm. The huge number of candidate patterns are divided into small groups, each of which can be analyzed by LPS. The variables in the patterns surviving the group step are analyzed by LPS one more time. We applied the method to gene expression barcode data, successfully finding a small number of interacting genes with very strong predictive power.

Acknowledgments

First and foremost, I would like to express my deep gratitude to my advisor Professor Grace Wahba, for her introduction to the field of penalized likelihood methods, and her guidance and encouragement through the entire course of my research. She shared generously with me her broad knowledge of statistics and her passion to find the truth in the complicated real world. It has been my great honor and privilege to have had the opportunity to work closely and learn from her. This dissertation would be still in infancy without insightful suggestions from her.

The work in this thesis is the product of a collaboration with a number of professors. In particular, I'd like to thank Professor Stephen J. Wright for his guidance in computation. Without him, our optimization algorithm could not be possible. I also want to thank Ms. Kristine Lee, Professor Ronald Klein, Professor Barbara Klein and Professor Rafael Irizarry for their original data sets, constructive advise and valuable insights on the application part of this dissertation.

I am also grateful to Professor Kam-Wah Tsui, Professor Wei-Yin Loh, and Professor Sündüz Keleş for their service in my defense committee. Professor Tsui has been a mentor to me in various ways. He guided me through difficult times during my PhD study. I learned decision tree from Professor Loh. His

ideas and way of teaching have inspired me in my own research. As one of the hosts of the Thursday group, Professor Keleş has given me many valuable suggestions during the last two years.

Former and current members of the Thursday group helped me in various ways these years. I thank Professor Yi Lin for his support and guidance during the first two years of my study, and for introducing me to the group. I thank Fan Lu, John Carew, Xianhong Xie, Ming Yuan, Yongho Jeon, Hyonho Chun, Pei-Fen Kuan, Kevin Eng, Hector Corrada Bravo, Xiwen Ma, Ted Wild and Alina Andrei. They made my life and study in Madison much more enjoyable.

Finally I would like to thank my dear parents Fulin Shi and Xidi Zhu, for their years of support and understanding, especially after my mother got one of the world's most deadly diseases. As the only child of the family, I owe them all I have and all I am about to have. I dedicate this dissertation to them.

Contents

List of Tables

List of Figures

Chapter 1

General Overview

1.1 Overview

In this thesis, we consider the problem which occurs in demographic and genomic studies when there are a large number of risk factors that potentially interact in complicated ways to induce elevated risk. The goal is to search for important patterns of multiple risk factors among a very large number of candidate patterns, with results that are easily interpretable. The LASSO-Patternsearch Algorithm (LPS) and its variant the Grouped LASSO-Patternsearch Algorithm (GLPS) are proposed for this task. All variables are binary, or have been dichotomized before the analysis, at the risk of some loss of information; this allows the study of much higher order interactions than would be possible with risk factors with more than several possible values, or with continuous risk factors. Thus LPS and GLPS may, if desired, be used as a preprocessor to select clusters of variables that are later analyzed in their pre-dichotomized form, see (Zhang, Wahba, Lin, Voelker, Ferris, Klein & Klein 2004). Along with demographic studies, a particularly promising application of LPS and GLPS is to the

analysis of patterns or clusters of SNPs (Single Nucleotide Polymorphisms) or other genetic variables that are associated with a particular phenotype such as the dichotomized gene expression data, when the attribute vectors are very large and there exists a very large number of candidate patterns. LPS and GLPS are designed specifically for the situation where the number of candidate patterns may be very large, but the solution, which may contain high order patterns, is believed to be sparse.

LPS consists of two steps, a LASSO step and a parametric logistic regression step. It works very well when the number of variables is manageable. GLPS can handle as many variables as we want if timing is not an issue. It begins with dividing all patterns into small groups and applying LPS on each group, followed by applying LPS on the variables in the surviving patterns.

1.2 Outline of the Thesis

The rest of the Thesis is organized as follows. Chapter 2 describes the details of the LASSO-Patternsearch Algorithm. We first introduce penalized log likelihood with the LASSO penalty. Then we describe the new criterion to choose the smoothing parameter, BGACV. Next we present the novel computational algorithm to solve the LASSO problem. After that we show the performance of our method through three simulated examples. And finally we apply LPS to a demographic study, the Beaver Dam Eye Study.

In Chapter 3 we apply LPS to data from a generative model of Rheumatoid Arthritis based on Problem 3 from the Genetic Analysis Workshop 15 (GAW). We also note some results from a preliminary version of our algorithm, which was actually presented in GAW.

In Chapter 4 we describle the details of GLPS. We present three simulation examples and the results from analysis of gene expression barcode data.

Chapter 5 gives some concluding remarks. Appendix A derives the BGACV score; Appendix B gives a lemma describing what happens when some of the variables are coded with the opposite direction; Appendix C shows the detailed results of Gene Expression Bar Code data; and finally Appendix D presents the Matlab code for LPS.

Chapter 2

LASSO-Patternsearch Algorithm

2.1 Introduction

In this chapter, we propose the LASSO-Patternsearch Algorithm (LPS) to search for important patterns of multiple risk factors among a very large number of candidate patterns, with results that are easily interpretable. LPS is based on the log linear parametrization of the multivariate Bernoulli distribution (Whittaker 1990) to generate all possible patterns, if feasible, or at least a large subset of all possible patterns up to some maximum order. LPS begins with a LASSO algorithm (penalized Bernoulli likelihood with an l_1 penalty), used with a new tuning score, BGACV. BGACV is a modified version of the GACV score (Xiang & Wahba 1996) to target variable selection, as opposed to Kullback-Liebler distance, which is the GACV target. A novel numerical algorithm is developed specifically for this step, which can handle an extremely large number of basis functions (patterns) simultaneously. This is in particular contrast to most of the literature in the area, which uses greedy or sequential

algorithms. The patterns surviving this process are then entered into a parametric linear logistic regression to obtain the final model, where further sparsity may be enforced via a backward elimination process using the BGACV score as a stopping criterion. Properties of LPS will be examined via simulation, and in demographic data by scrambling responses to establish false pattern generation rates.

There are many approaches that can model data with binary covariates and binary responses, see, for example CART (Breiman, Friedman, Olshen & Stone 1984), LOTUS (Chan & Loh 2004), Logic regression (Ruczinski, Kooperberg & LeBlanc 2003) and Stepwise Penalized Logistic Regression (SPLR) (Park & Hastie 2008). Logic regression is an adaptive regression methodology that constructs predictors as Boolean combinations of binary covariates. It uses simulated annealing to search through the high dimensional covariate space and uses five-fold cross validation and randomization based hypothesis testing to choose the best model size. SPLR is a variant of logistic regression with l_2 penalty to fit interaction models. It uses a forward stepwise procedure to search through the high dimensional covariate space. The model size is chosen by an AIC- or BIC-like score and the smoothing parameter is chosen by 5-fold cross validation. For Gaussian data the LASSO was proposed in (Tibshirani 1996) as a variant of linear least squares ridge regression with many predictor variables. As proposed there, the LASSO minimized the residual sum of squares subject to a constraint that the sum of absolute values of the coefficients of the basis

functions be less than some constant, say t. This is equivalent to minimizing the residual sum of squares plus a penalty which is some multiple λ (depending on t) of the sum of absolute values (l_1 penalty). It was demonstrated there that this approach tended to set many of the coefficients to zero, resulting in a sparse model, a property not generally obtaining with quadratic penalties. A similar idea was exploited in (Chen, Donoho & Saunders 1998) to select a good subset of an overcomplete set of nonorthogonal wavelet basis functions. The asymptotic behavior of LASSO type estimators was studied in (Knight & Fu 2000), and (Osborne, Presnell & Turlach 2000) discussed computational procedures in the Gaussian context. More recently (Efron, Hastie, Johnstone & Tibshirani 2004) discussed variants of the LASSO and methods for computing the LASSO for a continuous range of values of λ in the Gaussian case. Variable selection properties of the LASSO were examined in (Leng, Lin & Wahba 2006) in some special cases, and many applications can be found on the web. In the context of nonparametric ANOVA decompositions (Zhang et al. 2004) used an overcomplete set of basis functions obtained from a Smoothing Spline ANOVA model, and used ℓ_1 penalties on the coefficients of main effects and low order interaction terms, in the spirit of (Chen et al. 1998). The present paper uses some ideas from (Zhang et al. 2004), although the basis function set here is quite different. Other work has implemented ℓ_1 penalties along with quadratic (reproducing kernel square norm) penalties to take advantage of the properties of both kinds of penalties, see for example (Gunn & Kandola 2002) (Lee, Kim,

Lee & Koo 2006) (Zhang & Lin 2006)(Zou & Hastie 2005).

The rest of the Chapter is organized as follows. In Section 2.2 we describe the first (LASSO) step of the LPS including choosing the smoothing parameter by the B-type Generalized Approximate Cross Validation (BGACV), "B" standing for the prior belief that the solution is sparse, analogous to BIC. An efficient algorithm for the LASSO step is presented here. Section 2.3 gives details of the specially designed code for the LASSO which is capable of handling a very large number of patterns simultaneously. Section 2.4 describes the second step of the LASSO-Patternsearch algorithm, utilizing a parametric logistic regression, again tuned by BGACV. Section 2.5 presents three simulation examples, designed to demonstrate the properties of LPS as well as comparing LPS to Logic regression and SPLR. Favorable properties of LPS are exhibited in models with high order patterns and correlated attributes. Section 2.6 applies the method to myopic changes in refraction in an older cohort from the Beaver Dam Eye Study (Klein, Klein, Linton & DeMets 1991), where some interesting risk patterns including one involving smoking and vitamins are found.

2.2 The LASSO-Patternsearch Algorithm

2.2.1 The LASSO-Patternsearch Algorithm - Step 1

Considering n subjects, for which p variables are observed, we first reduce continuous variables to "high" or "low" in order to be able to examine *very many*

variables and *their interactions* simultaneously. We will assume that for all or most of the the p variables, we know in which direction they are likely to affect the outcome or outcomes of interest, if at all. For some variables, for example smoking, it is clear for most endpoints in which direction the smoking variable is likely to be "bad" if it has any effect, and this is true of many but not all variables. For some continuous variables, for example systolic blood pressure, higher is generally "worse", but extremely low can also be "bad". For continuous variables, we need to initially assume the location of a cut point on one side of which the variable is believed to be "risky" ("high") and the other side "not risky" ("low"). For systolic blood pressure that might, for example, be 140 mmHg. For an economic variable, that might be something related to the current standard for poverty level. If the "risky" direction is known for most variables the results will be readily interpretable. Each subject thus has an attribute vector of p zeroes and ones, describing whether each of their p attributes is on one side or the other of the cutoff point. The LASSO-Patternsearch approach described below is able to deal with high order interactions and very large p. The data is $\{y_i, x(i), i = 1, \cdots, n\}$, where $y_i \in \{0, 1\}$ codes the response, $x(i) = (x_1(i), x_2(i), \cdots, x_p(i))$ is the attribute vector for the ith subject, $x_j(i) \in \{0, 1\}$. Define the basis functions $B_{j_1 j_2 .. j_r}(x) = \prod_{\ell=1}^{r} x_{j_\ell}$, that is, $B_{j_1 j_2 .. j_r}(x) = 1$ if $x_{j_1}, ..., x_{j_r}$ are all 1's and 0 otherwise. We will call $B_{j_1 j_2 .. j_r}(x)$ an rth order pattern. Let q be the highest order we consider. Then there will be $N_B = \sum_{\nu=0}^{q} \binom{p}{\nu}$ patterns. If $q = p$, we have a complete set of $N_B = 2^p$

such patterns (including the constant function μ), spanning all possible patterns. If $q = 1$ only first order patterns (henceforth called "main effects") are considered, if $q = 2$ main effects and second order patterns are considered, and so forth. Letting $p(x) = \text{Prob}[y = 1|x]$ and the logit (log odds ratio) be $f(x) = \log[p(x)/(1 - p(x))]$, we estimate f by minimizing

$$I_\lambda(y, f) = \mathcal{L}(y, f) + \lambda J(f) \tag{2.1}$$

where $\mathcal{L}(y, f)$ is $\frac{1}{n}$ times the negative log likelihood:

$$\mathcal{L}(y, f) = \frac{1}{n}\sum_{i=1}^{n}[-y_i f(x(i)) + \log(1 + e^{f(x(i))})] \tag{2.2}$$

with

$$f(x) = \mu + \sum_{\ell=1}^{N_B-1} c_\ell B_\ell(x), \tag{2.3}$$

where we are relabeling the $N_B - 1$ (non-constant) patterns from 1 to $N_B - 1$, and

$$J(f) = \sum_{\ell=1}^{N_B-1} |c_\ell|. \tag{2.4}$$

If all possible patterns are included in (2.3) then f there is the most general form of the log odds ratio for y given x obtainable from the log linear parametrization of the multivariate Bernoulli distribution given in (Whittaker 1990). In Step 1 of the LASSO-Patternsearch we minimize (2.1) using the BGACV score to choose λ. The next section describes the BGACV score and the kinds of results

it can be expected to produce.

2.2.2 B-type Generalized Approximate Cross Validation (BGACV)

The tuning parameter λ in (2.1) balances the trade-off between data fitting and the sparsity of the model. The bigger λ is, the sparser the model is. The choice of λ is generally a crucial part of penalized likelihood methods and machine learning techniques like the Support Vector Machine. For smoothing spline models with Gaussian data, (Wahba & Wold 1975) proposed ordinary leave-out-one cross validation (OCV). Generalized Cross Validation (GCV), derived from OCV, was proposed in (Craven & Wahba 1979)(Golub, Heath & Wahba 1979), and theoretical properties were obtained in (Li 1986) and elsewhere. For smoothing spline models with Bernoulli data and quadratic penalty functionals, (Xiang & Wahba 1996) derived the Generalized Approximate Cross Validation (GACV) from an OCV estimate following the method used to obtain GCV. In (Zhang et al. 2004) GACV was extended to the case of Bernoulli data with continuous covariates and l_1 penalties.

The derivation of the GACV begins with a leaving-out-one likelihood to minimize an estimate of the comparative Kullback-Leibler distance (CKL) between the true and estimated model distributions. The ordinary leave-out-one cross

validation score for CKL is

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}[-y_i f_\lambda^{[-i]}(x(i)) + \log(1 + e^{f_\lambda(x(i))})], \tag{2.5}$$

where f_λ is the minimizer of the objective function (2.1), and $f_\lambda^{[-i]}$ is the minimizer of (2.1) with the ith data point left out. Through a series of approximations and an averaging step as described in Appendix A, we obtain the GACV score appropriate to the present context. It is a simple to compute special case of the GACV score in (Zhang et al. 2004):

$$GACV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}[-y_i f_{\lambda i} + \log(1 + e^{f_{\lambda i}})] + \frac{1}{n}\mathrm{tr}H\frac{\sum_{i=1}^{n} y_i(y_i - p_{\lambda i})}{(n - N_{B_0})}, \tag{2.6}$$

here $H = B^*(B^{*\prime}WB^*)^{-1}B^{*\prime}$, where W is the $n \times n$ diagonal matrix with iith element the estimated variance at $x(i)$ $(p_{i\lambda}(1-p_{i\lambda}))$ and B^* is the $n \times N_{B_0}$ design matrix for the N_{B_0} non-zero c_ℓ in the model. The quantity $\mathrm{tr}H\frac{\sum_{i=1}^{n} y_i(y_i-p_{\lambda i})}{(n-N_{B_0})}$ plays the role of degrees of freedom here.

As is clear from the preceding discussion, the GACV is a criterion whose target is the minimization of the (comparative) Kullback-Liebler distance from the estimate to the unknown "true" model. By analogy with the Gaussian case (where the predictive mean squared error and comparative Kullback-Liebler distance coincide), it is known that optimizing for predictive mean square error and optimizing for model selection *when the true model is sparse* are not in

general the same thing. This is discussed in various places, for example, see (George 2000) which discusses the relation between AIC and BIC, AIC being a predictive criterion and BIC, which generally results in a sparser model, being a model selection criterion, with desirable properties when the "true" model is of fixed (low) dimension as the sample size gets large. See also (Haughton 1988), (Leng et al. 2006) and particularly our remarks at the end of Appendix A. In the AIC to BIC transformation, if γ is the degrees of freedom for the model, then BIC replaces γ with $(\frac{1}{2}\log n)\gamma$. By analogy we obtain a model selection criterion, BGACV, from GACV as follows. Letting γ be the quantity playing the role of degrees of freedom for signal in the Bernoulli-l_1 penalty case,

$$\gamma = \mathrm{tr}H\frac{\sum_{i=1}^{n} y_i(y_i - p_{\lambda i})}{(n - N_{B_0})}, \tag{2.7}$$

γ is replaced by $(\frac{1}{2}\log n)\gamma$ to obtain

$$BGACV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}[-y_i f_{\lambda i} + \log(1+e^{f_{\lambda i}})] + \frac{1}{n}\frac{\log n}{2}\mathrm{tr}H\frac{\sum_{i=1}^{n} y_i(y_i - p_{\lambda i})}{(n - N_{B_0})}. \tag{2.8}$$

We illustrate the difference of empirical performances between GACV and BGACV on a "true" model with a small number of strong patterns. Let $(X_1^*, X_4^*)^T$, $(X_2^*, X_5^*)^T$ and $(X_3^*, X_6^*)^T$ be independently distributed from a bivariate normal distribution with mean 0, variance 1 and covariance 0.7. $X_i = 1$ if $X_i^* > 0$ and 0 otherwise, $i = 1, 2, \cdots, 6$. X_7 is independent of the others and takes two values $\{1, 0\}$, each with a probability of 0.5. $X = (X_1, \cdots, X_7)$. The

sample size $n = 800$. Three patterns that consist of six variables are important, and X_7 is noise. The true logit is

$$f(x) = -2 + 1.5B_1(x) + 1.5B_{23}(x) + 2B_{456}(x).$$

This problem is very small so we chose the maximum order $q = p = 7$, so 127 patterns plus a constant term are entered in the trial model. We ran the simulation 100 times and the result is shown in Table 1. Both GACV and BGACV select the three important patterns perfectly, but GACV selects more noise patterns than BGACV. Note that a total of $2^7 - 4$ noise patterns have been considered in each run. The maximum possible number in the last column is $100 \times (2^7 - 4) = 12,400$. Neither GACV or BGACV is doing a bad job but we will discuss a method to further reduce the number of noise patterns in Section 3. Figure 1 shows the scores of these two criteria in the first data set. BGACV selects a bigger smoothing parameter than GACV does. These scores are not continuous at the the point where a parameter becomes zero so we see jumps in the plots.

2.3 Computation

From a mathematical point of view, this optimization problem (2.1) is the same as the likelihood basis pursuit (LBP) algorithm in (Zhang et al. 2004), but with

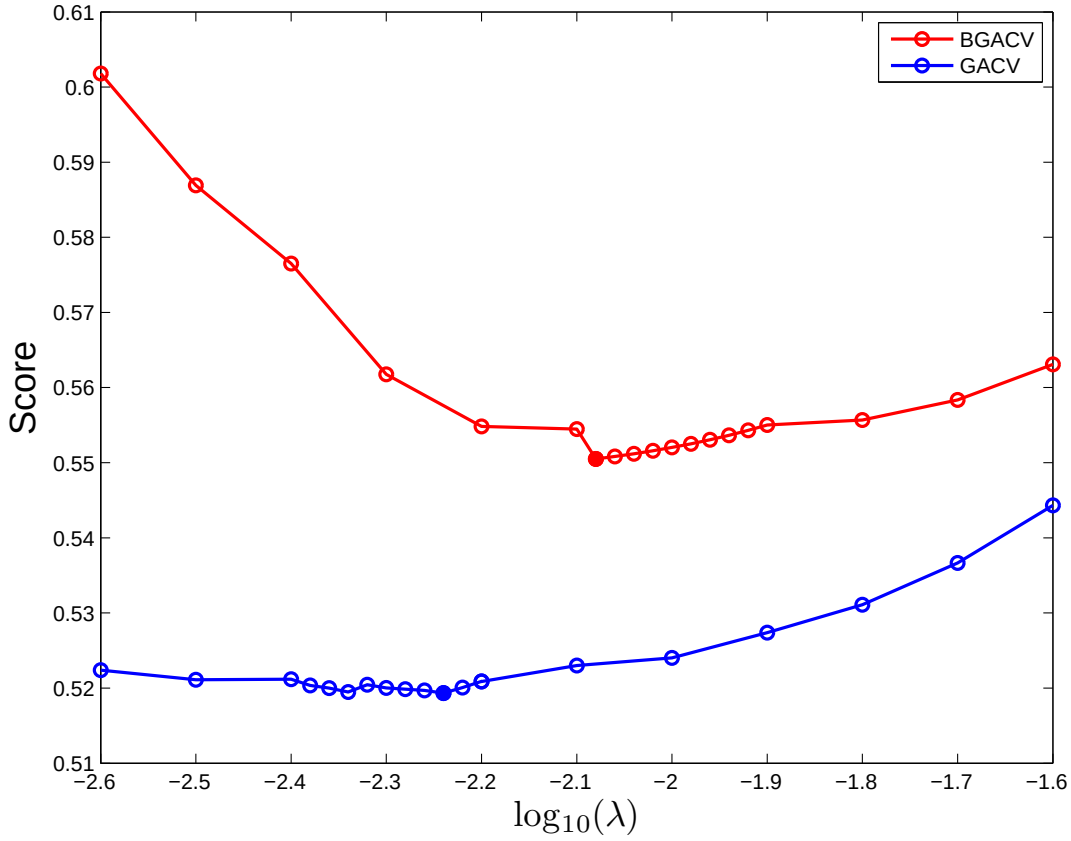

Figure 1: Comparison of BGACV and GACV in the first data set of Simulation Example 1. The solid dots are the minima. BGACV selects a bigger λ than GACV does.

Table 1: The results of Simulation Example 2.5.1. The second through fourth columns are the appearance frequencies of the three important patterns in the 100 runs. The last column is the total appearance frequency of all other patterns. The second and third row compare GACV and BGACV within the first step of LPS. The fourth through sixth rows, which compare the full LPS with Logic regression and SPLR will be discussed in Section 2.5.1.

Method	B_1	B_{23}	B_{456}	other
GACV	100	100	100	749
BGACV	100	100	100	568
LPS	97	96	98	34
Logic	100	94	94	64
SPLR	100	90	55	426

different basis functions. The solution can easily be computed via a general constrained nonlinear minimization code such as MATLAB's `fmincon` on a desktop, for a range of values of λ, provided n and N_B are not too large. However, for extremely large data sets with more than a few attributes p (and therefore a large number N_B of possible basis functions), the problem becomes much more difficult to solve computationally with general optimization software, and algorithms that exploit the structure of the problem are needed. We design an algorithm that uses gradient information for the likelihood term in (2.1) to find an estimate of the correct active set (that is, the set of components c_ℓ that are zero at the minimizer). When there are not too many nonzero parameters, the algorithm also attempts a Newton-like enhancement to the search direction, making use of the fact that first and second partial derivatives of the function in (2.1) with respect to the coefficients c_ℓ are easy to compute analytically once the

function has been evaluated at these values of c_ℓ. It is not economical to compute the full Hessian (the matrix of second partial derivatives), so the algorithm computes only the second derivatives of the log likelihood function $\mathcal{L}$ with respect to those coefficients c_ℓ that appear to be nonzero at the solution. For the problems that the LASSO-Patternsearch is designed to solve, just a small fraction of these N_B coefficients are nonzero at the solution. This approach is similar to the two-metric gradient projection approach for bound-constrained minimization, but avoids duplication of variables and allows certain other economies in the implementation. The algorithm is particularly well suited to solving the problem (2.1) for a number of different values of λ in succession; the solution for one value of λ provides an excellent starting point for the minimization with a nearby value of λ.

The function (2.1) is not differentiable with respect to the coefficients $\{c_\ell\}$ in the expansion (2.4), so most software for large-scale continuous optimization cannot be used to minimize it directly. We can however design a specialized algorithm that uses gradient information for the smooth term $\mathcal{L}(y, f)$ to form an estimate of the correct active set (that is, the set of components c_ℓ that are zero at the minimizer of (2.1)). Some iterations of the algorithm also attempt a Newton-like enhancement to the search direction, computed using the projection of the Hessian of $\mathcal{L}$ onto the set of nonzero components c_ℓ. This approach is similar to the two-metric gradient projection approach for bound-constrained minimization, but avoids duplication of variables and allows certain

other economies in the implementation.

We give details of our approach by simplifying the notation and expressing the problem as follows:

$$\min_{z \in R^m} T_\lambda(z) := T(z) + \lambda \|z\|_1. \tag{2.9}$$

When T is convex (as in our application), z is optimal for (2.9) if and only if the following condition holds:

$$\nabla T(z) + \lambda v = 0, \tag{2.10}$$

for some vector v in the subdifferential of $\|z\|_1$ (denoted by $\partial \|z\|_1$), that is,

$$v_i \begin{cases} = -1 & \text{if } z_i < 0 \\ \in [-1, 1] & \text{if } z_i = 0 \\ = 1 & \text{if } z_i > 0 \end{cases} \tag{2.11}$$

A measure of near-optimality is given as follows:

$$\delta(z) = \min_{v \in \partial \|z\|_1} \|\nabla T(z) + \lambda v\|. \tag{2.12}$$

We have that $\delta(z) = 0$ if and only if z is optimal.

In the remainder of this section, we describe a simplified version of the algorithm used to solve (2.9), finishing with an outline of the enhancements

that were used to decrease its run time.

The basic (first-order) step at iteration k is obtained by forming a simple model of the objective by expanding around the current iterate z^k as follows:

$$d^k = \arg\min_d \; T(z^k) + \nabla T(z^k)^T d + \frac{1}{2}\alpha_k d^T d + \lambda \|z^k + d\|_1, \tag{2.13}$$

where α_k is a positive scalar (whose value is discussed below) and d^k is the proposed step. The subproblem (2.13) is separable in the components of d and therefore trivial to solve in closed form, in $O(m)$ operations. We can examine the solution d^k to obtain an estimate of the active set as follows:

$$\mathcal{A}_k = \{i = 1, 2, \ldots, m \mid (z^k + d^k)_i = 0\}. \tag{2.14}$$

We define the "inactive set" estimate $\mathcal{I}_k$ to be the complement of the active set estimate, that is,

$$\mathcal{I}_k = \{1, 2, \ldots, m\} \setminus \mathcal{A}_k.$$

If the step d^k computed from (2.13) does not yield a decrease in the objective function T_λ, we can increase α_k and re-solve (2.13) to obtain a new d^k. This process can be repeated as needed. It can be shown that, provided z^k does not satisfy an optimality condition, the d^k obtained from (2.13) will yield $T_\lambda(z^k + d^k) < T_\lambda(z^k)$ for α_k sufficiently large.

We enhance the step by computing the restriction of the Hessian $\nabla^2 T(z^k)$

to the set $\mathcal{I}_k$ (denoted by $\nabla^2_{\mathcal{I}_k\mathcal{I}_k}T(z^k)$) and then computing a Newton-like step in the $\mathcal{I}_k$ components as follows:

$$(\nabla^2_{\mathcal{I}_k\mathcal{I}_k}T(z^k) + \delta_k I)p^k_{\mathcal{I}_k} = -\nabla_{\mathcal{I}_k}T(z^k) - \lambda w_{\mathcal{I}_k}, \tag{2.15}$$

where δ_k is a small damping parameter that that goes to zero as z^k approaches the solution, and $w_{\mathcal{I}_k}$ captures the gradient of the term $\|z\|_1$ at the nonzero components of $z^k + d^k$. Specifically, $w_{\mathcal{I}_k}$ coincides with $\partial\|z^k + d^k\|_1$ on the components $i \in \mathcal{I}_k$. If δ_k were set to zero, $p^k_{\mathcal{I}_k}$ would be the (exact) Newton step for the subspace defined by $\mathcal{I}_k$; the use of a damping parameter ensures that the step is well defined even when the partial Hessian $\nabla^2_{\mathcal{I}_k\mathcal{I}_k}T(z^k)$ is singular or nearly singular, as happens with our problems. In our implementation, we choose

$$\delta_k = \min\left(\delta(z^k), \text{mean diagonal of } \nabla^2_{\mathcal{I}_k\mathcal{I}_k}T(z^k)\right), \tag{2.16}$$

where $\delta(z)$ is defined in (2.12).

Because of the special form of $T(z)$ in our case (it is the function $\mathcal{L}$ defined by (2.2) and (2.3)), the Hessian is not expensive to compute once the gradient is known. However, it is dense in general, so considerable savings can be made by evaluating and factoring this matrix on only a reduced subset of the variables, as we do in the scheme described above.

If the partial Newton step calculated above fails to produce a decrease in the objective function T_λ, we reduce its length by a factor γ_k, to the point where

$z_i^k + \gamma_k p_i^k$ has the same sign as z_i^k for all $i \in \mathcal{I}_k$. If this modified step also fails to decrease the objective T_λ, we try the first-order step calculated from (2.13), and take this step if it decreases T_λ. Otherwise, we increase the parameter α_k, leave z^k unchanged, and proceed to the next iteration.

We summarize the algorithm as follows.

Algorithm 2.1

given *initial point* z^0, *initial damping* $\alpha_0 > 0$, *constants* $\texttt{tol} > 0$ *and* $\eta \in (0, 1)$;

for $k = 0, 1, 2, \ldots$

if $\delta(z^k) < \texttt{tol}$

stop *with approximate solution* z^k;

end

Solve (2.13) for d^k; *(* first-order step *)*

Evaluate $\mathcal{A}_k$ *and* $\mathcal{I}_k$;

Compute $p^k_{\mathcal{I}_k}$ *from (2.15); (* reduced Newton step *)*

Set $z^+_{\mathcal{I}_k} = z^k_{\mathcal{I}_k} + p^k_{\mathcal{I}_k}$ *and* $z^+_{\mathcal{A}_k} = 0$;

if $T_\lambda(z^+) < \min(T_\lambda(z^k + d^k), T_\lambda(z^k))$ *(* Newton step successful *)*

$z^{k+1} \leftarrow z^+$;

else

Choose γ_k *as the largest positive number such that* $(z^k + \gamma_k p^k)_i z_i^k > 0$

for all i *with* $z_i^k \neq 0$; *(* damp the Newton step *)*

Set $z^+_{\mathcal{I}_k} = z^k_{\mathcal{I}_k} + \gamma_k p^k_{\mathcal{I}_k}$ *and* $z^+_{\mathcal{A}_k} = 0$;

if $T_\lambda(z^+) < \min(T_\lambda(z^k + d^k), T_\lambda(z^k))$ *(* damped Newton step successful*

*)

$z^{k+1} \leftarrow z^+$;

else if $T_\lambda(z^k+d^k) < T_\lambda(z^k)$ *(* first-order step successful; use it if Newton steps have failed *)*

$z^{k+1} \leftarrow z^k + d^k$;

else *(* unable to find a successful step *)*

$z^{k+1} \leftarrow z^k$;

end

end

(increase or decrease* α *depending on success of first-order step *)*

if $T_\lambda(z^k + d^k) < T_\lambda(z^k)$

$\alpha_{k+1} \leftarrow \eta\alpha_k$; *(* first-order step decreased* T_λ*, so decrease* α **)*

else

$\alpha_{k+1} \leftarrow \alpha_k/\eta$;

end

end

We conclude by discussing some enhancements to this basic approach that can result in significant improvements to the execution time. Note first that evaluation of the full gradient $\nabla T(z^k)$, which is needed to compute the first-order step (2.13) can be quite expensive. Since in most cases the vast majority of components of z^k are zero, and will remain so after the next step is taken, we can economize by selecting just a subset of components of $\nabla T(z^k)$ to evaluate at each

step, and allowing just these components of the first-order step d to be nonzero. Specifically, for some chosen constant $\sigma \in (0, 1]$, we select σm components from the index set $\{1, 2, \ldots, m\}$ at random (using a different random selection at each iteration), and define the working set $\mathcal{W}_k$ to be the union of this set with the set of indices i for which $z_i^k \neq 0$. We then evaluate just the components of $\nabla T(z^k)$ for the indices $i \in \mathcal{W}_k$, and solve (2.13) subject to the constraint that $d_i = 0$ for $i \notin \mathcal{W}_k$.

Since $\delta(z^k)$ cannot be calculated without knowledge of the full gradient $\nabla T(z^k)$, we define a modified version of this quantity by taking the norm in (2.12) over the vector defined by $\mathcal{W}_k$, and use this version to compute the damping parameter δ_k in (2.16).

We modify the convergence criterion by forcing the *full* gradient vector to be computed on the next iteration $k+1$ when the threshold condition $\delta(z^k) < \texttt{tol}$ is satisfied. If this condition is satisfied again at iteration $k + 1$, we declare success and terminate.

A further enhancement is that we compute the second-order enhancement only when the number of components in $\mathcal{I}_k$ is small enough to make computation and factorization of the reduced Hessian economical. In the experiments reported here, we compute only the first order step if the number of components in $\mathcal{I}_k$ exceeds 500.

2.4 The LASSO-Patternsearch Algorithm - Step 2

In Step 2 of LASSO-Patternsearch algorithm, the N_{B_0} patterns surviving Step 1 are entered into a linear logistic regression model using `glmfit` in MATLAB and pattern selection is then carried out by the backward elimination method. We take out one of the N_{B_0} patterns at a time, fit the model with the remaining patterns and compute the tuning score. The pattern that gives the best tuning score to the model after being taken out is removed from the model. This process continues until there are no patterns in the model. A final model is chosen from the pattern set with the best tuning score. Note that all-subset selection is not being done, since this will introduce an overly large number of degrees of freedom into the process.

If copious data is available, then a tuning set can be used to create the tuning score, but this is frequently not the case. Inspired by the tuning method in the LASSO step, we propose the BGACV score for the parametric logistic regression. The likelihood function is smooth with respect to the parameters so the robust assumption that appears in Appendix A is not needed. Other than that, the derivation of the BGACV score for parametric logistic regression follows that in Appendix A. Let s be the current subset of patterns under consideration and B_s be the design matrix. The BGACV score for logistic regression is the same as (2.8) with $H = B_s(B_s'WB_s)^{-1}B_s'$.

$$BGACV(s) = \frac{1}{n}\sum_{i=1}^{n}[-y_i f_{si} + \log(1+e^{f_{si}})] + \frac{1}{n}\frac{\log n}{2}\mathrm{tr}H\frac{\sum_{i=1}^{n} y_i(y_i - p_{si})}{(n - N_{B_0})}, \quad (2.17)$$

where f_{si} is the estimated log odds ratio for observation i and p_{si} is the corresponding probability. The BGACV scores are computed for each model that is considered in the backward elimination procedure, and the model with the smallest BGACV score is taken as the final model.

The following is a summary of the LASSO-Patternsearch algorithm:

1. Solve (2.1) and choose λ by BGACV. Keep the patterns with nonzero coefficients.

2. Put the patterns with nonzero coefficients from Step 1 into a logistic regression model and select models by the backward elimination method with the selection criterion being BGACV.

For simulated data, the results can be compared with the simulated model. For observational data, a selective examination of the data will be used to validate the results. Other logistic regression codes, e. g. from R or SAS can be used instead of `glmfit` here.

2.5 Simulation Studies

In this section we study the empirical performance of the LPS through three simulated examples. The first example continues with simulated data in Section 2.2.2. There are three pairs of correlated variables and one independent variable. Three patterns are related to the response. The second example has only one high order pattern. The correlation within variables in the pattern is high and the correlation between variables in the pattern and other variables varies. The last example studies the performance of our method under various correlation settings. We compare LPS with two other methods, Logic regression (Ruczinski et al. 2003) and Stepwise Penalized Logistic Regression (SPLR) (Park & Hastie 2008). We use the R package `LogicReg` to run Logic regression and the R package `stepPlr` to run SPLR. The number of trees and number of leaves in Logic regression are selected by 5-fold cross validation. The smoothing parameter in SPLR is also selected by 5-fold cross validation, and then the model size is selected by a BIC-like score based on an approximation to a degrees of freedom reproduced in the Comments section of Appendix A.

2.5.1 Simulation Example 2.5.1

In this example we have 7 variables and the sample size is 800. The true logit is $f(x) = -2 + 1.5B_1(x) + 1.5B_{23}(x) + 2B_{456}(x)$. The distribution of the covariates was described in Section 2.2.2. We simulated 100 data sets according to this

model and ran all three methods on these data sets. The results are shown in the last three rows of Table 1.

Let's compare LPS with the LASSO step (third row in Table 1) first. LPS misses all three patterns a few times. However, these numbers are still very close to 100 and more importantly, LPS significantly reduced the number of noise patterns, from over 500 to 34. Here we see why a second step is needed after the LASSO step. Now let's look at LPS compared with the other two methods. Logic regression picks the first term perfectly but it doesn't do as well as LPS on the remaining two patterns. It also selects more noise patterns than LPS. SPLR does worse, especially on the last pattern. It is not surprising because this example is designed to be difficult for SPLR, which is a sequential method. In order for B_{456} to be in the model, at least one main effect of X_4, X_5 and X_6 should enter the model first, say X_4. And then a second order pattern should also enter before B_{456}. It could be B_{45} or B_{46}. However, none of these lower order patterns are in the true model. This makes it very hard for SPLR to consider B_{456}, and the fact that variables in the higher order pattern are correlated with variables in the lower order patterns makes it even harder. We also notice that SPLR selects many more noise patterns than LPS and Logic regression. Because of the way it allows high order patterns to enter the model, the smallest SPLR model has 6 terms, B_1, one of B_2 and B_3, B_{23}, one main effect and one second order pattern in B_{456}, and B_{456}. Conditioning on the appearance frequencies of the important patterns, SPLR has to select at least

$90 + 2 \times 55 = 200$ noise patterns. The difference, $426 - 200 = 226$ is still much bigger than 34 selected by LPS.

2.5.2 Simulation Example 2.5.2

We focus our attention on a high order pattern in this example. Let X_1^* through X_4^* be generated from a normal distribution with mean 1 and variance 1. The correlation between any of these two is 0.7. $X_i = 1$ if $X_i^* > 0$ and 0 otherwise for $i = 1, \cdots, 4$. $X_{i+4} = X_i$ with probability ρ and X_{i+4} will be generated from Bernoulli(0.84) otherwise for $i = 1, \cdots, 4$. ρ takes values $0, 0.2, 0.5$ and 0.7. $X = (X_1, \cdots, X_8)$. Note that $P(X_1 = 1) = 0.84$ in our simulation. The sample size is 2000 and the true logit is $f(x) = -2 + 2B_{1234}(x)$. We consider all possible patterns, so $q = p = 8$. We also ran this example 100 times and the results are shown in Table 2.

LPS does a very good job and it is very robust against increasing ρ, which governs the correlation between variables in the model and other variables. We selected the high order pattern almost perfectly and kept the noise patterns below 10 in all four settings. Logic regression selects the important pattern from 70 to 80 times and noise patterns over 130 times. There is a mild trend that it does worse as the correlation goes up, but the last one is an exception. SPLR is robust against the correlation but it doesn't do very well. It selects the important pattern from 50 to 60 times and noise patterns over 500 times. From this example we can see that LPS is extremely powerful in selecting high

Table 2: The results of simulation example 2.5.2. The numerators are the appearance frequencies of B_{1234} and the denominators are the appearance frequencies of all noise patterns.

ρ	0	0.2	0.5	0.7
LPS	98/10	100/9	97/8	98/5
Logic	82/132	73/162	72/237	74/157
SPLR	53/602	60/576	57/581	58/552

order patterns.

2.5.3 Simulation Example 2.5.3

The previous two examples have a small number of variables so we considered patterns of all orders. To demonstrate the power of our algorithm, we add in more noise variables in this example. The setting is similar to Example 2. Let X_1^* through X_4^* be generated from a normal distribution with mean 1 and variance 1. The correlation between any of these two is ρ_1 and ρ_1 takes values in 0,0.2, 0.5 and 0.7. $X_i = 1$ if $X_i^* > 0$ and 0 otherwise, $i = 1, 2, 3, 4$. $X_{i+4} = X_i$ with probability ρ_2 and X_{i+4} will be generated from Bernoulli(0.84) otherwise for $i = 1, 2, 3, 4$. ρ_2 takes values $0, 0.2, 0.5$ and 0.7 also. X_9 through X_{20} are generated from Bernoulli(0.5) independently. $X = (X_1, \cdots, X_{20})$. The sample size $n = 2000$ and the true logit is

$$f(x) = -2 + 2B_9(x) + 2B_{67}(x) + 2B_{1234}(x).$$

Unlike the previous two examples, we consider patterns only up to the order of 4 because of the large number of variables. That gives us a total of $\binom{20}{0} + \binom{20}{1} + \cdots + \binom{20}{4} = 6196$ basis functions.

Table 3: Results of Simulation Example 2.5.3. In each row of a cell the first three numbers are the appearance frequencies of the important patterns and the last number is the appearance frequency of noise patterns.

$\rho_2 \backslash \rho_1$		0	0.2	0.5	0.7
0	LPS	96/100/100/54	98/100/100/46	100/100/100/43	100/100/100/44
	Logic	100/98/96/120	98/95/93/107	99/94/92/83	100/98/83/134
	SPLR	100/100/100/527	100/100/98/525	100/100/98/487	100/100/97/489
0.2	LPS	99/100/100/46	100/100/100/49	100/100/100/39	100/100/98/36
	Logic	99/97/94/96	100/99/87/94	100/100/88/73	100/99/86/117
	SPLR	100/100/94/517	100/99/96/530	100/97/95/495	100/100/96/485
0.5	LPS	99/100/99/47	99/100/100/51	100/100/99/51	100/100/98/46
	Logic	99/96/86/162	100/95/87/109	100/96/78/122	100/99/80/143
	SPLR	100/98/75/548	100/96/80/552	100/99/80/531	100/98/78/518
0.7	LPS	100/99/96/44	99/99/97/51	100/99/96/67	100/99/94/65
	Logic	100/83/70/195	100/88/69/167	100/85/70/153	100/89/74/126
	SPLR	100/91/51/580	100/85/49/594	100/81/52/584	100/72/55/570

Figure 2 shows the appearance frequencies of the high order pattern B_{1234}. From the left plot we see that LPS dominates the other two methods. All methods are actually very robust against ρ_1, the correlation within the high order pattern. There is a huge gap between the two blue lines, which means SPLR is very sensitive to ρ_2 which governs the correlation between variables in the high order pattern and others. This is confirmed by the right plot, where the blue lines decrease sharply as ρ_2 increases. We see similar but milder behavior in Logic regression. This is quite natural because the problem becomes harder

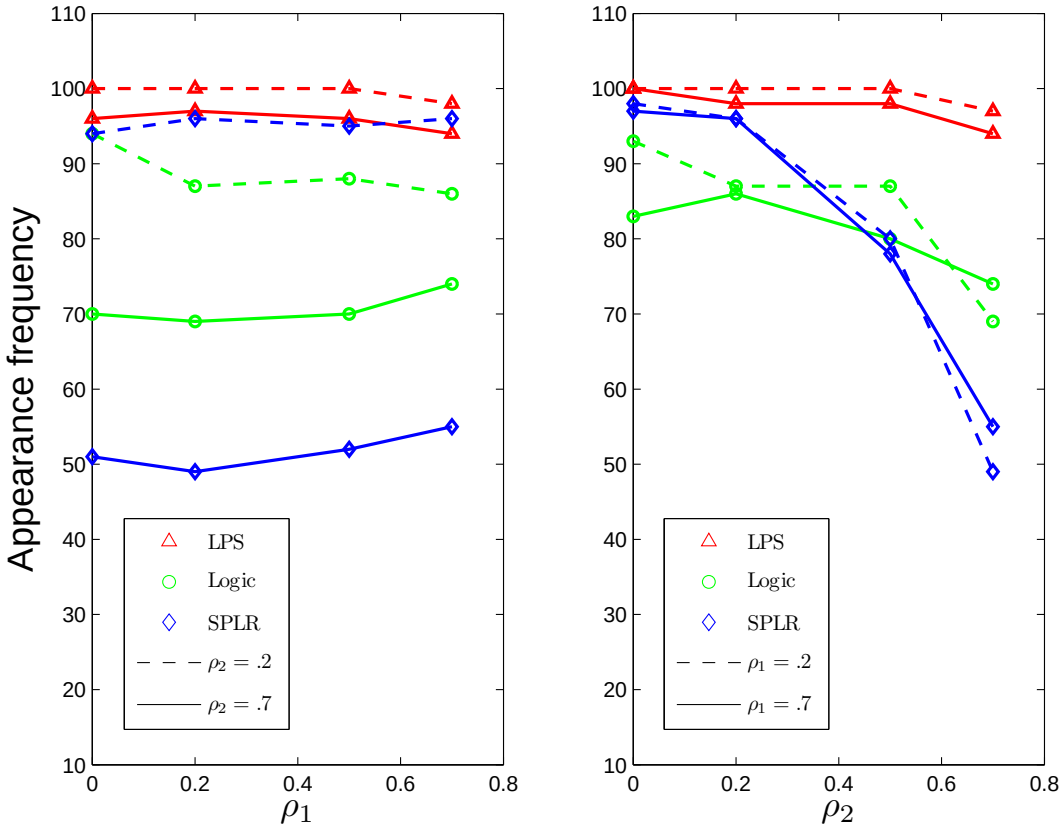

Figure 2: Appearance frequency of the high order pattern B_{1234} in simulation example 2.5.3. In the left panel, the x-axis is ρ_1. ρ_2 is 0.2 for the dashed line and 0.7 for the solid line. In the right panel, the x-axis is ρ_2. ρ_1 is 0.2 for the dashed line and 0.7 for the solid line. The red triangles represent LPS, the blue diamonds represent Logic regression and the green circles represent SPLR

as the the noise variables become more correlated with the important variables. However, LPS handles this issue quite well, at least in the current setting. We see a small decrease in LPS as ρ_2 goes up but those numbers are still very close to 100. The appearance frequencies of lower order patterns are shown in Table 3. The performance of these methods on the second order pattern B_{67} is generally similar but the trend is less obvious as a lower order pattern is easier for most methods. The main effect B_9 is selected almost perfectly by every method in all settings.

2.6 The Beaver Dam Eye Study

The Beaver Dam Eye Study (BDES) is an ongoing population-based study of age-related ocular disorders including cataract, age-related macular degeneration, visual impairment and refractive errors. Between 1987 and 1988, a private census identified 5924 people aged 43 through 84 years in Beaver Dam, WI. 4926 of these people participated the baseline exam (BD I) between 1988 and 1990. Five (BD II), ten (BD III) and fifteen (BD IV) year follow-up data have been collected and there have been several hundred publications on this data. A detailed description of the study can be found in (Klein et al. 1991).

Myopia, or nearsightedness, is one of the most prevalent world-wide eye conditions. Myopia occurs when the eyeball is slightly longer than usual from front to back for a given level of refractive power of the cornea and lens and

people with myopia find it hard to see objects at a distance without a corrective lens. Approximately one-third of the population experience this eye problem and in some countries like Singapore, more than 70% of the population have myopia upon completing college (Seet, Wong, Tan, Saw, Balakrishnan, Lee & Lim 2001). It is believed that myopia is related to various environmental risk factors as well as genetic factors. Refraction is the continuous measure from which myopia is defined. Understanding how refraction changes over time can provide further insight into when myopia may develop. Five and ten-year changes of refraction for the BDES population were summarized in (Lee, Klein & Klein 1999)(Lee, Klein, Klein & Wong 2002). We will study five-year myopic changes in refraction (hereinafter called "myopic change") in an older cohort aged 60 through 69 years. We focus on a small age group since the change of refraction differs for different age groups.

Based on (Lee et al. 2002) and some preliminary analysis we carried out on this data, we choose seven risk factors: *sex, inc, jomyop, catct, pky, asa* and *vtm* (sex, income, juvenile myopia, nuclear cataract, packyear, aspirin and vitamins). Descriptions and binary cut points are presented in Table 4. For most of these variables, we know which direction is bad. For example, male gender is a risk factor for most diseases and smoking is never good. The binary cut points are somewhat subjective here. Regarding *pky*, a pack a day for 30 years, for example, is a fairly substantial smoking history. *catct* has five levels of severity and we cut it at the third level. Aspirin (*asa*) and vitamin supplements (*vtm*)

are commonly taken to maintain good health so we treat not taking them as risk factors. Juvenile myopia *jomyop* is assessed from self-reported age at which the person first started wearing glasses for distance. For the purposes of this study we have defined myopic change as a change in refraction of more than -0.75 diopters from baseline exam to the five year followup; accordingly y is assigned 1 if this change occurred and 0 otherwise. There are 1374 participants in this age group at the baseline examination, of which 952 have measurements of refraction at the baseline and the five-year follow-up. Among the 952 people, 76 have missing values in the covariates. We assume that the missing values are missing at random for both response and covariates, although this assumption is not necessarily valid. However the examination of the missingness and possible imputation are beyond the scope of this study. Our final data consists of 876 subjects without any missing values in the seven risk factors.

Table 4: The variables in the myopic change example. The fourth column shows which direction is risky.

code	variable	unit	higher risk
sex	sex		Male
inc	income	\$1000	< 30
jomyop	juvenile myopia	age first wore glasses for distance	yes before age 21
catct	nuclear cataract	severity 1-5	4-5
pky	packyear	pack per day×years smoked	>30
asa	aspirin	taking/not taking	not taking
vtm	vitamins	taking/not taking	not taking

As the data set is small, we consider all possible patterns ($q = 7$). The first

step of the LASSO-Patternsearch algorithm selected 8 patterns, given in Table 5.

Table 5: Eight patterns selected at Step 1 in the myopic change data

	Pattern	Estimate		Pattern	Estimate
1	*constant*	-2.9020	6	$pky \times vtm$	0.6727
2	*catct*	1.9405	7	$inc \times pky \times vtm$	0.0801
3	*asa*	0.3000	8	$sex \times inc \times jomyop \times asa$	0.8708
4	$inc \times pky$	0.2728	9	$sex \times inc \times catct \times asa$	0.2585
5	$catct \times asa$	0.3958			

Figure 3 plots the coefficients of the 8 patterns plus the constant that survived Step 1 along with 90% confidence intervals. These patterns are then subject to Step 2, backward elimination, tuned via BGACV. The final model after the backward elimination step is

$$\begin{aligned} f &= -2.84 + 2.42 \times catct + 1.11 \times pky \times vtm + 1.98 \times \\ &\quad sex \times inc \times jomyop \times asa + 1.15 \times sex \times inc \times catct \times asa. \end{aligned} \quad (2.18)$$

The significance levels for the coefficients of the four patterns in this model (2.18) can be formally computed and are, respectively 3.3340e-21, 1.7253e-05, 1.5721e-04, and 0.0428. The pattern $pky \times vtm$ catches our attention because the pattern effect is strong and both variables are controllable. This model tells us that the distribution of y, myopic change conditional on $pky = 1$ depends

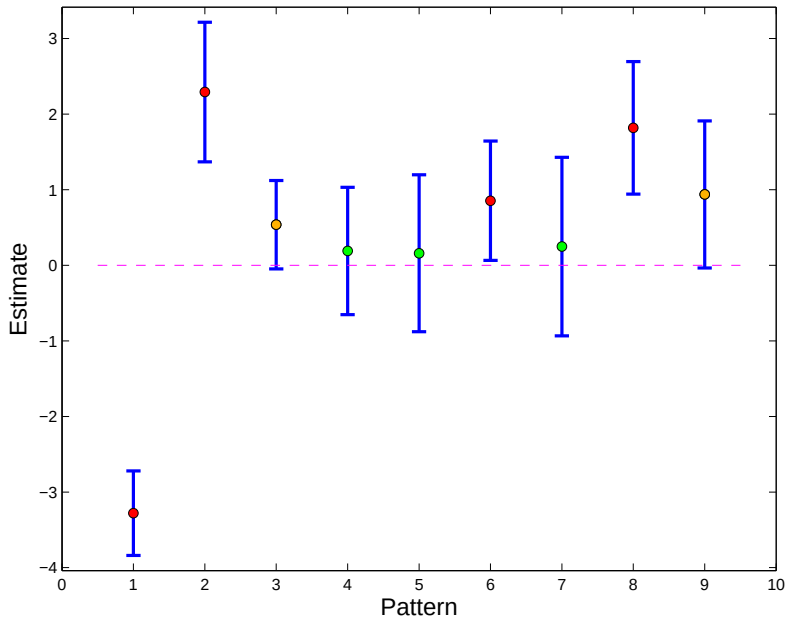

Figure 3: The eight patterns that survived Step 1 of LPS. The vertical bars are 90% confidence intervals based on linear logistic regression. Red dots mark the patterns that are significant at the 90% level. The orange dots are borderline cases, the confidence intervals barely covering 0.

on *catct*, as well as *vtm* and higher order interactions, but myopic change conditional on $pky = 0$ is independent of *vtm*. This interesting effect can easily be seen by going back to a table of the original *catct, pky* and *vtm* data (Table 6). The denominators in the risk column are the number of persons with the given pattern and the numerators are the number of those with $y = 1$. The first two rows list the heavy smokers with cataract. Heavy smokers who take vitamins have a smaller risk of having myopic change. The third and fourth rows list the heavy smokers without cataract. Again, taking vitamins is protective. The first four rows suggest that taking vitamins in heavy smokers is associated with a reduced risk of getting more myopic. The last four rows list all non-heavy smokers. Apparently taking vitamins does not similarly reduce the risk of becoming more myopic in this population. Actually, it is commonly known that smoking significantly decreases the serum and tissue vitamin level, especially Vitamin C and Vitamin E, for example (Galan, Viteri & et al 2005). Our data suggest a possible reduction in myopic change in persons who smoke who take vitamins. However, our data are observational and subject to uncontrolled confounding. A randomized controlled clinical trial would provide the best evidence of any effect of vitamins on myopic change in smokers.

Since the model is the result of previous data mining, caution in making significance statements may be in order. To investigate the probability of the overall procedure to generate significant false patterns, we kept the attribute

Table 6: The raw data for cataract, smoking and not taking vitamins.

catct	pky	no vitamins	risk
1	1	1	$17/23 = 0.7391$
1	1	0	$7/14 = 0.5000$
0	1	1	$22/137 = 0.1606$
0	1	0	$2/49 = 0.0408$
1	0	1	$18/51 = 0.3529$
1	0	0	$19/36 = 0.5278$
0	0	1	$22/363 = 0.0606$
0	0	0	$13/203 = 0.0640$

data fixed, randomly scrambled the response data and applied the LPS algorithm on the scrambled data. The procedure was repeated 1000 times, and in all these runs, 1 main effect, 10 second order, 5 third order and just 1 fourth order patterns showed up. We then checked on the raw data. There are 21 people with the pattern $sex \times inc \times jomyop \times asa$ and 9 of them have myopic change. The incidence rate is 0.4286, as compared to the overall rate of 0.137. Note that none of the variables in this pattern are involved with variables in the two lower order patterns *catct* and $pky \times vtm$ so it can be concluded that the incidence rate is contributed only by the pattern effect. People with the other size four pattern $sex \times inc \times catct \times asa$ have an incidence rate of 0.7727 (17 out of 22), which can be compared with the incidence rate of all people with *catct*, 0.4919.

We also applied Logic regression and SPLR on this data set. Logic selected both *catct* and $pky \times vtm$ but missed the two high order patterns. Instead,

it selected *asa* as a main effect. Note that *asa* is present in both size four patterns. SPLR selected the same patterns as Logic regression with an addition of *pky*, which is necessary for $pky \times vtm$ to be included. These results agree with what we have found in the simulation studies: they are not as likely as LPS in finding higher order patterns. It is noted that in the original version of LPS (Shi, Wahba, Wright, Lee, Klein & Klein 2006), Step 1 was tuned by GACV rather than BGACV, and resulted in the above eight patterns in Table 5 plus four more, but the final model after Step 2 was the same.

2.7 Discussion

A number of considerations enter into the choice of q. If the problem is small and the user is interested in high order patterns, it doesn't hurt to include all possible patterns; if the problem is about the size of Simulation Example 3, $q = 4$ might be a good choice; For genomic data the choice of q can be limited by extremely large attribute vectors. In genomic or other data where the existence of a very small number of important higher patterns is suspected, but there are too many candidates to deal with simultaneously, it may be possible to overcome the curse of dimensionality with multiple screening levels and multiple runs. For example, considering say, third or even fourth order patterns, variables could be assigned to doable sized runs so that every candidate triple or quadruple of variables is assigned to at least one run. With our purpose built algorithm, the

approach is quite amenable to various flavors of exploratory data mining. When a computing system such as Condor (`http://www.cs.wisc.edu/condor/`) is available, many runs can compute simultaneously.

The LASSO-Patternsearch algorithm brings together several known ideas in a novel way, using a tailored tuning and pattern selection procedure and a new purpose built computational algorithm. We have examined the properties of the LPS by analysis of observational data, and simulation studies at a scale similar to the observational data. The results are verified in the simulation studies by examination of the generated "truth", and in the observational data by selective examination of the observational data directly, and data scrambling to check false alarm rates, with excellent results. The novel computational algorithm allows the examination of a very large number of patterns, and, hence, high order interactions. We believe the LASSO-Patternsearch will be an important addition to the toolkit of the statistical data analyst.

Chapter 3

Genetics Analysis Workshop 15 - SNPs

3.1 Introduction

Rheumatoid arthritis (RA) is a complex disease with a moderately strong genetic component. Generally, females are at a higher risk than males and the mean onset of disease is in the fifth decade. Many studies (Thompson 2007) have implicated the HLA region on 6p21 with consistent evidence for several of the DR alleles contributing to risk. There remains much to learn about the genetic susceptibility for rheumatoid arthritis and possible gene and environmental interactions. Identification of disease-causing genes requires extensive evaluation of multiple potential genetic sites. The current trends in genetic epidemiology are to evaluate thousands of single-nucleotide polymorphisms (SNPs) along the chromosome to identify regions where the true diseasecausing gene may lie. SNPs are DNA sequence variations that occur when a single nucleotide in the genome sequence is changed. Many diseases are thought to be associated

with SNP changes at multiple sites that may interact, thus it is important to have tools that can ferret out groups of possibly interacting SNPs. The 15th Genetic Analysis Workshop (GAW 15, November 2006 (Cordell 2007)) focused on RA, and an extensive simulation data set of cases and controls with SNPs was provided to participants and is now publicly available (Miller, Lind, Li & Jang 2007).

Tree-structured methods such as CART (Breiman et al. 1984) and Logic regression (Ruczinski et al. 2003) usually select variables sequentially, and hence may miss the overall correlation structure of the variables. Random forest (Breiman 2001), which grows a large number of classification or regression trees with no trimming or pruning, has gained popularity in the analysis of genetic data. More recently a forward-stepwise penalized logistic regression (Park & Hastie 2008) has been developed for screening gene-gene interactions, which is also a sequential method. In this Chapter, we applied both the Modified LASSO-Patternsearch Algorithm (MLPS) and the original LASSO-Patternsearch Algorithm (LPS) on the simulated RA data from Genetic Analysis Workshop 15 (GAW15), to select SNPs, gene by gene interactions, and gene by environmental interactions. The core of our method is global, in that it deals with a very large number of patterns simultaneously, as opposed to sequential methods that constitute much of the literature in this area. When we first analyzed this data set, our computation algorithm was not powerful enough to include all SNPs simultaneously so we slightly modified LPS. However, the core

of the method is still global. The method has been modified in three places. First, we introduce a screen step to eliminate most of the noise SNPs and their interactions before applying the LASSO step. Second, we only consider the main effects and second-order interactions for ease of computation and interpretation. And last, we take advantage of the fact that we can extract separate training, tuning, and test data sets, all generated from the same (simulated) population. Therefore, we choose the tuning parameters by prediction accuracy on the tuning set, and, for quantitative comparison with other methods, estimate the prediction accuracy of the resulting model on the test set. When the computational algorithm became much more powerful later, we also applied LPS on the data.

We have chosen to use the simulated data (Problem 3) from GAW15. This data simulation was set up to mimic the familial pattern of RA, including a strong effect of DR type at the HLA 2 locus on chromosome 6. A large population of nuclear families (two parents and two offspring) was generated. This population contains close to 2 million sibling pairs (3.6 million subjects). RA affection status was determined for everyone from a complex genetic and environmental model. There were four loci (A on chromosome 16, B on chromosome 8, and C and D both on chromosome 6) in addition to a strong effect of the DR alleles that directly, or through interactions with smoking and gender, modeled RA status. Additional loci modeled severity and other related RA outcomes. From this population, a random sample of 1500 families was selected from among

families with two affected offspring (the affected sib-pair (ASP) group) and another random sample of 2000 families was selected from among families where no member was affected (control group). Within the 2000 families selected for the control group, one offspring was randomly selected to be in the final control group. A total of 100 independent (replicate) data sets were generated.

Microsatellites and SNPs were generated on 22 autosomes. These markers were designed to be like real human autosomes in terms of genetic and physical map lengths. The marker and trait loci were generated to have similar properties, such as linkage disequilibrium, to those observed in real data. We chose to analyze the SNPs, for all controls and the first sibling in the ASP group in Replicate 1. In addition, we used similar data from Replicates 2 and 3 as tuning and test data sets for MLPS.

3.2 Modified LASSO-Patternsearch Algorithm

3.2.1 Methods

We modify the original algorithm for use with genetic (SNP) data (add a screen step, consider only main effects and second-order interactions, and tune the smoothing parameters with a tuning set). Through the use of a series of basis functions described below, we can build a model for the relation between phenotype and variables that embodies main effects and two-factor interactions

("patterns"). The basis functions we use assume dichotomous risk factors. Responses are coded 1 for cases and 0 for controls; females are coded 1 and males 0; smokers as 1 and non-smokers as 0. Age is the only continuous risk factor and we code an elder group (≤ 55) as 1 and a younger group (<55) as 0. Because nearly all SNPs have three levels: normal, one variant allele, and two variant alleles, we retain this information by initially coding them as 0, 1, and 2, respectively. HLA DR also has three levels and we initially code them as DRX = 0, DR1 = 1, and DR4 = 2. For these three level variables, we define basis functions in a generalized way described below, which is equivalent to introducing two dummy variables. The modified algorithm has three steps:

• Step 0: The Screen step

We first define our coding basis functions to be used. Because each SNP can have three levels, we use slightly different notations for basis functions and coefficients in this step. Let x_j be the jth variable and x be $(x_1, x_2, \cdots, x_p)$ where p is the number of variables. Let $B_j^1(x) = 1$ if $x_j = 1$, and 0 otherwise, and let $B_j^2(x) = 1$ if $x_j = 2$, and 0 otherwise. We call these "main effects" basis functions. Let $B_{jk}^{rs} = 1$ if $x_j = r, x_k = s, r, s = 1, 2$, and 0 otherwise (so there are four basis functions for each pair (j, k)). We call these "two-factor interaction" basis functions. These basis functions will be used to code the variables into logistic or penalized logistic regression models. Let $p(x)$ be the probability that $y = 1$, given x, and let $f(x) = \log[p(x)/(1 - p(x))]$. The negative log likelihood

function is given by:

$$\mathcal{L}(y, f) = \sum_{i=1}^{n} [-y_i f(x(i)) + \log(1 + e^{f(x(i))})].$$

We will code the dependence on x by $f(x) = \mu + \sum c_\ell B_\ell(x)$, where the B_ℓ will be specified subsets of the basis functions defined above, and μ and $\{c_\ell\}$ are estimated by minimizing $\mathcal{L}(y, f)$. The goal is to select those basis functions that encode the variables or pairs of variables that best separate cases from controls. Because there are more than 9000 SNPs on all 22 chromosomes, incorporating 9000 main effects and 9000 choose 2 two-factor interactions, there will be more than 10^8 basis functions and we cannot deal with them all simultaneously. We first prescreen for main effects with a logistic regression model as follows. For each $j = 1, \cdots, p$, we find μ, c_j^1 and c_j^2 to minimize the negative log likelihood $\mathcal{L}(y, f_j)$, where $f_j(x) = \mu_j + \sum_{r=1}^{2} c_j^r B_j^r(x)$. We test the hypothesis at the 0.05 level that c_j^r is different from 0 and if it is, the jth variable will go to the second part of the prescreen step and the basis function $B_j^r(x)$ will go to Step 1. Note that each SNP may contribute two basis functions and they are not necessarily significant simultaneously. In that case, the significant basis function will go to the LASSO step and the SNP is still eligible for the screening of interactions. For each pair of variables (x_j, x_k), we construct the model

$$f_{jk}(x) = \mu_{jk} + \sum_{r=1}^{2} c_j^r B_j^r(x) + \sum_{s=1}^{2} c_j^s B_k^s(x) + \sum_{r,s=1,2} c_{jk}^{rs} B_{jk}^{rs}(x)$$

and minimize $\mathcal{L}(y, f_{jk})$. We test the hypotheses that the coefficients $c_{jk}^{rs}, r, s = 1, 2$ are different from 0 at the 0.002 level and the basis functions (patterns) that survive go to Step 1. At this point we would like to select the largest number of candidates that can be comfortably handled by the core global LASSO step. The significance level of 0.002 was chosen in an ad hoc manner to select candidates for the next step and resulted in a large set that, very roughly, met this goal.

• Step 1: The LASSO step

We relabel the basis functions that survive Step 0 as $B_\ell, \ell = 1, 2..., N_B$, where N_B is the total number of the basis functions. Now the notations are consistent with Chapter 2. We estimate f by minimizing (2.1) where f is defined by (2.3). We will choose the smoothing parameter by the prediction accuracy on a separate tuning set, which is different from LPS in Chapter 2. This is done as follows: for each trial value of λ, the minimizer of (2.1) produces $f_\lambda(x)$ and hence $p_\lambda(x)$. We make the important observation that the ratio of cases and controls in the training set is the same as the ratio of cases and controls in the tuning set. Thus, if one had a perfect estimate of $p(x)$ for the population that generated both the test and tuning set, and costs of misclassification were the same for both types of misclassification, then the Bayes rule (to minimize expected cost) for classifying members of the tuning set would be to classify a member as case if $p(x) > 0.5$ and as a control if $p(x) < 0.5$ (Wahba, Lin, Lee & Zhang 2002). Therefore we are motivated to examine the actual error rates on the tuning set, for each choice of λ, by using $p_\lambda(x) = 0.5$ as the classifier.

• Step 2: The logistic regression step

Step 1 produces a relatively sparse model, but we have seen a general tendency for the LASSO-Patternsearch to err on the conservative side in selecting basis functions, that is, there is a very high probability of including all relevant basis functions, at the expense of including some noise terms (Leng et al. 2006). Thus, we took a closer look at the terms that passed Step 1 by putting them into a parametric logistic regression and testing the significance of each term at level α. Rather than choose α on an ad hoc basis, it is selected based on prediction accuracy on the tuning set. The significant term goes into the logistic regression model again and that gives the final model. It is believed that this multi-step process is an effective procedure to meet two goals simultaneously, sparsity and generalizability, and the results below tend to bear this out.

It is believed that this multi-step process is an effective procedure ot meet two goals simultaneously, sparsity and generalizability, and the results below tend to bear this out.

3.2.2 Results

We selected the first replicate as the training set, the second replicate as the tuning set, and the third replicate as the test set. In our first pass, we examined age, smoking, and sex as environmental factors, and all chromosome 6 SNPs. The screen step identified 145 main effects and 1439 interactions, while the final

model included only 6 main effects and no interactions.

$$\begin{aligned} f &= -.62 + .87 \times smoking + 1.05 \times sex - 2.04 \times SNP6\_153\_1 \\ &\quad -1.45 \times SNP6\_154\_1 + 2.23 \times SNP6\_162\_1 - 5.60 \times SNP6\_153\_2, \end{aligned} \tag{3.1}$$

where $SNP6\_153\_1$ is SNP number 153 on chromosome 6 with 1 variant allele, $SNP6\_154\_1$ is SNP number 154 on chromosome 6 with 1 variant allele, $SNP6\_162\_1$ is SNP number 162 on chromosome 6 with 1 variant allele and $SNP6\_153\_2$ is SNP number 153 on chromosome 6 with 2 variant alleles. We found $SNP6\_153 - SNP6\_154$, which we later (after obtaining the answers) found out were close to locus C, and $SNP6\_162$, which was close to locus D. We also found *sex* and *smoking* as expected. Applying this model to predict the RA cases in Replicate 3 as any with an estimated probability > 0.5 resulted in a 13.8% error rate, with sensitivity of 85.3% and specificity of 87.0%. In fact, a plot of the prediction error rate as a function of the threshold (Figure 4) is essentially a convex curve with a minimum of 13.8% for any p between 0.41 and 0.56, verifying the appropriateness of the use of $p = 0.5$ as the threshold.

We then expanded our analysis to SNPs on all chromosomes and included the DR allele from each parent. That gave us 9192 variables, including 9187 SNPs, two DR alleles from parents, age, smoking, and sex. The main effect screen in Step 0 identified 880 basis functions, corresponding to 795 variables. We then

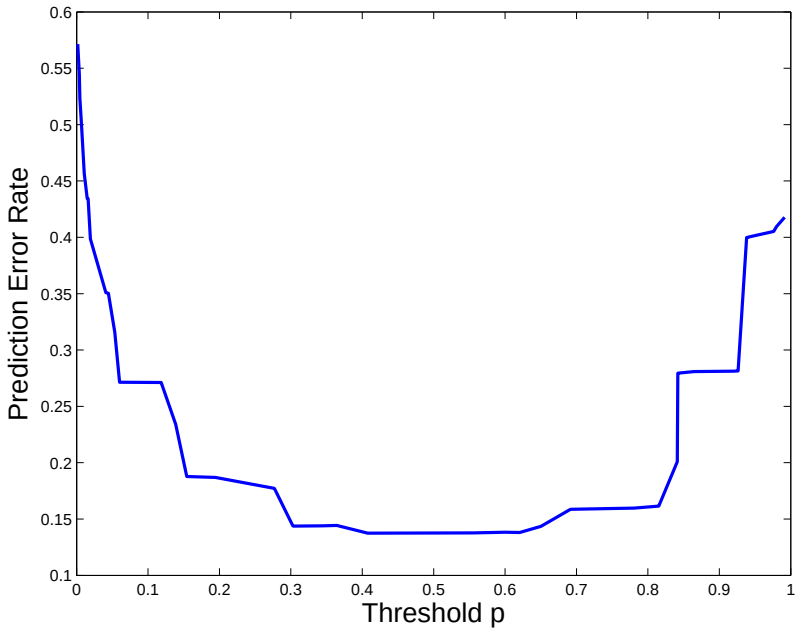

Figure 4: The plot of prediction error vs theshold p for the model on Chromosome 6.

screened for the interaction of these 795 variables and got 1679 interactions. Step 1 included 2559 (880 + 1679) basis functions. The final model identified eight main effects and three interactions (listed in Table 7). The main effects include DR allele from the parents, gender, and smoking, as well as the SNPs from chromosome 6 and an additional SNP on chromosomes 11. All of these were modeled in the simulation: $SNP6\_154$ is close to locus C, $SNP6\_162$ is close to locus D, and $SNP11\_389$ is close to locus F (which modeled severity of IgM). We have also identified three interaction terms, including one within-chromosome interaction on chromosome 2 and two betweenchromosome interactions. These interactions were not directly modeled in the simulation. The prediction error of this model on Replicate 3 is 12.6%, with sensitivity of 85.5% and specificity of 88.8%. A plot of the prediction error as a function of the threshold (not shown) is a convex curve with the minimum error rate of 12.6% for any p between 0.49 and 0.51.

Our method successfully selected many trait loci, but it also missed some. Locus B is on chromosome 8 and it increases the RA risk for smokers. We did not find this because locus B is at the end of the chromosome and none of the SNPs are close by. We also missed locus A, which affects the impact of HLA DR types. Another interaction we missed is sex and locus C. It turns out that this is due to an error in the data simulation, which will be discussed in details in the next section.

Table 7: DR is the HLA DR types. Level 1 is the level of variable 1 and level 2 is the level of variable 2. For SNPs, level = the number of variant alleles at that locus. For DR, level = 1 means DR1 and level = 2 means DR4.

	Variable 1	Level 1	Variable 2	Level 2	Coef
Main effects	constant	-	-	-	
	smoking	-	-	-	1.0434
	sex	-	-	-	1.0819
	$SNP6\_154$	1	-	-	-1.6228
	$SNP6\_162$	1	-	-	2.2717
	$DRfather$	2	-	-	2.3848
	$DRmother$	2	-	-	2.3443
	$SNP6\_154$	2	-	-	-3.0081
	$SNP11\_389$	2	-	-	0.9521
Interactions	$SNP2\_542$	1	$SNP2\_768$	1	-0.5061
	$SNP1\_673$	1	$SNP15\_77$	1	-0.8369
	$SNP8\_233$	1	$SNP16\_131$	2	-0.8044

3.3 Results by LPS

When the analysis in the previous section was done the computational algorithm was not able to solve very large problems. After we improved the algorithm we revisited this data set. This provided an opportunity to apply the original LPS in a context with large genetic attribute vectors, with a known genetic architecture, and to compare the results against the description of the architecture generating the data. We decided to use the GAW data to build a simulation study where we modified the model in (3.1) to introduce a third order pattern, and in the process deal with some anomalous minus signs in our fitted model, also observed by others (Schwartz, Szymczak, Ziegler & Konig 2007). We then

simulated phenotype data from the GAW genotypes and covariates, and can evaluate how well the LPS of this paper reproduces the model generating the data with the third order pattern. This section describes the results.

We take the 674 SNPs in chromosome 6 that were generated as a subset of the genome wide scan data and three environmental variables: *age*, *sex* and *smoking*. We first describe a reanalysis of this data using the original LPS algorithm. We began our analysis with a screen step. In this step, each variable is entered into a linear logistic regression model. For SNPs with three levels, both dummy variables are entered into the same model. We fit these models and keep the variables with at least one p-value less than 0.05. The screen step selected 74 variables (72 SNPs plus *sex* and *smoking*). We then ran LPS on these 74 variables with $q = 2$, which generates 10371 basis functions. The final model was exactly the same as (3.1)!

In MLPS presented in the previsous section both Step 1 and Step 2 were tuned against prediction accuracy using replicate 2 as a tuning set. We were pleased to find that the in-sample BGACV tuning here was just as good as having a separate tuning set. It is interesting to note that in this particular problem, tuning for prediction and for model selection apparently led to the same results, although in general this is not necessarily the case.

We were curious about the apparent counter-intuitive negative coefficients for both $SNP6\_153$ patterns and the $SNP6\_154\_1$ pattern, which appear to say that normal alleles are risky and variant alleles are protective. Others also

found an anomalous protective effect for $SNP6\_154$ normal alleles (Schwartz et al. 2007). We went back and looked at the raw data for $SNP6\_153$ and $SNP6\_154$ as a check but actually Table 4 of (Shi, Lee & Wahba 2007) shows that this protective effect is in the simulated data, for whatever reason, and it also shows that this effect is stronger for women than for men. We then recoded the $SNP6\_153$ and $SNP6\_154$ responses to reflect the actual effect of these two variables as can be seen in tables of the simulated data.

Table 8: Fitted and Simulated Models, see text for explanation.

	Variable 1	Level 1	Variable 2	Level 2	Coef	Est
Main effects	constant	-	-	-	-4.8546	-4.6002
	smoking	-	-	-	0.8603	0.9901
	$SNP6\_153$	1	-	-	1.8911	1.5604
	$SNP6\_162$	1	-	-	2.2013	1.9965
	$SNP6\_154$	2	-	-	0.7700	1.0808
2nd order patterns	*sex*	-	$SNP6\_153$	1	0.7848	0.9984
	sex	-	$SNP6\_154$	2	0.9330	0.9464
	$SNP6\_153$	2	$SNP6\_154$	2	4.5877	4.2465
	$SNP6\_153$	1	$SNP6\_553$	2	0.4021	0
	$SNP6\_154$	2	$SNP6\_490$	1	0.3888	0
Added						
3rd order pattern	$sex \times SNP6\_108\_2 \times SNP6\_334\_2$				3	2.9106

Table 8 shows the results. The new fitted model, above the double line, has four main effects and five second order patterns. The estimated coefficients are given in the column headed "Coef". The $sex \times SNP6\_153$ and $sex \times SNP6\_154$ are there as expected. Two weak second order patterns involving $SNP6\_553$

and $SNP6\_490$ are fitted, but do not appear to be explained by the simulation architecture. This model resulted from fitting with $q = 2$. Then the LPS algorithm was run to include all third order patterns ($q = 3$) of the 74 variables which passed the screen step. This generated 403,594 basis functions. No third order patterns were found, and the fitted model was the same as in the $q = 2$ case. To see if a third order pattern would be found if it were there, we created a generative model with the four main effects and five second order patterns of Table 1, with their coefficients from the "Coef" column, and added to it a third order pattern $sex \times SNP6\_108\_2 \times SNP6\_334\_2$ with coefficient 3. The two SNPs in the third order pattern were chosen to be well separated in chromosome 6 from the reported gene loci. LPS did indeed find the third order pattern.The estimated coefficients are found under the column headed "Est". No noise patterns were found, and the two weak second order patterns in the model were missed. However, the potential for the LPS algorithm to find higher order patterns is clear. Further investigation of the properties of the method in genotype-phenotype scenarios is clearly warranted, taking advantage of the power of the LASSO algorithm to handle a truly large number of unknowns simultaneously. Run time was 4.5 minutes on an AMD Dual-Core 2.8 GHz machine with 64 GB memory. Using multiple runs with clever designs to guarantee that every higher order pattern considered is in at least one run, will allow the analysis of much larger SNP data sets with tolerable computer cost.

3.4 Discussion

The original LASSO-Patternsearch algorithm was designed for demographic studies in which the data sets are smaller with fewer variables. It is a two-stage method whose core is global, as opposed to sequential methods, like trees and forward-stepwise penalized logistic regression (Park & Hastie 2008). We added a screen step to the front end in MLPS to handle the extremely large number of potential SNP patterns. We believe that this conservative screen step is unlikely to delete important patterns here. The results in selecting relevant SNP patterns here and the fact that LPS gives the same model certainly support that belief. The LASSO step took in the resulting large number of basis functions and returned a small fraction of them, retaining the flavor of a completely global algorithm, with the final tuning step removing less significant patterns, chosen as to maximize classification accuracy on the tuning set. The LASSO-Patternsearch method is complementary to random forest approaches. The random forest method is global, but operates quite differently. Thus, LASSO-Patternsearch provides a complimentary tool for the data analyst dealing with very large attribute vectors. LASSO-Patternsearch is also very efficient. Run time was 4.5 minutes on an AMD Dual-Core 2.8 GHz machine with 64 GB memory for 403,594 unknowns. Speed and capacity of the algorithm compare well with other methods discussed. Our method was able to identify important SNPs and covariates, and separate cases from controls similar to the

best results presented at the meeting. We believe that it provides a useful new tool for the analysis of genetic data.

Chapter 4

Grouped LASSO-Patternsearch Algorithm

4.1 Introduction

We showed that the LASSO-Patternsearch Algorithm works very well through simulations, a population based study and a genetic study in the previous two chapters. As a global method, it considers all possible patterns up to certain order. Therefore, it can handle very complicated correlation structures among the predictor variables, which can cause serious problems for the sequential methods. The novel computational algorithm makes LPS a very efficient method. We can solve more than 2 million parameters in less than 1.3 hours.

However, every method has a limitation due to the limited computer memory. For LPS, the limitation is more stringent because all possible patterns are being considered. For example, given that the maximum number of unknowns we can solve is 2 million, only 1999 variables can be included if we consider second order patterns. The current trends in genetic epidemiology are

to evaluate as many markers as possible. The number can easily exceed ten thousand or more (Valdar, Solberg, Gauguier, Burnett, Klenerman, Cookson, Taylor, Rawlins, Mott & Flint 2006). For these kinds of data sets, we will have to add a screen step before applying LPS, just like what we did in Chapter 3. In this Chapter, the Grouped LASSO-Patternsearch Algorithm (GLPS) is proposed to handle huge data sets. Like LPS, we assume that all predictor variables are binary, or have been dichotomized before the analysis. Although we consider myriad parameters, the solution is believed to be sparse. There are two steps in the Grouped LASSO-Patternsearch Algorithm, the group step and the post-group step. GLPS, as suggested by its name, begins with dividing the covariates into small groups. The number of variables in each group is the same. It is determined by the total number of unknowns we can solve in each run of LPS. Suppose we only want to consider main effects and size two patterns. For each group, we run LPS with $q = 2$ on all variables in this group; for each pair of groups, we run LPS with $q = 2$ on all variables in both groups, with the restriction that the size two patterns must have one variable from each group. We call this step of the algorithm the group step. All possible size two patterns have been considered in this step. We collect all the variables that show up in at least one of these runs, either as main effects or in second order patterns. In the post-group step we run LPS again on all variables surviving the group step, and the resulting model is our final model. For LPS in the group step, the tuning parameter is chosen by BGACV. For LPS in the post-group step, we use

two tuning parameters, one for main effects and one for interactions. They are chosen by BGACV2, which is a variation of BGACV. Properties of GLPS will be examined via simulations and in genetic data by evaluating its predictive power.

The rest of the chapter is organized as follows. In Section 4.2 we describe the details of the GLPS algorithm. Section 4.3 presents three simulation examples, designed to demonstrate the properties of GLPS as well as comparing GLPS with Logic regression, SPLR and Random forest. Section 4.4 applies the method to gene expression data dichotomized by the bar code method. And finally in Section 4.5 we present a discussion.

4.2 Grouped LASSO-Patternsearch Algorithm

Considering n subjects with p binary predictor variables, we first divide the p variables into k groups so that each group has g variables. For ease of presentation we assume that $p = kg$. If $(k-1)g < p < kg$, we can easily add $kg - p$ dummy variables that are not related to the response (like zeros) to the kth group. The data is $\{y, x_j, j = 1, \cdots, p\}$, where $y = (y_1, y_2, \cdots, y_n) \in \{0, 1\}$ is the response, $x_j = (x_j(1), x_j(2), \cdots, x_j(n))$ is the jth covariate, $x_j(i) \in \{0, 1\}$. We reorganize the p predictor variables and denote them by x_{st}, $s = 1, \cdots, k$ indexes groups and $t = 1, \cdots, g$ indexes variables within each group. Let q be the highest order of patterns we consider. Then there will be $N_B = \sum_{\nu=0}^{q} \binom{p}{\nu}$

patterns if we apply LPS. We use $q = 2$ to illustrate the GLPS algorithm. The case with larger q can easily be extended.

Let us look at the pair: the s_1th group and the s_2th group, where $1 \leq s_1 < s_2 \leq k$. The basis functions we consider are $\{B_{t_1t_2} = x_{s_1t_1} \times x_{s_2t_2}, t_1, t_2 = 1, \cdots, g\}$, $\{B_{t_1} = x_{s_1t_1}, t_1 = 1, \cdots, g\}$ and $\{B_{t_2} = x_{s_2t_2}, t_2 = 1, \cdots, g\}$. In words, all main effects and all size two patterns that consist of one variable from each group are included. There will be a total of $N_B = 2g + g^2 + 1$ patterns, including the constant. We first solve (2.1) with the basis functions defined above. The smoothing parameter λ is chosen by BGACV. We then build a logistic regression model on the remaining basis functions and select models by the backward elimination method with the selection criterion being BGACV. We call this step the between-group step because of the way we choose variables for the size two patterns. We loop through all possible pairs of groups here and record all the surviving variables.

Now we move to the size two patterns within each group. We call this step the within-group step. Between-group step and within-group step are named group steps. For the sth group, the basis functions we consider are $\{B_{t_1t_2} = x_{st_1} \times x_{st_2}, t_1, t_2 = 1, \cdots, g\}$ and $\{B_t = x_{st}, t = 1, \cdots, g\}$. There will be a total of $\frac{g(g+1)}{2} + 1$ patterns, including the constant. Note that this number is about half of that of the between-group step, so in practice we combine two neighboring groups, resulting in $N_B = g^2 + g + 1$ patterns. Similar to the above step, we use LASSO followed by logistic regression to do variable selection and

record all the surviving variables. Figure 5 is a graphical presentation of the group steps.

We take all variables that survive either the between-group step or the within-group step. For a surviving size two pattern, both variables are taken. Some of the variables may appear more than once and we only need one replicate. Usually only a small number of variables, denoted by p^*, will survive and they can be handled by LPS. We call this step the post-group step. Unlike the usual LPS, we use different smoothing parameters for main effects and size two patterns here. Suppose there are N_{B_1} $(= p^*)$ main effects and N_{B_2} $(= \binom{p^*}{2})$ size two patterns. Denote the main effects by $\{B_{1\ell}, \ell = 1, \cdots, N_{B_1}\}$ and the size two patterns by $\{B_{2\ell}, \ell = 1, \cdots, N_{B_2}\}$. The objective function becomes

$$I_\lambda(y, f) = \mathcal{L}(y, f) + \lambda_1 J_1(f) + \lambda_2 J_2(f), \tag{4.1}$$

where the link function f is

$$f(x) = \mu + \sum_{\ell=1}^{N_{B_1}} c_{1\ell} B_{1\ell}(x) + \sum_{\ell=1}^{N_{B_2}} c_{2\ell} B_{2\ell}(x), \tag{4.2}$$

and the penalties are

$$J_1(f) = \sum_{\ell=1}^{N_{B_1}} |c_{1\ell}|, \quad J_2(f) = \sum_{\ell=1}^{N_{B_2}} |c_{2\ell}|. \tag{4.3}$$

The choice of smoothing parameters, (λ_1, λ_2), in (4.1) are critical. We were

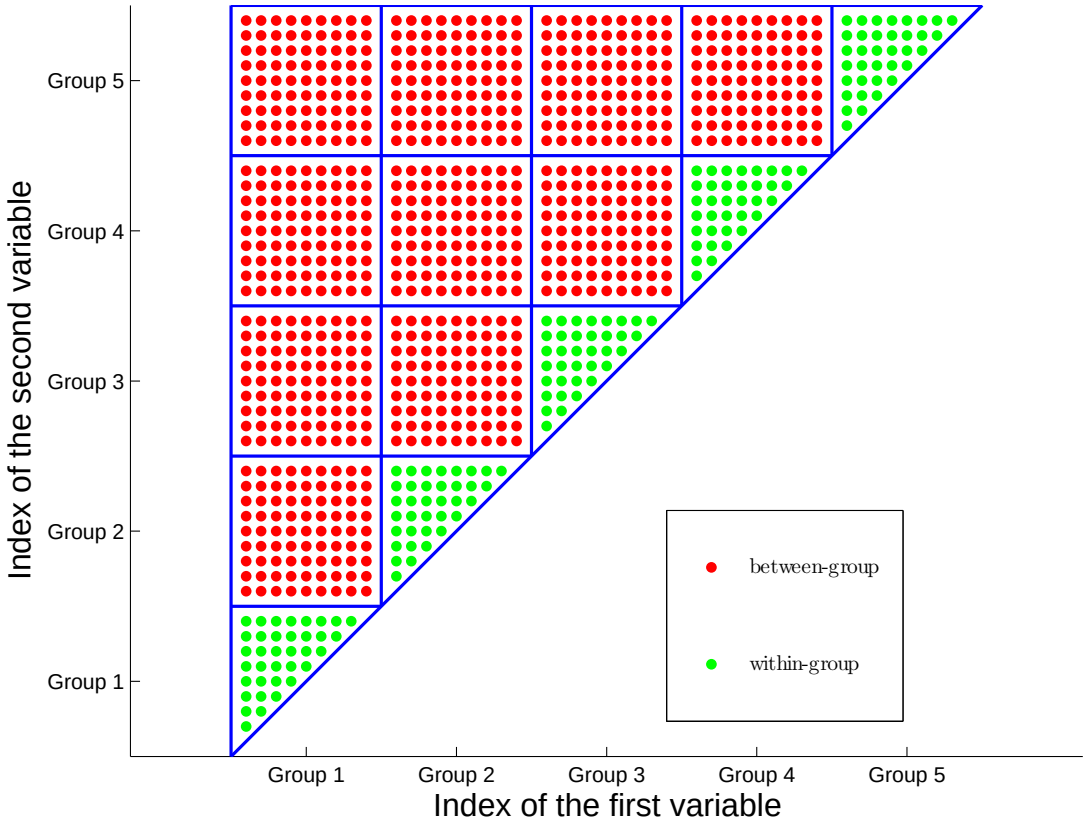

Figure 5: The plot of the group step of GLPS. We use 5 groups as an example. The length of the squares is the number of variables in every group. Each dot represents a size two pattern. The x axis indexes which group the first variable is from and the y axis indexes which group the second variable is from. The red squares are all runs in the between-group step and the green triangles are all runs in the within-group step.

surprised to find out that BGACV does not work very well. In a problem where both main effects and size two patterns are important, often times only size two patterns appear and very few times only main effects appear. In the first scenario true main effects appear in size two patterns with some other variables; in the second scenario true size two patterns appear as main effects. We believe that this phenomena is caused by the huge difference in the numbers of coefficients penalized by the two smoothing parameters. N_{B_2} is close to half of the square of N_{B_1}. The solution is very sensitive to small changes of smoothing parameters. Grid search method, which is commonly used in the search of smoothing parameters, can not find the exact optimal solutions. To fix this problem, we introduce the following penalty

$$BGACV2(\lambda_1, \lambda_2) = BGACV(\lambda_1, \lambda_2) \times (1 + 0.5\frac{|n_{b_1} - n_{b_2}|}{n_{b_1} + n_{b_2}}), \qquad (4.4)$$

where n_{b_1} is the number of nonzero coefficients of main effects and n_{b_2} is the number of nonzero coefficients of size two patterns. We add a penalty to force these two numbers to be close. Then the two scenarios described above can not happen. If the true model only has main effects the number of main effects showing up will be smaller than before due to the new penalty. However, BGACV is very conservative (see discussion in Section 2.2.2). We won't miss any important patterns by using the new penalty. We are just making it less conservative. Some size two patterns might show up as well. This can be taken

care of by the parametric logistic regression step followed by solving (4.1). The argument for a model with size two patterns only is similar.

Some minor changes need to be made if we are interested in size tree patterns ($q = 3$). Firstly, there will be four group level steps. Let $1 \leq s_1 \leq s_2 \leq s_3 \leq k$ be the groups from which the three variables are chosen. We have the following four cases: $s_1 < s_2 < s_3$, $s_1 = s_2 < s_3$, $s_1 < s_2 = s_3$ and $s_1 = s_2 = s_3$. In the post-group step, we will still be using two smoothing parameters, one for main effects and one for interactions, including size two patterns and size three patterns. The penalty added to BGACV is slightly different:

$$BGACV3(\lambda_1, \lambda_2) = BGACV(\lambda_1, \lambda_2) \times (1 + 0.5 \frac{|n_{b_1} - n_a| + |n_{b_2} - n_a| + |n_{b_3} - n_a|}{n_{b_1} + n_{b_2} + n_{b_3}}), \tag{4.5}$$

where n_{b_1} is the number of nonzero main effects, n_{b_2} is the number of nonzero size two patterns, b_{b_3} is the number of nonzero size three patterns and n_a is the average of the three.

In practice we only implemented $q = 2$ and $q = 3$. Higher order patterns are not very interesting given the huge number of variables we consider. However, they will not be hard to implement if necessary. From now on, we use GLPS to denote the case where $q = 2$. And we use GLPS3 to denote the case where $q = 3$. Sometimes, GLPS also means general GLPS with different q, depending on the setting.

The choice of g, the number of variables in each group, depends on the power

of the computer. On our super server (AMD Dual-Core 2.8 GHz with 64 GB memory), we usually set $g = 2,000$ for $q = 2$. This generates $N_B = 2,001,001$ basis functions, which can be handled by the super server comfortably. On a more standard computer (Intel(R) Pentium(R) 4 2.80GHz with 2 GB memory), we usually set $g = 200$ for $q = 2$ and $g = 35$ for $q = 3$. One big advantage of our algorithm is that the runs in group steps are parallel. We can run them simultaneously on different machines. The computing system Condor (`http://www.cs.wisc.edu/condor/`) provides us an excellent opportunity to do this job. We request all jobs to go to machines with at least 2 GB of memory so that we can use $g = 200$ for $q = 2$ and $g = 35$ for $q = 3$. Different g values usually give very similar results, if not the same. When better computers are available bigger g is preferred because it is usually faster.

4.3 Simulation Studies

In this section we study the empirical performance of GLPS through three simulated examples. In the first example the data set is relatively small and all variables are independent. Three patterns are included, one main effect and two second order interactions. The second example has a very large data set and strong correlations among neighboring variables. Two main effects and two second order interactions are included. The last example studies the performance of GLPS3. Two main effects, one second order interaction and

one third order interaction are included. We compare GLPS with three other methods, Logic regression (Ruczinski et al. 2003), Stepwise Penalized Logistic Regression (SPLR) (Park & Hastie 2008) and Random forest (RF) (Breiman 2001). We use the R package `LogicReg` to run Logic regression, the R package `stepPlr` to run SPLR, and the R package `randomForest` to run random forest. The number of trees and number of leaves in Logic regression are selected by 5-fold cross validation. The smoothing parameter in SPLR is also selected by 5-fold cross validation, and then the model size is selected by BIC.

4.3.1 Simulation Example 4.3.1

400 iid Bernoulli(0.5) random variables are generated. The sample size $n = 700$. The true logit function is

$$f(x) = -2 + 1.5X_{50} + 1.5X_{150}X_{250} + 1.5X_{251}X_{252}.$$

We simulated 100 data sets according to this model and ran all four methods on these data sets.

The results of this simulation are shown in Table 9. We use two numbers to show the appearance frequency of a main effect. The first number (outside the parenthesis) counts how many times the variable appears as a main effect and we call this number the pattern count; the second number (inside the parenthesis) counts how many times the variables appears in the model, either as a

Table 9: The result of Simulation 4.3.1. For each pattern, the number outside the parenthesis is the appearance frequency of the pattern; the two numbers inside the parenthesis are the appearance frequencies of the two variables in the pattern (they can appear as main effects, or in different patterns).

Methods	X_{50}	$X_{150}X_{250}$	$X_{251}X_{252}$	noise
GLPS	94 (100)	99 (99,99)	96 (97,97)	153
Logic	100 (100)	70 (88,91)	65 (84,90)	190
RF	NA (100)	NA (96,97)	NA (94,96)	517
SPLR	100 (100)	97 (100,97)	91 (100,98)	712

main effect or in some interactions and we call this number the variable count. Similarly, we need three numbers to show the appearance frequency of a size two pattern, one pattern count and two variable counts. The pattern count counts the number of times it appears as the exact pattern and the two variable counts count the number of times each variable appears in the model. Random forest doesn't generate a specific model. It only provides the importance scores of all variables. For each run, I took the top 10 variables as the selected variables. Therefore, we are not able to get the pattern count for random forest. They are denoted by 'NA' in the table.

GLPS selected all three patterns almost perfectly and the least number of noise patterns. Logic regression didn't do very well in the size two patterns and it selected slightly more noise. Random forest did well in selecting the important variables but it also selected lots of noise. Remember that we took top 10 variables for random forest. The number of noise variables will drop if we take less, but the appearance frequencies of the important variables will also

drop. SPLR performed similar to GLPS in selecting the patterns but it selected much more noise.

4.3.2 Simulation Example 4.3.2

This simulation studies the behavior of GLPS on a very large data set with correlations among the covarriates. The sample size $n = 1000$ and the number of variables $p = 8000$. $X_i^* \sim N(0,1)$, $i = 1, \cdots, p$. $\text{corr}(X_i^*, X_{i+1}^*) = 2/3$, $\text{corr}(X_i^*, X_{i+2}^*) = 1/3$, $i = 1, 2, \cdots, p-2$. X_i^* and X_j^* are independent if $|i-j| > 2$. $X_i = 1$ if $X_i^* > 0$ and 0 otherwise, $i = 1, \cdots, p$. The true logit function is

$$f(x) = -4 + 2X_{500} + 3X_{5000} + 2X_{1000}X_{3000} + 3X_{7000}X_{7002}.$$

The simulation was run 50 times. Logic regression has an upper limit on the number of predictor variables, 1000, so it can not be applied in this example. For random forest, we selected the top 12 variables. The results are shown in Table 10. GLPS did almost perfectly in selecting the important patterns. The number of noise patterns being selected was the smallest. RF did quite bad on the pattern $X_{1000}X_{3000}$. SPLR did perfectly on all four patterns but the number of noise patterns was huge. There is a parameter in SPLR that needs to be specified by the user, the maximum number of terms. We set it to be 20 here. Actually we reached the maximum in all 50 runs because $(50 \times 4 + 800)/50 = 20$. We would reach the maximum even if the maximum number of terms is set to

be 50.

Table 10: The result of Simulation Example 4.3.2. Logic regression not applied.

Methods	X_{500}	X_{5000}	$X_{1000}X_{3000}$	$X_{7000}X_{7002}$	noise
GLPS	50 (50)	50 (50)	48 (48,50)	50 (50,50)	278
RF	NA (50)	NA (50)	NA (28,37)	NA (50,50)	335
SPLR	50 (50)	50 (50)	50 (50,50)	50 (50,50)	800

4.3.3 Simulation Example 4.3.3

This simulation studies the behavior of GLPS3 on a large data set. The sample size $n = 1000$ and the number of variables $p = 500$. The correlation structure of these variables is the same as that of the previous example. In other words, two variables are highly correlated if they are next to each other; they are moderately correlated if they are separated by one variable; they are independent if they are separated by at least two variables. The marginal distribution of all 500 variables is Bernoulli(0.5). The true logit function is

$$f(x) = -4 + 2X_{100} + 3X_{200} + 2X_{300}X_{400} + 3X_{150}X_{450}X_{451}.$$

The simulation was run 50 times and the results are shown in Table 11. GLPS3 did quite well here, although it selected slightly more noise patterns than Logic regression. Logic regression didn't do very well on the two interaction terms. Random forest did well on the size three pattern but it missed the size two

pattern quite a few times. Similar to previous examples, SPLR did well in selecting important patterns, but it also selected lots of noise patterns.

Table 11: The result of Simulation Example 4.3.3.

Methods	X_{100}	X_{200}	$X_{300}X_{400}$	$X_{150}X_{450}X_{451}$	noise
GLPS3	47 (50)	50 (50)	47 (50,50)	47 (50,49,48)	204
Logic	50 (50)	50 (50)	34 (43,44)	30 (50,44,41)	151
RF	NA (50)	NA (50)	NA (36,40)	NA (49,47,49)	279
SPLR	50 (50)	50 (50)	45 (49,50)	50 (50,50,50)	554

We summarize the performance of all four methods here. Logic regression can't handle very large data sets and it doesn't do well on the interaction terms. Random forest doesn't give a specific model. It does well most of the time but it can be bad if the signal is not very strong (small coefficients). SPLR is good at picking the right patterns but it also picks lots of noise. GLPS can get the right patterns almost all times and keep the noise within a reasonable amount.

4.4 The Gene Expression Barcode Data

With current microarray technology we are able to measure thousands of RNA transcripts at one time. This allows for the characterization of cells and tissues in greater depth. However, feature characteristics such as probe sequence can cause the observed intensity to be far away from the actual expression. Even though the 'probe effect' is large, it is very consistent across different hybridizations, meaning that the probe effect is very similar when comparing

the intensities of different hybridizations for the same gene. Therefore, the majority of microarray data analysis uses relative expression rather than absolute expression. To overcome this, a gene expression bar code (GEBC) (Zilliox & Irizarry 2007) was proposed recently. They wanted to know what intensity relates to no expression for a given gene and microarray platform. GEBC starts with preprocessing all genes by Robust Multi-array Analysis (RMA) (Irizarry & Hobbs 2003). Then for each gene an empirical density smoother was used to estimate the density function of this gene across tissues. The smallest mode of the density function was considered as the expected intensity of an unexpressed gene. Gene expressions to the left of the mode were used to estimate the standard deviation of unexpressed genes. Lastly, they selected a constant K and considered genes to be expressed in tissues when the log expression estimates were K standard deviations larger than the unexpressed mean. K was chosen to be 6 by cross-validation. The expressed genes were coded as 1 and unexpressed genes coded as 0.

GEBC (Zilliox & Irizarry 2007) downloaded raw data that is public available from 40 different studies. This resulted in a database of 1092 human samples representing 118 different tissues. Of these, 498 were normal tissues, 500 were breast tumors, and 94 were other diseases. There were a total of 22,215 genes for each sample. In this section we applied GLPS on these data sets after the genes being dichotomized by GEBC. Many of these genes have unbalanced expression levels, i.e., only a small percentage are expressed or unexpressed.

These genes are not useful in building predictive models so we removed them from our analysis, with 7,654 genes remained. In our first analysis, we took all normal tissues as controls and all non-breast tumor samples as cases. In the second analysis we considered the survival time of breast cancer patients. We defined those died before 5 years as cases and those were alive after 10 years as controls.

4.4.1 Normal vs Cancer

In this analysis, all normal tissue and cancer tissue other than breast tumors are used. We excluded breast tumors because no normal breast tissue is available. There are 503 normal tissue samples and 70 cancer tissue samples. The incidence rate is 12.2%. We divided the data into a training set and a test set. The training set consists of 4/5 of the samples and the test set is the rest. We first applied GLPS on the training set with all 7,654 genes. The resulting model is shown in Table 12. We picked four main effects and two size two interactions. We tested the performance of this model on the test set, which yields a prediction accuracy of 96.5% with a sensitivity of 80.0% and a specificity of 97.0%.

We want to compare our model with models fitted by Logic regression and SPLR. However, Logic regression can handle at most 1,000 variables. Therefore we used a screen step to select a subset of most important genes. Here 636 genes were selected and we denoted this the reduced cancer data. Applying GLPS on these genes gives the model in Table 13. The prediction accuracy of this model

Table 12: The model by GLPS on cancer data with all genes

Genes			coefficients
constant			-8.811562
203238_s_at			1.514712
204351_at			2.885140
204779_s_at			1.218652
209215_at			1.669472
202546_at	×	209656_s_at	2.505314
210512_s_at	×	212543_at	3.172602

is 98.2% with a sensitivity of 86.7% and a specificity of 100%.

Table 13: The model by GLPS on the reduced cancer data

Genes			coefficients
constant			-8.147014
200755_s_at	×	202454_s_at	3.576007
200770_s_at	×	209771_x_at	1.929002
201818_at	×	202740_at	3.292240
202489_s_at	×	217850_at	3.747756
203238_s_at	×	208651_x_at	2.344783

The models in Tables 12 and 13 are quite different. They only share one gene in common. However, their predictive power is almost the same. The reason is that many of these genes are highly correlated. Choosing one important gene or another correlated gene doesn't make a big difference. Note that this is a special property of the l_1 norm. It tends to pick the most sparse model. An l_2 norm will pick all correlated genes that are predictive.

We also applied GLPS3, Logic regression and SPLR on the reduced data set.

The results are summarized in Table 14. We present the number of genes in the model (# Gene), the number of parameters in the model (# Para), the highest order of interactions (q), the summation of the previous three (Total), prediction accuracy (Pred), sensitivity (Sens) and specificity (Spec) of the model. The first three measure the complexity of the model. We add them together to get an overall criterion. We see that GLPS dominates in almost all criteria. GLPS3 has the smallest number of genes and SPLR has equally best specificity. The detailed models by these methods are listed in Appendix C.

Table 14: Summary of all methods on the reduced cancer data

Methods	# Gene	# Para	q	Total	Pred (%)	Sens (%)	Spec (%)
GLPS	10	**6**	**2**	**18**	**98.2**	**86.7**	**100**
GLPS3	**9**	7	3	19	96.5	80.0	99.0
Logic	10	**6**	4	23	95.6	80.0	98.0
SPLR	10	11	3	24	96.5	73.3	**100**

4.4.2 Breast Tumors with Survival Time

The survival time of breast cancer patients depends on many factors, such as grade, stage and oestrogen-receptor status. In this section we study the possible genetic effects on the survival time. we denote patients that died before 5 years as cases and patients that were alive after 10 years as controls. Patients with a censored death time less than 10 years and patients that died between 5 and 10 years are excluded. This gives us a total of 243 patients, among which 80 are

cases. The incidence rate is 80/243 = 32.9%. Similar to the previous section we used a screen step, which selected 592 genes and denote this set the reduced breast tumor data.

Table 15 shows the model fitted by GLPS. There are one main effect and four size two interactions. The prediction accuracy of this model is 77.1% with a sensitivity of 53.3% and a specificity of 87.1%.

Table 15: The model by GLPS on the reduced breast tumor data

Genes			coefficients
constant			3.206398
201578_at			-1.590480
202761_s_at	×	203156_at	-2.006628
202870_s_at	×	204313_s_at	2.052567
203010_at	×	203929_s_at	-1.880384
204041_at	×	209721_s_at	-1.889956

The models fitted by GLPS3, Logic regression and SPLR along with the one by GLPS are summarized in Table 16. The detailed models by these methods are also listed in Appendix C. Among the seven criteria we present in Table 16, GLPS does the best in four and Logic regression in three. Logic regression picked a size eight interaction which doesn't make much sense. Note that the incidence rate is 32.9% so we will get an accuracy of 67.1% if we predict all to be controls. The performance of Logic regression and SPLR is not quite different from this. GLPS is the only one that stands out. We do not get as high prediction accuracy as in the cancer data set, which implies that there are

other factors other than genes that are important to the survival time of breast cancer patients. One of them is grade. We have the grade information so we added it to the model along with all genes. However, it did not show up because it's highly correlated with some genes.

Table 16: Summary of all methods on the reduced breast cancer data

Methods	# Gene	# Para	q	Total	Pred (%)	Sens (%)	Spec (%)
GLPS	9	6	**2**	**17**	**77.1**	53.3	**87.1**
GLPS3	10	6	3	19	70.8	62.5	75.0
Logic	**8**	**2**	8	18	68.8	**75.0**	65.6
SPLR	10	11	3	24	68.8	43.8	81.3

4.5 Discussion

LASSO-Patternsearch Algorithm is a global method that handles binary data very well. However, it doesn't work when the number of parameters is huge. Traditional methods for small n large p problem deals with each variable separately. Grouped LASSO-Patternsearch Algorithm proposed in this chapter is in between a global method and a uni-variate method. It breaks the big problem into smaller ones that are much bigger than those in a uni-variate analysis. We show through simulations that GLPS is better than competing methods in selecting the right variables and controlling the amount of noise. We also show that GLPS gives smaller models with better prediction accuracy through two gene expression data sets.

Unlike LPS, we use two tuning parameters in GLPS and GLPS3. The tuning criterion is slightly different too. We impose a penalty on the difference between the number of main effects and the number of interactions for GLPS and a penalty on the difference among the numbers of main effects, size two interactions and size three interactions for GLPS3. These penalties prevent the extreme cases where only main effects or interactions come up in the LASSO step. The extreme cases appear way too often with the original criterion. If an extreme case is the truth, the LASSO step will select more noise but the parametric step will still get the right model.

Chapter 5

Concluding Remarks

In any problem where there are a large number of highly interacting predictor variables that are or can be reduced to dichotomous values, LPS and GLPS can be profitably used. If the "risky" direction (with respect to the outcome of interest) is known for all or almost all of the variables, the results are readily interpretable. If the risky direction is coded correctly for all of the variables, the fitted model can be expected to be sparser than that for any other coding. However, if a small number of risky variables are coded in the "wrong" way, this usually can be detected. The methods can be used as a preprocessor when there are very many continuous variables in contention, to reduce the number of variables for more detailed nonparametric analysis.

LPS, using the algorithm of Section 2.3 is efficient. On an Intel Quad-core 2.66 GHz machine with 8GB memory the LPS Steps 1 and 2 can do 90,000 basis functions in 4.5 minutes. On an AMD Dual-Core 2.8 GHz machine with 64 GB memory the algorithm did LPS with 403,594 basis functions in 4.5 minutes. It can do 2,000,000 basis functions in 1.25 hours. On the same AMD machine, problems of the size of the myopia data (128 unknowns) can be solved in a few

seconds and the GAW 15 problem data (10371 unknowns) was solved in 1.5 minutes.

Many generalizations are available. Two classes of models where the LASSO-Patternsearch approach can be expected to be useful are the multicategory end points model in (Lin 1998)(Wahba 2002), where an estimate is desired of the probability of being in class k when there are K possible outcomes; another is the multiple correlated endpoints model in (Gao, Wahba, Klein & Klein 2001). In this latter model, the correlation structure of the multiple endpoints can be of interest. Another generalization allows the coefficients c_ℓ to depend on other variables; however, the penalty functional must involve ℓ_1 penalties if it is desired to have a convex optimization problem with good sparsity properties with respect to the patterns. In studies with environmental as well as genomic data selected interactions between SNP patterns and continuous covariates can be examined (Zhang et al. 2004): the numerical algorithm can be used on large collections of basis functions that induce a reasonable design matrix, for example collections including splines, wavelets or radial basis functions.

Appendix A

The BGACV Score

We denote the estimated logit function by $f_\lambda(\cdot)$ and define $f_{\lambda i} = f_\lambda(x(i))$, $p_{\lambda i} = \frac{e^{f_{\lambda i}}}{1+e^{f_{\lambda i}}}$ $\sigma^2_{\lambda i} = \frac{e^{f_{\lambda i}}}{(1+e^{f_{\lambda i}})^2}$ for $i = 1, \cdots, n$. Now define

$$OBS(\lambda) = \frac{1}{n}\sum_{i=1}^{n}[-y_i f_{\lambda i} + \log(1 + e^{f_{\lambda i}})]. \tag{A.1}$$

From (Xiang & Wahba 1996)(Lin, Wahba, Xiang, Gao, Klein & Klein 2000) the leave-one-out CV is

$$\begin{aligned}
CV(\lambda) &= \frac{1}{n}\sum_{i=1}^{n}[-y_i f_{\lambda i}^{[-i]} + b(f_{\lambda i})] \\
&= OBS(\lambda) + \frac{1}{n}\sum_{i=1}^{n}[y_i(y_i - p_{\lambda i}^{[-i]})]\left[\frac{f_{\lambda i} - f_{\lambda i}^{[-i]}}{y_i - p_{\lambda i}^{[-i]}}\right] \\
&= OBS(\lambda) + \frac{1}{n}\sum_{i=1}^{n} y_i(y_i - p_{\lambda i})\left[\frac{f_{\lambda i} - f_{\lambda i}^{[-i]}}{y_i - p_{\lambda i}^{[-i]}}\right] / \left[1 - \left(\frac{p_{\lambda i} - p_{\lambda i}^{[-i]}}{f_{\lambda i} - f_{\lambda i}^{[-i]}}\right)\left(\frac{f_{\lambda i} - f_{\lambda i}^{[-i]}}{y_i - p_{\lambda i}^{[-i]}}\right)\right] \\
&\approx OBS(\lambda) + \frac{1}{n}\sum_{i=1}^{n} y_i(y_i - p_{\lambda i})\left[\frac{f_{\lambda i} - f_{\lambda i}^{[-i]}}{y_i - p_{\lambda i}^{[-i]}}\right] / \left[1 - \sigma^2_{\lambda i}\left(\frac{f_{\lambda i} - f_{\lambda i}^{[-i]}}{y_i - p_{\lambda i}^{[-i]}}\right)\right].
\end{aligned} \tag{A.2}$$

Here $\sigma^2_{\lambda i} = \sigma^2(f_{\lambda i})$ and the approximation in (A.2) follows upon recalling that $\frac{\partial p}{\partial f} = \sigma^2$.

Denote the objective function in (2.1) - (2.4) by $I_\lambda(y, c)$, let $B_{ij} = B_j(x(i))$ be the entries of the design matrix B, and for ease of notation denote $\mu = c_{N_B}$. Then the objective function can be written

$$I_\lambda(y, c) = \frac{1}{n}\sum_{i=1}^{n}[-y_i \sum_{j=1}^{N_B} c_j B_{ij} + \log(1 + e^{(\sum_{j=1}^{N_B} c_j B_{ij})})] + \lambda \sum_{j=1}^{N_B-1} |c_j|. \qquad \text{(A.3)}$$

Denote the minimizer of (A.3) by c_λ. We know that the l_1 penalty produces sparse solutions. Without loss of generality, we assume that the first s components of c_λ are nonzero. When there is a small perturbation ϵ on the response, we denote the minimizer of $I_\lambda(c, y + \epsilon)$ by c_λ^ϵ. The 0's in the solutions are robust against a small perturbation in the response. That is, when ϵ is small enough, the 0 elements will stay at 0. This can be seen by looking at the KKT conditions when minimizing (A.3). Therefore, the first s components of c_λ^ϵ are nonzero and the rest are zero. For simplicity, we denote the first s components of c by c^* and the first s columns of the design matrix B by B^*. Then let f_λ^y be the column vector with i entry $f_\lambda(x(i))$ based on data y, and let $f_\lambda^{y+\epsilon}$ be the same column vector based on data $y + \epsilon$.

$$f_\lambda^{y+\epsilon} - f_\lambda^y = B(c_\lambda^\epsilon - c_\lambda) = B^*(c_\lambda^{\epsilon *} - c_\lambda^*). \qquad \text{(A.4)}$$

Now we take the first-order Taylor expansion of $\frac{\partial I_\lambda}{\partial c^*}$:

$$\left[\frac{\partial I_\lambda}{\partial c^*}\right]_{(c_\lambda^\epsilon, y+\epsilon)} \approx \left[\frac{\partial I_\lambda}{\partial c^*}\right]_{(c_\lambda, y)} + \left[\frac{\partial^2 I_\lambda}{\partial c^* \partial c^{*\prime}}\right]_{(c_\lambda, y)} (c_\lambda^{\epsilon *} - c_\lambda^*)) + \left[\frac{\partial^2 I_\lambda}{\partial c^* \partial y^\prime}\right]_{(c_\lambda, y)} (y + \epsilon - y). \tag{A.5}$$

Define

$$U \equiv n \left[\frac{\partial^2 I_\lambda}{\partial c^* \partial c^{*\prime}}\right]_{(c_\lambda, y)} = B^{*\prime} \text{diag}([\frac{e^{f_{\lambda 1}}}{(1+e^{f_{\lambda 1}})^2}, \cdots, \frac{e^{f_{\lambda n}}}{(1+e^{f_{\lambda n}})^2}]) B^* = B*^\prime W B*, \text{ say,}$$

and

$$V \equiv -n \left[\frac{\partial^2 I_\lambda}{\partial c^* \partial y^\prime}\right]_{(c_\lambda, y)} = B^{*\prime}.$$

By the first-order conditions, the left-hand side and the first term of the right-hand side of (A.5) are zero. So we have

$$U(c_\lambda^{\epsilon *} - c_\lambda^*)) \approx V\epsilon. \tag{A.6}$$

Combine (A.4) and (A.6) we have $f_\lambda^{y+\epsilon} - f_\lambda^y \approx H\epsilon$, where

$$H \equiv B^* U^{-1} V \equiv B^* U^{-1} B^{*\prime}. \tag{A.7}$$

Now let ϵ be $\epsilon = (0, \cdots, y_i - p_{\lambda i}^{[-i]}, \cdots, 0)^\prime$; then $f_\lambda^{y+\epsilon} - f_\lambda^y \approx H_i \epsilon_i$, where $\epsilon_i = y_i - p_{\lambda i}^{[-i]}$ and H_i is the ith column of H. By the Leave-Out-One Lemma (stated

below), ${f_\lambda}^{[-i]} = {f_\lambda}^{y+\epsilon}$. Therefore

$$\frac{f_{\lambda i} - f_{\lambda i}^{[-i]}}{y_i - p_{\lambda i}^{[-i]}} = \frac{f_{\lambda i}^{y+\epsilon} - f_{\lambda i}}{y_i - p_{\lambda i}^{[-i]}} \approx h_{ii}$$

where h_{ii} is the iith entry of H. From the right hand side of (A.2), the approximate CV score is

$$CV(\lambda) \approx \frac{1}{n}\sum_{i=1}^{n}[-y_i f_{\lambda i} + \log(1 + e^{f_{\lambda i}})] + \frac{1}{n}\sum_{i=1}^{n} h_{ii}\frac{y_i(y_i - p_{\lambda i})}{(1 - \sigma_{\lambda i}^2 h_{ii})}. \tag{A.8}$$

The $GACV$ score is obtained from the approximate CV score in (A.8) by replacing h_{ii} by $\frac{1}{n}tr(H)$ and $\sigma_{\lambda i}^2 h_{ii}$ by $\frac{1}{n}tr(WH)$. It is not hard to see that $tr(WH) = trW^{1/2}HW^{1/2} = s \equiv N_{B_0}$, the number of basis functions in the model, giving

$$GACV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}[-y_i f_{\lambda i} + \log(1 + e^{f_{\lambda i}})] + \frac{1}{n}\mathrm{tr}H\frac{\sum_{i=1}^{n} y_i(y_i - p_{\lambda i})}{(n - N_{B_0})}. \tag{A.9}$$

Adding the weight $\frac{1}{2}\log n$ to the "optimism" part of the GACV score, we obtain the B-type GACV (BGACV):

$$BGACV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}[-y_i f_{\lambda i} + \log(1 + e^{f_{\lambda i}})] + \frac{1}{n}\frac{\log n}{2}\mathrm{tr}H\frac{\sum_{i=1}^{n} y_i(y_i - p_{\lambda i})}{(n - N_{B_0})}. \tag{A.10}$$

Lemma A.1 *(Leave-Out-One Lemma)*

Let the objective function $I_\lambda(y, f)$ be defined as before. Let $f_\lambda^{[-i]}$ be the minimizer of $I_\lambda(y, f)$ with the ith observation omitted and let $p_\lambda^{[-i]}$ be the corresponding probability. For any real number ν, we define the vector $z = (y_1, \cdots, y_{i-1}, \nu, y_{i+1}, \cdots, y_n)'$. Let $h_\lambda(i, \nu, \cdot)$ be the minimizer of $I_\lambda(z, f)$; then $h_\lambda(i, p_\lambda^{[-i]}, \cdot) = f_\lambda^{[-i]}(\cdot)$.

The proof of lemma $A.1$ is quite simple and very similar to the proof of the Leave-Out-One-Lemma in (Zhang et al. 2004) so we will omit it here.

We remark that in LPS we have employed the BGACV criterion twice as a stringent model selector under the assumption that the true or the desired model is sparse. Simulation experiments (not shown) suggest that the GACV criterion is preferable if the true model is not sparse and/or the signal is weak. The GACV and the BGACV selections probably bracket the region of interest of λ in most applications.

We are also interested in how BGACV might be compared to the more familiar $BIC = -\log \text{likelihood} + \frac{\log n}{2} df$. where df is the degrees of freedom in the case of Bernoulli data. The short answer to this question is that an exact expression for df does not, in the usual sense, exist in the case of Bernoulli data. Thus, only a hopefully good approximation to something that plays the role of df in the Bernoulli case can be found. This argument, which is independent of the nature of the estimate f_λ of f, is found in Section 2 of (Lin et al. 2000). We sketch the main idea. Let $KL(\lambda) = KL(f, f_\lambda)$ be the Kullback-Liebler distance between the distribution with the true but unknown canonical link f and the

distribution with link f_λ and let

$$CKL(\lambda) = KL(f, f_\lambda) - \frac{1}{n}[\sum_{i=1}^{n} E_\mu y_i f_i - b(f_i)] \equiv \sum_{i=1}^{n} -\mu_i f_{\lambda i} + b(f_{\lambda i}) \quad \text{(A.11)}$$

be the comparative Kullback-Liebler distance. The goal is to find an unbiased estimate of $CKL(\lambda)$ as a function of λ, which will then be minimized to estimate the λ minimizing the true but unknown CKL. Letting $OBS(\lambda) = \frac{1}{n}\sum_{i=1}^{n} -y_i f_{\lambda i} + b(f_{\lambda i})$ we can write $CKL(\lambda) = OBS(\lambda) + D(\lambda)$. where $D(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_i)f_{\lambda i}$. Then $E_\mu D(\lambda) = \frac{1}{n}\sum_{i=1}^{n} E_\mu(y_i - \mu_i)f_{\lambda i}$. Ye and Wong (1997) show, in exponential families, for *any* estimate f_λ of f

$$\frac{1}{n}\sum_{i=1}^{n} E_{\mu_i}(y_i - \mu_i)f_{\lambda i} = \frac{1}{n}\sum_{i=1}^{n} \sigma_i^2 \frac{\partial}{\partial \mu_i} E_{\mu_i}(f_{\lambda i}). \quad \text{(A.12)}$$

Here $E_{\mu_i}(f_{\lambda i})$ is the expectation with respect to y_i conditional on the $y_j, j \neq i$ being fixed. (Their proof is reproduced in (Lin et al. 2000).) Ye and Wong call n times the right hand side of (A.12) the generalized degrees of freedom (GDF). and it does indeed reduce to the usual trace of the influence matrix in the case of Gaussian data with quadratic penalties. Unbiased estimates of the GDF can be found for Poisson, Gamma, Binomial distribution taking on three or more values, and other distributions, using the results in (Hudson 1978) but Ye and Wong show that *no unbiased estimate of the GDF in the Bernoulli case exists.* See (Wong 2006). Thus, in the absence of a bona fide unbiased risk method of estimating the $CKL(\lambda)$ the alternative GACV based on leave-one-out to target

the CKL has been proposed. Thus, in this paper $\frac{1}{n}\mathrm{tr}H\frac{\sum_{i=1}^{n} y_i(y_i-p_{si})}{(n-N_{B_0})} = \hat{D}(\lambda)$, say, plays the role of *df*. Since no exact unbiased estimate for *df* exists, the issue of the accuracy of the approximations in obtaining $\hat{D}$ reduces to the issue of to what extent the minimizer of $GACV(\lambda)$ is a good estimate of the minimizer of the (unobservable) $CKL(\lambda)$. The GACV for Bernoulli data was first proposed in (Xiang & Wahba 1996) for RKHS (quadratic) penalty functionals, where simulation results demonstrated the accuracy of this approximation. Further excellent favorable results for a randomized version of the GACV with RKHS penalties were presented in (Lin et al. 2000). In (Zhang et al. 2004) the GACV was derived for Bernoulli data with l_1 penalties with a nontrivial null space, and favorable results for the randomized version were obtained. The derivation was rather complicated, and a simplified derivation as well as a simpler expression for the result which is possible in the present context are presented above. A recent work involving the LASSO in the Bernoulli case with l_1 penalty uses a tuning set to choose the smoothing parameters. SPLR (Park & Hastie 2008) uses $tr(B*'WB*+\lambda I*)^{-1}(B*'WB*)$, where I is the diagonal matrix with all 1's except in the position of the model constant, as their proxy for df in their BIC-like criteria for model selection after fixing λ. In the light of the Ye and Wong result it is no surprise that an exact definition of *df* in this case cannot be found in the literature.

Appendix B

Effect of Coding Flips

Proposition: Let $f(x) = \mu + \sum c_{j_1 j_2 .. j_r} B_{j_1 j_2 .. j_r}(x)$ with all $c_{j_1 j_2 .. j_r}$ which appear in the sum strictly positive. If $x_j \to 1 - x_j$ for $j \in$ some subset of $\{1, 2, ..., p\}$ such that at least one x_j appears in f, then the resulting representation has at least one negative coefficient and at least as many terms as f. This follows from the

Lemma: Let $g_k(x)$ be the function obtained from f by transforming $x_j \to 1 - x_j, 1 \leq j \leq k$. Then the coefficient of $B_{j_1 j_2 .. j_r}(x)$ in $g_k(x)$ is

$$(-1)^{|\{j_1,..,j_r\} \cap \{1,...,k\}|} \sum_{\{j_1,\cdots,j_r\} \subseteq T \subseteq \{j_1,\cdots,j_r\} \cup \{1,\cdots k,\}} c_T$$

where $|\cdot|$ means number of entries.

Appendix C

More Results on Gene Expression Bar Code Data

Table 17: The model by GLPS3 on the reduced cancer data

Genes					coefficients
constant					-6.493502
200770_s_at	×	202408_s_at	×	212543_at	4.027583
200770_s_at	×	202740_at	×	208407_s_at	2.621721
201818_at	×	202740_at	×	208407_s_at	1.087071
201818_at	×	209771_x_at	×	212543_at	2.317507
201818_at	×	209771_x_at	×	221927_s_at	2.585422
202740_at	×	218531_at	×	221927_s_at	2.443283

The model fitted by Logic Regression on the reduced breast tumor data is

$$\begin{aligned}\text{logit}(x) &= 1.40 - 4.22 \times (((212485\_at \vee 213645\_at) \wedge (204094\_s\_at \vee 201752\_s\_at)) \\ &\quad \wedge((203362\_s\_at^c \wedge 34858\_at) \vee (203104\_at \wedge 205345\_at^c)))\end{aligned}$$

where $gene^c = 1 - gene$, $\vee$ means "or" and $\wedge$ means "and". In this model, only one tree is fitted and this tree has 8 leaves. It would be very complicated to

Table 18: The model by Logic Regression on the reduced cancer data

Genes	coefficients
constant	-7.89
203108_at	-5.33
220751_s_at	-5.33
208161_s_at	4.56
205157_s_at×215177_s_at	6.48
216973_s_at×219684_at	6.48
203108_at×220751_s_at	5.33
205597_at×212194_s_at	4.56
201115_at×202489_s_at×212335_at	7.91
205597_at×208161_s_at×212194_s_at	-4.56
205157_s_at×215177_s_at×216973_s_at×219684_at	-6.48

Table 19: The model by SPLR on the reduced cancer data

Genes					coefficients
constant					-4.251617
219121_s_at					1.4442
221529_s_at					1.1393
202820_at					1.4263
211429_s_at	×	221529_s_at			0.4959
210541_s_at	×	219121_s_at			1.3055
201818_at	×	221529_s_at			1.5010
209796_s_at	×	219121_s_at			1.3890
208999_at	×	211429_s_at	×	221529_s_at	1.1865
208941_s_at	×	210541_s_at	×	219121_s_at	1.0551
201818_at	×	209771_x_at	×	221529_s_at	1.3634

Table 20: The model by GLPS3 on the reduced breast tumor data

Genes					coefficients
constant					0.5187764
201578_at					-1.4915942
219192_at					1.5793844
209391_at	×	213671_s_at			2.5934726
201752_s_at	×	209256_s_at	×	212914_at	-2.0885824
203394_s_at	×	203439_s_at	×	222200_s_at	-2.3821726

write the tree in terms of our basis functions ("and" is used, not "or").

Table 21: The model by SPLR on the reduced breast tumor data

Genes					coefficients
constant					0.1094
203010_at					-0.3689083
204021_s_at					-0.3407206
219192_at					-0.5187957
202587_s_at					0.4959507
201578_at	×	203010_at			-0.5454934
212075_s_at	×	219192_at			0.3529248
204021_s_at	×	210042_s_at			0.5808981
201946_s_at	×	219192_at			0.5444631
201578_at	×	203010_at	×	209500_x_at	-0.6153894
204021_s_at	×	210042_s_at	×	218396_at	-0.6143376

Appendix D

Matlab Code for LPS

```
function [ff, beta, loglike, Tlambda, numNonzeros, iterations]...
    =  cv(lambda_input,beta_init,method,output)
% method = 1 bgacv, method = 2 gacv
global lambda_global;
global fBasis;
global Y_global;
[sampsize nbeta] = size(fBasis);
lambda_global=lambda_input;


%-----------------------------------------------------------------
% Define some algorithmic parameters
%-----------------------------------------------------------------
maxiter = 1000;
tol = 1.e-6;
alphaMax = 1.e12;
iter = 0;
```

```
useNewton=1;      % set to 1 if want to have second-order scaling
                  % set to 0 if always use a gradient projection
two_metric_threshold = min(500,nbeta);
                  % ordinary GP is used above this threshhold
beta = beta_init;
Tlambda = Functions(beta);
alpha = 1.e1;
                  % set initial value of the damping parameter
beta_step=zeros(nbeta,1);
                  % initialize beta_step;
loop_on=1;        % set to zero to stop the loop


%-----------------------------------------------------------------
% parameters to control randomized evaluation:
% sigma = fraction of random gradient components to evaluate for
%         zero components of beta. Set sigma=1 to get full
%         gradient evaluation.
% evalFullGrad = flag indicating whether full gradient should be
%                evaluated.Turned on when gpnorm drops below tol.
%-----------------------------------------------------------------
sigma = 0.01;
evalFullGrad = 0;
```

```
if output == 1
    nz=length(find(beta_init(1:nbeta)~=0));
    fprintf(' Initial point: nz=%d, f= %15.12g\n', nz, Tlambda);
end

%-----------------------------------------------------------------
% Start the loop of the algorithm
%-----------------------------------------------------------------
while (iter<maxiter) && (loop_on==1)
    iter = iter+1;
    % initialize beta_new to be the current beta (subsequently,
    % of its entries will be modified by the step)
    beta_new=beta;

    %-------------------------------------------------------------
    % evaluate the gradient for (i) the current nonzero indices
    % and (ii) a fraction sigma of the other indices.
    %-------------------------------------------------------------
    if evalFullGrad
      beta_inx=1:nbeta;
    else
```

```
  beta_nz = find(beta);
  beta_rand = find(rand(nbeta,1)<sigma);
  beta_inx = union(beta_nz,beta_rand);
end

Grad = zeros(nbeta,1);
Grad(beta_inx) = Gradient(beta, beta_inx);

%-------------------------------------------------------------
% Evaluate convergence test quantity: subgradient of minimal
% norm
%-------------------------------------------------------------
subGradMin = Grad - lambda_global*(beta<0) + ...
    lambda_global*(beta>0);
km = find(beta==0 & Grad<=0);
kp = find(beta==0 & Grad> 0);
subGradMin(km) = min(0,Grad(km)+lambda_global);
subGradMin(kp) = max(0,Grad(kp)-lambda_global);
subGradMin(nbeta) = Grad(nbeta);
              % gradient of the last component remains same
gpnorm = norm(subGradMin);
```

```
    %-------------------------------------------------------------
    % for interest, calculate the nonzero set for this beta
    %-------------------------------------------------------------
    InonZero0 = find(beta);
    InonZero0 = union(InonZero0, nbeta);
    InonZero0_complement = setdiff((1:nbeta),InonZero0)';
    numberNonZero0 = length(InonZero0);

    % print out information every NI-th iteration
    NI=10000;
    if (mod(iter,NI)==0) || (loop_on==0)
    fprintf('\n iter %d, gpnorm=%8.4g, nonzero=%d (%5.1f%%)',...
  iter, gpnorm, numberNonZero0, 100*numberNonZero0/nbeta);
    fprintf('function=%15.12g, alpha=%12.5e\n', Tlambda, alpha);
    end

    if gpnorm<tol
      if evalFullGrad==1
break             % break if convergence test is met
      else
evalFullGrad=1;   % evaluate full gradient on next iteration
      end
```

```
else
  evalFullGrad=0; % turn off full grad evaluation
end

%-------------------------------------------------------------
% find the Barzilai-Borwein parameter, which approximates the
% curvature of the Hessian along the latest step
%-------------------------------------------------------------
if iter>1 && norm(beta_step)~=0
  alphaBB = (Grad-Grad_old)'*beta_step/(beta_step'*beta_step);
  % reset alpha if it appears to be too small - that is, make
  % it at least equal to curvature of the actual Hessian
  % along the latest step. This should keep the first-order
  % step and reduced Newton step in the same ballpark.
  if alphaBB > alpha
      alpha = alphaBB;
  end
end


%-------------------------------------------------------------
% Calculate the first-order step
```

```
%-------------------------------------------------------------

betaGrad = Grad - alpha*beta;

kn = find( betaGrad< -lambda_global);

kp = find( betaGrad>  lambda_global);

beta_new(betaGrad>=-lambda_global & betaGrad<=...

    lambda_global) = 0.0;

beta_new(kn) = beta(kn) - (Grad(kn)+lambda_global)/alpha;

beta_new(kp) = beta(kp) - (Grad(kp)-lambda_global)/alpha;

beta_new(nbeta) = beta(nbeta) - Grad(nbeta)/alpha;

beta_step_first = beta_new-beta;


%-------------------------------------------------------------

% evaluate the function at the first-order point. The Newton

% step, if tried, has to do better than Tlambda_first in order

% to be accepted.

%-------------------------------------------------------------

Tlambda_first = Functions(beta+beta_step_first);

InonZero=find(beta_new);

InonZero=[InonZero;nbeta];

InonZero_complement = setdiff(1:nbeta,InonZero)';

InonZero_diff = setxor(InonZero0,InonZero);

numNonZero_diff = length(InonZero_diff);
```

```
%-------------------------------------------------------------
% Calculate the second-order component
%-------------------------------------------------------------
newtonSuccess=0;
if (useNewton==1) && (length(InonZero)<=two_metric_threshold)
    reducedGradient = Grad - lambda_global*(beta_new<0) + ...
        lambda_global*(beta_new>0);
    reducedGradient(nbeta) = Grad(nbeta);
    reducedGradient = reducedGradient(InonZero);
    partialHessian  = Hessian(beta, InonZero);
    scale_factor = mean(diag(partialHessian));
    damping = min(scale_factor, 1.0*gpnorm);
    beta_step(InonZero) = -(partialHessian + ...
damping*eye(length(InonZero))) \ reducedGradient;
    beta_step(InonZero_complement) = beta_step_first...
        (InonZero_complement);

    %-------------------------------------------------------
    % Do a kind of line search
    %-------------------------------------------------------
    newtonStepLength=1;
```

```
beta_newton=zeros(nbeta,1);
beta_newton(InonZero) = beta(InonZero) + ...
    newtonStepLength*beta_step(InonZero);
          % take the line-search Newton step
          % in nonzero components;
beta_newton(InonZero_complement)=0;
          % set to zero those components that appear to
          % be zero in the first-order subproblem
iCrossings=(beta_newton.*beta<0);
iCrossings(nbeta)=0;
numCrossings=sum(iCrossings);
          % number of kinks that are crossed by this step

%-----------------------------------------------------------
% Evaluate function and check for decrease over the step
% obtained in the first-morder model
%-----------------------------------------------------------
Tlambda_new = Functions(beta_newton);
if Tlambda_new< min(Tlambda_first, Tlambda)
    newtonSuccess=1;
end
```

```
%-------------------------------------------------------
% If kinks were crossed, rescale the step to stop at the
% first kink evaluate function and check for decrease
% over the step from the first-order model
%-------------------------------------------------------
if numCrossings>0 && newtonSuccess==0
    findCrossings=find(iCrossings==1);
    newtonStepLength = min( abs(beta(findCrossings))./...
        (abs(beta(findCrossings)-beta_newton...
        (findCrossings))));
    beta_newton(InonZero) = beta(InonZero)+...
        newtonStepLength*beta_step(InonZero);
    beta_newton(InonZero_complement)=0;
    Tlambda_new = Functions(beta_newton);
    if Tlambda_new< min(Tlambda_first, Tlambda)
        newtonSuccess=1;
    end
end

%-------------------------------------------------------
% End of Newton segment. If a good step was found in this
% segment, take it. Otherwise try some alternatives.
```

```
%-------------------------------------------------------


if newtonSuccess
    beta_new = beta_newton;
    if Tlambda_first < Tlambda
        alpha = 0.8*alpha;
    else
        alpha = 2*alpha;
        if alpha > alphaMax
            loop_on = 0;
        end
    end
else      % try just the first order step
    if Tlambda_first < Tlambda
        beta_step = beta_step_first;
        beta_new = beta + beta_step;
        alpha = 0.8*alpha;
    else
        beta_new = beta;
        alpha = 2*alpha;
        if alpha > alphaMax
            loop_on = 0;
```

```
            end
        end
    end % newtonSuccess
else          % useNewton = 0 or two metric threshold
              % exceeded, just try the first-order step
    beta_step = beta_step_first;
    if Tlambda_first < Tlambda
        beta_new = beta + beta_step;
        alpha = 0.8*alpha;
    else
        beta_new = beta;
        alpha = 2*alpha;
        if alpha > alphaMax
            loop_on = 0;
        end
    end
end
Tlambda_new = Functions(beta_new);   % reevaluate function


%-------------------------------------------------------------
% Update everything for the next iteration
%-------------------------------------------------------------
```

```
    Grad_old = Grad;
    beta_step = beta_new - beta;
    beta = beta_new;
    Tlambda = Tlambda_new;
end               % end the algorithm

[Tlambda, loglike] = Functions(beta); % reevaluate final function

%-----------------------------------------------------------------
% Calculate GACV or BGACV score
%-----------------------------------------------------------------
logit=fBasis*beta;
prob=exp(logit)./(1+exp(logit));
OBS=-1/sampsize*sum(Y_global.*log(prob)+(1-Y_global).*log(1-prob));
W=diag(prob./(1+exp(logit)));
nzindex=find(abs(beta)>1e-6);
numNonzeros = length(nzindex);
iterations = iter;
U = transpose(fBasis(:,nzindex))*W*fBasis(:,nzindex);
[EV EU] = eig(U);                        % use generalized inverse
eu = diag(EU);
smeig_ind = find(abs(eu)<1e-10);
```

```
inv_eu(smeig_ind) = 0;
inv_eu(setdiff(1:length(eu),smeig_ind)) = 1./...
    eu(setdiff(1:length(eu),smeig_ind));
H=fBasis(:,nzindex)*(EV*diag(inv_eu)*EV')*...
    transpose(fBasis(:,nzindex));
if method == 2                          % GACV
    ff=OBS+1/sampsize*sum(Y_global.*(Y_global-prob))*trace(H)/...
        (sampsize-length(nzindex));
elseif method == 1                      % BGACV
    ff=OBS+log(sampsize)/sampsize/2*sum(Y_global.*...
        (Y_global-prob))*trace(H)/(sampsize-length(nzindex));
end
return
```

Bibliography

Breiman, L. (2001), 'Random forests', *Machine Learning* **45**, 5–32.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984), *Classification and Regression Trees*, Wadsworth.

Chan, K.-Y. & Loh, W.-Y. (2004), 'Lotus: An algorithm for building accurate and comprehensible logistic regression trees', *Journal of Computational and Graphical Statistics* **13**, 826–852.

Chen, S., Donoho, D. & Saunders, M. (1998), 'Atomic decomposition by basis pursuit', *SIAM J. Sci. Comput.* **20**, 33–61.

Cordell, H. e. a. (2007), 'Genetic analysis workshop 15: gene expression analysis and approaches to detecting multiple functional loci', *BMC Proceedings* **S1**, 1–4.

Craven, P. & Wahba, G. (1979), 'Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation', *Numer. Math.* **31**, 377–403.

Efron, B., Hastie, T., Johnstone, I. & Tibshirani, R. (2004), 'Least angle regression', *Ann. Statist.* **32**, 407–499.

Galan, P., Viteri, F. & et al, S. B. (2005), 'Serum concentrations of beta-carotene, vitamins C and E, zinc and selenium are influenced by sex, age, diet, smoking status, alcohol consumption and corpulence in a general French adult population', *Eur. J. Clin. Nutr.* **59**, 1181–90.

Gao, F., Wahba, G., Klein, R. & Klein, B. (2001), 'Smoothing spline ANOVA for multivariate Bernoulli observations, with applications to ophthalmology data, with discussion', *J. Amer. Statist. Assoc.* **96**, 127–160.

George, E. (2000), 'The variable selection problem', *J. Amer. Statist. Assoc.* **95**, 1304–1308.

Golub, G., Heath, M. & Wahba, G. (1979), 'Generalized cross validation as a method for choosing a good ridge parameter', *Technometrics* **21**, 215–224.

Gunn, S. & Kandola, J. (2002), 'Structural modelling with sparse kernels', *Machine Learning* **48**, 137–163.

Haughton, D. (1988), 'On the choice of a model to fit data from an exponential familly', *Ann. Statist.* **16**, 342–355.

Hudson, M. (1978), 'A natural identity for exponential families with applications in multiparameter estimation', *Ann. Statist.* **6**, 473–484.

Irizarry, R. & Hobbs, B. (2003), 'Exploration, normalization, and summaries of high density oligonucleotide array probe level data', *Biostatistics* **4**, 249–64.

Klein, R., Klein, B. E. K., Linton, K. & DeMets, D. (1991), 'The Beaver Dam eye study: Visual acuity', *Ophthalmology* **98**, 1310–1315.

Knight, K. & Fu, W. (2000), 'Asymptotics for LASSO-type estimators', *Ann. Statist* **28**, 1356–1378.

Lee, K., Klein, B. & Klein, R. (1999), 'Changes in refractive error over a 5-year interval in the Beaver Dam Eye Study', *Investigative Ophthalmalogy & Visual Science* **40**, 1645–1649.

Lee, K., Klein, B., Klein, R. & Wong, T. (2002), 'Changes in refraction over 10 years in an adult population: the Beaver Dam Eye Study', *Investigative Ophthalmology Visual Science* **43**, 2566–71.

Lee, Y., Kim, Y., Lee, S. & Koo, J. (2006), 'Structured multicategory support vector machines with analysis of variance decomposition', *Biometrika* **93**, 555–571.

Leng, C., Lin, Y. & Wahba, G. (2006), 'A note on the LASSO and related procedures in model selection', *Statistica Sinica* **16**, 1273–1284.

Li, K. C. (1986), 'Asymptotic optimality of C_L and generalized cross validation in ridge regression with application to spline smoothing', *Ann. Statist.* **14**, 1101–1112.

Lin, X. (1998), Smoothing spline analysis of variance for polychotomous response data, Technical Report 1003, PhD thesis, Department of Statistics, University of Wisconsin, Madison WI. Available via G. Wahba's website.

Lin, X., Wahba, G., Xiang, D., Gao, F., Klein, R. & Klein, B. (2000), 'Smoothing spline ANOVA models for large data sets with Bernoulli observations and the randomized GACV', *Ann. Statist.* **28**, 1570–1600.

Miller, M., Lind, G., Li, N. & Jang, S. (2007), 'Genetic analysis workshop 15: Simulation of a complex genetic model for rheumatoid arthritis in nuclear families including a dense snp map with linkage disequilibrium between marker loci and trait loci', *BMC Proceedings* **1(Suppl 1)**, S4, 1–7.

Osborne, M., Presnell, B. & Turlach, B. (2000), 'On the LASSO and its dual', *J. Comp. Graph. Stat.* **9**, 319–337.

Park, M. & Hastie, T. (2008), 'Penalized logistic regression for detecting gene interactions', *Biostatistics* **9**, 30–50.

Ruczinski, I., Kooperberg, C. & LeBlanc, M. (2003), 'Logic regression', *J. Computational and Graphical Statistics* **12**, 475–511.

Schwartz, D., Szymczak, S., Ziegler, A. & Konig, R. (2007), 'Picking single-nucleotide polymorphisms in forests', *BMCProceedings* **1(Suppl 1)**, S59, 1–5.

Seet, B., Wong, T., Tan, D., Saw, S., Balakrishnan, V., Lee, L. & Lim, A. (2001), 'Myopia in Singapore: taking a public health approach', *Br. J. Ophthalmol.* **85**, 521–526.

Shi, W., Lee, K. & Wahba, G. (2007), 'Detecting disease-causing genes by LASSO-patternsearch algorithm', *BMC Proceedings* **1(Suppl 1)**, S60, 1–5. PMCID: PM2367607.

Shi, W., Wahba, G., Wright, S., Lee, K., Klein, R. & Klein, B. (2006), LASSO-Patternsearch algorithm with application to ophthalmalogy data, Technical Report 1131, Department of Statistics, University of Wisconsin, Madison WI. Under revision.

Thompson, W. e. a. (2007), 'Rheumatoid arthritis association at 6q23', *Nature Genetics* **doi:10.1038/ng.2007.32**.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the LASSO', *J. Roy. Stat. Soc, B* **58**, 267–288.

Valdar, W., Solberg, L., Gauguier, D., Burnett, S., Klenerman, P., Cookson, W., Taylor, M., Rawlins, J., Mott, R. & Flint, J. (2006), 'Genome-wide genetic association of complex traits in heterogeneous stock mice', *Nature Genetics* **38**, 879–887.

Wahba, G. (2002), 'Soft and hard classification by reproducing kernel Hilbert space methods', *Proceedings of the National Academy of Sciences* **99**, 16524–16530. Open Source at www.pnas.org/content/99/26/16524.full.pdf+html.

Wahba, G., Lin, Y., Lee, Y. & Zhang, H. (2002), Optimal properties and adaptive tuning of standard and nonstandard support vector machines, *in* D. Denison, M. Hansen, C. Holmes, B. Mallick & B. Yu, eds, 'Nonlinear Estimation and Classification', Springer, pp. 129–148.

Wahba, G. & Wold, S. (1975), 'A completely automatic French curve', *Commun. Stat.* **4**, 1–17.

Whittaker, J. (1990), *Graphical Models in Applied Mathematical Multivariate Statistics*, Wiley.

Wong, W. (2006), Estimation of the loss of an estimate, *in* J. Fan & H. Koul, eds, 'Frontiers of Statistics', Imperial College Press, London, pp. 491–506.

Xiang, D. & Wahba, G. (1996), 'A generalized approximate cross validation for smoothing splines with non-Gaussian data', *Statistica Sinica* **6**, 675–692.

Zhang, H. & Lin, Y. (2006), 'Component selection and smoothing for nonparametric regression in exponential families', *Statistica Sinica* **16**, 1021–1042.

Zhang, H., Wahba, G., Lin, Y., Voelker, M., Ferris, M., Klein, R. & Klein, B.

(2004), 'Variable selection and model building via likelihood basis pursuit', *J. Amer. Statist. Assoc.* **99**, 659–672.

Zilliox, M. J. & Irizarry, R. A. (2007), 'A gene expression bar code for microarray data', *Nature Methods* **4**, 911–913.

Zou, H. & Hastie, T. (2005), 'Regularization and variable selection via the elastic net', *Journal of the Royal Statistical Society: Series B* **67**, 301–320.