Variable Ranking by Solution-path Algorithms

by

Bo Wang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Statistics

Waterloo, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Variable Selection has always been a very important problem in statistics. We often meet situations where a huge data set is given and we want to find out the relationship between the response and the corresponding variables. With a huge number of variables, we often end up with a big model even if we delete those that are insignificant. There are two reasons why we are unsatisfied with a final model with too many variables[4]. The first reason is the prediction accuracy. Though the prediction bias might be small under a big model, the variance is usually very high. The second reason is interpretation. With a large number of variables in the model, it's hard to determine a clear relationship and explain the effects of variables we are interested in.

A lot of variable selection methods have been proposed, such as Forward- and Backward-Stepwise Selection, Forward-Stagewise Regression, shrinkage methods like Ridge Regression, Lasso and Least Angle Regression. However, one disadvantage of variable selection is that different sizes of model require different tuning parameters in the analysis, which is hard to choose for non-statisticians. Xin and Zhu [8] advocate variable ranking instead of variable selection. Once variables are ranked properly, we can make the selection by adopting a threshold rule. One possible variable ranking can be constructed using Random Forest, which gives two different variable importance measures. We can use either one to get a ranking and usually they lead to similar results.

In this thesis, we try to rank the variables using Least Angle Regression (LARS). Some shrinkage methods like Lasso and LARS can shrink the coefficients to zero. The advantage of this kind of methods is that they can give a solution path which describes the order that variables enter the model. This provides an intuitive way to rank variables based on the path. However, Lasso can sometimes be difficult to apply to variable ranking directly. This is because that in a Lasso solution path, variables might enter the model and then get dropped. This dropping issue makes it hard to rank based on the order of entrance. However, LARS, which is a modified version of Lasso, doesn't have this problem. We'll make use of this property and rank variables using LARS solution path.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. Mu Zhu for his supervision, encouragement and support. He taught me how to think and gave me valuable advice and comments which helped me a lot during the research.

Thank Prof. Ali Ghodsi and Prof. Guangzhe Fan for being my second readers and giving valuable comments.

Special thanks to Mary Lou Dufton who helped me a lot during my master study.

I would also like to thank my dear friends Feng He, Wei Wei, Xiao Hao, Haocheng Li, Zhiyue Huang and my best partner Yang Yu for their friendship and support.

Finally, I owe my deepest gratitude to my parents and sister who bring love and hope to my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Ranking and selection have a lot in common. They both intend to end up with a smaller model which sacrifices a little bias to get better prediction variance and interpretation. Ranking can be regarded as an advanced form of selection. Once variables are ranked properly, selection can be performed by thresholding[8]. On the other hand, selection provides the basis for ranking. We often get insights from model selection on assigning a proper rank to each of the variables. Various model selection algorithms have been proposed in history.

1.1 Forward- and Backward-Stepwise Selection

In Forward-Stepwise Selection, we start from the null model. All variables that are not currently in the model are added one at a time. The one that improves the objective function (e.g. AIC or BIC) most is kept and added into the model. In Backward-Stepwise Selection, we start from the full model. All variables that are currently in the model are deleted one at a time. The one that improves the objective function (e.g. AIC or BIC) most is deleted from the model.

The advantages of Forward- and Backward-Stepwise Selection are: First, the computation cost is relatively low compared to best subset selection. When the number of variables

is large, best subset selection is not applicable. However, we can always compute Stepwise Selection. Second, Stepwise Selection is a more constrained method compared to best subset selection[4]. Some prediction accuracy is sacrificed for lower variance.

The disadvantages of Forward- and Backward-Stepwise Selection are: First, they are greedy algorithms and the resulted sequence is nested. They are actually selecting suboptimal subsets. Second, Backward selection can only be used when $N > p$.

1.2 Stochastic Stepwise Selection

Xin and Zhu [8] proposed a stochastic version of the stepwise selection: the Stochastic Stepwise(ST2) selection. Instead of adding or deleting one variable at a time, the ST2 algorithm adds or deletes a group of variables at a time. The size of the group is randomly decided. In order to avoid the huge number of possible groups when group size is greater than one, ST2 algorithm randomly assesses some groups and adds or deletes the optimal one. The algorithm is described as follows:

Repeat

1. (Forward Step) Start with variables that are not in the current model.
   - (i) Determine the size of group that are to be added into the model, denoted by d_f.
   - (ii) Determine the number of groups that are to be assessed, denoted by k_f.
   - (iii) Randomly select k_f groups of variables of size d_f.
   - (iv) Assess each group by adding it into the current model. The group that improves the objective function (e.g. AIC or BIC) most is added into the model.
2. (Backward Step) Start with variables that are in the current model.
   - (i) Determine the size of group that are to be deleted from the model, denoted by d_b.
   - (ii) Determine the number of groups that are to be assessed, denoted by k_b.
   - (iii) Randomly select k_b groups of variables of size d_b.

(iv) Assess each group by deleting it from the current model. The group that improves the objective function (e.g. AIC or BIC) most is deleted from the model.

Until neither forward nor backward step can provide any additional improvement.

The stochastic stepwise algorithm can be used to construct a variable selection ensemble (VSE). Variable ranking can be obtained based on VSE.

1.3 Forward-Stagewise Regression

Forward-Stagewise Regression is a cautious version of Forward-Selection. The algorithm assigns small updates to the coefficient of one variable at a time which results in much more steps to get the full least square estimate compared to Stepwise Selection which only takes p steps. The algorithm can be described as follows (assume predictors are standardized):

(i) Start with $r = y, \beta_1, \beta_2, \ldots \beta_p = 0$. Here r denotes the residual, $\beta_1, \beta_2, ...\beta_p$ are the coefficients.

(ii) Find the predictor x_j most correlated with r.

(iii) Update $\beta_j \leftarrow \beta_j + \delta_j$.

(iv) Set $r \leftarrow r - \delta_j \cdot x_j$.

Forward-Stagewise Regression seems to be inefficient. But the slow fitting feature is usually quite competitive in high-dimensional problems[4]. Forward-Stagewise Regression is closely related to the Least Angle Regression (LARS). A modified version of LARS will lead to Forward-Stagewise Regression and allows it to be implemented using relatively larger steps.

1.4 Shrinkage methods Ridge Regression and Lasso

Ridge Regression and Lasso have similar form. They both impose a penalty on the size to the regression formula. They can be written in the following expression:

$$\text{Minimize} \quad \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{i=1}^{p} x_{ij}\beta_j)^2 \tag{1.1}$$

Subject to (**penalty**).

In Ridge Regression, the penalty is $\sum_{j=1}^{p} \beta_j^2 < t$. In Lasso, the penalty is $\sum_{j=1}^{p} |\beta_j| < t$. Different penalty forms result in completely different properties of Ridge Regression and Lasso.

In Ridge Regression, the L_2 penalty shrinks the coefficients proportionally. So no variable coefficient will be shrunk to zero. The penalty form of Ridge Regression can be equivalently written in the following way

$$\text{Minimize} \quad (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta \tag{1.2}$$

The λ here is the penalty parameter and can be determined by t. In this way we can easily get the solution: $\hat{\beta} = (X^TX + \lambda I)^{-1}X^Ty$. The advantage of Ridge is that adding λI to X^TX makes the regression problem non-singular, even if X^TX is not of full rank. However, as no coefficient is shrunk to zero, we cannot get a smaller model in this way.

In Lasso, there's no closed form of expression for the solution. However, Tibshirani [6] pointed out that if the design matrix X is orthonormal, then the solution can be written as

$$\hat{\beta}_j = sign(\hat{\beta}_j^0)(|\hat{\beta}_j^0| - r)^+ \tag{1.3}$$

where $\hat{\beta}_j^0$ is the full least square estimate and r is determined by the condition $\sum_{j=1}^{p} |\beta_j| < t$. When t is sufficiently small, r is large so that some of the coefficients are shrunk to zero. In this way we get a smaller model. Lasso also gives a solution path when the penalty t varies from 0 to $\hat{t} = \sum_{j=1}^{p} \hat{\beta}_j^0$. This is the basis how we rank the variables based on Lasso.

Tibshirani [6] also pointed out that the different properties of Lasso and Ridge can be

explained from a geometric perspective. The residual sum of square $(y - X\beta)^T(y - X\beta)$ equals $(\beta - \hat{\beta}_0)^T X^T X(\beta - \hat{\beta}_0)$ plus a constant. The solution of Lasso and Ridge Regression is actually the first place where the elliptical contours touch the bound of the penalty area. The L_2 penalty of Ridge Regression forms a penalty area which is a sphere while the L_1 penalty of Lasso forms a penalty area which is a square. When the touch is occurred at a corn, then some coefficients will be zero. However, in the penalty area of Ridge Regression, there's no corner so zero coefficients rarely happen. This geometric view is shown below in a two-dimensional case in Figure 1.1 and 1.2.

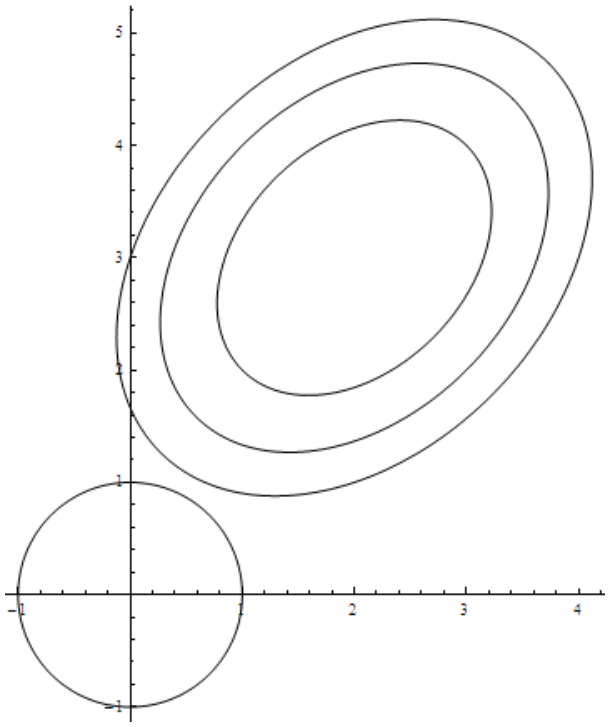

Figure 1.1: Ridge Regression

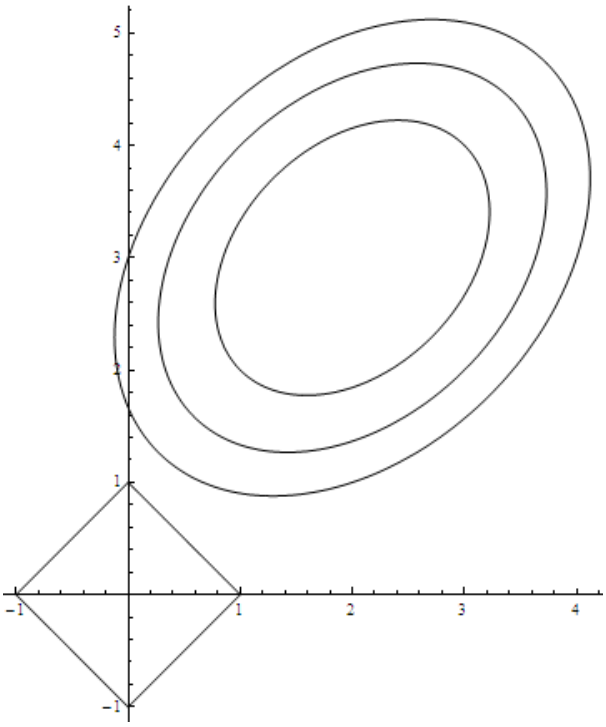

Figure 1.2: Lasso

1.5 Least Angle Regression

Least Angle Regression (LARS) is proposed by Efron et al. in 2004. LARS is an algorithm between Stagewise Regression and Stepwise-Selection. Like Forward-Stagewise Regression, none of the other variables are adjusted when a variable is added. However, LARS has relatively larger steps in updating the coefficients compared to Stagewise Regression. The LARS algorithm is described as follows (assume the predictors have been standardized):

(i) Start with the residual $r = y - \bar{y}, \beta_1, \beta_2, \ldots, \beta_p = 0$.

(ii) Find the variable x_j most correlated with the residual r.

(iii) Increase the coefficient of variable x_j, from 0 towards its least square coefficient until some other variable x_k has as much correlation as x_j with the current residual.

(iv) Move β_j and β_k in the direction which has the same angle with x_j and x_k, until some other competitor x_l has as much correlation with the current residual.

(v) Move the coefficients in the direction which has the same angle with x_j, x_k and x_l, until some other competitor has as much correlation with the current residual.

(vi) Continue until all variables are added.

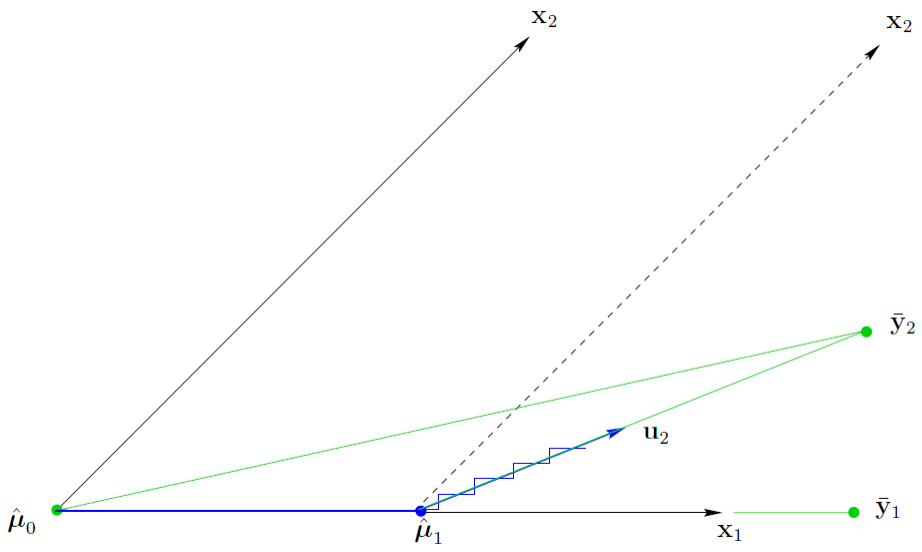

Figure 1.3: Least Angle Regression

Figure 1.3 [3] is an example of LARS when there exist two predictors. Suppose $\bar{\mathbf{y}}_2$ is the projection of the response $\mathbf{y}$ on the space spanned by $\mathbf{x_1}$ and $\mathbf{x_2}$. In this example the angle between $\bar{\mathbf{y}}_2$ and $\mathbf{x_1}$ is smaller than the angle between $\bar{\mathbf{y}}_2$ and $\mathbf{x_2}$, which means that the correlation between $\bar{\mathbf{y}}_2$ and $\mathbf{x_1}$ is larger than that between $\bar{\mathbf{y}}_2$ and $\mathbf{x_2}$. Then LARS algorithm proceeds as follows. First we start with $\hat{\boldsymbol{\mu}}_0 = 0$. Next we update $\hat{\boldsymbol{\mu}}$ in the direction of $\mathbf{x_1}$. The updated amount is chosen such that the residual has the same correlation with $\mathbf{x_1}$ and $\mathbf{x_2}$, which is $\hat{\boldsymbol{\mu}}_1$ in Figure 1.3. At last we update $\hat{\boldsymbol{\mu}}$ in the direction of the equiangular vector $\mathbf{u}_2$ until $\hat{\boldsymbol{\mu}} = \bar{\mathbf{y}}_2$. In contrast, for ordinary linear regression, the updated value for the first step is $\bar{\mathbf{y}}_1$, which is the projection of $\bar{\mathbf{y}}_2$ on $\mathbf{x_1}$.

LARS is a very important algorithm because some modifications will lead to effective implementation of Lasso and Forward-Stagewise Regression. We'll discuss this more in Chapter 3.

1.6 Random Forest

Different from the methods mentioned above, Random Forest[2] is a highly non-linear method. The idea is to build many correlated trees and take the average. The algorithm is:

(i) Take a bootstrap sample from the training data.

(ii) Grow a tree on the bootstrapped data. However, in splitting each node, only a randomly selected subset of variables is considered instead of the full variable set.

(iii) Repeat (i) and (ii) many times.

A very important aspect of random forest is that it gives two different variable importance measures which are very useful for variable ranking. One importance measure is defined as the sum of the improvement in the split criterion of the variable over all trees. The other one is constructed based on out-of-bag (OOB) samples. It's calculated as the average decrease in accuracy over all trees by randomly permuting the value of a certain variable of the OOB samples. We can rank variables based on these two importance measures which usually lead to similar results.

1.7 Outline

We proceed as follows. In Chapter 2 we'll first point out that Zhu et al.[10] doesn't consider the variable dropping issue when they use Lasso in variable ranking. Some empirical study is provided to show that it's very hard to explain variable dropping from the perspective of data structure. So we advocate ranking based on LARS instead of Lasso. In Chapter 3,

we give mathematical details of the subtle difference between Lasso and LARS and an explanation of variable dropping from the perspective of LARS. In Chapter 4 we test the performance of LARS ranking by several simulation examples. A real-data application for the Info-Rehab program is also given. In Chapter 5 we make a brief summary and discuss some potential methods to overcome the limitation of LARS ranking.

Chapter 2

Variable Dropping in Lasso

First we start with an application example. In Zhu et al.[10], machine learning algorithms are applied to clinical practice. The objective is to identify the most important variables for predicting whether to receive rehabilitation or not. Instead of ranking the variables using just one Lasso solution path, they use the ensemble approach of Lasso. 100 random samples, $S_1, S_2, ..., S_{100}$, of size $n = 10,000$ are taken from the original data set. Denote $r(b, j)$ as the rank of variable j in the b_{th} sample S_b. The final rank of variable j is calculated as

$$\bar{r}(j) = \frac{1}{100} \sum_{b=1}^{100} r(b, j) \tag{2.1}$$

The standard deviation $\sigma(j)$ of the rank of variable j can also be calculated and ranking can be obtained based on $\bar{r}(j)$ and $\sigma(j)$.

By adopting the ensemble approach of Lasso, they get a more stable variable ranking. However, they ignore a very important problem of Lasso. The sequence of models generated in the Lasso solution path is not necessarily nested. In some situations, a variable might first enter the model and then get dropped. With the existence of such dropping issue, it's hard to apply Lasso directly to rank the variables. In this chapter, we'll first discuss this dropping issue and then advocate ranking based on LARS instead of Lasso.

2.1 A Dropping Example

First we look at an example where variable dropping happens in Lasso solution path. The diabetes example is first used by Efron et al.[3]. In the diabetes data set, the response is a measure of disease progression one year after baseline, and the covariates are age, sex, body mass index, average blood pressure and six blood serum measurements named "tc", "ldl", "hdl", "tch", "ltg", "glu". The entire Lasso solution path is given below.

```
LARS Step 1 :  Variable 3  added
LARS Step 2 :  Variable 9  added
LARS Step 3 :  Variable 4  added
LARS Step 4 :  Variable 7  added
LARS Step 5 :  Variable 2  added
LARS Step 6 :  Variable 10  added
LARS Step 7 :  Variable 5  added
LARS Step 8 :  Variable 8  added
LARS Step 9 :  Variable 6  added
LARS Step 10 :  Variable 1  added
Lasso Step 11 :  Variable 7  dropped
LARS Step 12 :  Variable 7  added
```

We see that variable dropping does exist in Lasso. So we cannot directly apply Lasso to variable ranking.

2.2 Empirical Study of Dropping Mechanism

Because Lasso provides a solution path which is an intuitive way to rank variables, we still hope that we can make use of this solution path. Then the most important thing is to solve the dropping issue. Once the data set is given, Lasso solution path is completely determined. So if we can figure out a condition on the data set under which variables will be dropped, we might be able to come up with certain methods to solve the dropping issue.

2.2.1 A Possible Explanation: Multicollinearity

When running simulations, we do find that dropping is usually related to multicollinearity. As shown in (1.3), in the special case where X is orthogonal, the solution can be written as $\hat{\beta}_j = sign(\hat{\beta}_j^0)(|\hat{\beta}_j^0| - r)^+$. No dropping will occur under such assumption. This proves that dropping might be related to multicollinearity.

We still use the diabetes data as an example. From the solution path we can see that variable 7 ("hdl") is dropped at step 11 and at the same time variable 1 is added. Before this dropping step, every variable except variable 1 ("age") is in the model. If the penalty parameter increases by a small amount, we might not be able to add an "entire" variable to our model. However, if under the new penalty condition, the candidate variable increases the performance of the model more than an existing one, the candidate variable might be added into the model and one existing variable might be dropped. In the diabetes example, before the dropping step, variable 2 to 10 are in the model. So when variable 1 is added, all the other variables might be dropped.

We think that dropping is possibly related to the correlation structure of the data set in two aspects. We use i to denote the variable index. So variable indices at risk are $i = 2, 3, ..., 10$. We use C to denote the set of indices of variable that are current in the model and X_C to denote the corresponding part of the data matrix. The first possible relevant aspect is the multiple correlation between variable i and all the other variables currently in the model. We denote this value by $multi\_cor_i$,that is,

$$multi\_cor_i = cor(X_i, X_{C-\{i\}}) \tag{2.2}$$

where variable i is at risk of being dropped. If this value (consider absolute value) is large, then we can say that the effect of variable i can be explained by the other variables in the model. So variable i is more likely to be dropped.

The second possible relevant aspect is the correlation between variable i and the residuals of the regression between the response y and all the variables currently in the model

Table 2.1: Dropping in Lasso

Variable Index	$multi\_cor_i$	res_cor_i
2	0.2098	-0.1663
3	0.3374	0.2867
4	0.2891	0.1981
5	0.9831	-0.0118
6	0.9744	0.0108
7	0.9349	0.0058
8	0.8875	0.0177
9	0.9004	0.0650
10	0.3145	0.0403

except variable i. We denote this value by res_cor_i, that is,

$$res\_cor_i = cor(X_i, residual(y \sim X_{C-\{i\}})) \tag{2.3}$$

If this value (consider absolute value) is small, it means that there's little information we can get from adding the variable i. Then variable i might be dropped.

We calculate these two values for variables $i = 2, 3, ..., 10$ which are at risk and hope to find out the reason why variable 7 is the one that is actually dropped. The result is shown in table 2.1.

From the perspective of $multi\_cor_i$, variable 5,6,7 have the highest multiple correlation with the other variables. From the perspective of res_cor_i, variable 7 has the least correlation with the regression residuals. This might explain why variable 7 is dropped.

We think that there might be some tradeoff between $multi\_cor_i$, res_cor_i and some other potential factors which determine which variable will be dropped. In general, variables with high $multi\_cor_i$ and low res_cor_i are likely to be dropped.

One possible additional factor which might have an impact on dropping is the coefficient. Since the penalty is on the L_1 norm of the coefficients, variables with smaller coefficients (absolute value) might be added and those with larger coefficient might be dropped. In the diabetes example, from the Lasso result we can see that the coefficient of variable 7 is

-134.5994 before being dropped, while the coefficient of variable 1 is only -5.718948 after being added. This might also be a reason why variable 1 is added and variable 7 is dropped.

However, these effects need to be considered as a whole. We cannot get a reasonable explanation by considering just one of them. For example, for the coefficient effect mentioned above, the coefficient (absolute value) of variable 7 is actually not the largest among the current variables. But combined with other effects like the residual correlation, we might explain why variable 7 is dropped.

2.2.2 Sensitivity to the design matrix and response

In the previous section we try to explain dropping from the data structure and hope to find a sufficient condition on the data set under which variable dropping will happen. However, in our simulation study, we also find that dropping is very sensitive to the data structure. Even a very small perturbation might change the dropping situation in the path. Two examples are given below to illustrate this.

First we look at a very famous example which is first used by Tibshirani[6]. The model is

$$\boldsymbol{y} = \beta^T \boldsymbol{X} + \sigma \boldsymbol{\epsilon} \tag{2.4}$$

$\boldsymbol{\epsilon}$ is standard normal. $\boldsymbol{X}$ follows a multi-normal distribution and the correlation between x_i and x_j is $\rho^{|i-j|}$. In the original example by Tibshirani, the parameters are set as: $\beta = (3, 1.5, 0, 0, 2, 0, 0, 0)$, $\rho = 0.5$ and $\sigma = 3$. So there are three true variables in the model and the other five are noise.

We make some minor changes to the original parameters. We set $\beta = (3, 1.5, 0.01, 0.01, 2, 0.01, 0.01, 0.01)$ which makes the noise to be very weak signals. The correlation is changed to $cor(x_i, x_j) = 0.9^{|i-j|}$.

First we generate the design matrix $\boldsymbol{X}$ and response variable $\boldsymbol{y}$ according to the model. Then we get the Lasso solution path. Next we add i.i.d normal perturbation $N(0, 10^{-4})$ to the response $\boldsymbol{y}$ and recalculate the Lasso solution path to see if there's any change. The details of the simulation can be found in Appendix A.1. In order to show the structure change more clearly, we also calculate the correlation between response variable $\boldsymbol{y}$ and the

Table 2.2: Correlation between $\boldsymbol{y}$,$\boldsymbol{y_1}$ and $\boldsymbol{X}$, where $\boldsymbol{y_1}$ is the response after the perturbation is added.

Correlation	$\boldsymbol{X}[,1]$	$\boldsymbol{X}[,2]$	$\boldsymbol{X}[,3]$	$\boldsymbol{X}[,4]$	$\boldsymbol{X}[,5]$	$\boldsymbol{X}[,6]$	$\boldsymbol{X}[,7]$	$\boldsymbol{X}[,8]$
$\boldsymbol{y}$	0.8932	0.8786	0.8415	0.8087	0.7887	0.7007	0.5778	0.5161
$\boldsymbol{y_1}$	0.8931	0.8785	0.8414	0.8086	0.7887	0.7006	0.5778	0.5160

design matrix $\boldsymbol{X}$ for the original data and after the perturbation is added. The result is shown in Table 2.2.

As we can see from the results (A.1), the Lasso solution path changes when a small normal perturbation $N(0, 10^{-4})$ is applied to the response. Variable 6 is dropped in the original Lasso path while there's no dropping after the perturbation is applied. We should mention that $\boldsymbol{y}$ and $\boldsymbol{X}$ are fixed. However, a small perturbation added to the response could completely change the solution path. From Table 2.2 we see that the correlation structure between the response and design matrix hardly change. This illustrates that dropping is very sensitive to the change in response variable. Another point worth mentioning is that more simulations show that such sensitivity exists only in those weak signals.

Similarly, we use simulation to investigate the sensitivity to the design matrix. We use the design matrix $\boldsymbol{X}$ and response $\boldsymbol{y}$ generated above. Then i.i.d normal noise $N(0, 10^{-4})$ is added to the design matrix and Lasso solution path is recalculated. Details about the simulation can be found in Appendix A.2. We also compare the correlation between $\boldsymbol{y}$ and the original design matrix $\boldsymbol{X}$ and the correlation between $\boldsymbol{y}$ and the new design matrix $\boldsymbol{X_1}$. The result is shown in Table 2.3 and Table 2.4. Correlation matrices of $\boldsymbol{X}$ and $\boldsymbol{X_1}$ are also calculated and shown in Table 2.5 and Table 2.6.

Similar result is obtained. When a small change is applied to the design matrix, dropping situation changes in the solution path. From Table 2.3 2.4 2.5 2.6 we can see that the correlation structure hardly changes. This illustrates that dropping is very sensitive to the change in design matrix. The change in variable dropping also only applies to the weak signals.

From the above simulation results we see that Lasso solution path is very sensitive to the

Table 2.3: Correlation between $\boldsymbol{y}$ and $\boldsymbol{X}$

Correlation	$\boldsymbol{X}[,1]$	$\boldsymbol{X}[,2]$	$\boldsymbol{X}[,3]$	$\boldsymbol{X}[,4]$	$\boldsymbol{X}[,5]$	$\boldsymbol{X}[,6]$	$\boldsymbol{X}[,7]$	$\boldsymbol{X}[,8]$
$\boldsymbol{y}$	0.8932	0.8786	0.8415	0.8087	0.7887	0.7007	0.5778	0.5161

Table 2.4: Correlation between $\boldsymbol{y}$ and $\boldsymbol{X_1}$, where $\boldsymbol{X_1}$ is the design matrix after the perturbation is added.

Correlation	$\boldsymbol{X_1}[,1]$	$\boldsymbol{X_1}[,2]$	$\boldsymbol{X_1}[,3]$	$\boldsymbol{X_1}[,4]$	$\boldsymbol{X_1}[,5]$	$\boldsymbol{X_1}[,6]$	$\boldsymbol{X_1}[,7]$	$\boldsymbol{X_1}[,8]$
$\boldsymbol{y}$	0.8931	0.8787	0.8416	0.8085	0.7894	0.7000	0.5776	0.5153

Table 2.5: Correlation matrix of $\boldsymbol{X}$

1.000	0.919	0.837	0.754	0.694	0.601	0.455	0.434
0.919	1.000	0.903	0.827	0.732	0.639	0.523	0.469
0.837	0.903	1.000	0.904	0.822	0.751	0.669	0.602
0.754	0.827	0.904	1.000	0.933	0.850	0.785	0.719
0.694	0.732	0.822	0.933	1.000	0.923	0.832	0.762
0.601	0.639	0.751	0.850	0.923	1.000	0.913	0.848
0.455	0.523	0.669	0.785	0.832	0.913	1.000	0.918
0.434	0.469	0.602	0.719	0.762	0.848	0.918	1.000

Table 2.6: Correlation matrix of $\boldsymbol{X_1}$

1.000	0.919	0.837	0.754	0.695	0.601	0.454	0.432
0.919	1.000	0.904	0.827	0.734	0.639	0.524	0.468
0.837	0.904	1.000	0.904	0.822	0.749	0.669	0.600
0.754	0.827	0.904	1.000	0.934	0.849	0.785	0.719
0.695	0.734	0.822	0.934	1.000	0.923	0.832	0.761
0.601	0.639	0.749	0.849	0.923	1.000	0.913	0.848
0.454	0.524	0.669	0.785	0.832	0.913	1.000	0.918
0.432	0.468	0.600	0.719	0.761	0.848	0.918	1.000

change of response variable and design matrix, especially when weak signal or noise exists. This indicates that it's very hard to find a dropping condition on the data set. However, Efron et al.[3] proposes a modified version of Lasso which doesn't have this dropping issue while maintaining the solution path. So we think it might be more appropriate to rank based on LARS solution path. Moreover, LARS algorithm gives a clear condition under which variable dropping will happen in Lasso.

Chapter 3

Least Angle Regression

It is mentioned by Efron et al.[3] that a simple modification of LARS will lead to entire solution path of Lasso. This important feature of LARS reveals the dropping mechanism in Lasso from a mathematical point of view.

3.1 LARS Algorithm

Below is a description of the full LARS algorithm in Efron et al. [3]. The LARS algorithm is like Stagewise Regression. However, the step of LARS is bigger than that of Stagewise.

We begin with $\hat{\boldsymbol{\mu}}_0 = 0$ and build $\hat{\boldsymbol{\mu}}$ by LARS steps. Suppose $\mathcal{A}$ is a subset of the indices $\{1, 2, \ldots, m\}$ which represents the variables currently in the model and define $\hat{\boldsymbol{\mu}}_{\mathcal{A}}$ to be the current estimate. The current correlation is defined as $\hat{\boldsymbol{c}} = X'(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_{\mathcal{A}})$. Let the set $\mathcal{A}$ be the variables which corresponds to the largest absolute correlation, that is, $\mathcal{A} = \{j : |\hat{c}_j| = \hat{C}\}$, where $\hat{C} = max\{|\hat{c}_j|\}$. Let $s_j = sign\{\hat{c}_j\}$ for $j \in \mathcal{A}$ and define the matrix

$$X_{\mathcal{A}} = (\ldots s_j \mathbf{x}_j \ldots)_{j \in \mathcal{A}} \tag{3.1}$$

Now we need to find the vector which has equal angles with the existing variables and we'll

update the estimate in this direction. Define

$$\mathcal{G}_{\mathcal{A}} = {X_{\mathcal{A}}}' X_{\mathcal{A}} \quad \text{and } A_{\mathcal{A}} = ({\mathbf{1}_{\mathcal{A}}}' {\mathcal{G}_{\mathcal{A}}}^{-1} \mathbf{1}_{\mathcal{A}})^{-1/2} \tag{3.2}$$

$\mathbf{1}_{\mathcal{A}}$ is a vector of $1's$ of length equaling $|\mathcal{A}|$, the size of $\mathcal{A}$. The equiangular vector is defined as:

$$\boldsymbol{u}_{\mathcal{A}} = X_{\mathcal{A}} \omega_{\mathcal{A}} \quad \text{where } \omega_{\mathcal{A}} = A_{\mathcal{A}} {\mathcal{G}_{\mathcal{A}}}^{-1} \mathbf{1}_{\mathcal{A}} \tag{3.3}$$

This is true because

$${X_{\mathcal{A}}}' \boldsymbol{u}_{\mathcal{A}} = A_{\mathcal{A}} \mathbf{1}_{\mathcal{A}} \tag{3.4}$$

And $\boldsymbol{u}_{\mathcal{A}}$ defined here is a unit vector since $||\boldsymbol{u}_{\mathcal{A}}||^2 = 1$. The inner product between $\boldsymbol{u}_{\mathcal{A}}$ and the full set of variables is $\boldsymbol{a} = X' \boldsymbol{u}_{\mathcal{A}}$. Now the next LAR step is determined as $\hat{\boldsymbol{\mu}}_{\mathcal{A}_+} = \hat{\boldsymbol{\mu}}_{\mathcal{A}} + \hat{\gamma} \boldsymbol{u}_{\mathcal{A}}$, where

$$\hat{\gamma} = \min_{j \in \mathcal{A}^c}{}^{+} \{ \frac{\hat{C} - \hat{c}_j}{A_{\mathcal{A}} - a_j}, \frac{\hat{C} + \hat{c}_j}{A_{\mathcal{A}} + a_j} \} \tag{3.5}$$

The "min$^+$" here means that only positive values are considered for a minimum. a_j is the jth element of the vector $\boldsymbol{a}$. The meaning of formulas (3.4) and (3.5) is as follows: We are updating the estimate along the direction defined by $\boldsymbol{u}_{\mathcal{A}}$. So

$$\boldsymbol{\mu}(\gamma) = \boldsymbol{\mu}_{\mathcal{A}} + \gamma \boldsymbol{u}_{\mathcal{A}} \tag{3.6}$$

the $\gamma > 0$ here is the "amount" that we should update. The updated correlation is

$$c_j(\gamma) = {\boldsymbol{x}'}_j (\boldsymbol{y} - \boldsymbol{\mu}(\gamma)) = {\boldsymbol{x}'}_j (\boldsymbol{y} - \boldsymbol{\mu}_{\mathcal{A}} - \gamma \boldsymbol{u}_{\mathcal{A}}) = \hat{c}_j - \gamma a_j \tag{3.7}$$

For $j \in \mathcal{A}$, using (3.2), (3.3)and (3.4) we get

$$|c_j(\gamma)| = \hat{C} - \gamma A_{\mathcal{A}} \tag{3.8}$$

We can see that the correlation between the residual and the variables in the active set is decreasing at the same speed. We choose γ so that some other variable has the same absolute correlation with the active set. So we have either $\hat{c}_j - \gamma a_j = \hat{C} - \gamma A_{\mathcal{A}}$ or $\hat{c}_j - \gamma a_j =$

$-\hat{C} + \gamma A_{\mathcal{A}}$, which lead to values of γ to be

$$\gamma = \frac{\hat{C} - \hat{c}_j}{A_{\mathcal{A}} - a_j} \quad \text{and} \quad \gamma = \frac{\hat{C} + \hat{c}_j}{A_{\mathcal{A}} + a_j} \tag{3.9}$$

respectively. As γ is the smallest positive number which makes the correlation of some other variables equal that of the active set, we have (3.5).

3.2 LARS and Lasso

Efron et al.[3] proved in the LARS paper that Lasso is actually a modified version of LARS. This important property reveals why variables are dropped in Lasso.

Lasso has an important property: Suppose $\hat{\beta}$ is a Lasso solution in (1.1), and $\hat{\mu} = X\hat{\beta}$. The current correlation is defined as $\hat{c}_j = \boldsymbol{x}'_j(\boldsymbol{y} - \hat{\boldsymbol{\mu}})$. Then we have

$$sign(\hat{\beta}_j) = sign(\hat{c}_j) = s_j \tag{3.10}$$

The proof is given in [3]. However, LARS doesn't have this requirement and can be modified to get Lasso solution path.

In the LARS algorithm, suppose in one LARS step, we get the active set $\mathcal{A}$ and the LARS estimate $\hat{\boldsymbol{\mu}}_{\mathcal{A}} = X\boldsymbol{\beta}$. Define $\omega_{\mathcal{A}}$ as in (3.3). Let $d_j = s_j\omega_{\mathcal{A}_j}$ for $j \in \mathcal{A}$ and zero otherwise. When we update μ along the direction of the equiangular vector, we have $\boldsymbol{\mu}(\gamma) = \boldsymbol{\mu}_{\mathcal{A}} + \gamma\boldsymbol{u}_{\mathcal{A}}$ as in (3.6). Because $\boldsymbol{\mu}(\gamma) = X\boldsymbol{\beta}(\gamma)$, we have

$$\beta_j(\gamma) = \hat{\beta}_j + \gamma\hat{d}_j \tag{3.11}$$

for $j \in \mathcal{A}$. We see that $\beta_j(\gamma)$ will change sign at $\gamma_j = -\frac{\hat{\beta}_j}{\hat{d}_j}$. Notice that $\gamma > 0$, so the first place that $\beta_j(\gamma)$ changes sign is at $\tilde{\gamma} = \min_{\gamma_j>0}\{\gamma_j\}$.

Another constrain on $\tilde{\gamma}$ is that $\tilde{\gamma} < \hat{\gamma}$, where $\hat{\gamma}$ is defined in (3.5). So if $\tilde{\gamma} < \hat{\gamma}$, then $\beta_j(\gamma)$ can't be a Lasso path for $\gamma > \tilde{\gamma}$ because (3.10) is not satisfied. This is because under such situation, $\beta_{\tilde{j}}(\gamma)$ has changed sign. However the sign of the continuous function $c_{\tilde{j}}(\gamma)$ has not changed since $|c_{\tilde{j}}(\gamma)| = \hat{C} - \gamma A_{\mathcal{A}} > 0$.

Efron et al. [3] proposed a LARS Modification which is: If $\tilde{\gamma} < \hat{\gamma}$, stop the ongoing LARS step an $\gamma = \tilde{\gamma}$ and remove $\tilde{j}$ from the calculation of the next equiangular direction, that is

$$\hat{\boldsymbol{\mu}}_{\mathcal{A}_+} = \hat{\boldsymbol{\mu}}_{\mathcal{A}} + \tilde{\gamma}\boldsymbol{u}_{\mathcal{A}} \quad \text{and} \quad \mathcal{A}_+ = \mathcal{A} - \{\tilde{j}\} \tag{3.12}$$

Efron et al. [3] proved the following theorem which states that the modified LARS yields Lasso under certain conditions.

Theorem 3.2.1. *Under the Lasso modification, and assuming the "one at a time*[1]*" condition, the LARS algorithm yields all Lasso solutions.*

So it's quite clear that when $\tilde{\gamma} < \hat{\gamma}$, variables will be dropped from Lasso solution path.

[1]One at a time means that the increase and decrease of the active set $\mathcal{A}$ never involve more than one variable

Chapter 4

Ranking based on LARS

Ranking based on LARS means that we'll assign an importance measure to each of the variables according to the order they enter the model. Variables that enter earlier are assigned a higher rank. Taking the diabetes data as an example, the LARS solution path is shown in Figure 4.1. The order that variables enter the model is $\{3, 9, 4, 7, 2, 10, 5, 8, 6, 1\}$. This is also the rank that we get based on LARS.

In this chapter, we use simulated data to test the ranking based on LARS. Like in the diabetes data, as we don't know what are the true signals and noise, it's hard to decide whether LARS ranking actually works. So we use simulated data where all information is available. First we'll introduce the rule we use in assessing the ranking quality: AUC.

4.1 Area under ROC curve

Area under ROC curve (AUC) [5] is a common measure of ranking quality. The receiver operating characteristic (ROC) curve is defined as a plot of the true positive rate versus the false positive rate. In a binary classification problem, we have two classes which we denote as positive (P) and negative (N). Then four probabilities are defined as below:

- True positive (TP) rate: the probability that a object from class P is classified as class P

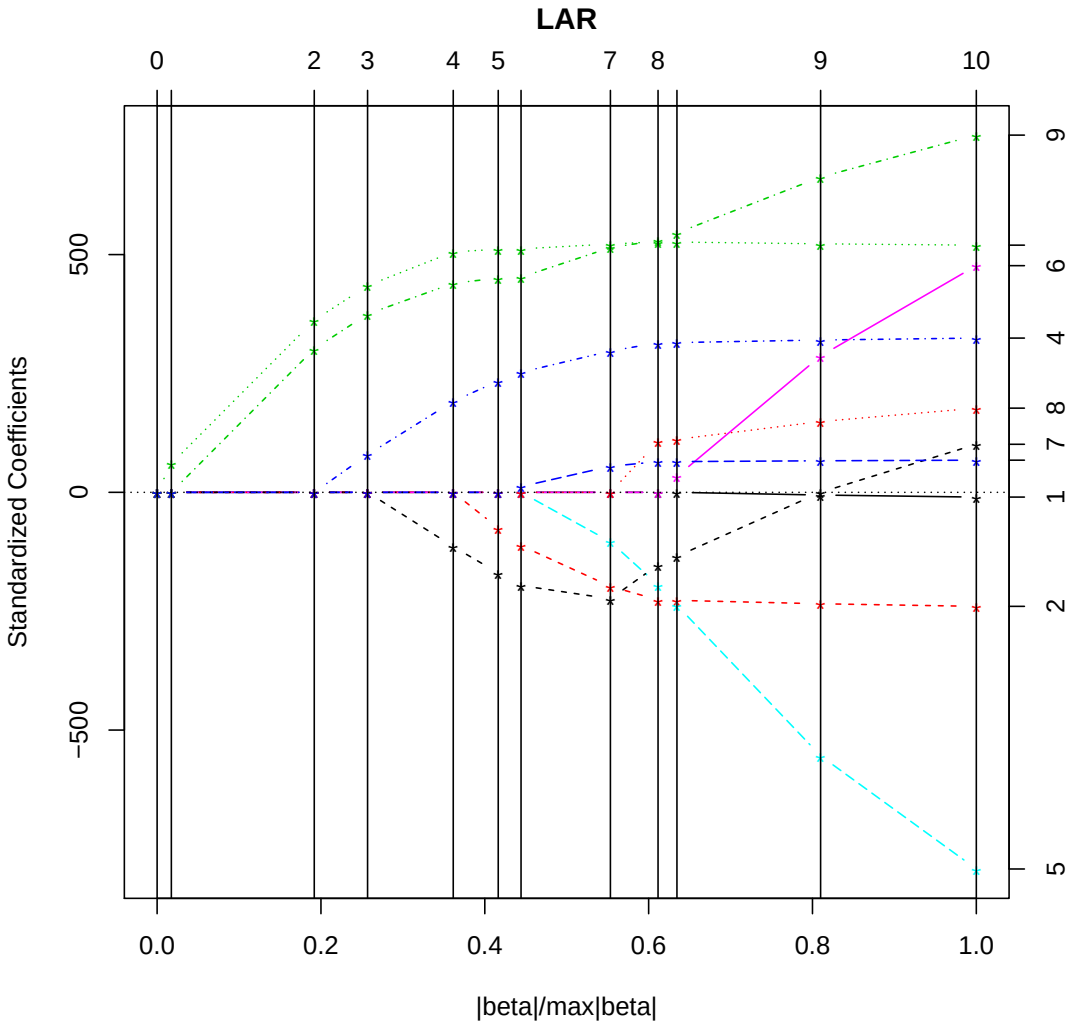

Figure 4.1: LARS solution path of diabetes data

- False positive (FP) rate: the probability that a object from class N is classified as class P
- True negative (TN) rate: the probability that a object from class N is classified as class N, $P(TN) = 1 - P(FP)$

Table 4.1: Average AUC with $\beta = (3, 1.5, 0, 0, 2, 0, 0, 0)$

Parameter	Average AUC
$n = 50, \rho = 0.5$	0.9880
$n = 100, \rho = 0.5$	0.9987
$n = 1000, \rho = 0.5$	1.0000
$n = 10000, \rho = 0.5$	1.0000
$n = 50, \rho = 0.9$	0.8280
$n = 100, \rho = 0.9$	0.9160
$n = 1000, \rho = 0.9$	0.9793
$n = 10000, \rho = 0.9$	1.0000

- False negative (FN) rate: the probability that a object from class P is classified as class N, $P(FN) = 1 - P(TP)$

The AUC function is defined as the area under the ROC curve. AUC can be regarded as the probability that a random chosen positive item will be ranked higher than a randomly chosen negative item. In the our variable ranking case, the positive items represents the signals and the negative items represents the noise. The value of AUC lies between 0 and 1. A random classifier yields AUC value of 0.5. From the perspective of AUC, higher value means higher quality of the classifier.

4.2 Example 1: A widely used benchmark

First we use the example by Tibshirani[6] (2.4). Denote n as the number of points simulated in one simulation. The simulation is replicated 100 times. Simulation results with different ρ and n are given in Table 4.1.

We can see that the result of LARS ranking is good, especially when n is relatively large. We also notice that when ρ increases from 0.5 to 0.9, the performance of LARS ranking decreases a little. This suggests that the performance of LARS ranking might be related to the correlation structure. We'll test LARS ranking in a highly-correlated case in Section 4.3.

Table 4.2: Average AUC with $\beta = (3, 1.5, 0.1, 0.1, 2, 0, 0, 0)$

Parameter	Average AUC
$n = 50, \rho = 0.5$	0.8246
$n = 100, \rho = 0.5$	0.8300
$n = 1000, \rho = 0.5$	0.9020
$n = 10000, \rho = 0.5$	0.9947
$n = 50, \rho = 0.9$	0.8033
$n = 100, \rho = 0.9$	0.8453
$n = 1000, \rho = 0.9$	0.8493
$n = 10000, \rho = 0.9$	0.9473

Next we change the parameter to test the ranking performance in situations where weak signals exist. We change β in (2.4) to $\beta = (3, 1.5, 0.1, 0.1, 2, 0, 0, 0)$ which converts two noise to weak signals. The corresponding result is shown in Table 4.2.

We can see that when n is small, the existence of weak signals decreases the average AUC of LARS ranking. However, when n is large enough, say 10000, the result is almost at the same level with the case without weak signals. When n is small, because of the bias of simulation, the correlation matrix of the simulated data is not exactly the same with the designed correlation matrix. When n is large enough, the bias is negligible. So the decrease of AUC might be due to the bias of simulation instead of the LARS method. When n is large enough, LARS ranking performs well in catching weak signals.

4.3 Example 2: A highly-correlated example

We use the example in Wang et al.[7] to test the performance of LARS ranking in a high correlation structure. The predictors in this example are highly-correlated and the sign of the signals are opposite. First we general 40 random variables $x_1, x_2, ..., x_{40} \overset{\text{id}}{\sim} N(0, 1)$. The response y is generated by

$$y = 3x_1 + 3x_2 - 2x_3 + 3x_4 + 3x_5 - 2x_6 + \sigma\epsilon$$

Table 4.3: Highly-correlated predictors

Number of points generated	Average AUC
$n = 50$	0.6418
$n = 100$	0.7130
$n = 1000$	0.8544
$n = 10000$	0.9952

The covariance matrix between the predictors is give by

$$cov = \begin{bmatrix} \mathbf{C_{3\times 3}} & - & - \\ - & \mathbf{C_{3\times 3}} & - \\ - & - & - \end{bmatrix}$$

where

$$\mathbf{C} = \begin{bmatrix} 1 & 0.9 & 0.9 \\ 0.9 & 1 & 0.9 \\ 0.9 & 0.9 & 1 \end{bmatrix}$$

So there are 6 signals and 34 noise. The signals are divided into two groups $\{1, 2, 3\}$ and $\{4, 5, 6\}$. The correlation within the group is very high and the two groups are independent. Signals and noise are also independent. The simulation is replicated 100 times and results are given in Table 4.3.

We get similar result here. When n is relatively small, the performance of LARS ranking is not very good in the high-correlation example. However, when n is large enough, LARS ranking still have very high accuracy. As long as we have enough data, ranking result based on LARS is guaranteed.

Though this is a case where there exists high correlation in the data set, we should notice that the high correlation only exists in the signals. In the next section we'll use an example where signal and noise are highly correlated to see if LARS can separate them.

Table 4.4: Inconsistent example

Number of points generated	Average AUC
$n = 50$	0.005
$n = 100$	0
$n = 500$	0
$n = 1000$	0
$n = 10000$	0

4.4 Limitation of LARS: An inconsistent example

Next we discuss an example mentioned in Zhao and Yu[9]. In this example, LARS doesn't give a consistent model selection and we want to investigate the performance of LARS ranking in an incorrect model.

First we generate $x_{i1}, x_{i2}, e_i, \epsilon_i \overset{\text{iid}}{\sim} N(0,1)$, $i = 1, 2, ..., n$. x_{i3} is correlated with x_{i1} and x_{i2}:

$$x_{i3} = \frac{2}{3}x_{i1} + \frac{2}{3}x_{i2} + \frac{1}{3}e_i$$

The response is given by

$$Y_i = 2x_{i1} + 3x_{i2} + \epsilon_i$$

Under such design, x_{i3} is noise but highly correlated with signals x_{i1} and x_{i2}. The simulation is replicated 100 times and result is given in Table 4.4.

We can see from the results that the LARS ranking is completely wrong for this example. Different from the previous two examples, when n increases, the performance of LARS ranking doesn't change at all. In Zhao and Yu[9], they point out Lasso doesn't give a consistent model selection in this case. As a result, the solution path of LARS is also misleading and LARS ranking is completely wrong.

In Zhao and Yu[9], they also give a necessary and sufficient condition for model selection consistency of Lasso. However, the condition is based on that we know exactly what are the signals and noise. In real cases where such information is not available, it's hard to determine whether LARS is consistent in advance. This is a limitation of LARS ranking.

We'll discuss some possible improvements in Chapter 5.

4.5 A real-data application: Info-Rehab Program

At last we apply what we discuss above to an applied task: the Info-Rehab Program. In Zhu et al.[10], they apply Lasso to identify the most important variables for the binary outcome of receiving rehabilitation services[10]. Here we investigate a slightly different task.

The data set we have is called RAI-HC which contains 1,043,700 observations of rehabilitation clients. Each observation is a record of assessment for a client. The outcome variable is called Cognitive Performance Scale (CPS). CPS is an assessment of the cognitive status and takes value in $\{0, 1, 2, 3, 4, 5, 6\}$. 0 means not impaired, or normal cognitive status. 6 means severely impaired cognition. 217 variables are included in the analysis. We are interested in the change of CPS for a client after receiving rehabilitation for approximately one year and try to find out the most important variables associated with CPS change. With these variables we can predict whether rehabilitation will be effective for a client with certain health indices and thus identify those who are most appropriate to receive rehabilitation.

In addition to LARS ranking, we compare the result with ranking given by importance measure in random forest. The result is shown in Table 4.5 and Table 4.6. Common variables selected by LARS and random forest are listed in Table 4.7.

There are 11 variables ranked in top 20 by both LARS and random forest. The top variables, for example "B1a" which represents that short memory is OK, make sense intuitively.

Table 4.5: Top 20 Variables ranked by LARS

Rank	Variable name	Variable Explanation
1	B1a	Short memory OK
2	H1db	Difficulty of manage medications on own
3	J1g	Alzheimer's
4	approx_age	Approximate Age
5	H1cb	Difficulty of manage finances on own
6	K4b	Intensity of pain
7	H1da	Performance of managing medications
8	H1eb	Difficulty for phone use
9	K4d	Character of pain
10	H1ca	Performance of managing finances
11	J1h	Dementia other than Alzheimer's
12	B2a	Decisions about organizing the day
13	CC2	Reason for referral
14	Q1	number of medications
15	K4a	Frequency client shows pain
16	F3a	Length of time client is alone during the day
17	H1aa	Performance of meal preparation
18	B1b	Procedural memory OK
19	J1m	Arthritis
20	P4a	Number of times in hospital with overnight stay

Table 4.6: Top 20 Variables ranked by Random Forest: Rank given by Random Forest is based on 30 runs each with random sample of size 10000.

Rank	Variable name	Average rank	SD	Variable Explanation
1	B1a	1.0000	0	Short memory OK
2	B2a	2.0000	0	Decisions about organizing the day
3	H1db	3.2667	0.5208	Difficulty of manage medications on own
4	H1da	4.1667	0.9499	Performance of managing medications
5	FADIndex	5.4000	1.3544	Frailty index
6	C2	6.8333	2.0186	Making self understood
7	H1cb	8.0667	1.4126	Difficulty of manage finances on own
8	H1ca	8.7000	2.5072	Performance of managing finances
9	H1ea	10.033	2.7604	Performance of phone use
10	B1b	10.100	3.4874	Procedural memory OK
11	H1eb	10.633	2.4280	Difficulty for phone use
12	H1aa	11.500	1.9253	Performance of meal preparation
13	C3	13.367	2.4980	Ability to understand others
14	approx_age	13.833	1.8019	Approximate age
15	J1h	15.933	5.1390	Dementia other than Alzheimer's
16	H1ab	16.033	2.9767	Difficulty for meal preparation
17	B2b	16.267	6.0111	Worsening of decision making
18	H2f	19.200	2.0410	Performance of dressing lower body
19	H2e	19.967	3.1237	Performance of dressing upper body
20	H2d	23.100	3.3046	Locomotion outside home

Table 4.7: Common variables selected by LARS and random forest

Variable name	Rank in LARS	Rank in random forest
B1a	1	1
H1db	2	3
approx_age	4	14
H1cb	5	7
H1da	7	4
H1eb	8	11
H1ca	10	8
J1h	11	15
B2a	12	2
H1aa	17	12
B1b	18	10

Chapter 5

Summary and Future Work

Our motivation comes from the Lasso solution path which is a natural way to rank variables. However, variable dropping issue of Lasso makes it difficult to use the solution path directly. In this thesis, we first point out that dropping is actually very common in Lasso. Least Angel Regression gives a mathematical explanation about when variable dropping happens. In order to get a more straightforward and understandable idea about the dropping issue, we try to explain variable dropping from data structure. However, we find that dropping is very sensitive to the change in response variable and design matrix, which makes it hard to find a dropping condition on data structure. Least Angle Regression, which is a modified version of Lasso, maintains the solution path but doesn't have this dropping issue. So we advocate ranking by LARS instead of Lasso. Several simulation examples are used to test the performance of LARS ranking. We find that LARS ranking perform well in the first two examples. Even if there exist high correlation structure and weak signals, LARS can still give a satisfactory ranking result. However, we also find that there exist inconsistent examples where Lasso or LARS fails to give a consistent model selection. In such cases, LARS ranking is completely misleading.

One possible way to overcome this limitation of LARS ranking is to combine LARS solution path with some other importance measure in ranking. A potential method is to rank based on LARS solution path and the ordinary least square estimates. The idea here is very simple. In linear regression, the larger the variable coefficient is, the bigger

influence it has on the response. So the coefficient is also a kind of importance measure. We can combine the OLS estimate with the order in LARS solution path to get a "mixed" importance measure. In simulations, we test the following form of mixed importance measure:

$$im_k = -order + log|\beta_k| \tag{5.1}$$

where "*order*" is the order of variable k in the solution path and β_k is the OLS estimate. Simulation result shows that the "mixed" importance measure performs quite well in all the examples mentioned in Chapter 4, including the inconsistent case. It seems that the OLS estimates can sometimes make up for the inconsistency of LARS. However, we haven't found enough theoretical support for such kind of importance measure. We get 5.1 from intuition and as an imitation of the AIC rule[1]. We hope that some reasonable support, like the explanation of AIC, can be found for such a measure. Then we can improve LARS ranking to avoid the misleading inconsistent case.

APPENDICES

Appendix A

Simulation Results in Section 2.2.2

A.1 Simulation about the sensitivity to response

First we generate the design matrix $\mathbf{X}$ from a multivariate normal distribution. Then we calculate response $\mathbf{y}$ from the model and add a normal $N(0, 10^{-4})$ noise to the response. Details are given below.

```
mu<-rep(0,8)
cov<-matrix(rep(1,64),8,8)
for(i in 1:8){
  for(j in 1:8){
    cov[i,j]<-0.9^(abs(i-j))
  }
}
x<-mvrnorm(100,mu,cov)
pa=c(3,1.5,0.01,0.01,2,0.01,0.01,0.01)
sigma=3
```

After setting the parameters, we generate the first response $\mathbf{y}$.

```
eps<-rnorm(100,0,1)
y<-x%*%pa+sigma*eps
lars(x,y,trace=T)
```

The Lasso solution path is:

```
LASSO sequence
Computing X'X .....
LARS Step 1 :    Variable 1     added
LARS Step 2 :    Variable 2     added
LARS Step 3 :    Variable 5     added
LARS Step 4 :    Variable 8     added
LARS Step 5 :    Variable 7     added
LARS Step 6 :    Variable 6     added
LARS Step 7 :    Variable 4     added
Lasso Step 8 :   Variable 6     dropped
LARS Step 9 :    Variable 3     added
LARS Step 10 :   Variable 6     added
Computing residuals, RSS etc .....

Call:
lars(x = x, y = y, trace = T)
R-squared: 0.802
Sequence of LASSO moves:

Var  1 2 5 3 7 -3 6 3 8  4
Step 1 2 3 4 5  6 7 8 9 10
```

Then we add a normal $N(0, 10^{-4})$ noise to the response and recalculate the Lasso sequence.

```
noise=0.01*rnorm(100,0,1)
```

```
y1=y+noise
object1=lars(x,y1,trace=T)
```

The Lasso solution path is:

```
LASSO sequence
Computing X'X .....
LARS Step 1 :    Variable 1     added
LARS Step 2 :    Variable 2     added
LARS Step 3 :    Variable 5     added
LARS Step 4 :    Variable 8     added
LARS Step 5 :    Variable 7     added
LARS Step 6 :    Variable 4     added
LARS Step 7 :    Variable 3     added
LARS Step 8 :    Variable 6     added
Computing residuals, RSS etc .....

Call:
lars(x = x, y = y1, trace = T)
R-squared: 0.802
Sequence of LASSO moves:

Var  1 2 5 3 7 6 8 4
Step 1 2 3 4 5 6 7 8
```

A.2 Simulation about the sensitivity to design matrix

We use the design matrix and response generated in A.1. Then $i.i.d$ normal noise $N(0, 10^{-4})$ is added to the design matrix and Lasso solution path is recalculated.

```
x1=x+0.01* mvrnorm(100,rep(0,8),diag(8))
lars(x1,y,trace=T)
```

```
LASSO sequence
Computing X'X .....
LARS Step 1 :    Variable 1     added
LARS Step 2 :    Variable 3     added
LARS Step 3 :    Variable 2     added
LARS Step 4 :    Variable 4     added
LARS Step 5 :    Variable 5     added
LARS Step 6 :    Variable 7     added
LARS Step 7 :    Variable 6     added
LARS Step 8 :    Variable 8     added
Computing residuals, RSS etc .....

Call:
lars(x = x1, y = y, trace = T)
R-squared: 0.7
Sequence of LASSO moves:

Var  1 3 2 4 5 7 6 8
Step 1 2 3 4 5 6 7 8
```

References

[1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Second international symposium on information theory*, pages 267–281. Springer Verlag, 1973.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[4] Trevor. Hastie, Robert. Tibshirani, and JH (Jerome H.) Friedman. *The elements of statistical learning*. Springer, 2009.

[5] Kent A. Spackman. Signal detection theory: valuable tools for evaluating inductive learning. In *Proceedings of the sixth international workshop on Machine learning*, pages 160–163. Morgan Kaufmann Publishers Inc., 1989.

[6] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[7] S. Wang, B. Nan, S. Rosset, and J. Zhu. Random lasso. *The Annals of Applied Statistics*, 5(1):468–485, 2011.

[8] L. Xin and M. Zhu (in press). Stochastic stepwise ensembles for variable selection. *Journal of Computational and Graphical Statistics, accepted and to appear*, 2010.

[9] P. Zhao and B. Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

[10] Mu Zhu, Lu Cheng, Joshua J. Armstrong, Jeff W. Poss, John P. Hirdes, and Paul Stolee. Using machine learning to plan rehabilitation for home care clients: Beyond black-box predictions. To appear in *Machine Learning in Healthcare Informatics, Springer*, 2011.