이학박사 학위논문

A Modified Least Angle Regression Algorithm for Hierarchical Interaction

계층적 교호작용을 고려한 수정된 LARS 알고리즘 개발

2016년 2월

서울대학교 대학원

통계학과

김 우 성

A Modified Least Angle Regression Algorithm for Hierarchical Interaction

By

Woosung Kim

A Thesis
Submitted in fulfillment of the requirements
for the degree of
Doctor of Philosophy
in Statistics

Department of Statistics
College of Natural Sciences
Seoul National University
February, 2016

Abstract

Variable selection is important in high dimensional regression. Traditional variable selection methods such as stepwise selection are unstable which means that the set of the selected variables is sensitive according to the change of data sets. As an alternative to those methods, a series of sparse penalized methods are used for estimation and variable selection simultaneously. The full set of LASSO solutions can be calculated by a minor modification of the LARS algorithm.

In many important practical problems, the main effect alone may not be enough to capture the relationship between the response and predictors, and high-order interactions are often of interest to scientific researchers. In considering two-way interaction models with a large number of covariates, we often would like to determine a smaller subset that exhibits strong effects on the response variable have been suggested.

Considering all possible interactions, however, is almost impossible due to computational burden when the number of covariate is large. To resolve this problem, the heredity structure between the main and interaction effect can be considered, algorithms for LASSO with heredity structure. However these algorithms cannot be executed if the number of main effects

is large since still computational burden is large. To resolve this issue, we suggest a hierarchical LARS algorithm which can be parallelized easily with MPI.

The proposed hierarchical LARS is a modified version of LARS, but it is more faster than LARS and has comparable prediction accuracy. It can be scaled up since it is possible to be executed in parallel process. MPI is a well known parallel model and we suggest a MPI-version of hierarchical LARS.

Keywords: High dimensional regression, Two way interaction model, LASSO, LARS, Massage passing interface

Student Number: 2011 – 30088

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Variable selection is important on high dimensional models, which affects model interpretation and prediction accuracy. In high dimensional models, the sparse penalized method has received a great attention recently as an alternative method for classical subset selections. Traditional variable selection methods including stepwise and the best subset selection using information criterion such as Mallows' Cp by Mallows (1973), Akaike information criterion (AIC) by Akaike (1973) and Bayesian information criterion (BIC) by Schwarz et al. (1978) have various drawbacks; computationally intensive, unstable and difficult to draw sampling properties. See Breiman et al. (1996) for detail. In contrast, sparse penalized methods give stable estimators with automatic variable selection and hence resulting estimators perform well in prediction.

There are several sparse penalized methods such as least absolute

shrinkage and selection operator (LASSO) by Tibshirani (1996), smoothly clipped absolute deviation (SCAD) by Fan and Li (2001) and etc. When the true model is sparse, sparse penalized methods have desirable large sample properties such as selection consistency, the oracle property and optimal convergence rate.

In particular, LASSO (Tibshirani, 1996) has gained much attention in recent years. Least Angle Regression (LARS) by Efron et al. (2004) is a promising algorithm to solve the LASSO problem. It is related to the classic model-selection method known as forward selection described in Weisberg (2005). Forward stagewise is a much mores cautious version of the forward selection. It is known that the LARS algorithm is closely related forward stagewise algorithm and this was the original motivation for the LARS algorithm. Furthermore, the full set of LASSO solutions can be calculated by modifying the LARS algorithm and it is faster than quadratic program suggested in Tibshirani (1996).

In many important practical problems, the main effect alone may not be enough to capture the relationship between the response and predictors, and higher-order interactions are often of interest to scientific researchers. For example, many complex diseases, such as cancer, involve multiple genetic and environmental risk factors, and scientists are particularly interested in assessing gene-gene and gene-environment interactions. (Choi et al., 2010)

In considering a regression model with main effects and all possible two-way interaction effects, which is called the two-way interaction model, there is an important challenge; computational burden. We would like to have our model accurately predict the future data. Prediction accuracy

can often be improved by shrinking the regression coefficients. Shrinkage sacrifices unbiasedness to reduce the variance of the predicted value and hence improve the overall prediction accuracy. In addition, we want to have a model which is easily interpretable. Interpretability is often realized via variable selection. With a large number of variables (including both the main terms and the interaction terms), possibly larger than the number of observations, we often would like to determine a smaller subset that exhibits the strongest effects (Choi et al., 2010). To achieve good prediction accuracy and intepretability, we often use sparse penalized approaches where the estimator is obtained by minimizing the penalized sum of squared residuals with sparse penalties such as LASSO or SCAD. However, when we consider the all two-way interactions as well as the main effects, minimizing the penalized sum of squared residuals is computationally difficult in particular when the number of covariates is large.

To reduce computational burden, one way is to consider the heredity between the main and interaction effects. It is well established in practice among statisticians that when we fit a linear regression model with main and interaction effect, we allow an interaction into the model only if the corresponding main effects are also in the model. Such restrictions are known under various names, including "heredity", "marginality", and being "hierarchical well-formulated" (Chipman, 1996, Hamada and Wu, 1992, Nelder, 1977). In the words of Cox (1984), "Large component main effects are more likely to lead to appreciable interactions than small components. Also, the interactions corresponding to larger main effects may be in some sense of more practical importance." In other words, rather than looking at all possible interactions, it may be useful to focus our search

on those interactions that have large main effects (Bien et al., 2013).

To reflect the heredity constraint into sparse penalized approaches, Choi et al. (2010) and Bien et al. (2013) suggested algorithms for LASSO with special sparse penalties. However, there exists still an issue to overcome. If the number of main effects is very large, we need to fit a two-way interaction model with ultra high dimensional data since the number of total effects is proportional to the square of the number of main effects. In particular, if the number of main effects is so large than the data cannot be allocated in a single machine, the algorithms suggested by Choi et al. (2010) and Bien et al. (2013) cannot be executed. To resolve this issue, we have to consider algorithms being able to be executed in parallel process. There exist some parallel process techniques such as massage passing interface (MPI), HADOOP which can settle the lack of memories. Especially, MPI is widely used in Mathematics, Physics, Computational Science and so on because it is fast, stable and possible to be scaled up.

MPI is a message-passing library interface specification (Lusk et al., 2009). MPI addresses primarily the message-passing parallel programming model, in which data is moved from the address space of one process to that of another process through cooperative operations on each process. Extensions to the "classical" message-passing model are provided in collective operations, remote-memory access operations, dynamic process creation, and parallel I/O. MPI is a specification, not an implementation; there are multiple implementations of MPI. This specification is for a library interface; MPI is not a language, and all MPI operations are expressed as functions, subroutines, or methods, according to the appropriate language bindings which, for C and Fortran, are part of the MPI standard. The stan-

dard has been defined through an open process by a community of parallel computing vendors, computer scientists, and application developers. See more details in Lusk et al. (2009).

The main advantages of establishing a message-passing standard are portability and ease of use. In a distributed memory communication environment in which the higher level routines and/or abstractions are built upon lower level message-passing routines, the benefits of standardization are particularly apparent. Furthermore, the definition of a message-passing standard provides vendors with a clearly defined base set of routines that they can implement efficiently, or in some cases for which they can provide hardware support, thereby enhancing scalability. Simply stated, the goal of MPI is to develop a widely used standard for writing message-passing programs. As such the interface should establish a practical, portable, efficient, and flexible standard for message passing.

In this thesis, first we develop an algorithm similar to the LARS algorithm with considering hierarchy between the main effect and interaction effect to overcome scale up problem which we call the hierarchical LARS (HLARS). In LARS, we need to compute correlations between residuals and covariates and if we consider two-way interaction model, we have to calculate all of the correlations corresponding the main and interaction effects which require huge computational costs. To resolve this problem we propose a modified LARS algorithm which takes account of hierarchy between main and interaction effects. Second, we suggest a parallel version of the HLARS algorithm. For some cases, data containing interactions cannot be allocated in a single machine memory. We modify further the HLARS algorithm so that we can split the data into the memories of

multiple machines and calculate correlations in parallel.

The contributions of this thesis are as follows.

- We suggest the HLARS algorithm which is a modified version of LARS. Our algorithm is faster than its competitors and has comparable prediction accuracy.
- We extend further the HLARS algorithm to MPI environment to analyze very large data. Some modification of HLARS makes it possible to execute the algorithm in parallel processing. We implement our algorithm in the MPI environment.
- In numerical studies, by simulation, we show that HLARS yields comparable prediction accuracies to the other competing algorithms such as LARS and HLASSO. In addition, we illustrate by analyzing a real data set about SNP that HLARS can deal with very large data efficiently in the MPI environment.

1.2 Outline of the thesis

This thesis is organized as follows. In Chapter 2, we review the relevant penalized methods for two-way interaction linear regression models and parallel processing - massage passing interface. In Chapter 3, we develop the HLARS algorithm and modify it to very large data which cannot be allocated in a single machine. In Chapter 4, we present the results of analysis of simulated and real data sets. Concluding remarks follow in Chapter 5. 0

Chapter 2

Literature Review : Variable Selection Methods on High Dimensions and MPI

2.1 Sparse regularization methods

Let $(y_1, \mathbf{x}_1), \cdots, (y_n, \mathbf{x}_n)$ be n pairs of observations where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^p$ is a covariate vector and $y_i \in \mathcal{Y} \subseteq \mathbb{R}$ is a response variable. We assume that $(y_1, \mathbf{x}_1), \cdots, (y_n, \mathbf{x}_n)$ are independent copies of a random vector (Y, X). The objective of regression analysis is to find the regression coefficient vector, $\beta \in \mathbb{R}^p$, which minimizes the prediction error evaluated by the expected loss $E[l(Y; X'\beta)]$, where $l : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$ is a given loss function. Examples of the loss function are squared loss $l(y, \mathbf{x}'\beta) = (y - \mathbf{x}'\beta)^2$).

Unfortunately, the prediction error is not available since the underlying distribution of (Y, X) is unknown. An alternative technique to resolve this

problem is to estimate β by minimizing the empirical expected loss $\mathcal{L}(\beta)$ defined by $\frac{1}{n}\sum_{i=1}^{n} l(y_i, \mathbf{x}_i^T\beta)$. However, directly minimizing the empirical expected loss suffers from the so-called 'over-fitting', especially when p is large compared to n. To avoid over-fitting, we minimize the penalized empirical expected loss. That is we estimate the regression coefficient β by minimizing the penalized empirical expected loss given as

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \mathcal{L}(\beta) + \sum_{j=1}^{p} J_\lambda(\beta_j) \right\},$$

where $J_\lambda(\cdot)$ is a penalty function and λ is a regularization parameter to control the effect of the penalty to the estimator.

An earlier research of penalized methods is the ridge penalty by Hoerl and Kennard (1970). The ridge estimator is defined as

$$\hat{\beta}^{Ridge} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \mathcal{L}(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2 \right\},$$

where $\lambda \geq 0$ is a tunning parameter that shrinks the estimator toward 0. The larger the value of λ is, the greater the amount of shrinkage is.

Typically, the shrinkage reduces the variance while it increases the bias. Let $\hat{\beta}_j$ is the ordinary least square estimator that minimizes $\mathcal{L}(\beta)$. It can be shown that $\|\hat{\beta}^{Ridge}\|_2 \leq \|\hat{\beta}\|_2$, where $\|\cdot\|_2$ is L_2 norm. This phenomenon is called the shrinkage which is helpful when the variance of the ordinary estimator is large compared to the bias. When there are many correlated covariates, the ordinary least square estimator has a large variance due to the multicollinearity. Hence, the ridge estimator achieves better prediction performance by reducing variance significantly while introducing a small amount of bias. For high dimensional models, typically

the ordinary least square estimator has a large variance and the shrinkage improves the performance of the estimator much.

But, the ridge regularization method has two critical drawbacks. First, the ridge estimator penalizes heavily the large coefficients that leads to serious biases to large coefficients. Second, the ridge estimator does not produce a sparse solution. That is, none of the estimated coefficients is exactly zero and thus the model interpretation is difficult.

Tibshirani (1996) proposed the least absolute shrinkage and selection operator (LASSO) to overcome these drawbacks of the ridge penalty. The LASSO estimator is defined as

$$\hat{\beta}^{LASSO} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \mathcal{L}(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

Due to the singularity of the objective function at the origin, LASSO penalty can shrink some of the estimated coefficients to be exact zero when the tuning parameter λ is sufficiently large. Figure 2.1 (Tibshirani, 1996) illustrates why the coefficient can be exact zero with the LASSO penalty.

The LASSO estimator varies continuously as λ varies. So, the LASSO estimator can be understood as a continuous subset selection procedure that results in lower variance. Furthermore, the LASSO estimator is less biased than the ridge estimator since the effect of the LASSO penalty to the estimator is smaller. Computing the LASSO solution is a quadratic programming problem, which may not be easy. However an efficient algorithm is available for computing the entire path of solutions as λ is varied. One search algorithm is modified least angle regression (LARS) algorithm suggested by Efron et al. (2004). LARS is a relative newcomer, and can be viewed as a kind of "democratic" version of forward stepwise regression.

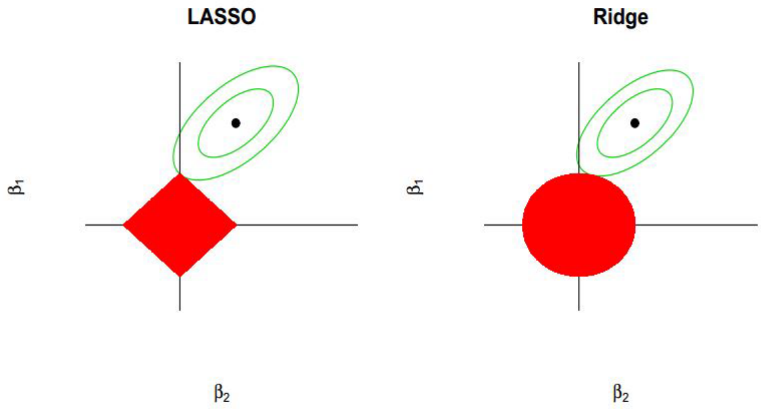

Figure 2.1: Why the LASSO solution can be exact zero

As we will see, LARS is intimately connected with the LASSO, and in fact provides an extremely efficient algorithm for computing the entire LASSO solution path.

Forward stepwise regression builds a model sequentially, adding one variable at a time. At each step, it identifies the best variable to include in the *active set*, and then updates the least squares fit to include all the active variables (the set of coefficient in the model).

LARS uses a similar strategy, but only enters "as much" of a predictor as it deserves. At the first step it identifies the variable most correlated with the response. Rather than fitting this variable completely, LARS moves the coefficient of this variable continuously toward its least squares value (causing its correlation with the evolving residual to decrease in absolute value). As soon as another variable "catches up" in terms of

correlation with the residual, the process is paused. The second variable then joins in the active set, and their coefficients are moved together in a way that keeps their correlations tied and decreasing. This process is continued until all the variables are in the model, and ends at the full least-squares fit (Friedman et al., 2001).

Detail of the LARS algorithm suggested by Efron et al. (2004) is as follows. By location and scale transformations we can always assume that the covariates have been standardized to have mean 0 and unit length, and that the response has mean 0,

$$\sum_{i=1}^{n} y_i = 0, \ \sum_{i=1}^{n} x_{ij} = 0, \ \sum_{i=1}^{n} x_{ij}^2 = 1 \text{ for } j = 1, 2, \cdots, p.$$

For a subset $\mathcal{A}$ of the i ndices $\{1, \cdots, p\}$, $\hat{\mu}_{\mathcal{A}} = \sum_{j \in \mathcal{A}} X_j \hat{\beta}_j$ is a current LARS estimate, $\hat{c} = X'(Y - \hat{\mu}_{\mathcal{A}})$ is the vector of correlation between covariate and residuals. Define the matrix

$$X_{\mathcal{A}} = (\cdots s_j X_j \cdots)_{j \in \mathcal{A}},$$

where the signs s_j equal to 1 or -1. Let $\mathcal{G}_{\mathcal{A}} = X'_{\mathcal{A}} X_{\mathcal{A}}$ and $A_{\mathcal{A}} = (1'_{\mathcal{A}} \mathcal{G}'_{\mathcal{A}} 1_{\mathcal{A}})^{-1/2}$. Then equiangular vector is given as $u_{\mathcal{A}} = X_{\mathcal{A}} \omega_{\mathcal{A}}$ where $\omega_{\mathcal{A}} = A_{\mathcal{A}} \mathcal{G}_{\mathcal{A}}^{-1} 1_{\mathcal{A}}$. The LARS algorithm proceed as follows.

Algorithm:LARS

Let $\mathcal{A} = \emptyset$ and $\beta = 0$.

1. Compute $\hat{c} = \mathbf{X}'(y - \hat{\mu}_{\mathcal{A}})$
2. Compute $\hat{C} = \max_j\{|\hat{c}_j|\}$, $\mathcal{A} = \{j : |\hat{c}_j| = \hat{C}\}$ and $s_j = \text{sign}(\hat{c}_j)$

3. Compute $X_{\mathcal{A}}$, $A_{\mathcal{A}}$ and $u_{\mathcal{A}}$

4. Compute $a = X' u_{\mathcal{A}}$

5. Compute $\hat{\delta} = \min^{+}_{j \in \mathcal{A}^c} \left\{ \frac{\hat{C} - \hat{c}_j}{A_{\mathcal{A}} - a_j}, \frac{\hat{C} + \hat{c}_j}{A_{\mathcal{A}} + a_j} \right\}$

6. Update $\hat{\beta}_{\mathcal{A}} = \hat{\beta}_{\mathcal{A}} + \hat{\delta} w_{\mathcal{A}}$

7. Repeat 1 ∼6 until $\hat{\beta}_{\mathcal{A}}$ equals to the OLS estimate for the full set of p covariate.

LARS algorithm keeps adding variables, when a regression coefficient reaches at 0, we delete the corresponding variable and recalculate the equiangular direction. Such a modification makes the LARS algorithm to solve the LASSO problem. The full set of the LASSO solutions can be obtained by a minor modification of the LARS algorithm. It closely parallels the homotopy method in the paper by Osborne et al. (2000a,b), though the LARS approach is somewhat more direct. Let $\hat{\beta}^{LASSO}$ be a LASSO solution with $\hat{\mu} = X\hat{\beta}^{LASSO}$. Then it is easy to show that the sign of any nonzero coordinate $\hat{\beta}_j^{LASSO}$ must agree with the sign s_j of the current correlation $\hat{c}_j = \mathbf{x}_j'(Y - \hat{\mu})$,

$$\text{sign}(\hat{\beta}_j) = \text{sign}(\hat{c}_j) = s_j.$$

The LARS algorithm does not enforce above restriction, but it can be easily be modified as follow.

Suppose that we have just completed a LARS step giving a new active set $\mathcal{A}$ and that the corresponding LARS estimate $\hat{\mu}_{\mathcal{A}}$ corresponds to a LASSO solution $\hat{\mu} = X\hat{\beta}^{LASSO}$. Let $\omega_{\mathcal{A}} = A_{\mathcal{A}} \mathcal{G}_{\mathcal{A}}^{-1} 1_{\mathcal{A}}$, a vector of length

the size of $\mathcal{A}$, and define $\hat{\mathbf{d}}$ to be the p-dimensional vector equaling $s_j \omega_{\mathcal{A}_j}$ for $j \in \mathcal{A}$ and zero elsewhere. Moving in the positive δ direction along the LARS line, we see that

$$\mu(\delta) = X\beta(\delta), \quad \text{where } \beta_j(\delta) = \hat{\beta}_j + \delta \hat{d}_j$$

for $j \in \mathcal{A}$. Therefore $\beta_j(\delta)$ will change its sign at

$$\delta_j = -\hat{\beta}_j / \hat{d}_j,$$

the first such changes occurring at

$$\tilde{\delta} = \min_j \{\delta_j\},$$

say for covariate $x_{\tilde{j}}$.

If $\tilde{\delta}$ is less than $\hat{\delta}$, then $\beta_j(\delta)$ cannot be a LASSO solution for $\delta > \tilde{\delta}$ since the sign restriction is violated. The next LASSO modification and Remark are given in the Theorem 1 of Efron et al. (2004).

LASSO MODIFICATION. If $\tilde{\delta} < \hat{\delta}$, stop the ongoing LARS step at $\delta = \tilde{\delta}$ and remove $\tilde{j}$ from the calculation of the next equiangular direction. That is,

$$\hat{\mu}_{\mathcal{A}_+} = \hat{\mu}_{\mathcal{A}} + \tilde{\delta} u_{\mathcal{A}} \text{ and } \mathcal{A}_+ = \mathcal{A} - \{\tilde{j}\}.$$

Remark 1. *Under the LASSO modification, and assuming the "one at a time" condition discussed below, the LARS algorithm yields all LASSO solutions.*

The active set $\mathcal{A}$ grows monotonically as the original LARS algorithm progresses, but the LASSO modification allows $\mathcal{A}$ to decrease. Here, "one at a time" means that the increases and decreases never involve more that a single index j. This is the usual case for quantitative data and can always be realized by adding a little jitter to the Y values.

2.2 Sparse regularization methods for two-way interaction linear model

The problem we are to tackle is a linear regression with high order interactions:

$$y = \beta_0 + \sum_{j=1}^{p} \beta_j x_j + \sum_{j \neq k} \beta_{jk} x_j x_k + \sum_{j \neq k \neq l} \beta_{jkl} x_j x_k x_l + \cdots + \epsilon,$$

where $\epsilon \sim N(0, \sigma^2)$. Fitting regression models with interactions is challenging when one has even a moderate number, p, of covariates, since there are $\binom{p}{k}$ interactions of order k. We focus on the two-way interaction model,

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \sum_{j<k} \beta_{jk} X_j X_k + \epsilon. \tag{2.1}$$

Our goal is to select a subset of the $p + p(p-1)/2$ main and interaction effects.

When p is large, there are too many terms in the model and thus an efficient variable selection method is needed. LASSO is powerful for variable selection and prediction, but if we consider interactions, we need an advanced method accounting the relationship between main and interaction effects. Bien et al. (2013) and Choi et al. (2010) suggested that. To model

heredity of interactions, they only allow an interaction into the model if the corresponding main effects are also in the model. Such restriction is known under various names, including "heredity", "marginality" and "hierarchically well-formulated" (Chipman, 1996, Hamada and Wu, 1992, Peixoto, 1987).

There are two types of restrictions, strong and weak hierarchy:

$$\text{strong hierarchy} \ : \ \beta_{jk} \neq 0 \text{ only if } \beta_j \neq 0 \text{ and } \beta_k \neq 0$$

$$\text{weak hierarchy} \ : \beta_{jk} \neq 0 \text{ only if } \beta_j \neq 0 \text{ or } \beta_k \neq 0$$

These hierarchies have to do with statistical power. In the words of Cox (1984),
"Large component main effects are more likely to lead to appreciable interactions than small components. Also, the interactions corresponding to larger main effects may be in some sense of more practical importance." It means that it may be useful to focus our search on interactions that have large main effects (Bien et al., 2013).

Choi et al. (2010) suggests a strong heredity interaction model. They reparametrized the coefficients for the interaction terms as

$$\beta_{jk} = \alpha_{jk}\beta_k\beta_k, \ j < k.$$

Then the two-way interaction model can be rewritten as

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \sum_{j<k} \alpha_{jk}\beta_j\beta_k X_j X_k. \tag{2.2}$$

Notice that the difference in the coefficients of the interaction term between (2.1) and (2.2). In (2.2), the coefficient for the interaction term

(X_jX_k) is expressed as the product of α_{jk}, β_j and β_k, instead of a single parameter β_{jk}. By writing the coefficient as a product, the model itself enforces the heredity constraint. That is, whenever the coefficient of either X_j or X_k, that is, β_j or β_k. is equal to zero, the coefficient for the interaction term (X_jX_k) automatically set to zero; vice versa, if the coefficient for (X_jX_k) is not equal to zero, it implies that both β_j and β_k are not equal to zero.

Let $\alpha = (\alpha_{jk})$ and $Z = (X_jX_k),\ j < k$. They suggested a following penalized least square estimate:

$$\text{argmin}_{\beta,\alpha}\|Y - X\beta - Z\alpha\|^2 + \lambda_\beta\|\beta\|_1 + \lambda_\alpha\|\alpha\|_1, \tag{2.3}$$

There are two tuning parameters, λ_β and λ_α. The first tuning parameter λ_β controls the estimates at the main effect level: if β_j is shrunken to zero, variable X_j and all its "descendants," that is, the corresponding interaction terms that involve X_j are removed from the model. The second tuning parameter λ_α controls the estimates at the interaction effect level: if β_j and β_k are not equal to zero but the corresponding interaction effect is not strong, α_{jk} still has the possibility of being zero, so it has the flexibility of selecting only the main terms.

To obtain the estimators β_k and α_{jk}, they used an iterative approach, that is, first fix β_j and estimate α_{jk}, then fix α_{jk} and estimate β_j, and iterate between these steps until the solution coverages. Since at each step, the value of the objective function decreases, the solution is guaranteed to converge.

When $\beta_j,\ j = 1, \cdots, p$ are fixed, (2.3) becomes a LASSO problem, hence we can use either the LARS/LASSO algorithm or a quadratic pro-

gramming package to efficiently solve for $\alpha_{jk}, j < k$. When $\alpha_{jk}, j < k$ are fixed, we can sequentially solve for $\beta_j, j = 1, \cdots, p$ similarly.

However the problem is computational cost because the number of parameters is large and the objective function is not convex. So it is almost impossible to be scaled-up for very high dimensional data.

Bien et al. (2013) also assigned a hierarchy structure to the interaction effects. They wanted to estimate the two-way interaction model using LASSO. Let $B = (\beta_{jk}),\ j, k = 1, \cdots, p$ with $B_{jj} = 0$ for $j = 1, \cdots, p$. Then all-pairs LASSO for the two-way interaction model find β and B as

$$\text{argmin}_{\beta \in R^p, B \in R^{p \times p}} \|Y - X\beta - \frac{1}{2} ZB\|_2^2 + \lambda \|\beta\|_1 + \frac{\lambda}{2} \|B\|_1. \qquad (2.4)$$

where $B = B'$ and $\|B\|_1 = \sum_{j \neq k} |\beta_{jk}|$. The factor of one half before the interaction summation is a consequence of notational decision to deal with a symmetric matrix B of interactions rather than a vector of length $p(p-1)/2$. They referred this model as *all-pairs LASSO* since it is simply the LASSO applied to a data matrix which includes all pairs of interactions (as well as all main effects). It is common with the LASSO to standardized the predictors so that they are on the same scale. They standardized X so that its columns have mean 0 and standard deviation 1 and then formed Z from these standardized predictors and, finally, centered the resulting columns of Z. By centering Y and (X, Z), they take $\hat{\beta}_0 = 0$.

The notions of strong and weak hierarchy represent situations in which they wanted to exclude certain sparsity patterns. There had been a growing literature focusing on methods that produce structured sparsity (Bach et al., 2012a,b, Jenatton et al., 2010, 2011, Yuan and Lin, 2006, Zhao et al., 2009). These methods make use of the group LASSO penalty which, given

a predetermined grouping of the parameters, induces entire groups of parameters to be set to zero (Yuan and Lin, 2006).

Bien et al. (2013) proposed a LASSO-like procedure that produces sparse estimates of β and B while satisfying the strong or weak hierarchy constraint. In contrast to many of the structured sparsity which are based on group LASSO penalties, they were involved adding a set of convex constraints to the LASSO. A key advantage of their specific choice of penalty structure is that it admits a simple interpretation of the effect of the hierarchy demand. Unlike other hierarchical sparsity methods, which do not pay much attention to the particular choice of norms, their formulation was carefully tailored to allow it to be related directly back to the LASSO, permitting one to understand specifically how hierarchy alters the solution. Furthermore, their characterization suggested that the demand for hierarchy is a form of "regularization."

As a motivation for their proposal, consider building hierarchy into the optimization problem as a constraint,

$$\text{argmin}_{\beta\in R^p, B\in R^{p\times p}} \|Y - X\beta - \frac{1}{2}ZB\|_2^2 + \lambda\|\beta\|_1 + \frac{\lambda}{2}\|B\|_1. \tag{2.5}$$

such that $B = B'$ and $\|B_j\|_1 \leq |\beta_j|$ for $j = 1, \cdots, p$. Notice that if $\hat{\beta}_{jk} \neq 0$ then $\|\hat{B}_j\|_1 > 0$ and $\|\hat{B}_k\|_1 > 0$ and thus $\hat{\beta}_j \neq 0$ and $\hat{\beta}_k \neq 0$. While the added constraints enforce strong hierarchy, they are not convex and hence is (2.5) undesirable. So Bien et al. (2013) proposed a straightforward convex relaxation, which called strong hierarchical LASSO and weak hierarchical LASSO. Relaxed LASSO problems are as follows:

1. Strong hierarchical LASSO: Find β and B as

$$\text{argmin}_{\beta\in R^p, B\in R^{p\times p}} \|Y - X\beta^+ + X\beta^- - ZB\|_2^2 + \lambda 1'(\beta^+ + \beta^-) + \frac{\lambda}{2}\|B\|_1 \tag{2.6}$$

such that $B = B'$, $\|B_j\|_1 \leq \beta_j^+ + \beta_j^-$, $\beta_j^+ \geq 0$ and $\beta_j^- \geq 0$ for $j = 1, \cdots, p$.

2. Weak hierarchical LASSO : Find β and B as

$$\text{argmin}_{\beta\in R^p, B\in R^{p\times p}} \|Y - X\beta^+ + X\beta^- - ZB\|_2^2 + \lambda 1'(\beta^+ + \beta^-) + \frac{\lambda}{2}\|B\|_1 \tag{2.7}$$

such that $\|B_j\|_1 \leq \beta_j^+ + \beta_j^-$, $\beta_j^+ \geq 0$ and $\beta_j^- \geq 0$ for $j = 1, \cdots, p$.

The main idea is to replace the optimization variable $\beta \in \mathbb{R}^p$ by two vectors $\beta^+, \beta^- \in \mathbb{R}^p$. After solving the above problem, the fitted model is of the form $\hat{f}(x) = \hat{\beta}_0 + x'(\hat{\beta}^+ - \hat{\beta}^-) + x'\hat{B}x/2$.

The hierarchy constraints can be seen as an embedding into their method of David Cox's "principle" that "large component main effects are more likely to appreciable interactions than small components." The constraint

$$\|B_j\|_1 \leq \beta_j^+ + \beta_j^-$$

budgets the total amount of interactions involving variable X_j according to the relative importance of X_j as a main effect. One additional advantage of the convex relaxation is that the constraint is less restrictive. If the best fitting model would have $\|B_j\|_1$ large but $|\beta_j|$ only moderate, this can be accommodated by making β_j^+ and β_j^- both large.

Also, they suggested including an elastic net term, $(\epsilon/2)(\|B\|_F^2 + \|\beta^+\|_2^2 + \|\beta^-\|_2^2)$, to ensure uniqueness of the solution (Zou and Hastie, 2005). They

thought of $\epsilon > 0$ as a fixed tiny fraction of λ, such as $\epsilon = 10^{-8}\lambda$, rather than as an additional tuning parameter. Such a modification does not complicate the algorithm, but simplifies the study of the estimator.

A key advantage of formulating an estimator as a solution to a convex problem is that it can be completely characterized by a set of optimality conditions, known as the Karush-Kuhn-Tucker (KKT) conditions. These conditions are useful for understanding the effect that the hierarchy constraint in (2.6) and (2.7) has on their solutions.

Let

$$\begin{aligned} r^{(-j)} &= y - \hat{y} + x_j\hat{\beta}_j, \\ r^{(-jk)} &= y - \hat{y} + (x_j * x_k)(\hat{B}_{jk} + \hat{B}_{kj})/2 \end{aligned}$$

denote partial residuals (where $*$ denotes elementwise multiplication, $\hat{y}$ the vector of fitted values and x_j the jth predictor), and assume that $\|x_j\|_2 = 1$. For linear regression, the KKT conditions are known as the normal equations and can be written as,

$$\hat{\beta}_j = x_j' r^{(-j)}, \;\; \hat{B}_{jk} = \frac{(x_j * x_k)' r^{(-jk)}}{\|x_j * x_k\|_2^2}.$$

The all-pairs LASSO solution satisfies

$$\hat{\beta}_j = \mathcal{S}(x_j' r^{(-j)}, \lambda), \;\; \hat{B}_{jk} = \frac{\mathcal{S}[(x_j * x_k)' r^{(-jk)}, \lambda]}{\|x_j * x_k\|_2^2},$$

where $\mathcal{S}$ denotes the soft-thresholding operator defined by $\mathcal{S}(c, \lambda) = \text{sign}(c)(|c| - \lambda)_+$. Written this way, we see that the LASSO is similar to linear regression, but all coefficients are shrunken toward 0, with some coefficients set to zero. It is instructive to examine the corresponding statements for the strong and weak hierarchical LASSO methods and it is stated as property 1 in Bien et al. (2013) which is restated as next Remark 2.

Remark 2. *The coefficients of the strong and weak hierarchical LASSOs with $\lambda > 0$ with linear regression model satisfy:*

- Strong:

$$\begin{aligned} \hat{\beta}_j^+ - \hat{\beta}_j^- &= \mathcal{S}(x_j' r^{(-j)}, \lambda - \hat{\alpha}_j), \\ \hat{B}_{jk} &= \frac{\mathcal{S}[(x_j * x_k)' r^{(-jk)}, \lambda + \hat{\alpha}_j + \hat{\alpha}_k]}{\|x_j * x_k\|_2^2} \end{aligned}$$

- Weak:

$$\begin{aligned} \hat{\beta}_j^+ - \hat{\beta}_j^- &= \mathcal{S}(x_j' r^{(-j)}, \lambda - \hat{\alpha}_j), \\ \frac{\hat{B}_{jk} + \hat{B}_{kj}}{2} &= \frac{\mathcal{S}[(x_j * x_k)' r^{(-jk)}, \lambda + 2\min\{\tilde{\alpha}_j, \tilde{\alpha}_k\}]}{\|x_j * x_k\|_2^2} \end{aligned}$$

for some $\hat{\alpha}_j \geq 0,\ j = 1, \cdots, p$ with $\hat{\alpha}_j = 0$ when $\|\hat{B}_j\|_1 < \hat{\beta}_j^+ + \hat{\beta}_j^-$ (and likewise for $\tilde{\alpha}_j$).

Remark 2 reveals that the overall form of the all-pairs LASSO and hierarchical LASSO methods are identical. The difference is that the hierarchy constraint leads to a reduction in the shrinkage of certain main effects and an increase in the shrinkage of certain interactions. In particular, we see that when the hierarchy constraints are loose at the solution, that is, $\|\hat{B}_j\|_1 < \hat{\beta}_j^+ + \hat{\beta}_j^-$, the weak hierarchical LASSO's optimality conditions become identical to the all-pairs LASSO (since $\tilde{\alpha}_j = 0$) for all coefficients involving x_j. For the strong hierarchical LASSO, when both the jth and kth constraints are looses, the optimality conditions match those of the all-pairs LASSO for the coefficients of x_j, x_k and $x_j * x_k$. The

methods differ when constraints are active, that is, when $\|\hat{B}_j\|_1 = \hat{\beta}_j^+ + \hat{\beta}_j^-$, which allow $\hat{\alpha}_j$ (or $\tilde{\alpha}_j$), to be nonzero. Intuitively, this case corresponds to the situation in which hierarchy would not have held without the constraint, and the corresponding dual variable plays the role of reducing $\hat{B}_j$ in l_1-norm and increasing $\hat{\beta}_j^+ + \hat{\beta}_j^-$ until the constraint is satisfied. The way in which the weak and strong hierarchical LASSO methods perform this shrinkage is different, but both are identical to the all-pairs LASSO when all constraints are loose.

Bien et al. (2013) also has shown that hierarchy is guaranteed for convex relaxation's constraints. In particular they studied (2.6) and (2.7) included an elastic net term. The importance of this modification is that it ensures uniqueness, simplifying the analysis and the results can be summarized as following Remark 3, 4.

Remark 3. *Suppose Y is absolutely continuous with respect to the Lebesque measure on $\mathbb{R}^n$. If $(\hat{\beta}^+, \hat{\beta}^-, \hat{B})$ solve (2.6) with an $\epsilon > 0$ ridge penalty, then strong hierarchy holds with probability 1, that is,*

$$\hat{B}_{jk} \neq 0 \;\Rightarrow\; \hat{\beta}_j^+ - \hat{\beta}_j^- \neq 0 \text{ and } \hat{\beta}_k^+ - \hat{\beta}_k^- \neq 0.$$

Remark 4. *Suppose Y is absolutely continuous with respect to the Lebesque measure on $\mathbb{R}^n$. If $(\hat{\beta}^+, \hat{\beta}^-, \hat{B})$ solve (2.7) with an $\epsilon > 0$ ridge penalty, then weak hierarchy holds with probability 1, that is,*

$$\frac{\hat{B}_{jk} + \hat{B}_{kj}}{2} \neq 0 \;\Rightarrow\; \hat{\beta}_j^+ - \hat{\beta}_j^- \neq 0 \text{ and } \hat{\beta}_k^+ - \hat{\beta}_k^- \neq 0.$$

Bien et al. (2013) suggest generalized gradient descent approach to solve (2.6) and (2.7). For weak hierarchical LASSO, the algorithm is as follow (with elastic net penalty ϵ).

Algorithm:Weak hierarchical LASSO

Input: $X \in \mathbb{R}^{n\times p}$, $Z \in \mathbb{R}^{n\times p(p-1)}$, $\lambda > 0$. Initialize $(\hat{\beta}^{+(0)}, \hat{\beta}^{-(0)}, \hat{B}^{(0)})$.

For $k = 1, 2, \cdots$, until convergence:

Compute residual: $\hat{r}^{(k-1)} \leftarrow Y - X(\hat{\beta}^{+(k-1)} - \hat{\beta}^{-(k-1)}) - Z\hat{B}^{(k-1)}/2$.

For $j = 1, \cdots, p$:

$$(\hat{\beta}_j^{+(k)}, \hat{\beta}_j^{-(k)}, \hat{B}_j^{(k)}) \leftarrow \mathsf{ONEROW}(\quad \delta\hat{\beta}_j^{+(k+1)} - tX_j'\hat{r}^{(k-1)},\ \delta\hat{\beta}^{-(k-1)} + tX_j'\hat{r}^{(k-1)},\ \delta\hat{B}_j^{(k-1)} - tZ_{(j,\cdot)}'\hat{r}^{(k-1)}),$$

where ONEROW is given in Algorithm ONEROW, t is a suitably chosen step size, $\delta = 1 - t\epsilon$, and $Z_{(j,\cdot)} \in \mathbb{R}^{n\times(p-1)}$ denotes the columns of Z involving X_j.

Algorithm:ONEROW

Input:$\tilde{\beta}_j^+, \tilde{\beta}_j^- \in \mathbb{R}$, $\tilde{B}_j \in \mathbb{R}^{p-1}$, $\lambda \geq 0$.

1. Find $\hat{\alpha}$. Define $f(\alpha) = \|S(\tilde{B}_j, t(\lambda/2, \alpha))\|_1 - [\tilde{\beta}_j^+ + t\alpha]_+ - [\tilde{\beta}_j^- + t\alpha]_+$.

   (a) If $f(0) \leq 0$, take $\hat{\alpha} = 0$ and go to step 2.

(b) Form knot the set $\mathcal{P} = \{|\tilde{B}_{jk}|/t - \lambda/2\}_{k=1}^{p} \cup \{-\tilde{\beta}^{\pm}/t\}$, and let $\mathcal{P}^{+} = \mathcal{P} \cap [0, \infty)$.

(c) Evaluate $f(p)$ for $p \in \mathcal{P}^{+}$

(d) If $f(p) = 0$ for some $p \in \mathcal{P}^{+}$, take $\hat{\alpha} = p$ and go to step 2.

(e) Find adjacent knots, $p_1, p_2 \in \mathcal{P}^{+}$, such that $f(p_1) > 0 > f(p_2)$. Take

$$\hat{\alpha} = -f(p_1)[f(p_2) - f(p_1)]/(p_2 - p_1)$$

2. Return $\hat{B}_j = \mathcal{S}[\tilde{B}_j, t(\lambda/2 + \hat{\alpha})]$ and $\hat{\beta}_j^{\pm} = [\tilde{\beta}_j^{\pm} + t\hat{\alpha}]_{+}$.

The R package hierNet provides implementations of their strong and weak methods. The methods suggested by Bien et al. (2013) has some advantage that it admits a simple characterization of the effect of imposing hierarchy. However it requires few thousands iteration until convergence while p^2 many computations is required in each interaction. Hence the algorithm can be applicable only when p is less than few hundreds.

2.3 Message passing interface

To put our discussion of message passing in perspective, we briefly review informally the principal parallel computational models. We focus then on the advantages of the message-passing models (Gropp et al., 2014).

2.3.1 Parallel computational models

A computational model is a conceptual view of the types of operations available to a program. It does not include the specific syntax of a particular programming language or library, and it is (almost) independent of the underlying hardware that supports it. That is, any of the models we discuss can be implemented on any modern parallel computer, given a little help from the operating system. The effectiveness of such an implementation, however, depends on the gap between the model and the machine.

Parallel computational models form a complicated structure. They can be differentiated along multiple axes: where the memory is physically shared or distributed, how much communication is in hardware or software, exactly what the unit of execution is, and so forth.

Data parallelism

Although parallelism occurs in many cases and at many levels in a modern computer, one of the first places it was made available to the programmer was in vector processors. Indeed, the vector machine began the current age of supercomputing. The vector machine's notion of operating on an array of similar data items in parallel during a single operation was extended to include the operation of whole programs on collection of data structures. The parallelism need not necessarily proceed instruction by unstruction in lock step for it to be classified as data parallel.

Shared memory

Parallelism that is not determined implicitly by data independence but is explicitly specified by the programmer is control parallelism. One simple model of control parallelism is the shared-memory model, in which each processor has access to all of a single, shared address space at the usual level of load and store operations. Access to locations manipulated by multiple processes is coordinated by some form of locking, although high-level languages may hide the explicit use of locks.

Making ”true” shared-memory machines with more than a few tens of processors is difficult (and expensive). To achieve the shared-memory model with large numbers of processors, one must allow some memory references to take longer than others. The most common shared-memory systems today are single-chip multicore processors or nodes consisting of a few multicore processors. Such nodes can be assembled into very large distributed-memory machines.

Message passing

The message-passing model posits a set of processes that have only local memory but are able to communicate with other processes by sending and receiving messages. It is a defining feature of the message-passing model that data transfer from the local memory of one process to the local memory of another requires operations to be performed by both processes. Since MPI is a specific realization of the message-passing model, message-passing models are implemented on a wide variety of hardware architectures.

Gropp et al. (1996) give a brief overviews of MPI. The message-passing model of parallel computation has emerged as an expressive, efficient, and well-understood paradigm for parallel programming. Until 1994, the syntax and precise semantics of each message-passing library implementation were different from the others, although mean of the general semantics were similar. The proliferation of message-passing library designs from both vendors and users was appropriate for a while, but eventually it was seen that enough consensus on requirements and general semantics for message-passing had been reached that an attempt at standardization might usefully be undertaken.

The project to provide a portable implementation of MPI began at the same time as the MPI definition process itself. The idea was to provide early feedback on decisions being made by the MPI Forum and provide an early implementation to allow users to experiment with the definitions even as they were being developed. Targets for the implementation were to include all systems capable of supporting the message-passing model. MPICH is a freely available, complete implementation of the MPI specification, designed to be both portable and efficients. The "CH" in MPICH stands for "Chameleon", symbol of adaptability to one's environment and thus of portability.

MPICH is thus both a research project and a software development project. As a research project, its goal is to explore methods for narrowing the gap between the programmer of a parallel computer and the performance deliverable by its hardware. In MPICH, we adopt the constraint that the programming interface will be MPI, reject constraints on the architecture of the tangent machine, and retain high performance (measured

in terms of bandwidth and latency for message-passing operations) as a goal. As a software project, MPICH's goal is to promote the adoption of the MPI Standard by providing users with a free, high-performance implementation on a diversity of platforms, while aiding vendors in providing their own customized implementations.

MPI is a message-passing application programmer interface, together with protocol and semantic specifications for how its features must behave in any implementation(such as a message buffering and message delivery progress requirement). MPI includes point-to-point message passing and collective (global) operations, all scoped to a user-specified group of processes. Furthermore, MPI provides abstractions for processes at two levels. First, processes are named according to the rank of the group in which the communication is being processes that help relate the application semantics to the message passing semantics in a convenient, efficient way. Communicators, which house groups and communication context (scoping) information, provide an important measure of safety that necessary and useful for building up library-oriented parallel code.

MPI also provides three additional classes of services: environmental inquiry, basic timing information for application performance measurement, and a profiling interface for external performance monitoring. MPI makes heterogeneous data conversion transparent part of its services by requiring datatype specification for all communication operations. Both built-in and user-defined datatypes are provided.

MPI accomplishes its functionality with opaque objects, with well-defined constructors and destructors, giving MPI an object-based look and feel. Opaque objects include groups (the fundamental container for pro-

cesses), communicators(which contain groups and are used as arguments to communication calls), and request objects for asynchronous operations. User-defined and predefined datatypes allow for heterogeneous communication and elegant description of gather/scatter semantics in send/receive operations as well as in collective operations.

2.3.2 Advantages of the MPI

University

The message-passing model fits well on separate processors connected by a communication network. Thus, it matches the highest level of the hardware of most of today's parallel supercomputers, as well as workstation networks and dedicated PC clusters. Where the machine supplies extra hardware to support a shared-memory model, the message-passing model can take advantage of this hardware to speed data transfer. Use of a GPU can be orthogonal to the use of MPI.

Expressivity

Message passing has been found to be a useful and complete model in which to express parallel algorithms. It provides the control missing from the data-parallel and compiler-based models in dealing with data locality. Some find its anthropomorphic flavor useful in formulating a parallel algorithm. It is well suited to adaptive, self-scheduling algorithms and to programs that can be made tolerant of the imbalance in process speeds found on shared network.

Ease of debugging

Debugging of parallel programs remains a challenging research area. While debuggers for parallel programs are perhaps easier to write for the shared-memory model, it is arguable that the debugging process itself is easier in the message-passing paradigm. The reason is that one of the most common causes of error is unexpected overwriting memory. The message-passing model, by controlling memory references more explicitly than any of the other models, make it easier to locate erroneous memory reads and writes. Some parallel debuggers even can display message queues, which are normally invisible to the programmer.

Performance

The most completing reason that massage passing will remain a permanent part of the parallel computing environment is performance. As modern CPUs have become faster, management of their caches and the memory hierarchy in general has become the key to getting the most out of these machines. Message passing provides a way for the programmer to explicitly associate specific data with processes and thus allow the compiler and cache-management hardware to function fully. Indeed, one advantage distributed-memory computers have over even the largest single-processor machines is that they typically provide more memory and more cache. Memory-bound applications can exhibit superlinear speedups when proted to such machines. And even on shared-memory computers, use of the message-passing model can improve performance by providing more programmer control of data locality in the memory hierarchy.

2.3.3 Basic MPI concepts

In the message-passing model of parallel computation, the processes executing in parallel have separate address spaces. Communication occurs when a portion of one process's address space is copied into another process's address space. This operation is cooperative and occurs only when the first process executes a *send* operations and the second process executes a *receive* operation.

For the sender, the obvious arguments that must be specified are the data to be communicated and the destination process to which the data is to be sent. The minimal way to describe data is to specify a starting address and a length. Any sort of data item might be used to identify the destination, typically it has been an integer. On receiver's side, the minimum arguments are the address and length of an area in local memory where the received variable is to be placed, together with a variable to be filled in with the identity of the sender, so that the receiving process can know which process sent it the message.

Although an implementation of this minimum interface might be adequate for some applications, more features usually are needed. One key notion is that of *matching*: a process must be able to control which messages it receives, by screening them by means of another integer, called the *type* or *tag* of the message. A message-passing system is expected to supply queuing capabilities so that a receive operation specifying a tag will complete successfully only when a message sent with a matching tag arrives. This consideration adds the tag as an argument for both sender and receiver. It is also convenient if the source can be specified on a receive operation as an additional screening parameter.

Now minimal message interface has become

- send(address, length, destination, tag)
- receive(address, length, source, tag, actlen)

where source and tag in the receive can be either input arguments used to screen messages or special values used as "wild cards" to indicate that messages will be matched from any source or with any tag, in which case they could be filled in with the actual tag and destination of the message received. The argument actlen is the length of the message received. Typically it is considered an error if a matching message is received that is too long, but no if it is too short.

In decide the destination and source, we need to naming processes. These processes belong to groups. If a group contains n processes, then its processes are identified within the group by *ranks*, which are integers from 0 to $n - 1$. All processes in an MPI implementation belong to an initial group. Within this group, processes are numbered similarly to the way in which they are numbered in many previous message-passing systems.

The notions of context and group are combined in a single object called a *communicator*, which becomes an argument to most point-to-point and collective operations. Thus the destination or source specified in a send or receive operation always refers to the rank of the process in the group identified with the given communicator.

MPI the basic *send* and *receive* operation is follows,

- MPI_Send(address, count, datatype, destination, tag, comm)
- MPI_Recv(address, count, datatype, source, tag, comm, status)

where

- (address, count, datatype) describes count occurrences of items of the form datatype starting at address,
- destination is the rank of the destination in the group associated with the communicator comm,
- source is the rank of the source of the message in the group associated with the communicator comm, or a wildcard matching any source,
- tag is an integer used for message matching,
- comm identifies a group of processes and a communication context, and
- status holds information about the actual message size, source, and tag, useful when wild cards have been used.

Our focus so far has been on the basic *send* and *receive* operations. However MPI is a large specification and offers many other advanced features, including collective communications, virtual topologies, debugging and profiling, communication modes, support for libraries, support for heterogeneous networks and etc. For details, see Gropp et al. (2014).

Chapter 3

LARS for hierarchical interaction

3.1 HLARS algorithm

If we want to find interaction effects using the LARS algorithm, it cannot take account of hierarchy between main effects and interaction effects and need much of computational time. In this thesis, we will suggest a modified LARS algorithm to do variable selection and prediction simultaneously by considering weak heredity and extend it to ultra high dimensional data using parallel processing to scale up. Strong heredity can be done easily.

As reviewed previously, using the standard LASSO algorithm with hierarchy takes much of computational time. So we focus on developing a fast and scalable algorithm by hierarchical LARS which is called HLARS.

Note that the LARS algorithm has two sets active $\mathcal{A}$ and inactive $\mathcal{A}^c$. The main idea of the proposed the HLARS algorithm is to have a

candidate set $\mathcal{B}$ in between $\mathcal{A}$ and $\mathcal{A}^c$ as

1. LARS
$$\mathcal{A} \rightleftarrows \mathcal{A}^c$$
2. HLARS
$$\mathcal{A} \rightleftarrows \mathcal{B} \rightleftarrows \mathcal{A}^c$$

where $\mathcal{B}$ = main effect$\cup$interactions with nonzero main effects. In LARS. "new" variable comes from inactive set. If we fit the regression model only with the main effect, it is suffice to having active set and inactive set. However when we consider the regression model not only main effects but also interaction effects, the inactive set has $p + p(p-1)/2 - |\mathcal{A}|$ variables and it need much of times to search new variable in the inactive set. So we propose a "candidate set" $\mathcal{B}$ to reduce searching time of new variable by weak heredity restriction.

Brief explanation of the idea of HLARS algorithm is given follows and is summarized Figure 3.1.

Idea of HLARS algorithm

1. Let u be the variable (including interactions) in $\mathcal{B}$ which has the largest correlation with the current residuals.
2. If u is an interaction, move it to the active set.
3. If u is a main effect
   (a) Add the main effect and corresponding interactions into $\mathcal{B}$.

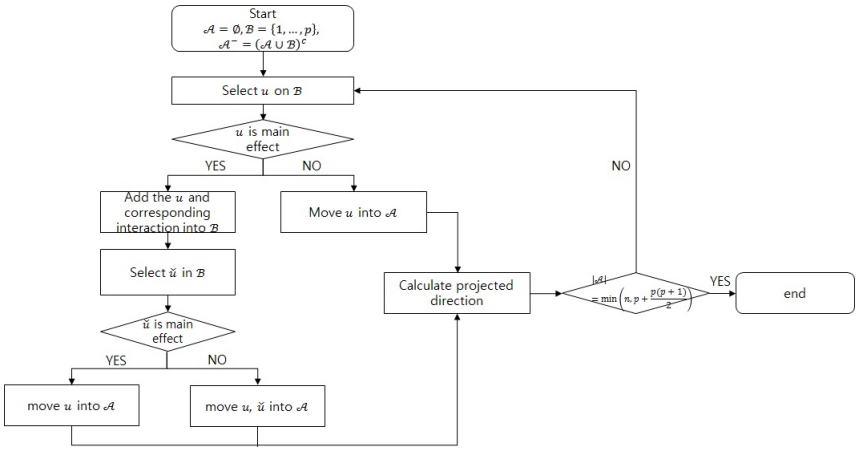

Figure 3.1: Flow chart for HLARS algorithm

(b) Select the variable in $\mathcal{B}$ having the largest correlation with the current residual.

(c) If the selected variable is a main effect, move it to $\mathcal{A}$.

(d) Otherwise, move the selected interaction and corresponding main effect to $\mathcal{A}$ to meet weak heredity.

4. Calculate the new direction so called the projected direction only with variables in $\mathcal{A}$.

5. Move the solution until there exists a variable in $\mathcal{B}$ having a larger correlation with the current residual.

To explain how the HLARS algorithm proceed, let define some notations and calculations.

- Notations
  - $\mathcal{I}^M = \{1, \cdots, p\}$
  - $\mathcal{I}_j^L = \{(1, j), \cdots, (j-1, j)\}$
  - $\mathcal{I}_j^U = \{(j, j+1), \cdots, (j, p)\}$
  - $X_v = X_j X_k$ for $v = (j, k) \in \mathcal{I}_j^U$
  - $Z_j = (X_v)$ for all $v \in \mathcal{I}_j^U$
  - $\mathbf{Z} = (Z_1, Z_2, \cdots, Z_{p-1})$
  - $\tilde{\mathbf{X}} = (\mathbf{X}, \mathbf{Z})$
  - $\mathcal{A}$: active set
  - $\mathcal{B}$: candidate set
  - r : residuals
- Calculations
  - $\hat{c} = \tilde{\mathbf{X}} r$
  - $s_u = \text{sign}\{\hat{c}_u\}$ for $u \in \mathcal{A}$.
  - $\tilde{\mathbf{X}}_{\mathcal{A}} = (\cdots s_u X_u \cdots)_{u \in \mathcal{A}}$
  - $\beta_{\mathcal{A}} = (\beta_u)_{u \in \mathcal{A}}$

The HLARS algorithm works as follows,

HLARS algorithm

Let $\mathcal{A} = \emptyset, \mathcal{B} = \{1, \cdots, p\}$ and $\beta = 0$.

1. Let $\hat{\beta}_{\mathcal{A}}$ is current HLARS estimate.

2. Let $r = Y - \tilde{\mathbf{X}}_{\mathcal{A}}\hat{\beta}_{\mathcal{A}}$

3. Compute $\hat{c} = \tilde{\mathbf{X}}'r$

4. Choose $u = \text{argmax}_v\{|\hat{c}_v| : v \in \mathcal{B}\}$

5. If $u \in \{1, \cdots, p\}$, update $\mathcal{B} = \mathcal{B} \bigcup \left[v : v \in \mathcal{I}_u^L \cup \mathcal{I}_u^U\right]$.

   (a) Choose $\breve{u} = \text{argmax}_v\{|\hat{c}_v| : v \in \mathcal{B}\}$

   (b) Update $\mathcal{A} = \mathcal{A} \bigcup \{u, \breve{u}\}$

6. Else update $\mathcal{A} = \mathcal{A} \bigcup \{u\}$

7. Compute

$$\begin{aligned} S &= (\cdots, s_j, \cdots)_{j\in\mathcal{A}},\ s_j = Sign(\hat{c}_j),\ s_j \in \{-1, 1\} \\ w_{\mathcal{A}} &= (\tilde{\mathbf{X}}'_{\mathcal{A}}\tilde{\mathbf{X}}_{\mathcal{A}})^{-1}\tilde{\mathbf{X}}'_{\mathcal{A}}r \\ Cmax &= \max_i \tilde{\mathbf{X}}'_{\mathcal{A}}r \end{aligned}$$

8. Let $r(\delta) = r - \delta\tilde{\mathbf{X}}_{\mathcal{A}}w_{\mathcal{A}}$

9. Compute

$$\begin{aligned} \hat{\delta} &= \min\left\{\delta : \max_{i\in\mathcal{A}} |\tilde{\mathbf{X}}'_i r(\delta)| = |\tilde{\mathbf{X}}'_j r(\delta)|,\ \text{for } j \in \mathcal{B}\right\} \\ &= \min\{\nu_j : \nu_j > 0, j \in \mathcal{B}\} \end{aligned}$$

   where $\nu_j = \min_{j\in\mathcal{B}}\left(\frac{Cmax+\tilde{\mathbf{X}}'_j r}{Cmax+\tilde{\mathbf{X}}'_j\tilde{\mathbf{X}}_{\mathcal{A}}w_{\mathcal{A}}}, \frac{Cmax-\tilde{\mathbf{X}}'_j r}{Cmax-\tilde{\mathbf{X}}'_j\tilde{\mathbf{X}}_{\mathcal{A}}w_{\mathcal{A}}}\right)$

10. Update $\hat{\beta}_{\mathcal{A}} = \hat{\beta}_{\mathcal{A}} + \hat{\delta}w_{\mathcal{A}} \otimes S$

11. Repeat 1∼10 until $|\mathcal{A}| = \min(n, p + p(p-1)/2)$.

Let consider the first step in the HLARS algorithm. Active set $\mathcal{A}$ is NULL and candidate set $\mathcal{B}$ has all of the main effects. For example, let $p = 5$ then $\mathcal{B} = \{1, 2, 3, 4, 5\}$ and let corresponding correlation between residuals be $\{10, 20, 15, -20, 25\}$. Then the HLARS choose the 5th main effect which is most correlated with the residuals and updates $\mathcal{B} = \{1, 2, 3, 4, 5, (1, 5), (2, 5), (3, 5), (4, 5)\}$. The next step is to recompute correlations related to interactions in $\mathcal{B}$. Let updated correlations be $\{10, 20, 15, -20, 25, 30, 17, 13, 21\}$. Interaction $(1, 5)$ is more correlated to the residuals than the 5th main effect and so the 5th main effect and interaction $(1, 5)$ are added to active set together. After that, HLARS computes the projected direction $\omega_{\mathcal{A}}$ and the amount of move $\hat{\delta}$ and finally updates β accordingly.

As we can see the HLARS algorithm, it is very similar to the LARS algorithm. However there exist some differences. First, $\mathcal{B}$ is not the inactive set but it is smaller than the inactive set in LARS and it makes HLARS search the solution faster than the LARS. This difference makes it possible to scale up which will be described in next section.

Second, the direction of moving $\omega_{\mathcal{A}}$ is not equiangular as in the LARS. In the LARS, δ, the size of movement, is determined by the correlation between residuals and newly activated variable and the variable is as correlated as variables in active set. It means all variables in the active set has same correlations between variables and current residuals. However in the HLARS if a main effect and the interaction corresponding the main effect enter the active set simultaneously, the correlation related to main effect is less than correlation of the corresponding interaction term. So the next variable which is enters the active set has same correlation as the

interaction effect, but not as main effect. That makes $\omega_{\mathcal{A}}$ in-equiangular direction.

Third, when we determine the amount of move $\hat{\delta}$, we compute $\max_{k\in\mathcal{A}} |\tilde{X}_k' r(\delta)|$. In the LARS, we do not have to calculate the maximum of correlations between current residuals and covariates in active set since all correlations are same. However in the HLARS some variables in the active set have different correlations. We choose $\hat{\delta}$ as maximum correlation in the active set and is same in the candidate set. So we choose $\hat{\delta}$ such that as some variables in the candidate set $\mathcal{B}$ has as much as similar to the maximum correlation.

Similar to the LARS algorithm, HLARS algorithm can be modified to remove some variables in the active set $\mathcal{A}$ to $\mathcal{B}$. If a coefficient becomes 0, then the corresponding variable in the active set $\mathcal{A}$ can move to candidate set $\mathcal{B}$. There exists 3-cases where the zero coefficient can move to $\mathcal{B}$. First case is that the a coefficient of interaction in active set become zero. In this case, as the LARS algorithm, the variable moves to $\mathcal{B}$. Second case is that the coefficient of a main effect become zero, some of corresponding interactions are in $\mathcal{A}$, then we do not move the main effect to $\mathcal{B}$ to make weak heredity hold. Finally, if the coefficient of a main effect become zero, corresponding interactions are not in $\mathcal{A}$, then we remove the main effect from $\mathcal{A}$. we summarize this modification below.

HARS modification

- HLARS updates $\beta_u(\delta) = \hat{\beta}_u + \delta w_u$ for $u \in \mathcal{A}$

- Therefore $\beta_u(\delta)$ will changes sign at

$$\delta_u = -\hat{\beta}_u / w_u.$$

- The first such change occurring at

$$\tilde{\delta} = \min_{u>0}\{\delta_u\}$$

say for covariate $X_{\tilde{u}}$.

1. If $\tilde{u}$ is main effect
   (a) If $\exists v \in \mathcal{I}_{\tilde{u}}^L \cup \mathcal{I}_{\tilde{u}}^R$ such that $v \in \mathcal{B} - \mathcal{A}$,
   we move $\beta_{\mathcal{A}_+} = \beta_{\mathcal{A}} + \delta w \otimes s$.
   (b) If $\nexists v \in \mathcal{I}_{\tilde{u}}^L \cup \mathcal{I}_{\tilde{u}}^R$ such that $v \in \mathcal{B} - \mathcal{A}$,
   we move $\beta_{\mathcal{A}_+} = \beta_{\mathcal{A}} + \tilde{\delta} w \otimes s$ and $\mathcal{A}_+ = \mathcal{A} - \{\tilde{u}\}$
2. If $\tilde{u}$ is interaction, we move $\beta_{\mathcal{A}_+} = \beta_{\mathcal{A}} + \tilde{\delta} w \otimes s$ and $\mathcal{A}_+ = \mathcal{A} - \{\tilde{u}\}$
3. After deletion step, repeat (1)$\sim$(10) in HLARS algorithm.

3.2 MPI algorithm for HLARS

In this section, we will develop a MPI-version algorithm of HLARS. If the number of covariates is very large like SNP data, or market basket data for recommendation system, we cannot execute the HLARS algorithm in a single machine due to various reasons. First, data cannot be allocated in a single machine memory. Frequently SNP data and market basket data

are larger than 1GB and if we want to fit two way interaction model, the transformed data is much larger than 1GB and, it cannot be allocated in a single machine. Second, it is very hard to calculate correlation between covariates and residuals since it needs to compute $n \times (p + p(p-1)/2)$ multiplication and n times summation.

In these case, we resolve above two problem by a distributed algorithm. There are many computational tools for distributed processes and MPI is a widely used library for this purpose. By some modification of the HLARS algorithm, we can execute the HLARS algorithm in MPI environment and it resolves above two problems.

3.2.1 Modification of HLARS

For very high dimensional data it is hard to allocate data on a single machine and to solve this problem we have to decompose the data into several pieces of sub data and split it each slave node. In the modified HLARS algorithm all of our calculation will be done in slave nodes. Figure 3.2 shows how to decompose very large data into slave nodes. Each slave node has some main effects on each memory.

In calculating the correlations, we need two steps. the first is to calculate correlation between main effects with residuals. To do this, master node sends residuals to slave node. Each slave node calculates correlations for its main effects and then finds maximum correlation and corresponding main effect. After that slave nodes send the information about maximum correlations and related main effects to the master node and the master node receives maximum correlations and corresponding main effects from slave nodes. Figure 3.3 shows computation of correlations between main

effect and residuals.

For calculating correlation corresponding interaction effects related with covariate X_j, master node receives X_j from slave node which has X_j, and sends it to all the other slave nodes. Each slave node computes correlation between X_jX_k and current residuals, and find maximum correlation and its interaction effects. Finally master node receives maximum correlations and interaction effects from slave nodes.

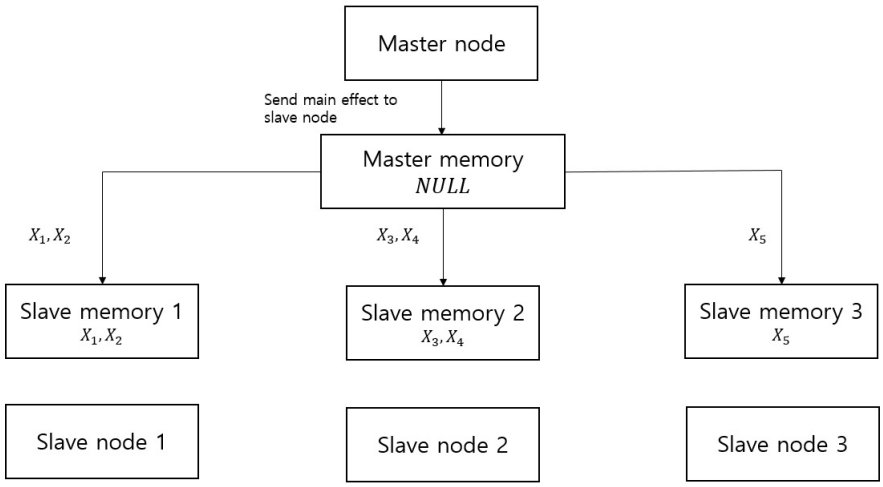

Figure 3.2: How to split data

Figure 3.2 - Figure 3.4 show the flow chart of MPI modified HLARS algorithm's concept. Brief explanation of MPI-modified HLARS algorithm (MPI-HLARS) is described below.

Idea of MPI-HLARS

1. At each slave, calculate correlation between residuals and their

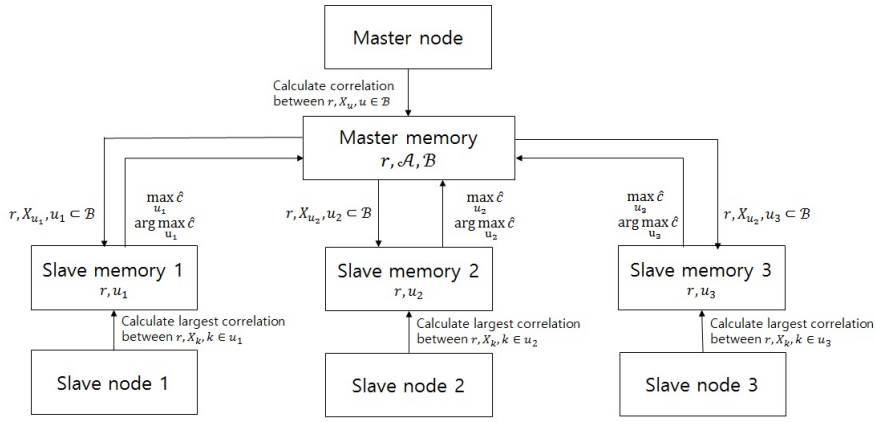

Figure 3.3: Compute correlation between main effect and residuals

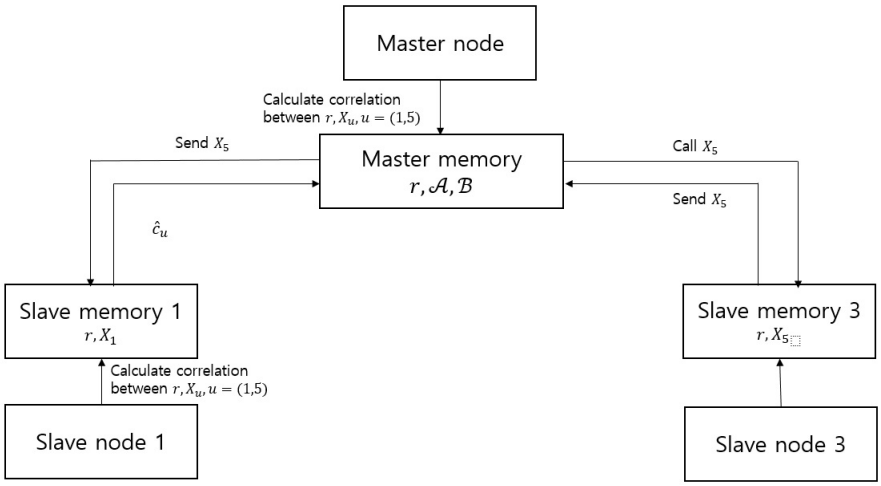

Figure 3.4: Compute correlation between interaction effect and residuals

main effect except $\mathcal{A}$.

2. Let u be the variable (only main effect) in $\mathcal{A}$.

3. At master node receive X_u from some node and send it to other nodes.

   (a) At slave node receive X_u, calculate correlation of interaction effect except $\mathcal{A}$ and send the maximum correlation and its interaction to master node.

4. At master node calculate maximum correlation of main and interaction effect and update β.

In MPI-HLARS, neither master and nor slave nodes should not keep candidate set $\mathcal{B}$. If we execute HLARS in a single machine with relatively small data, having $\mathcal{B}$ makes HLARS much faster. However when we analyze large data with high dimension, $\mathcal{B}$ needs lots of memory($k \times p$ byte where k is number of main effect in $\mathcal{A}$). Since $\mathcal{B}$ can be calculated from $\mathcal{A}$, we don't have to hold $\mathcal{B}$ in memory but it suffices to calculate correlation corresponding main effects not in $\mathcal{A}$ and corresponding interaction effects not in $\mathcal{A}$.

The MPI-HLARS algorithm is as follows. Let $\mathcal{A} = \emptyset$, $\mathcal{M} = \{1, \cdots, p\}$, $X_{\mathcal{A}} = \emptyset$ and $\beta = 0$. Define the rank of the master node as 0 and rank of the kth slave node as k, $k = 1, \cdots, P$.

MPI-HLARS algorithm

1. Let $\hat{\beta}_{\mathcal{A}}$ is current HLARS estimate.

2. If master node

   (a) Compute $r = Y - \tilde{\mathbf{X}}_{\mathcal{A}}\hat{\beta}_{\mathcal{A}}$

   (b) Send r to slave node

3. If slave node

   (a) Receive r from master node

   (b) Compute $\hat{c}_k = \max_{j\in\mathcal{M}-\mathcal{A}} |X_j' r|$, $\hat{j}_k = \text{argmax}_{j\in\mathcal{M}-\mathcal{A}} |X_j' r|$

   (c) Send $\hat{c}_k$, $\hat{j}_k$ and $X_{\hat{j}_k}$ to master node

4. If master node

   (a) Compute $\hat{c} = \max_k \hat{c}_k$, $\hat{k} = \text{argmax}_k \hat{c}_k$, $u = \hat{j}_{\hat{k}}$ and $X_{\mathcal{A}} = X \cup X_{\hat{j}_{\hat{k}}}$

   (b) If $u \in \{1, \cdots, p\}$, $\mathcal{A} = \mathcal{A} \cup \{u\}$ and send X_u to slave node.

      i. If slave node

         A. Recieve $X_{\hat{j}_{\hat{k}}}$ from master node

         B. Compute $\hat{c}_k^I = \max_{(u,j)\in\mathcal{M}-\mathcal{A}} |X_u X_j' r|$, $(u, \hat{j}_k) = \text{argmax}_{(u,j)\in\mathcal{M}-\mathcal{A}} |X_u X_j' r|$

         C. Send $\hat{c}_k^I$, $(u, \hat{j}_k)$ and $X_u X_{\hat{j}_k}$ to master node

   (c) Compute $\hat{c}^I = \max_k \hat{c}_k^I$, $\hat{k} = \text{argmax}_k \hat{c}_k^I$ and $\tilde{u} = (u, \hat{j}_{\hat{k}})$

   (d) if $\hat{c}^I > \hat{c}$, $\mathcal{A} = \mathcal{A} \cup \tilde{u}$ and $X_{\mathcal{A}} = X \cup X_u X_{\hat{j}_{\hat{k}}}$

   (e) Compute

$$
\begin{aligned}
S &= (\cdots, s_j, \cdots)_{j\in\mathcal{A}},\ s_j = Sign(\hat{c}_j),\ s_j \in \{-1, 1\} \\
w_{\mathcal{A}} &= (\tilde{\mathbf{X}}_{\mathcal{A}}' \tilde{\mathbf{X}}_{\mathcal{A}})^{-1} \tilde{\mathbf{X}}_{\mathcal{A}}' r \\
Cmax &= \max_i \tilde{\mathbf{X}}_{\mathcal{A}}' r
\end{aligned}
$$

(f) Let $r(\delta) = r - \delta \tilde{\mathbf{X}}_{\mathcal{A}} w_{\mathcal{A}}$

(g) Compute

$$\begin{aligned} \hat{\delta} &= \min \left\{ \delta : \max_{i \in \mathcal{A}} |\tilde{\mathbf{X}}_i' r(\delta)| = |\tilde{\mathbf{X}}_j' r(\delta)|, \text{ for } j \in \mathcal{B} \right\} \\ &= \min \{ \nu_j : \nu_j > 0, j \in \mathcal{B} \} \end{aligned}$$

where $\nu_j = \min_{j \in \mathcal{B}} \left(\frac{Cmax + \tilde{\mathbf{X}}_j' r}{Cmax + \tilde{\mathbf{X}}_j' \tilde{\mathbf{X}}_{\mathcal{A}} w_{\mathcal{A}}}, \frac{Cmax - \tilde{\mathbf{X}}_j' r}{Cmax - \tilde{\mathbf{X}}_j' \tilde{\mathbf{X}}_{\mathcal{A}} w_{\mathcal{A}}} \right)$

(h) Update $\hat{\beta}_{\mathcal{A}} = \hat{\beta}_{\mathcal{A}} + \hat{\delta} w_{\mathcal{A}} \otimes S$

5. Repeat (1)$\sim$(4) until $|\mathcal{A}| = \min(n, p + p(p-1)/2)$.

Chapter 4

Numerical studies

4.1 Simulation studies

In this section, we investigate the performance of the proposed HLARS algorithm. In particular, we compare our algorithm with other methods, LARS with full interactions, LARS with pre-screening main effects and corresponding interactions (LARS-M) and LASSO with hierarchical interaction(HLASSO) suggested by Bien et al. (2013). For LARS-M, we fit LARS with main effects only and using cross validation choice subset of main effect. After that execute the LARS algorithm with selected main effects and related interactions which satisfy the weak heredity.

We consider 4 scenarios as Bien et al. (2013);

1. Weak hierarchy: $\beta_{jk} \neq 0 \Rightarrow \beta_j \neq 0$ or $\beta_k \neq 0$
2. Weak hierarchy with $\|B_j\|_1 > |\beta_j|$ for some $j = 1, \cdots, p$
3. Only interactions:$\beta_j = 0$ for all j

4. Only has main effects:$\beta_{jk} = 0$ for all j, k

We compare 4 method - HLARS, LARS, LARS-M, HLASSO with the 4 Scenarios. The number of main effects set to be $p \in \{5, 50, 100\}$ and the sample size is set to be $n = 200$. For each Scenario, Data are generated as follow.

1. $X \sim MVN(0, \Sigma)$
2. Generate β_j, β_{jk} from $U(-5, 5)$ to meet 4 scenarios.
   (a) scenario1 : choose 5 interaction which are not sparse and generate $\beta_{jk} \sim U(-5, 5), j < k$ and generate $\beta_j \sim U(-5, 5), j = 1, \cdots, p$.
   (b) scenario2 : choose 5 interaction which are not sparse and generate $\beta_{jk} \sim U(-5, 5), j < k$ and generate $\beta_j \sim U(-\|B_j\|_1, \|B_j\|_1), j = 1, \cdots, p$.
   (c) scenario3 : choose 5 interaction which are not sparse and generate $\beta_{jk} \sim U(-5, 5), j < k$
   (d) scenario4 : choose 5 main effect which are not sparse and generate $\beta_j \sim U(-5, 5), j = 1, \cdots, p$
3. Generate $Y \sim \sum_j \beta_j X_j + \sum_{j<k} \beta_{jk} X_j X_k + N(0, 1)$

For the variance-covariance matrix Σ of X, we consider two forms; (1) independent $\Sigma = \sigma^2 I$ and (2) compound symmetry $\Sigma = \sigma^2 \Sigma^0$ with $\Sigma^0_{jj} = 1, \Sigma^0_{jk} = \rho$.

We choose the tuning parameter λ by 5-fold cross validation for HLARS $\lambda = \|\beta\|_1$. Results based on 100 replications of simulation are summarized in the following 2 subsections.

4.1.1 Simulation 1

In this subsection, we compare 4 algorithms for 4 Scenarios when the variance-covariance matrix of X is $\sigma^2 I$. For Scenario 1, figure 4.1 shows

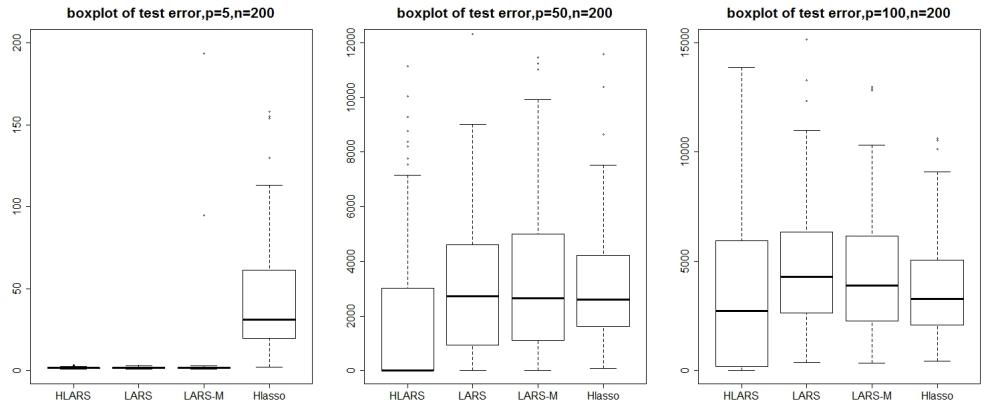

Figure 4.1: Test error: Scenario 1, independent covariance

that HLARS has performance as good as LARS, LARS-M, and HLASSO algorithms.

For Scenario 2, Figure 4.2 shows LARS has best performance while HLARS has similar performance to HLASSO and LARS-M.

For Scenario 3, Figure 4.3 shows LARS is best. Because of Scenario 3 assume no main effect but only interaction, HLARS, LARS-M and HLASSO have poor performance since they assume the weak hierarchy of interaction.

Finally for Scenario 4, Figure 4.4 shows HLARS has best performance. In scenario 4, there exist only main effect. Since the candidate set of HLARS consists of all of the main effects and some interactions, but in

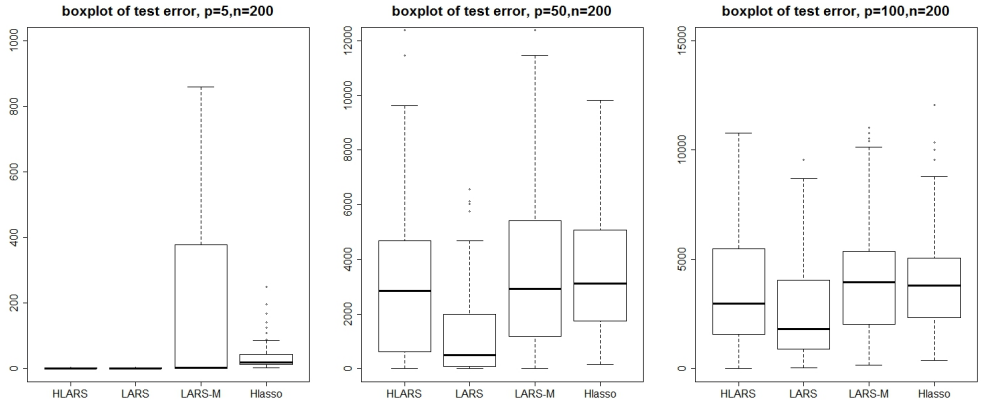

Figure 4.2: Test error: Scenario 2, independent covariance

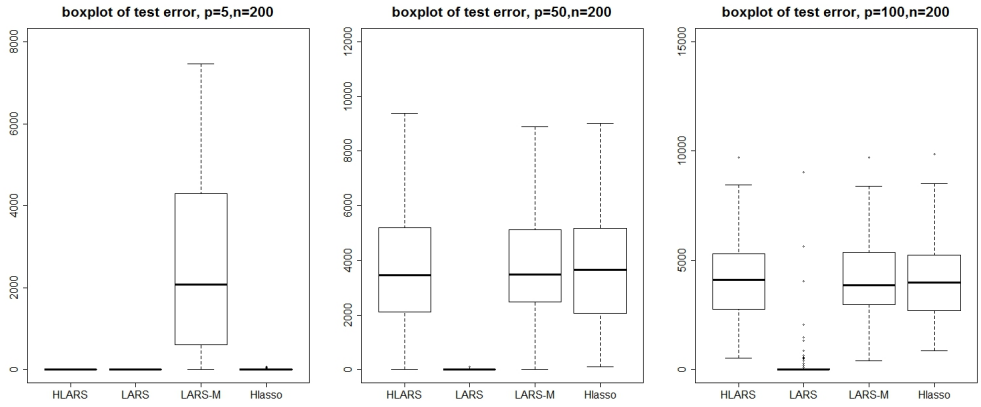

Figure 4.3: Test error: Scenario 3, independent covariance

LARS the inactive set is consist of some main effects and all of interactions. So HLARS can choose main effects better than LARS does.

Table 4.1 show the sensitivity and specificity for variable selection.

Table 4.1: Sensitivity and specificity for Simulation 1

Scenario		sensitivity			specificity		
		p=5	p=50	p=100	p=5	p=50	p=100
1	HLARS	1.0000	0.7732	0.5689	0.1761	0.9807	0.9956
	LARS	1.0000	0.6729	0.3692	0.1929	0.9747	0.9951
	LARS-M	0.9822	0.5863	0.3876	0.2559	0.9797	0.9955
	HLASSO	0.9750	0.8095	0.6474	0.7170	0.9613	0.9902
2	HLARS	0.9943	0.4206	0.3608	0.1998	0.9885	0.9973
	LARS	0.9965	0.6859	0.4686	0.2589	0.9732	0.9935
	LARS-M	0.8150	0.3267	0.2682	0.4431	0.9879	0.9967
	HLASSO	0.9739	0.6380	0.5086	0.7337	0.9687	0.9914
3	HLARS	1.0000	0.1940	0.0780	0.1750	0.9923	0.9988
	LARS	0.9980	0.9840	0.9200	0.3990	0.9750	0.9930
	LARS-M	0.3980	0.1120	0.0440	0.8030	0.9957	0.9992
	HLASSO	0.9760	0.4280	0.2820	0.4420	0.9723	0.9942
4	HLARS	0.9940	0.9780	0.9820	0.6710	0.9880	0.9963
	LARS	0.9960	0.9620	0.9280	0.4620	0.9767	0.9931
	LARS-M	0.9960	0.9680	0.9720	0.4620	0.9781	0.9933
	HLASSO	0.9920	0.9820	0.9860	0.8070	0.9815	0.9941

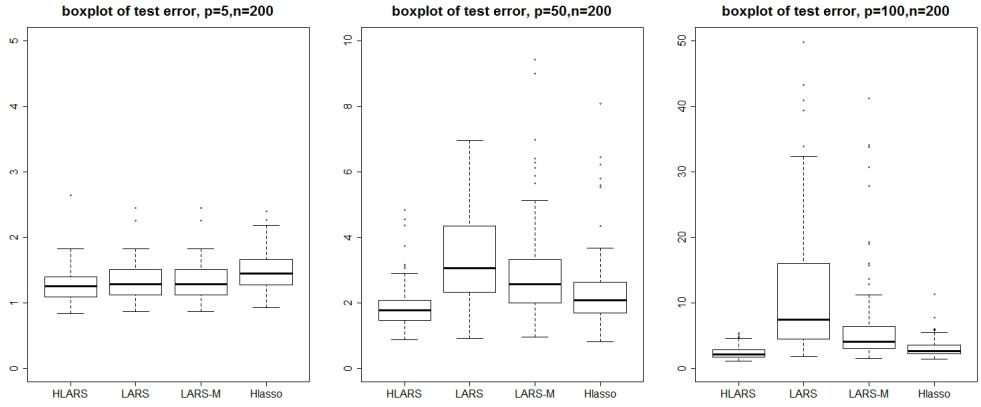

Figure 4.4: Test error: Scenario 4, independent covariance

HLARS has lower sensitivities than LARS and HLASSO but bigger sensitivities than LARS-M. While LARS and HLASSO searches all of main effects and interaction effects, HLARS searches in candidate set only, which makes HLARS has lower sensitivity.

Figure 4.5 - 4.8 show solution paths of HLARS and LARS for 4 scenarios. In the case of scenario 1 and 2, both solution paths given in Figure 4.5 and 4.6 are similar but in HLARS, we can notice that 2 variables be joined in active set simultaneously at the first step. It makes HLARS searches the solution quickly rather than LARS does. For Scenario 3, two solution paths given in Figure 4.7 looks different. Especially solution path of HLARS is strange. Note that in Scenario 3, there exists significant interaction effects and without significant main effect. That is Scenario 3 violates weak heredity, hence it is expected HLARS does not work well. For scenario 4, two solution paths given in Figure 4.8 are almost same.

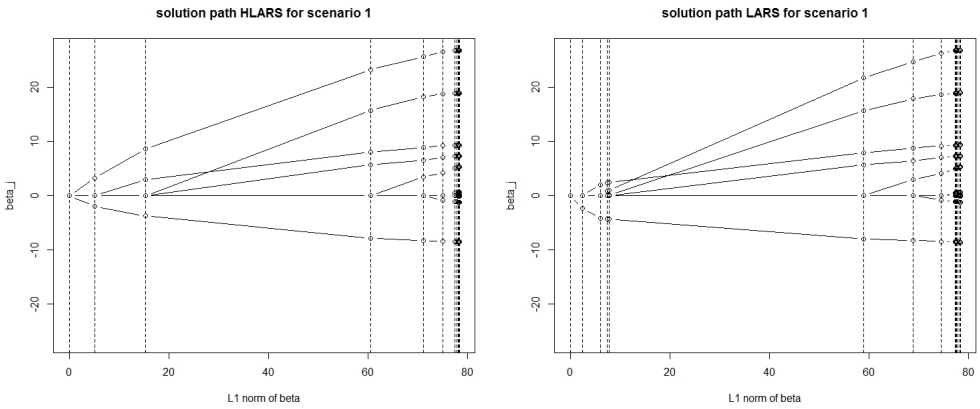

Figure 4.5: Solution path for HLARS and LARS for Scenario 1

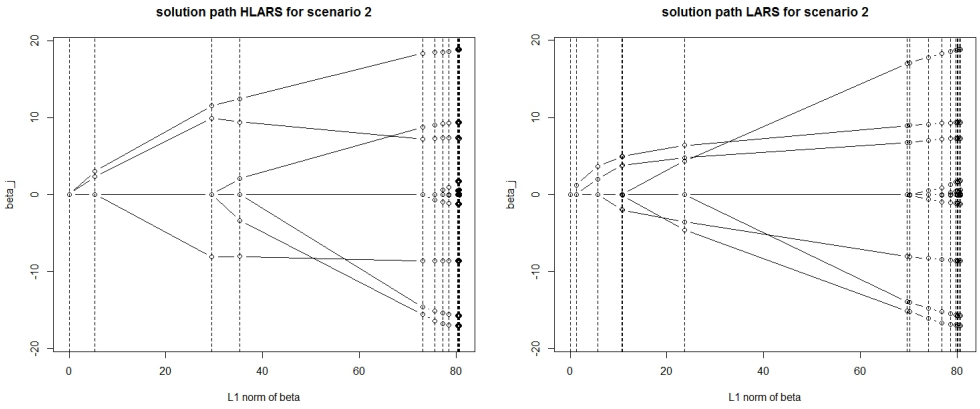

Figure 4.6: Solution path for HLARS and LARS for Scenario 2

4.1.2 Simulation 2

In this subsection, we compare 4 algorithm for 4 Scenarios when the variance-covariance matrix of X is $\Sigma = \sigma^2\Sigma^0$. Compare with simulation

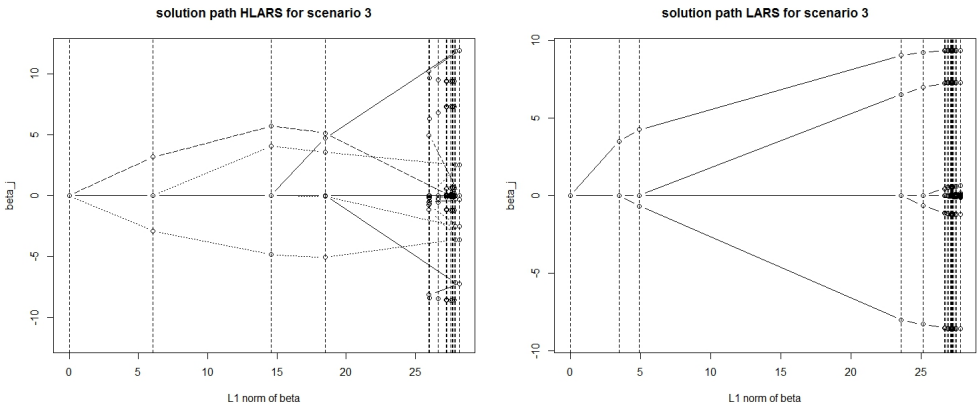

Figure 4.7: Solution path for HLARS and LARS for Scenario 3

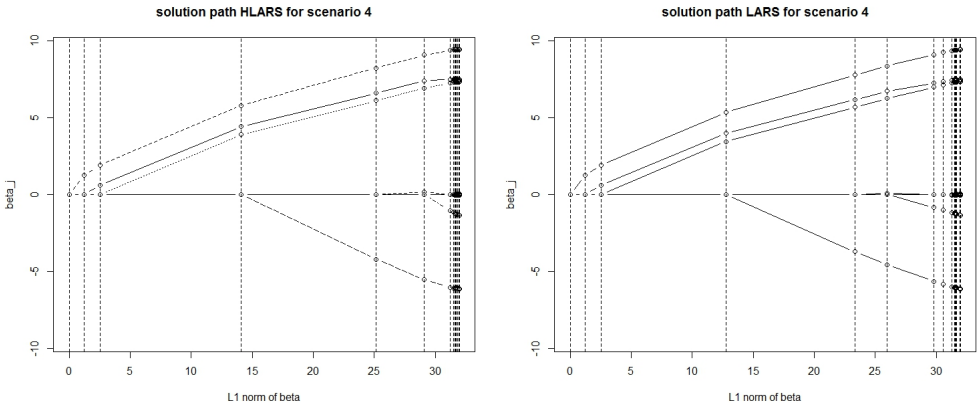

Figure 4.8: Solution path for HLARS and LARS for Scenario 4

1, LARS has better performance for test error and variable selectivity. We conjecture that the algorithms with hierarchical interaction may be barrable for the multicollinearity and we will do it in the future.

Table 4.2: Sensitivity and specificity for Simulation 2

Scenario		sensitivity			specificity		
		p=5	p=50	p=100	p=5	p=50	p=100
1	HALRS	1.0000	0.5654	0.4794	0.3638	0.9782	0.9945
	LARS	1.0000	0.7505	0.5970	0.3383	0.9719	0.9927
	LARS-M	0.8873	0.4013	0.3499	0.4807	0.9777	0.9946
	HLASSO	0.9810	0.6863	0.5628	0.6398	0.9579	0.9889
2	HALRS	0.9947	0.2903	0.2513	0.4195	0.9827	0.9960
	LARS	0.9907	0.6408	0.5003	0.3871	0.9714	0.9928
	LARS-M	0.5890	0.1491	0.1616	0.6070	0.9896	0.9963
	HLASSO	0.9837	0.5735	0.4467	0.6379	0.9614	0.9892
3	HALRS	0.9967	0.1417	0.0700	0.2792	0.9862	0.9968
	LARS	0.9967	0.9333	0.8280	0.4892	0.9719	0.9928
	LARS-M	0.4367	0.0467	0.0340	0.7800	0.9968	0.9986
	HLASSO	0.9767	0.3733	0.3020	0.4900	0.9587	0.9889
4	HALRS	1.0000	0.9780	0.9880	0.7438	0.9897	0.9965
	LARS	1.0000	0.9700	0.9220	0.6354	0.9794	0.9938
	LARS-M	1.0000	0.9760	0.9680	0.6362	0.9806	0.9941
	HLASSO	0.9900	0.9740	0.9840	0.8531	0.9886	0.9960

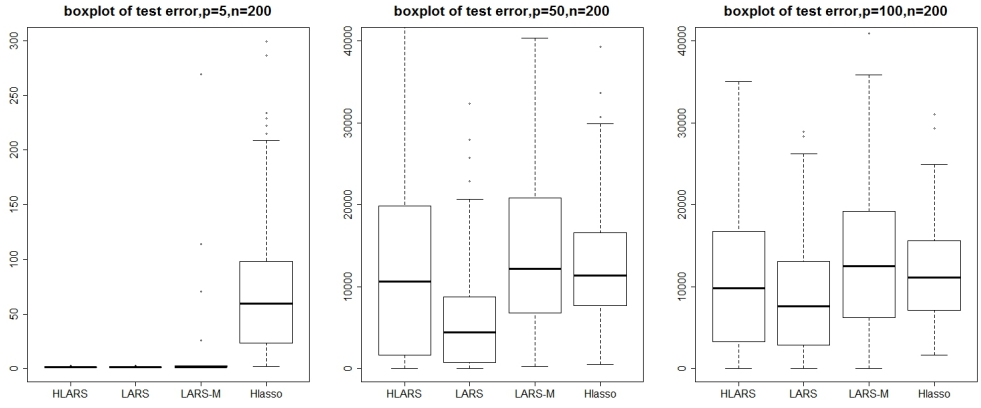

Figure 4.9: Test error: Scenario 1, compound symmetry

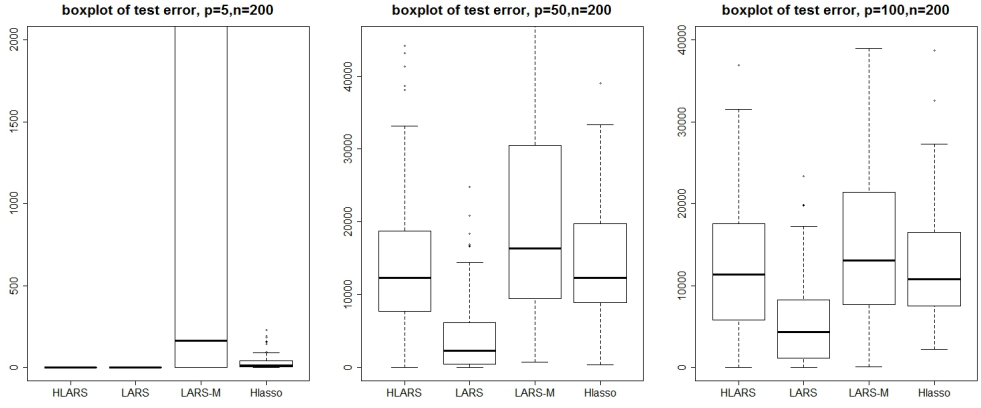

Figure 4.10: Test error: Scenario 2, compound symmetry

4.1.3 Simulation result for MPI modified HLARS

In this subsection we calculate the computational time for MPI modified HLARS. Computer node specification is like this. We use 8 Intel 3210 Xeon

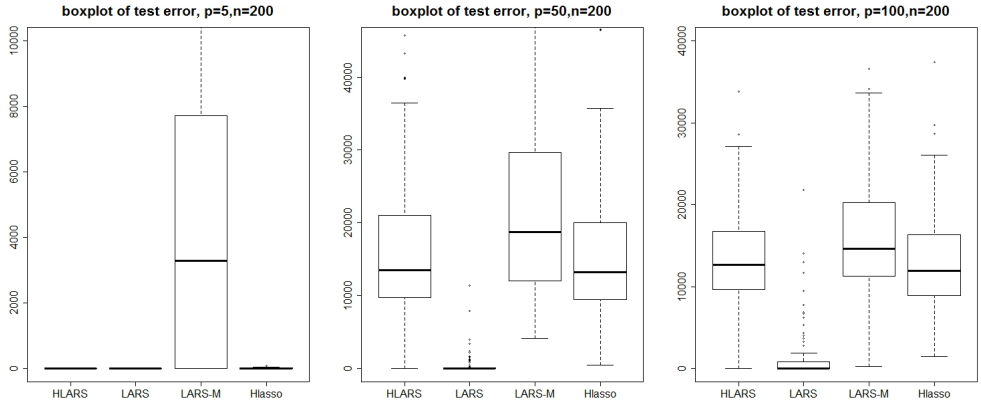

Figure 4.11: Test error: Scenario 3, compound symmetry

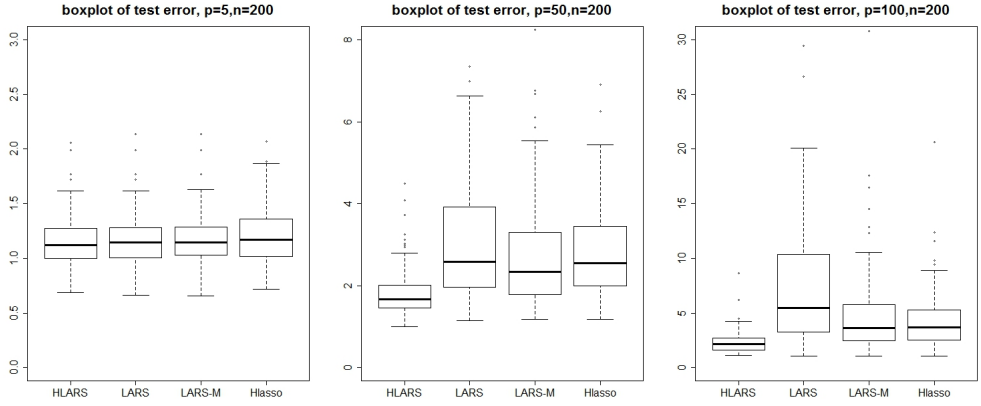

Figure 4.12: Test error: Scenario 4, compound symmetry

2.13GHz Quad Core CPU(32 cores) and 8 GB memory at each nodes. OS is Windows HPC server 2008 R2 and software is Windows HPC cluster pack. We simulate $n = 300$ and $p = 200, 500, 1000$. Figure 4.13 show the

result of simulations. As the number of cores increases we need more communication times. However computational cost of each node is decreasing and thus overall computational time does not increasing linearly. Our conjecture is that if the number of nodes are sufficiently large, computational time will decrease. It make us to use MPI modified HLARS to very high dimensional data, i.e. we can use HLARS algorithm to estimate regression coefficient by scale-up.

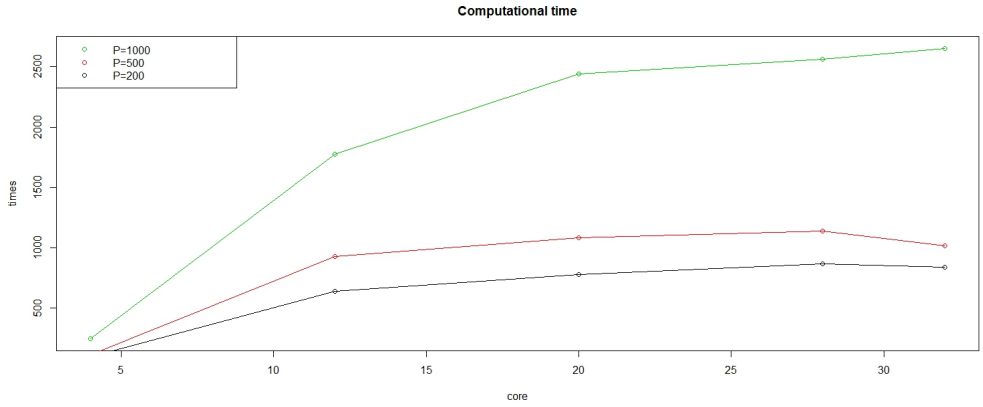

Figure 4.13: Computational time for MPI HLARS

4.2 Real data analysis

We analyze heights of 7000 individuals with 304245 SNPs. We fit simple linear regression for pre-screening and choose 10000, 50000 main effects which has less sum of squared residuals than the others. After that we execute the HLARS algorithm for 2 cases, 10000 main effects and 50000 main effects. In the HLARS algorithm, if the number of observation and

dimension of covariates are large together, it take much of times to execute the HLARS algorithm. So we restrict the number of steps less than 100. Since it takes much of times to choice tuning parameter λ by cross-validation, we choice λ in a validation set. In estimating β by HLARS algorithm, first, we randomly divide 7000 individuals into 3500 training set, 1250 validation set and 1250 test set. Second execute the HLARS algorithm with the training set and estimate the tuning parameter λ and β in the validation set. Finally we calculate test error for test set.

In the HLARS algorithm, we have to normalize the main and interaction effects. However our data has only main effects and at each steps normalize operation executed repeatedly. So we modify this job by doing normalize all of main effects at slave node and estimate coefficients of interactions without normalize. It is denoted by HLARS-WIN. Table 4.3 shows test error of the HLARS algorithm and LARS (without interactions) based 10 repetition. Table 4.4 shows computational time for execute the HLARS and HLARS-WIN. Three methods as similar test error but the HALRS has a best performance. Computational time of the HLARS is much better than HLARS-WIN since HLARS-WIN does not normalize interaction effects and it makes effect of interaction smaller corresponding large component main effect. So the HLARS-WIN has active set with main effects only and it make $\mathcal{B}$ larger than HLARS.

Table 4.3: Test error for real data analysis

number of main effects	HLARS	HLARS-WIN	LARS
10000	75.2116	75.5698	75.3021
50000	75.1125	75.2558	75.2009

Table 4.4: Computational time (sec)

number of main effects	HLARS	HLARS-WIN
10000	412.3340	6223.8990
50000	476.8510	8910.6990

Chapter 5

Concluding remarks

In this thesis, We suggested hierarchical LARS algorithm called HLARS to solve linear regression problem with interactions. The proposed method has several advantages over existing methods for hierarchical linear model. Advantages of our model are (1) computationally fast than other methods, (2) parallelization is possible and (3) parallelizing and scale up make it possible to analyze very ultra high dimensional data. In the near future, we will extend the HLARS algorithm to the logistic regression problems.

Bibliography

Hirotugu Akaike. Information theory and an extension of the maximum. In *Second International Symposium on Information Theory*, pages 267–281, 1973.

Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012a.

Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012b.

Jacob Bien, Jonathan Taylor, Robert Tibshirani, et al. A lasso for hierarchical interactions. *The Annals of Statistics*, 41(3):1111–1141, 2013.

Leo Breiman et al. Heuristics of instability and stabilization in model selection. *The annals of statistics*, 24(6):2350–2383, 1996.

Hugh Chipman. Bayesian variable selection with related predictors. *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, pages 17–36, 1996.

Nam Hee Choi, William Li, and Ji Zhu. Variable selection with the strong heredity constraint and its oracle property. *Journal of the American Statistical Association*, 105(489):354–364, 2010.

David R Cox. Interaction. *International Statistical Review/Revue Internationale de Statistique*, pages 1–24, 1984.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel computing*, 22(6):789–828, 1996.

William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 2014.

Michael Hamada and CF Jeff Wu. Analysis of designed experiments with complex aliasing. *Journal of Quality Technology*, 24(3):130–137, 1992.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Rodolphe Jenatton, Julien Mairal, Francis R Bach, and Guillaume R Obozinski. Proximal methods for sparse hierarchical dictionary learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 487–494, 2010.

Rodolphe Jenatton, Jean-Yves Audibert, and Francis Bach. Structured variable selection with sparsity-inducing norms. *The Journal of Machine Learning Research*, 12:2777–2824, 2011.

Ewing Lusk, S Huss, B Saphir, and M Snir. Mpi: A message-passing interface standard, 2009.

Colin L Mallows. Some comments on c p. *Technometrics*, 15(4):661–675, 1973.

JA Nelder. A reformulation of linear models. *Journal of the Royal Statistical Society. Series A (General)*, pages 48–77, 1977.

Michael R Osborne, Brett Presnell, and Berwin A Turlach. On the lasso and its dual. *Journal of Computational and Graphical statistics*, 9(2): 319–337, 2000a.

Michael R Osborne, Brett Presnell, and Berwin A Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis-Institute of Mathematics and its Applications*, 20 (3):389–404, 2000b.

Julio L Peixoto. Hierarchical variable selection in polynomial regression models. *The American Statistician*, 41(4):311–313, 1987.

Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Sanford Weisberg. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497, 2009.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

국문초록

변수선택은 고차원 회귀모형에서 중요하다. 단계별 변수선택법과 같은 전통적인 변수 선택 방법은 자료에 따라서 선택된 변수들이 변하므로 불안정하다. 이에 대한 대안으로 변수선택과 추정을 동시에 하는 성긴 벌점화 방법들이 사용된다. 라쏘는 대표적인 성긴 벌점화 방법 중 하나이며 라스 알고리즘의 수정을 통해 라쏘의 추정량을 쉽게 계산할 수 있다.

고차원 회귀모형에서 중요한 문제중 하나는 설명변수와 반응변수간의 관계를 주효과 만으로 쉽게 설명하기 힘든 경우가 발생한다는 것이며 많은 연구자들이 설명변수와 반응변수의 관계를 설명하기 위하여 고차 상호작용을 모형에 포함시키는 것에 대해 관심을 가지고 있다. 특히 공변량이 많은 이원요인 상호작용 모형에서 반응변수에 대한 설명력이 높은 주효과 혹은 상호작용 효과를 결정하는 변수선택 방법들이 제안되고 있다.

그러나 공변량의 갯수가 많은 경우에 모든 가능한 상호작용 효과를 모형에 포함시키는 것은 그 계산량이 매우 많은 문제가 있다. 성긴 벌점화 방법론에서는 이러한 문제를 해결하기 위하여 주효과와 상호작용 효과간에 유전적 구조를 부여하여 계산량을 줄임과 동시에 주효과와 상호작용 효과를 추정하고자 한다. 하지만 성긴 벌점화 방법론의 알고리즘들은 주효과의 갯수가 매우 많은 경우에 여전히 계산량이 많은 문제를 가지고 있다. 본 연구에서는 이를 해결하기 위해 계층적 라스 알고리즘을 개발하였다.

제안한 계층적 라스 알고리즘은 기존의 라스 알고리즘을 수정한 것이며 라스 알고리즘 보다 빠르면서 예측에 있어서 비슷한 수준의 정확성을 보여준다. 또한 병렬처리가 가능한 장점을 가지고 있어서 병렬처리 규모를 확대 가능하다. 메시지 패싱 인터페이스는 병렬처리 모형중에서 가장 잘 알려져 있으며 본 논문에서는 메시지 패싱 인터페이스를 활용한 계층적 라스 알고리즘을 제안하였다.

주요어 : 고차원 회귀모형, 이원요인 상호작용 모형, 라쏘, 라스 알고리즘, 메시지 전달 인터페이스

학 번 : 2011 – 30088