Stochastic Stepwise Ensembles for Variable Selection

by

Lu Xin

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, 2009

AUTHOR'S DECLARATION

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

Ensembles methods such as AdaBoost, Bagging and Random Forest have attracted much attention in the statistical learning community in the last 15 years. Zhu and Chipman (2006) proposed the idea of using ensembles for variable selection. Their implementation used a parallel genetic algorithm (PGA). In this thesis, I propose a stochastic stepwise ensemble for variable selection, which improves upon PGA.

Traditional stepwise regression (Efroymson 1960) combines forward and backward selection. One step of forward selection is followed by one step of backward selection. In the forward step, each variable other than those already included is added to the current model, one at a time, and the one that can best improve the objective function is retained. In the backward step, each variable already included is deleted from the current model, one at a time, and the one that can best improve the objective function is discarded. The algorithm continues until no improvement can be made by either the forward or the backward step.

Instead of adding or deleting one variable at a time, Stochastic Stepwise Algorithm (STST) adds or deletes a *group* of variables at a time, where the group size is randomly decided. In traditional stepwise, the group size is one and each candidate variable is assessed. When the group size is larger than one, as is often the case for STST, the total number of variable groups can be quite large. Instead of evaluating all possible groups, only a few randomly selected groups are assessed and the best one is chosen.

From a methodological point of view, the improvement of STST ensemble over PGA is due to the use of a more structured way to construct the ensemble; this allows us to better control over the strength-diversity tradeoff established by Breiman (2001). In fact, there is no mechanism to control this fundamental tradeoff in PGA. Empirically, the improvement is most prominent when a true variable in the model has a relatively small coefficient (relative to other true variables). I show empirically that PGA has a much higher probability of missing that variable.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Professor Mu Zhu for his insight, guidance, constant encouragement, and unlimited support in every aspect of my study, research and personal life. His rigorous scholarship and enthusiasm in statistics have inspired and enriched my growth as a student and a researcher.

I am also very grateful for the assistance and valuable advice from my thesis readers Professor Hugh Chipman and Professor Ali Ghodsi.

Many thanks to my friends, Michelle Zhou, Baojiang Chen, Pengfei Li, Yanqiao Zhang, Hui Zhao, Chao Qiu, Hua Shen, Le Li, Jing Zhou, Lihua Wang for their help during my time in Waterloo. They make my life and study in Canada much easier and more colorful. I also want to take this opportunity to thank the graduate studies coordinator Mary Lou Dufton for her patience and advices.

Finally, I am particularly grateful for the help from Y. Celia Huang during my master study. Special thanks to my parents for their support and encouragement throughout my whole life. Without them, nothing is possible.

To the memory of my grandfather

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Statistical Machine Learning Problems and Ensemble Methods

"Vast amounts of data are being generated in many fields, and the statistician's job is to make sense of it all: to extract important patterns and trends, and understand 'What the data says.' We call this learning from data." (Hastie et al, 2001)

Typically, we have a response measurement and some predictors, quantitative or categorical. In traditional linear regression, forms of the functions linking the predictors to the response are determined before the fitting process begins. These procedures are all parametric. We have to make different kinds of "reasonable" assumptions about the models. However, in statistical learning, people rely far less on the prior information of the link function. Although sometimes there are still constraints, the functional forms are mostly arrived from the data. In that sense, statistical learning procedures can be seen as nonparametric. It is more about learning from data rather than learning from assumptions. There are two kinds of statistical learning problems: supervised and unsupervised. Regarding the supervised learning, the response variable is measured so that the learning process can be guided by the outcome, e.g. regression problems. In the unsupervised learning problem, we can only observe the features but no measurements of the outcome, e.g. cluster analysis.

To a certain extent, statistical learning can be viewed as the modern version of Exploratory Data Analysis (Tukey, 1977) developed along with the enormous developments of computing power and computer algorithms over the past two decades.

In statistical learning, many methods and algorithms have been created. Generally, many algorithms developed in the past 15 years fall into two categories: kernel methods and ensemble methods. We will only focus on the ensemble methods in the following sections. Rather than using a strong classifier, the main idea behind ensemble methods is to combine the outputs from many weak classifiers. Bagging (Breiman, 1996), Random Forest (Breiman, 2001) and AdaBoost (Freund and Schapire, 1996) are the most important representatives of ensemble methods.

1.2 Bagging

Bagging stands for "Bootstrap Aggregation". It was first proposed by Leo Breiman as an extension of CART (Classification and Regression Tree). A number of classification trees are built on different

Bootstrap samples, and the outputs are aggregated to produce the final results. Bagging is perhaps the earliest procedure to exploit a combination of fitted values based on Bootstrap samples.

With a dataset having categorical response, the Bagging algorithm can be easily described as the following key steps:

1. Take a Bootstrap sample from the data.
2. Construct a classification tree on the Bootstrap sample.
3. Repeat steps 1-2 for many times.
4. For an observation, usually from the test data, drop it down all the trees and count the number of times that it is classified in all categories. Assign it to a final category by the majority vote.

Bootstrapping is the only stochastic mechanism of Bagging. Because of Bootstrapping, the averaging tends to cancel out results shaped by idiosyncratic features of the data. A large number of fitting attempts can produce flexible fitting functions which are able to respond to systematic, but highly localized, feature of the data. One can obtain more stable fitted values and more honest assessments of how good the fit really is. By averaging a number of low-bias, high-variance predictions, Bagging can effectively maintain the low bias and reduce the prediction variance.

Bagging also suffers from several problems. First, it cannot obviously depict how the predictors are related to the response. Second, the bootstrap sampling can lead to problems when categorical predictors or outcomes are highly unbalanced. For example, for the rare target problems, Bootstrap can make the data even more unbalanced. Third, since the same predictors are used from tree to tree, the sets of fitted values are highly correlated. This may make the output of the trees similar. If the fitting function is substantially inappropriate, bagging can no longer cancel out the negative effect. At that time, bagging is a weak classifier that makes things worse.

After all, Bagging is a useful tool in practice and an important conceptual advance in statistical learning. It can be regarded as an important step of what Leo Breiman called “algorithmic modeling”. Algorithmic models are computer algorithms designed to solve very particular data analysis problems, linking inputs to outputs in the purpose of making the classification errors small. Although they may be bad at describing how the inputs are linked to the outputs, the accuracy of prediction and the high level of automation are attractive.

1.3 Random Forest

Random Forest (Breiman, 2001) is an extension of Bagging. The main difference is, as each tree is constructed, only a random sample of predictors is considered before each node is split. Thus, each tree is produced using Bootstrap sampling of the observations, and at each split a random sample of the predictors is considered. Two stochastic mechanisms exist. Although the difference seems small, some problems of Bagging are solved.

The Algorithm of Random Forest

The random forest algorithm is very much like the Bagging algorithm. Let P be the number of predictors and assume the response is categorical variable.

1. Take a Bootstrap sample from the data.
2. Construct a classification tree without pruning. During the process, before each split is made, instead of using all the predictors, use a sample of the predictors randomly chosen with the same size $M<P$.
3. Drop the out-of-bag data (the observations that are not used to build the tree) down the tree. Store the class assigned to each observation in the out-of-bag data.
4. Repeat steps 1-3 for a number of times.
5. For each observation, usually from the test data, drop it down all the trees and count the number of times that it is classified in all categories. Assign it to a final category by the majority vote.

During the forest building progress, the out-of-bag data are dropped down the trees and are assigned classes i.e. the classes are determined by the trees that are built without them. This can be used to generate an internal unbiased estimate of the generalization error. So misclassification error can truly assess the performance of the Random Forest. Bagging can also do the same procedure.

By working with a random sample of predictors at each split, the trees have more variety, thus, are more independent. Breiman (2001) theoretically established the benefit of this, which we will review later.

Breiman's Theorems about Random Forest (Breiman, 2001)

Considering an ensemble of K classifiers, $f_1(x), f_2(x), ..., f_K(x)$. Suppose the data point is *y, x*, where *y* denotes the response and x is the predictor vector. Then the margin function is defined as

$$mg(x, y) = av_k I(f_k(x) = y) - \max_{j \neq y} av_k I(f_k(x) = j),$$

where *I* is the indicator function, *j* is an incorrect class and '*av*' denotes taking average over all the classifiers. The margin shows the difference between the proportions of the votes for the correct class and the most voted incorrect class. Generalization error is defined by

$$g = P_{x,y}(mg(x, y) < 0),$$

where *P* denotes probability. The generalization error is the probability over the population that the correct class does not receive the most votes, thus, the class is assigned incorrectly.

Now, let's move to Random Forest, where the classifiers are trees $f_k(x) = f(x, \theta_k), k = 1,2,3,...$ Breiman proved that as the number of trees increases, for almost surely all sequences $\theta_1, \theta_2, ...$, *g* converges to

$$P_{x,y}(P_\theta(f(x, \theta) = y) - \max_{j \neq y} P_\theta(f(x, \theta) = j) < 0).$$

The importance of the convergence is that it indicates Random Forest does not overfit as more trees are added. This is an important result regarding the problem of overfitting.

Breiman showed an upper bound for the generalization error which enlightens why Random Forest is more powerful than Bagging. Before we show the form of the upper bound, we need more definitions. The margin function for a given data point in a Random Forest is defined as

$$mr(x, y) = P_\theta(f(x, \theta) = y) - \max_{j \neq y} P_\theta(f(x, \theta) = j).$$

The strength of the set of classifiers $\{f(x, \theta)\}$ is $s = E_{x,y} mr(x, y)$. The strength of a Random Forest is the average margin over the data population, the larger the better. The strength is an indicator of the power for the classifiers in the Random Forest.

Breiman defined $\bar{\rho} = E_{\theta,\theta'}(\rho(\theta, \theta') sd(\theta) sd(\theta')) / E_{\theta,\theta'}(sd(\theta) sd(\theta'))$ as the mean value of the correlation. The parameter θ here is the random mechanism of the classifiers. Regarding Random

Forest, it is Bootstrap sampling of the data set and predictors sampling before each split. Strictly speaking, this definition of the mean correlation is not 100% correct from a technical point of view.

Finally, Breiman showed

$$g \le \bar{\rho}(1-s^2)/s^2.$$

In order to make Random Forest work well i.e. make *g* smaller, we need the strength to be bigger and $\bar{\rho}$ to be smaller. Consequently, an individual tree should be as strong as possible and the trees should be as independent as possible. That is why the trees in Random Forest are built without pruning. The introduction of the predictor sampling before each split increases the diversity of the trees, thus, make them more uncorrelated. And that is the advantage of Random Forest compared to Bagging.

Generally, the principle of increasing strength of individual paths and reducing the dependence not only applies to Random Forest, but also applies to all the ensemble methods. And how to balance the strength-diversity tradeoff is always an open question in the field of ensemble methods.

1.4 AdaBoost

AdaBoost is the most popular boosting procedure introduced by Freund and Schapire (1996). Consider a classification problem with binary response coded as 1 and -1. The algorithm is as following:

1. Give the initial weights $w_i = 1/N, i = 1,2,3,...,N$ to each observation
2. For *m*=1 to *M*

   Fit a classifier $f_m(X)$ to the observations using the weights w_i.

   Compute $err_m = \dfrac{\sum_{i=1}^{N} w_i I(y_i \neq f_m(x_i))}{\sum_{i=1}^{N} w_i}$ and $\alpha_m = \log[(1-err_m)/err_m]$.

   Set the new weights $w_i^{new} = w_i \cdot \exp[\alpha_m \cdot I(y_i \neq f_m(x_i))], i = 1,2,...,N$.
3. The final output is $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

All people firstly seeing the algorithm of AdaBoost must have the question why should the weights be defined in this way. Let W and R denote the set of observations that are wrongly and correctly classified by classifier $f_m(X)$, respectively. We have $err_m = \sum_{i \in W} w_i / \sum_{all} w_i$, which indicates

$$\sum_{i \in W} w_i = err_m \sum_{all} w_i \text{ and } \sum_{i \in R} w_i = (1 - err_m) \sum_{all} w_i .$$

If we reweigh the observations by

$$w_i^{new} = w_i \cdot \exp[\alpha_m \cdot I(y_i \neq f_m(x_i))], i = 1,2,...,N ,$$

where $\alpha_m = \log[(1 - err_m)/err_m]$, only the observations wrongly classified by classifier $f_m(X)$ get new weights. The interesting result is

$$\sum_{i \in W} w_i^{new} = \sum_{i \in W} w_i \left(\frac{1 - err_m}{err_m} \right) = (1 - err_m) \sum_{all} w_i = \sum_{i \in R} w_i = \sum_{i \in R} w_i^{new} ,$$

whick indicates that given the new weights, the current classifier $f_m(X)$ has 50% error rate. In that sense, the next classifier $f_{m+1}(X)$ is "decoupled" from $f_m(X)$.

Friedman, Hastie and Tibshirani (2000) proved that AdaBoost substantially is a forward stagewise modeling approach that minimizes the exponential loss $L(y, f) = \exp(-yf)$. By replacing the empirical average by the population expectation, it is easy to general AdaBoost to "Real AdaBoost" and "Gentle AdaBoost". Indeed, many other Boosting variations are possible as long as we pick a different loss function.

To summarize, AdaBoost combines a large number of weighted classifiers, and each classifier is built on the weighted observations. The observation weights are functions of how poorly such observation was fitted using the previous classifier. The classifier weights are functions of how well the classifier fits the data. Once a classifier is fitted, it will not be re-adjusted. Unlike Bagging or Random Forest where individual classifiers are independently constructed, in AdaBoost the individual classifiers are constructed sequentially. However, because of the reweighing of the observations in each step, the next classifier is always "decoupled" from the former one. So the diversity of classifiers increases.

1.5 Conclusion

Ensemble methods appear to be efficient and successful. The power of aggregate of a number of weak classifiers is usually greater than that of one strong classifier. From the review of the well-known ensemble methods, we find that in order to build an effective ensemble, the strength-diversity tradeoff must be handled appropriately. If all the ensemble members are very strong, they may be similar with each other, so that the ensemble has no much sense. On the other hand, if all the ensemble members are very weak, although the diversity is big, each ensemble member is too weak to be eligible. Generally, how to find the balance of the strength-diversity tradeoff depends on the situation.

In this thesis, we try to use ensembles to deal with variable selection problem. From the beginning to the end, our research is guided by how to well handle the strength-diversity tradeoff.

Chapter 2
Review of Variable Selection Using Ensembles

We now turn to the idea of using ensembles for variable selection, which is the problem that we will study in this thesis. Firstly, we will review the variable selection problem. Then, we will show our motivation of why using ensembles and take an overview of Parallel Genetic Algorithm (Zhu and Chipman, 2006).

2.1 Variable Selection Problem

Regression models are widely applied in statistical inference. Variable selection is a central problem in this field. Consider the simple linear regression model

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2),$$

where the response Y is linearly related to p explanatory variables $X_1, X_2, ..., X_p$. Random error ε follows the distribution $N(0, \sigma^2)$. Generally, not all the predictors actually affect the response, so variable selection is necessary before any statistical estimation or prediction. The traditional procedure of variable selection is that we first choose a criterion, then, try to find the subset of predictors which optimizes the criterion. Clearly, the best subset is unique. And usually, this subset may not be the best when examined by other criterion. Zhu (2008) summarized that two main challenges for subset selection are:

Computation

Let Ω be the space of all the subsets of $S = \{X_1, ..., X_p\}$. The size of the space is $|\Omega| = 2^p$. When p is large, exhaustive search becomes impossible. Many efficient algorithms were developed in order to find the relatively "best" subset. The most well known one is the "stepwise regression" proposed by Efroymson (1960).

Criterion

This problem is more substantial. How to determine which subset is "better" or one variable is "more important" than others? AIC (Akaike, 1973) and BIC (Schwarz, 1978) are the most representative and widely used variable selection criteria.

AIC and BIC have the expressions

$$AIC = 2k - \ln(L),$$

$$BIC = k\ln(n) - \ln(L),$$

where k is the number of parameters, n is the number of observations and L is the maximized value of the likelihood function of the estimated model. It is well known that AIC tends to choose more variables and BIC tends to choose less. And they both have their own properties. BIC is consistent in selecting the true model and AIC is minimax-rate optimal for estimating the regression function (Yang, 2005). Different criteria always give us different selections. Obviously, which criterion to use depends on the data and the objects. It is usually a difficult problem.

2.2 Motivation

Zhu and Chipman (2006) designed an ensemble procedure – Parallel Genetic Algorithm (PGA) to boost up the performance of various variable-selection criteria such as the AIC and GCV. It is the first try of using ensembles to deal with variable selection. Zhu (2008) used a small toy example to explain the gist of their idea, i.e., how can they use ensemble to boost up the performance of AIC. He considered the simple linear regression model

$$y_i = x_{i,2} + x_{i,5} + x_{i,8} + \varepsilon_i, x_{i,1},...,x_{i,10}, \varepsilon_i \overset{iid}{\sim} N(0,1), i = 1,2,...,50.$$

There are 10 potential predictors but only x_2, x_5, x_8 are contained in the true model. All the predictors and errors are generated independently from $N(0,1)$. It is affordable to exhaustively compute the AIC values for all the 1024 (2^{10}) subsets. The distribution of AIC values in the subset space is showed by Figure 2.1. Each point in the figure represents a subset.

We find that the subset with the smallest AIC is not $\{x_2, x_5, x_8\}$, but contains 5 variables. If we choose the "best" subset, we will include two more junk variables. We applied the traditional variable selection algorithms (forward, backward and stepwise) to the toy example using AIC as objective function and they all output the subset $\{x_2, x_5, x_8, x_6, x_{10}\}$. It is partly efficiency and partly lucky that they could find the best subset without exhaustive search. However, they failed to choose the true subset. The traditional algorithms like forward, backward and stepwise try to find the global optimal point but ignore the whole distribution of AIC in the subset space.

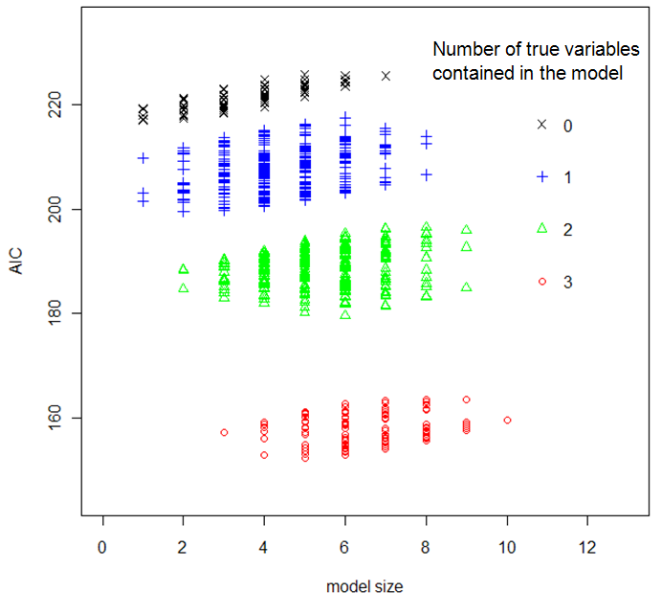

Figure 2.1 AIC values vs subset sizes

The points in the figure are separated into several parts. Significant gaps exist between different groups of points. Indeed, the red ($\circ$) group consists of the subsets that contain all 3 true variables x_2, x_5, x_8. Green (Δ), blue (+) and black ($\times$) groups consist of the subsets containing 2, 1 and 0 of the 3 true variables, respectively. This is strong evidence showing that x_2, x_5, x_8 are more important than other variables. PGA tries to use ensemble to catch such information about AIC, thus, boost up the performance of AIC.

2.3 Parallel Genetic Algorithm

In this section, we take an overview of Parallel Genetic Algorithm. Genetic Algorithm (GA) mimics the mechanisms of natural reproduction processes. It is firstly used by Kuncheva (1993) to do feature selection for parallel classifiers. In the context of variable selection, generally speaking, GA starts with an initial population set which are all subsets of predictors. The subsets then produce offspring which are also subsets by crossing-over, mutation and combination with certain probabilities in order to fit the criterion better. The offspring continue producing another generation of offspring. This process is regarded as evolution. The whole family is called a universe. The last-generation offspring of a universe are supposed to be the best subsets. Kuncheva (1993) uses GA to select a number of

subsets for the multiple-classifier system in order to increase the predictive accuracy of the system. After all, GA is a search algorithm for finding optimal subsets.

Instead of using a single GA to search the best subset, Zhu and Chipman (2006) use PGA which is an ensemble of parallel GA paths to deal with variable selection. The main idea is to create a number of parallel universes by Single-Path GA (Zhu and Chipman, 2006), each of which gives a score to each of the predictors. The predictors with relatively high average scores are chosen.

Suppose the potential variables are $\{x_1, x_2, ..., x_p\}$ and we have a data set of the observations. The algorithm can be described as:

1. Start a universe i.e. a path

a) Run a Single-Path Genetic Algorithm on the data set using AIC as objective function for a few generations before it converges to produce offspring which are all subsets of $\{x_1, x_2, ..., x_p\}$.

b) Consider the last-generation subsets. For each variable, calculate the percentage of the subsets that contain it as its score. For example, variable x_1 being contained by 50% last-generation subsets indicates that it gets a score 0.5 from this universe.

2. Produce a number of universes, so that each variable has a number of scores. Use the average of those scores as the final score for the corresponding variable.

3. Order the final scores of the variables from big to small and find the largest gap of the two neighborhoods. The variables with the scores above the gap are output.

Due to the randomness of Genetic Algorithm, the universes are generated variously. The ensemble of universes determines the destiny of the variables.

The most important parameter of the algorithm is how many generations to evolve in each universe, i.e., when to stop.

- If we stop early, the universes are not well evolved and the last-generation offspring may stay in the green (Δ) group, blue ($\times$) group even black (+) group. The scores of such universe are not good enough to be eligible. Our purpose is to let the offspring all or mostly fall into the red ($\circ$) group, i.e. they should contain all the 3 true variables. Thus, the 3 true variables can all have high scores. Actually, since the evolutionary algorithm is a heuristic stochastic search algorithm that mimics Darwin's "natural selection" to optimize any given objective function

(Goldberg 1989), the offspring are very likely to fall into the red (∘) group if the universes evolve long enough.

- The evolution will not stop until the offspring are all competitive. If we let the universes evolve too many generations, the universe tends to go farther than just falling into the red (∘) group. The offspring may all be near the lowest AIC point. Because the subset with smallest AIC includes junk variables, the universes may score some junk variables highly, which misleads us to choose junk variables.

As an ensemble member, for the individual GA path, each universe should evolve deeply to make the last-generation offspring competitive, i.e. the last-generation subsets correspond to the small AIC values. But in that way, the offspring may be similar with each other, so that the efficiency of the ensemble is reduced. This is exactly the strength-diversity tradeoff mentioned in 1.1.2. Because the genetic algorithm is heuristic, it contains too much randomness which makes PGA hard to control. Zhu and Chipman used an entropy method to choose the parameter, but it is in general tricky. We will show later in a simulation study that PGA is unstable.

2.4 Conclusion

The exhaustive search is the greediest algorithm that exactly outputs the subset optimizing the criterion. Other searching algorithms like stepwise algorithm, Genetic algorithm also aim to find the global optimal point. From the point of finding the optimal point, exhaustive search is always the best. However, from Figure 2.1, we find that the subset with smallest AIC does not correspond to the "true" model. In that sense, simply finding the optimal point of a criterion should not be the only object. PGA shows us that ensemble methods are able to catch more information of the criterion like AIC and boost the performance of it, thus, are powerful in finding the "true" subset.

Chapter 3
Stochastic Stepwise Ensemble

From the discussion of PGA, we find that the single-path genetic algorithm makes it an algorithm hard to control. We want more structure. Instead of using the genetic algorithm, we construct a Stochastic Stepwise Ensemble by using a stochastic stepwise selection algorithm in each path. The stochastic stepwise algorithm is built on the highly structured approach of stepwise selection, but is inserted stochastic elements so that it can be used in an ensemble. We will show the details about the stochastic elements later.

3.1 The Stochastic Stepwise Algorithm (STST)

Traditional stepwise regression combines forward and backward selection. One step of forward selection is followed by one step of backward selection. In the forward step, each variable other than those already included is added to the current model, one at a time, and the one that can best improve the objective function is retained. In the backward step, each variable already included is deleted from the current model, one at a time, and the one that can best improve the objective function is discarded. The algorithm continues until no improvement can be made by either the forward or the backward step. Instead of adding or deleting one variable at a time, Stearns (1976) develops the Plus-l-Minus-r (l-r) search method which always adds l variables in the forward step and deletes r variables in the backward step. Pudil, Novovicova and Kittler (1994) remove the constraint of fixing l and r by proposing floating search method. All the algorithms are all determinate searching algorithm.

The stochastic stepwise algorithm (STST) is developed for building ensemble. STST also adds or deletes a *group* of variables at a time, but the group size is randomly decided. In traditional stepwise, the group size is one and each candidate variable is assessed. When the group size is larger than one, as is often the case for STST, the total number of variable groups can be quite large. Instead of evaluating all possible groups, only a few are assessed and the best one is chosen. Table 3.1 contains a detailed description of the STST algorithm.

Figure 3.1 illustrates a possible stochastic stepwise path. Each floor of the ellipse stands for the effect of a step (forward or backward). Left ellipse nodes indicate being included in the model and right ellipse nodes indicate being excluded from the model. Initially, we have 10 potential predictors $\{x_1, x_2, x_3, ..., x_{10}\}$. In the first forward step, three variables $\{x_1, x_2, x_5\}$ are added into the

model. Then, one variable $\{x_5\}$ is deleted in the first backward step. In the second forward step, variables $\{x_3, x_4\}$ are added… The algorithm continues doing forward and backward steps till no improvement can be made.

Repeat

1. (Forward Step) Suppose d variables, $\{x_{l_1}, x_{l_2}, ..., x_{l_d}\}$ are not in the current model. Initially, $d = p$.

   a) Determine the number of variables we want to add into the model, or the group size, say $g_f < d$.*

   b) Determine the number of candidate groups that will be assessed, k_f.*

   c) Generate k_f candidate groups of size g_f, each by randomly choosing g_f variables without replacement from the set, $\{x_{l_1}, x_{l_2}, ..., x_{l_d}\}$.

   d) Assess each candidate group by adding it into the current model, one group at a time. The one that can best improve the AIC is added into the model.

2. (Backward Step) Suppose h variables, $\{x_{l_1}, x_{l_2}, ..., x_{l_h}\}$ are not in the current model.

   a) Determine the number of variables we want to delete from the model, or the group size, say $g_b < h$.*

   b) Determine the number of candidate groups that will be assessed, k_b.*

   c) Generate k_b candidate groups of size g_b, each by randomly choosing g_f variables without replacement from the set, $\{x_{l_1}, x_{l_2}, ..., x_{l_h}\}$.

   d) Assess each candidate group by deleting it from the current model, one group at a time. The one that can best improve the AIC is deleted from the model.

Until

No improvement can be made by either the forward or the backward step.

Table 3.1: The stochastic stepwise (STST) algorithm for variable selection

* Details for how the numbers $g_f, k_f, g_b\ k_b$ are determined are given in Section 3.2 and 3.3.

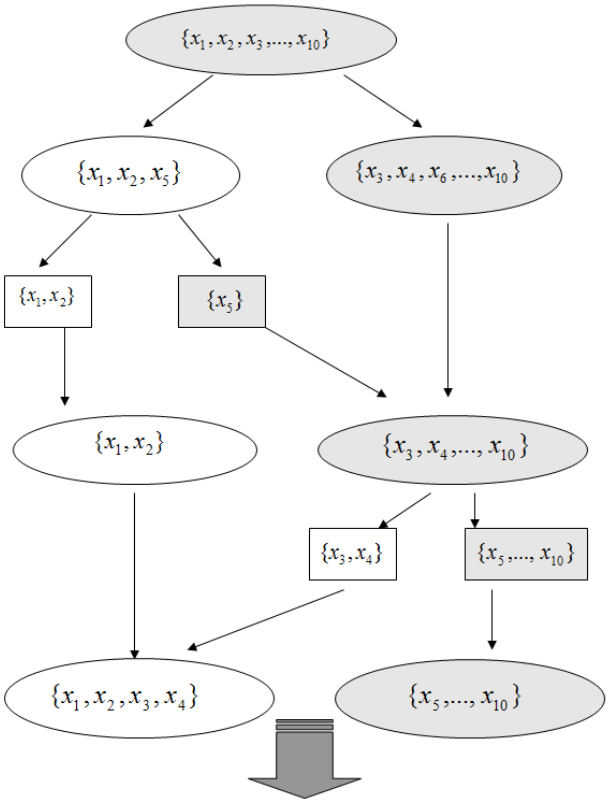

Figure 3.1: An illustration of STST

3.2 Tuning Functions

In this section and the next, we explain how the numbers $g_f, k_f, g_b\ k_b$ are determined. Suppose we are doing a forward (backward) step and the potential predictors to be added (deleted) are $\{x_1, x_2, ..., x_m\}$. First, we need to determine *g*, the number of variables to add (delete). Intuitively, it seems reasonable that *g* should depend on *m,* say

$$g = gsf(m),$$

where "gsf" stands for "group size function". Second, given g, we have a total of $\binom{m}{g}$ possible groups of variables and need to determine k, the number of groups to assess. Intuitively, it also seems reasonable that k should depend on $\binom{m}{g}$, say

$$k = cnf(m, g),$$

where "cnf" stands for "candidate number function".

Because both g and k are integers, we first define a function, *round*: $\Re \to \aleph$, which outputs the nearest integer of a number, e.g., *round*(3.7)=4 and *round*(2.4)=2. Middle points such as 3.5 and 4.5 are rounded up, i.e., *round*(n+0.5)=n+1 for all $n \in \aleph$. Suppose $g, m \in \aleph$ and $g < m$. Let Ψ_m denote the set of all the integers on the interval

$$[round(m^{1/2\lambda}),\ \ round(m^{1/\lambda})]$$

for some λ>1. Our definitions of the functions *gsf* and *cnf* are:

$$gsf(m) \sim Unif(\Psi_m),$$

and

$$cnf(m, g) = round\left(\binom{m}{g}^{1/\kappa}\right)$$

for some κ>1. We discuss how to choose λ and κ in the next section (Section 3.3). Here, we first explain why the functions *gsf* and *cnf* are chosen to have these particular forms.

First, it is important that $g = gsf(m)$ is a stochastic but not a deterministic function. Consider the first forward step and the first backward step. Suppose there are m=p=20 potential predictors. If *gsf(m)* is a deterministic function, say *gsf*(20)=7 and *gsf*(7)=3, then, for all parallel STST paths, only models with 7 predictors are assessed by the forward step and those with 7-3=4 predictors are assessed by the backward step. Many models, such as those with 2 or 3 predictors, are never assessed. Clearly, more flexibility is needed. According to our definition, suppose λ=1.5, then

$gsf(20) \sim Unif[3,4,5,6,7]$. In the first forward step, some STST paths may add 3 variables, while others may add 4, 5, 6 or 7 variables. This increases the diversity of our ensemble.

Second, we want *gsf(m)* to grow with *m*, but we don't want it to grow too fast. This is why functions of the form m^{α} with $\alpha<1$ are considered. A natural upper bound is *gsf(m)* *m*, so the parameter λ must not be smaller than 1. On the other hand, it is easy to see that $gsf(m) \to 1$ for all fixed *m* as $\lambda \to \infty$, which means λ should never be too large, either. The choice of $m^{1/2\lambda}$ as the lower bound is largely empirical, because we want the set Ψ_m to have a reasonable range, one that is neither too big nor too small, so that our overall ensemble is effective

Finally, we want the function *cnf(m,g)* to be monotonically increasing in $\binom{m}{g}$ – as more subsets become available, more candidate groups should be assessed in order to have a reasonable chance of finding an improvement. However, we cannot afford to let this grow linearly in $\binom{m}{g}$ since it can be a very large number and that's why we use $\binom{m}{g}^{1/\kappa}$ for some $\kappa>1$.

These choices are not unique, as long as they are reasonable. Our choices may not be the best ones, but it works well if λ and κ are chosen appropriately.

3.3 Tuning Parameters

In this section, we explain how to choose the parameters λ and κ. Notice that, for variable selection, we cannot resort to cross validation to choose these parameters because we don't know what the true variables are. Recall that, for the PGA (Zhu and Chipman, 2006), the main parameter is how many generations to evolve in each universe, i.e., the depth of each path or simple "path depth". Zhu and Chipman used an entropy method to choose this parameter (Zhu and Chipman 2006; Section 3.2), but it is in general tricky to choose this parameter properly. For STST, each path stops automatically when no improvement can be made. That is, we do not directly control the "path depth" i.e. the number of steps until algorithm stops, but the "path depth" is highly related to the parameters λ and κ.

To illustrate how these parameters affect the "path depth", we use a simulated example from Zhu and Chipman (2006):

$$y_i = x_{i,5} + 2x_{i,10} + 3x_{i,15} + \varepsilon_i, \quad x_{i,1},\ldots,x_{i,20}, \varepsilon_i \overset{iid}{\sim} N(0,1), \ i = 1,2,\ldots,40$$

There are 20 predictors, but only three of them are actually used in the model to generate y. For every $\lambda \in \{1, 1.1, 1.2,\ldots, 3.9, 4\}$ and $\kappa \in \{1.5, 2, 2.5, \ldots, 14.5, 15\}$, we ran STST for 300 times and recorded the depth of each path. Figure 3.1 plots the average depths against all (λ, κ)-pairs. From the figure, we can see:

First, at the left of the dotted line, the depths are small (around 2). Because the value of λ is close to 1, the group sizes are relatively large compared with the total number of available variables. Correspond to the STST algorithm (Table 3.1), the group sizes g_f and g_b are very likely to be larger than $d/2$ and $h/2$, respectively. In that sense, the probability that the candidate groups contain important variables is very high. For the first one or two forward steps, because of the existence of important variables, we always end with adding one of the candidate groups, even though such candi-

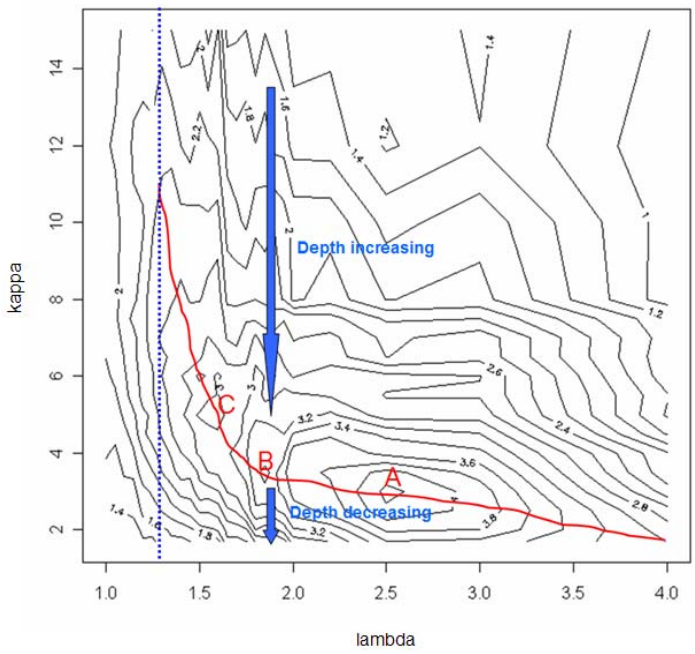

Figure 3.2: Contour plot of the average depths vs λ and κ

date contains many junk variables. Also because of the existence of important variables in the candidate groups, backward steps always end with deleting nothing. So, the paths stop at a subset containing most of the variables (all important variables and most junk variables) within several forward steps. In our particular case, there are 20 potential predictors and 3 of them are true. If we add a group of 10 predictors at the first forward step, and a group of 5 predictors at the second forward step after their competition with some other candidate groups, we should believe that all 3 true variables are included in those 10+5=15 variables. That is why the average depths in this area are around 2. This area is not preferred because the paths select too many junk variables.

Second, at the right side of the blue dotted line, fixing λ, the depth first increases then decreases along with the increase of κ. Because λ is fixed, the group sizes are relatively stable. When the number of candidates is too small (κ is big), only a small number of groups have chance to be assessed. It is very likely that none of the candidate groups are useful. That keeps both forward step and backward step from working effectively, so the paths stop early. In that sense, the paths are too weak to be used as an ensemble. On the other hand, when there are too many candidates (κ is small), accessing a large number of candidate groups makes the algorithm too greedy. It is always able to add the best variables in the forward step and delete the worst ones in the backward step. It makes full use of each step and finish quickly. The paths become too strong so that they are similar to each other. We don't want κ to be neither too large nor too small which both indicate small depths of the paths. Intuitively, fix the value of , the κ value corresponding to the peak of depths is preferred.

The line in Figure 3.1 links the peaks of depths. Which (λ, κ)-pair in the line should we choose? Three parts A, B and C as labeled are attractive.

Part A is around λ=2.5, κ=3. Let's take a look at the possible group sizes i.e., Ψ_m (Section 3.2) with respect to *m*—the total number of available variables.

M	Ψ_m
10-20	{2,3}
8-9	{2}
3-7	{1,2}
1-2	{1}

Table 3.2: Ψ_m for different *m* when λ=2.5

From Table 3.2, we find that the group sizes are very small. Especially in the first couple of forward steps, with almost 20 available potential variables, only the groups containing 2 or 3 variables are assessed. If a small number of candidate groups are assessed, the probability that no important variables are contained is very high, thus, the algorithm stops early. In order to guarantee at least one of the important variables are assessed at each step, i.e., be contained by one of the candidate groups, the number of candidate groups should be large enough (κ be small). However, the bigger the number of candidate groups, the more STST algorithm is similar with traditional stepwise algorithm. Being similar with traditional stepwise indicates loss of diversity. With small group sizes, STST algorithm can easily fall into being too weak or too strong. Even though points around part A look like balance points, they are relatively quite sensitive with respect to the change of κ, thus are not preferred.

Let's move to part B, which is around λ=1.8, κ=3.5. Although the value of λ is larger indicating that the paths have bigger group sizes than part A, the value of κ is 3.5. It is only slightly bigger than 3 of part A. With κ=3 and small group sizes (part A), the STST algorithm suffers from the danger of being too greedy. Now with a similar κ value but bigger group sizes, the chance of each variable being assessed increase greatly. At each step, the STST algorithm can always add the best variables or delete the worst ones. Empirically speaking, the STST paths are too greedy around part B.

Part C has λ around 1.6 and κ around 5 which are both relatively appropriate. And the values of depths nearby C are quite stable with respect to both λ and κ.

Our choice of the (λ, κ)-pair for this data set is λ=1.6, κ=5.

The choice of the (λ, κ)-pair must vary for different data sets, so the similar procedure should be repeated for every new data set.

3.4 Conclusion

STST algorithm is an extension of the traditional stepwise algorithm. It allows adding or eliminating a group of variables in the forward or backward step, respectively and contains two stochastic mechanisms. One is the random determination of the group size to be added or eliminated in each step. The other is the random pick of the candidate groups to be assessed. STST Algorithm is a stochastic searching algorithm which is quite weak. However, stochastic stepwise ensemble is the aggregate of a number of single STST paths. It should be emphasized that the stochastic stepwise

ensemble is no longer a simple searching algorithm. The selecting of the variables is determined by the majority vote.

Two tuning parameters λ and κ are the control of the group size and the number of candidate groups. Different values of the parameters correspond to different strength and diversity of the single STST paths. The variable selection problem is unsupervised, so we cannot apply cross-validation to choose the parameters. We use the contour plot of the average path depths to do that. It is less computationally expensive than cross-validation. The process of choose the parameters is actually the process of finding the balance of strength-diversity tradeoff.

Chapter 4
Simulation Study

In this chapter, we conduct empirical experiments to assess the performance of stochastic stepwise ensembles. The predictive effect is recognized as the golden standard to evaluate a model (Miller, 2002). People always care mostly about the predictive accuracy of a model. The mean squared error of prediction (MSEP) is a popular metric to evaluate the predictive effect. It is defined as:

$$MSEP = E(f(x) - \hat{f}(x))^2,$$

where x is a new observation of the predictors.

4.1 Theoretical Results for MSEP

Let's first take a look at the characteristics of MSEP on linear regression models.

Assume Y is the $n \times 1$ observation vector; β is the $m \times 1$ unknown coefficient vector and X is the $n \times m$ covariate matrix. X, β can be separated into $X = (X_p \quad X_q)$ and $\beta = \begin{pmatrix} \beta_p \\ \beta_q \end{pmatrix}$, where X_p is $n \times p$ matrix; X_q is $n \times q$ matrix; β_p is $p \times 1$ vector and β_q is $q \times 1$ vector.

The full model is

$$Y = X\beta + \varepsilon, \quad \varepsilon \sim N_n(0, \sigma^2 I_n).$$

The partial model is

$$Y = X_p\beta_p + \varepsilon, \quad \varepsilon \sim N_n(0, \sigma^2 I_n).$$

We define $\hat{\beta}$ to be the least square estimate of β based on the full model and $\widetilde{\beta}_p$ to be the least square estimate of β_p based on the partial model which only contains the first p variables. So $\hat{\beta}$ and $\widetilde{\beta}_p$ can be expressed as $\hat{\beta} = (X'X)^{-1}X'Y, \quad \widetilde{\beta}_p = (X_p'X_p)^{-1}X_p'Y$, respectively. Let $\hat{\beta}_q$ be

the last q components of $\hat{\beta}$. Assume x is a new observation of the potential predictors which can be partitioned into $(x_p \quad x_q)$.

We consider two cases:

1. **Theorem 1:**

Suppose the full model $Y = X\beta + \varepsilon$ is true. All the components of β are nonzero. We have

1) $E(\hat{y}) = E(x\hat{\beta}) = x\beta, \quad E(\tilde{y}_p) = E(x_p\tilde{\beta}_p) \neq x\beta \quad except \quad X'_p X_q = 0$

2) $If \quad \mathrm{cov}(\hat{\beta}_q) - \beta_q\beta_q{}' \geq (<)0, \quad E(x\beta - x\hat{\beta})^2 \geq (<)E(x\beta - x_p\tilde{\beta}_p)^2$

Proof: see Appendix

This theorem means if we only choose part of the true variables, we always get a biased prediction. Whether MSEP decreases or increases depends on the condition $\mathrm{cov}(\hat{\beta}_q) - \beta_q\beta_q \geq 0'$ or < 0.

To illustrate the condition, it is helpful to consider the simple 1-dimensional linear regression

$$Y = \beta_1 + X\beta_2 + \varepsilon, \quad \varepsilon \sim N_n(0, \sigma^2 I_n),$$

where Y is the $n \times 1$ observation vector; β_1 and β_2 are unknown coefficients which are all constants and X is the $n \times 1$ vector of observations of the single predictor. Assume $\beta_2 \neq 0$, which means the full model is true. Consider fitting both full model and null model, the sizes of MSEP depends on the value of β_2. The condition becomes $\mathrm{var}(\hat{\beta}_2) - \beta_2{}^2 \geq 0$ or < 0. Notice that $\mathrm{var}(\hat{\beta}_2)$ is a constant determined by X and σ. If $|\beta_2| < \sqrt{\mathrm{var}(\hat{\beta}_2)}$, which means the absolute value of β_2 is relatively small, the fitted model based on the null model has lower MSEP. On the other hand, if $|\beta_2| > \sqrt{\mathrm{var}(\hat{\beta}_2)}$, the absolute value of β_2 is relatively big, we'd better keep the single variable and fit the full model in order to have a lower MSEP.

Moving back to multiple regression, although the value of an individual coefficient cannot determine the property of the matrix, the main principle still holds. If a predictor has a large

coefficient (absolute value), which means such predictor is important, we definitely should select it. Although the coefficient of a predictor is nonzero, its absolute value is close to zero. It also indicates that such predictor is almost useless, so we should not select it.

2. **Theorem 2:**

Suppose the partial model $Y = X_p\beta_p + \varepsilon$ is true. All the components of β_p are nonzero while all the components of β_q are zero. We have

1) $E(\hat{y}) = E(x\hat{\beta}) = x_p\beta_p, \quad E(\tilde{y}_p) = E(x_p\tilde{\beta}_p) = x_p\beta_p$

2) $E(x_p\beta_p - x\hat{\beta})^2 \geq E(x_p\beta_p - x_p\tilde{\beta}_p)^2$

Proof: see Appendix

If we include more variables into the model besides the true ones, we can still get an unbiased prediction. However, MSEP will definitely increase. The second conclusion can be seen as the special case of 2) in Theorem 1. If the coefficients of some predictors are all zero, the condition automatically becomes $\mathrm{cov}(\hat{\beta}_q) - \beta_q\beta_q' = \mathrm{cov}(\hat{\beta}_q) \geq 0$, so including such predictors will increase the MSEP.

As said at the beginning of this chapter, we use MSEP to assess the performance of the STST ensemble. Based on the results of the theorems, if the coefficients of the true predictors are all big enough, in order to get lowest MSEP, we should include all of the true variables and exclude all junk variables. So, to assess STST ensemble, we just simply check if it can choose the true subset. We conduct such experiments in section 4.2. If the coefficients of some true predictors are close to zero, it is hard to determine whether selecting them is better or discarding them is better. In this sense, to assess STST ensemble, we have to compute the MSEP values directly. This part is showed in section 4.3.

4.2 Study 1

In this section, we consider the model:

$$y_i = x_{i,5} + 2x_{i,10} + 3x_{i,15} + \varepsilon_i, \quad x_{i,1},...,x_{i,20},\varepsilon_i \quad iid \sim N(0,\ 1) \quad i = 1,2,...,40$$

We have 20 potential variables and only 3 of them are true. The sample size is 40. The coefficients of the true variables are 1, 2 and 3, obviously nonzero. So according to the results in last section, in order to achieve better MSEP, we have to choose exactly the subset $\{x_5, x_{10}, x_{15}\}$.

When Zhu and Chipman (2006 Section 4.3) compared PGA with Traditional Stepwise AIC or BIC, Exhaustive Search using AIC or BIC and a Bayesian method SSVS (George and McCulloch, 1993), the same model was used. They repeated the simulation 100 times and generated 100 data sets, then, applied all the mentioned variable selection algorithms to the data sets. For each data set, if an algorithm successfully chose $\{x_5, x_{10}, x_{15}\}$, it was recorded as a "hard" success. If an algorithm ranked all the true variables ahead of the junk variables, it was recorded as a "soft" success. We applied STST ensemble to the same data sets and also output the time of hard successes and soft successes. We quote the results from Zhu and Chipman (2006) and add the results of STST.

Method	Time of Successes	Method	Time of Successes
Traditional Stepwise AIC	1	SSVS (1,5; hard)	81
Traditional Stepwise BIC	20	SSVS (1,10; hard)	88
Exhaustive AIC	3	SSVS (10, 100; hard)	58
Exhaustive BIC	28	SSVS (10, 500; hard)	84
PGA (hard)	77	SSVS (1,5; soft)	97
PGA (soft)	97	SSVS (1, 10; soft)	97
STST (hard	74	SSVS (10, 100; soft)	100
STST (soft)	98	SSVS (10, 500; soft)	100

Table 4.1: Performance of different algorithms

From the result, we find that STST is a competitive algorithm. It performs much better than the traditional stepwise and exhaustive search using AIC or BIC as objective function. For both "hard" and "soft" metrics, STST is at the same level as PGA and SSVS.

Both PGA and STST give about 75 out of 100 "hard" successes. It is not bad comparing with the SSVS algorithm. However, we have to think what happened to the rest 25 simulations?

Let's take a look at the illustration in Section 2.2 again. In Figure 2.1, the bottom red ( ) part is the subsets containing all the three true variables. However, there is a small gap that separates this part into two parts.

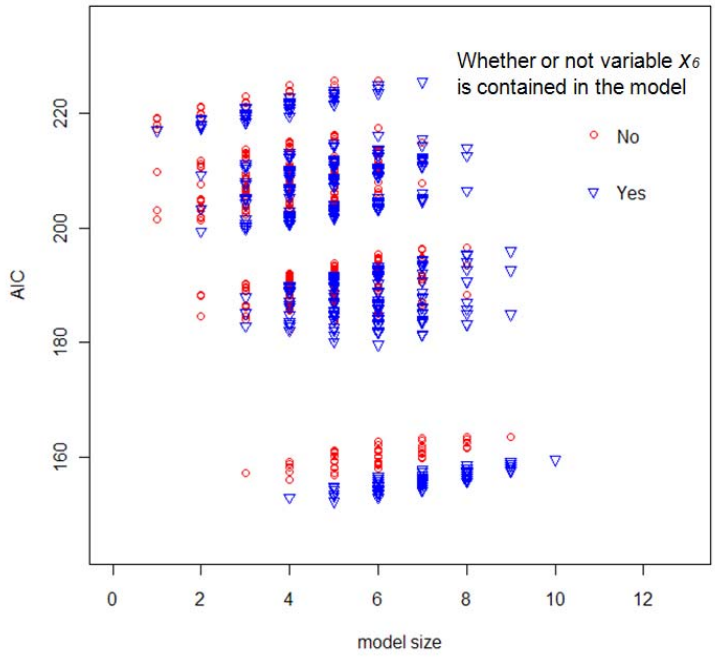

Figure 4.1: AIC values vs subset sizes

From Figure 4.1, we see that the gap is caused by variable x_6. The top part does not contain x_6 but the bottom part does. The subsets containing x_2, x_5, x_6, x_8 are at the bottom. So if only consider the distribution of AIC, the variable x_6 together with x_2, x_5, x_8 determine the models with the smallest AIC values. The variable x_6 performs just like a true variable. Technically, it is impossible for the algorithms like PGA and STST to figure out x_6 is different with x_2, x_5, x_8. Facing such problem, PGA and STST have to give a "wrong" result containing x_6.

Why is x_6 so important? We find that variable x_6 has a larger correlation with the error ε than the other junk variables does (Table 4.2). We generated the predictors and errors independently. However, the sample correlations may not definitely be close to zero. If the correlation of a junk variable and a true variable or the error is large, such junk variable becomes "important". The distribution of the sample correlation is given by the following result:

Junk Variable	Correlation With ε
x_1	-0.138
x_3	-0.175
x_4	-0.0017
x_6	**0.330**
x_7	0.033
x_9	-0.104
x_{10}	0.224

Table 4.2: Correlations of junk variables with error

Theorem 3 (Kern O. Kymn 1968):

Suppose a pair of random variables (*x, y*) have a bivariate normal distribution and assume that the correlation coefficient ρ between *x* and *y* is given as ρ=0. We take a random sample of size *n* of a pair of independent variates $(X_1, Y_1),...,(X_n, Y_n)$ from the above population and define

$$r = \frac{S_{XY}}{\sqrt{S_{XX} S_{YY}}},$$

where S_{XY} is the sample covariance and S_{XX}, S_{YY} are the sample variance. Then the statistic $S=(1+r)/(1-r)$ is distributed as *F* with degree of freedom (*n-2, n-2*).

The statistic *S* is a monotone increasing function with respect to *r*. Figure 4.2 displays 5 cumulative distribution functions of *r* at n=30, 40, 50, 100, 200 in yellow, red, blue, green and black, respectively.

From Figure 4.2, we find that the sample correlations are almost between (-0.5, 0.5). And as the sample size increases, the range decreases. This is reasonable because the larger the sample size is, the harder the sample of two variables to be highly correlated.

In our simulation, we had 20 predictors and an error. The sample size was 40. Check the corresponding line, the probability of the sample correlation beyond (-0.3, 0.3) is quite big. It is easy for the sample of a junk variable to be highly correlated with the sample of a true variable or the error.

We tried doing the same simulations with larger sample sizes (more than 100). We could get more than 95 "hard" successes for both PGA and STST.

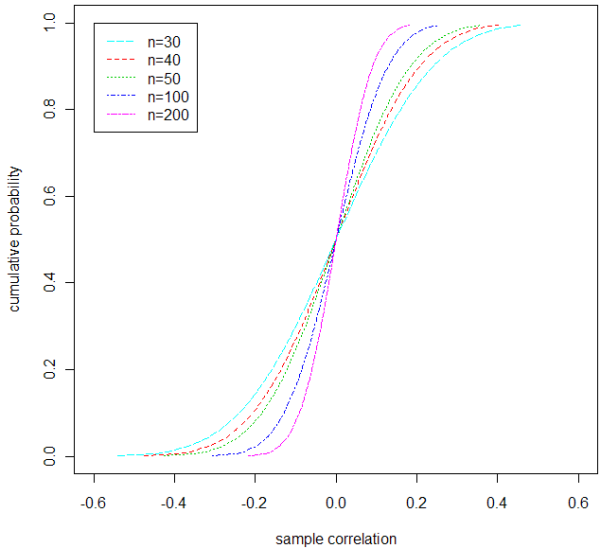

Figure 4.2: Cumulative probability functions of sample correlations with different sample size

Generally, when the sample size is small, none of the variable selection criteria or algorithms can totally get rid of the high correlation problem.

4.3 Study 2

In last section, we use model

$$y_i = x_{i,5} + 2x_{i,10} + 3x_{i,15} + \varepsilon_i, \quad x_{i,1},\ldots,x_{i,20},\varepsilon_i \quad iid \sim N(0,\ 1) \quad i = 1,2,\ldots,40,$$

where the coefficients of the true variables are obviously nonzero. So it is quite clear that $y = x_5 + 2x_{10} + 3x_{15} + \varepsilon$ can be treated as the true model. From the aspect of MSEP metric, such model is also preferred. However, when some coefficients of the true variables are close to zero, it is hard to identify the right model. According to Theorem 1, if we use MSEP metric, which model is better depends on the values of the coefficients. Yang (2007) discusses a motivating numerical example where he considered the simplest 1-dimentional linear regression model

$$y_i = \beta_1 + \beta_2 x_i + \varepsilon_i, i = 1,2,...,n .$$

There are only two models: full model and null model. It is possible to have the simple expressions of MSEP for different criteria like AIC, BIC. The expressions depend on the value of β_1, the distribution of $\{\varepsilon_i\}, i = 1,2,...,n$ and the sample values of $\{x_i\}, i = 1,2,3,...,n$. The MSEP of different criteria are compared under different values of β_1.

Our purpose of this section is to compare STST with PGA and traditional stepwise with AIC and BIC as objective functions. We follow the same routine but considered a more complicated model so that STST and PGA could be applied. We considered the model:

$$y_i = \alpha x_{i,5} + 2x_{i,10} + 3x_{i,15} + \varepsilon_i, \quad x_{i,1},...,x_{i,20},\varepsilon_i \quad iid \sim N(0, \ 1) \quad i = 1,2,...,n .$$

We let α change from 0 to 1 and fitted the model based on the outcome of AIC, BIC, STST and PGA, respectively. Then we used the fitted model to predict the response of a new observation of the predictors (generated from $N_{20}(0, \ I_{20})$). However, because we did not have simple expressions for the MSEP, for a certain value of α, we simulated 100 times and calculate the MSEP values of all the competitors for 100 times. (See some notes in Appendix). We also recorded the times that variable x_5 were included in the final model for each algorithm. We repeated the procedure for different sample size n=150, 300, 700 and plotted the average MSEP values against α and times x_5 being included against α.

From Figure 4.3, 4.4, 4.5, we find that there are three regions of α, α small, α large and α in-between. When α is near zero, regarding the MSEP values, STST ≈ PGA<BIC<AIC. When α becomes larger, AIC<BIC<PGA ≈ STST. When α is very large, STST ≈ BIC<AIC<PGA.

According to Theorem 1 (section 4.1), theoretically, there exists a threshold value for α, say C. When $\alpha < C$, including variable x_5 indicates the increase of MSEP. When α >C, not including x_5 increases MSEP.

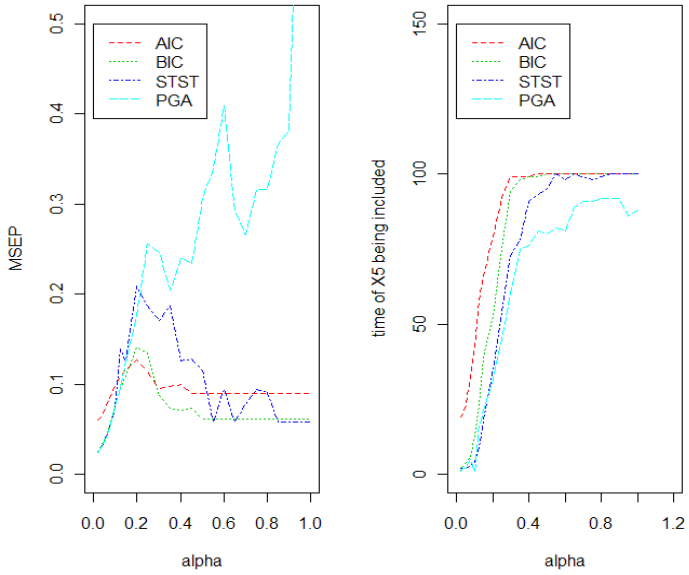

Figure 4.3: MSEP versus α (left) and times of x_5 being included versus α (right) for different algorithms when sample size n=150

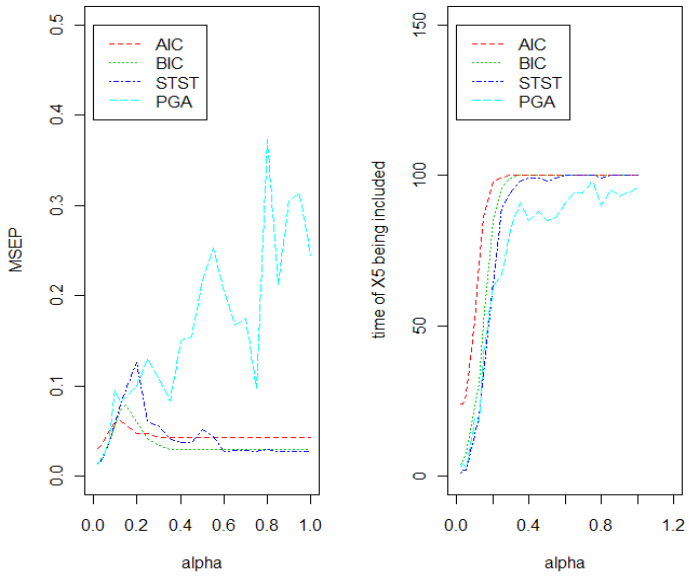

Figure 4.4: MSEP versus α (left) and times of x_5 being included versus α (right) for different algorithms when sample size n=300

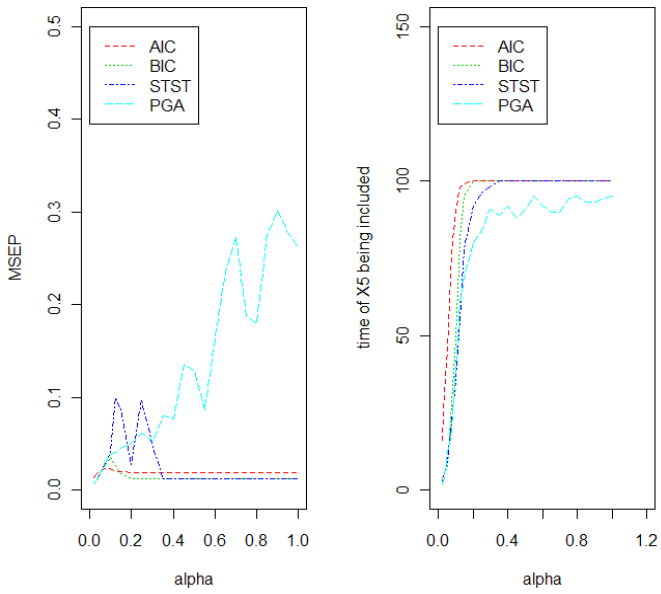

Figure 4.5: MSEP versus α (left) and times of x_5 being included versus α (right) for different algorithms when sample size n=700

When α is small, e.g., α < C, models containing x_5 will have larger MSEP values. STST and PGA are most unlikely to contain x_5 and AIC tends to choose x_5 more often than BIC does. So for the MSEP values, STST ≈ PGA<BIC<AIC.

As α increases, e.g., when α exceeds C, models containing x_5 will have smaller MSEP values. All the algorithms still perform in the same way. The order of the MSEP values is inverted to AIC<BIC<PGA ≈ STST.

When α is large enough, PGA starts to perform differently from the other methods. It always has a non-zero probability of missing x_5 , although x_5 is quite important (with a large coefficient). Actually, the probability that PGA misses x_5 is not very large. However, missing x_5 is disastrous for MSEP. Regarding other algorithms, AIC and BIC 100% contain x_5 and STST is almost 100% except for some small fluctuations. From that point of view, they are the same. But AIC tends to include more junk variables, and according Theorem 2, that increases MSEP. So we have STST ≈ BIC<AIC<PGA.

Although STST is an algorithm based on AIC, it performs like BIC. This is not a surprise for us. It is well known that BIC tends to choose a subset of predictors with small size. And BIC is a consistent variable selection criterion which means it will asymptotically choose the true model as the sample size goes to infinity. STST also tends to choose less variables comparing with AIC. STST is an algorithm that tries to choose the true model when the sample size is relatively small. When the value of α is large, the lines of STST and BIC seem to overlap with each other. Actually, the line of STST is a little bit lower than that of BIC. STST tends to choose less variables even comparing with BIC.

PGA seems out of control in our simulations when the value of α is close to 1. It always loses x_5 for several times out of 100 times. Although PGA has a high probability to choose the true model (section 4.2), it also has a relatively high probability to lose important variables comparing with STST. If we continue increasing the value of α, PGA selects x_5 more often but still less than STST does. If α becomes greater than 2, then, variable x_{10} with coefficient 2 takes the place of x_5. PGA tends to lose the important variables with relatively small coefficients sometimes. PGA is an algorithm using genetic algorithm to produce each path. But genetic algorithm is hard to control. STST is an algorithm based on stepwise algorithm which is more structural, thus, easier to control. Our method of using contour plot of depths to choosing the parameters appears to work well. For the ensemble methods with stochastic mechanisms, uncertainty always exists. That is why STST also has some small fluctuations.

The middle region shrinks with increase of the sample size. The larger the sample size is, the more information we can get from the data. The algorithms become more sensitive to the change of α.

Chapter 5
Conclusions and Future Research

The objective of the thesis is to propose a Stochastic Stepwise ensemble method for variable selection. We use a stochastic stepwise algorithm to build the individual paths of the ensemble. The final selection is made by aggregating the results from many Stochastic Stepwise paths. In Chapter 1, we take an overview of statistical learning and briefly introduce three popular and powerful ensemble methods: Bagging, Random Forest and AdaBoost. The strength-diversity tradeoff plays an enormous role in guiding our research. Chapter 2 first reviews the variable selection problem and its two big challenges – computation and criterion. Then, we discuss the Parallel Genetic Algorithm. In Chapter 3, we propose Stochastic Stepwise Algorithm. The key point is instead of adding or deleting one variable at each step, we randomly add or delete a group of variables at each step. We use STST algorithm to produce the individual paths of the ensemble. The stepwise algorithm is more structural than genetic algorithm, thus, is easier to control. Contour plot of depths of the STST paths is used to choose the tuning parameters. The tuning process is actually the process of finding the balance of the strength and diversity. In simulation studies, we compare STST to other algorithms. When all the coefficients of the true variables are obviously nonzero, STST performs as well as PGA and SSVS and are much more likely to choose the true model than the traditional methods do. When some coefficients of the true variables are close to zero, we find that STST performs like BIC and is more stable than PGA.

In the future, we plan to investigate the following topics:

1. Although our definitions of the tuning functions $gsf(\ )$, $cnf(\ ,\ )$ work well if we choose the proper parameters, they may not be the best. We will try other possible ways to define the functions. If the tuning functions and tuning parameters are changed, the contour plot of depth will look different. The trick to choose the proper parameters may be different as well, but it basically has to follow the same thought of finding the balance of strength and diversity.
2. Although the tuning process is not complicated, we still cannot handle to do it each time we generate a new dataset in the large simulation study. In that sense, it would be better if the algorithm can do the tuning process automatically.

3. In the current version of STST, we only use AIC as the objective function. Actually, the paths are not necessarily to have a unique objective function. Different paths can use different criteria, say some use AIC; some use BIC; some use Cp…This will bring the algorithm larger diversity.

4. STST, as a variable selection algorithm, can deal with the large p small n problem i.e. the number of potential predictors is greater than that of the observations. Generally, in that case, it is impossible to fit the model. However, for STST, when build a path, we can randomly choose d predictors ($d<n<p$) and only use them in the procedure. Although for a single path, only d predictors have effect, the consolidation of the paths contains the information of all predictors. All the predictors play an equivalent game and the ones get high scores are regarded as better than others.

5. In this thesis, as illustrations, we only considered the simplest linear regression model. It is easy to apply STST to other generalized linear regression models. The only difference is to change the model fitting procedure.

Appendix

A.1 Proofs of Theorem 1 and 2

Lemma:

Assume A is a nonsingular symmetric matrix and it can be separated as

$$A = \begin{pmatrix} B & C \\ C' & D \end{pmatrix}$$

where $B : p \times p,\ \ C : p \times q,\ \ D : q \times q$.

If $B^{-1},\ \ D^{-1}$ exist, we have

$$A^{-1} = \begin{pmatrix} B_1 & C_1 \\ C_1' & D_1 \end{pmatrix} = \begin{pmatrix} \left(B - CD^{-1}C'\right)^{-1} & -B_1CD^{-1} \\ -D^{-1}C'B_1 & D^{-1} + D^{-1}C'B_1CD^{-1} \end{pmatrix}$$

$$= \begin{pmatrix} B^{-1} + B^{-1}CD_1C'B^{-1} & -B^{-1}CD_1 \\ -D_1C'B^{-1} & \left(D - C'B^{-1}C\right)^{-1} \end{pmatrix}$$

The proof of the lemma can be found in most linear algebra text books or linear regression books, e.g. Hocking (2003).

Before the proof of the theorems, we define

$$X'X = \begin{pmatrix} X_p' \\ X_q' \end{pmatrix} \begin{pmatrix} X_p & X_q \end{pmatrix} = \begin{pmatrix} X_p'X_p & X_p'X_q \\ X_q'X_p & X_qX_q \end{pmatrix} = \begin{pmatrix} B & C \\ C' & D \end{pmatrix},$$

where $B = X_p'X_p,\ \ C = X_p'X_q,\ \ D = X_q'X_q$

and

$$(X'X)^{-1} = \begin{pmatrix} B_1 & C_1 \\ C_1' & D_1 \end{pmatrix},$$

where $B_1 : p \times p,\ \ C_1 : p \times q,\ \ D_1 : q \times q$.

Proof of Theorem 1:

The true linkage is $Y = X\beta + \varepsilon$.

1) $E(\hat{y}) = E(x\hat{\beta}) = xE(\hat{\beta}) = xE\left((X'X)^{-1}X'Y\right) = x(X'X)^{-1}X'X\beta = x\beta$

$$E(\tilde{y}_p) = E(x_p\tilde{\beta}_p) = x_pE(\tilde{\beta}_p) = x_p({X_p}'X_p)^{-1}{X_p}'X\beta$$
$$= x_p({X_p}'X_p)^{-1}{X_p}'(X_p \quad X_q)\begin{pmatrix}\beta_p\\ \beta_q\end{pmatrix}$$
$$= x_p({X_p}'X_p)^{-1}{X_p}'(X_p\beta_p + X_q\beta_q) = x_p\beta_p + x_p({X_p}'X_p)^{-1}{X_p}'X_q\beta_q$$

If $\beta_q = 0,\ \ E(\tilde{y}_p) = x_p\beta_p = x\beta$, otherwise $E(\tilde{y}_p) \neq x\beta$.

2) $E(x\beta - x\hat{\beta})^2 = Var(x\hat{\beta}) = x(X'X)^{-1}x'\sigma^2 = (x_pB_1{x_p}' + x_pC_1{x_q}' + x_qC_1{x_p}' + x_qD_1{x_q}')\sigma^2$

$$E(x\beta - x_p\tilde{\beta}_p)^2 = Var(x\beta - x_p\tilde{\beta}_p) + E^2(x\beta - x_p\tilde{\beta}_p)$$
$$= x_p({X_p}'X_p)^{-1}{x_p}'\sigma^2 + (x_pB^{-1}C\beta_q - x_q\beta_q)^2$$
$$= x_pB^{-1}{x_p}'\sigma^2 + (x_pB^{-1}C - x_q)\beta_q{\beta_q}'(x_pB^{-1}C - x_q)'$$

$$\therefore E(x\beta - x\hat{\beta})^2 - E(x\beta - x_p\tilde{\beta}_p)^2$$
$$= (x_p(B_1 - B^{-1}){x_p}' + x_pC_1{x_q}' + x_qC_1{x_p}' + x_qD_1{x_q}')\sigma^2 - (x_pB^{-1}C - x_q)\beta_q{\beta_q}'(x_pB^{-1}C - x_q)'$$
$$= (x_pB^{-1}CD_1C'B^{-1}{x_p}' - x_pB^{-1}CD_1{x_q}' - x_qD_1C'B^{-1}{x_p}' + x_qD_1{x_q}')\sigma^2$$
$$-(x_pB^{-1}C - x_q)\beta_q{\beta_q}'(x_pB^{-1}C - x_q)'$$

$$= (x_pB^{-1}C - x_q)D_1\sigma^2(x_pB^{-1}C - x_q)' - (x_pB^{-1}C - x_q)\beta_q{\beta_q}'(x_pB^{-1}C - x_q)'$$
$$= (x_pB^{-1}C - x_q)(D_1\sigma^2 - \beta_q{\beta_q}')(x_pB^{-1}C - x_q)'$$
$$= (x_pB^{-1}C - x_q)(\mathrm{cov}(\hat{\beta}_q) - \beta_q{\beta_q}')(x_pB^{-1}C - x_q)'$$

$$\therefore If\ \ \mathrm{cov}(\hat{\beta}_q) - \beta_q{\beta_q}' \geq 0,\ \ E(x\beta - x\hat{\beta})^2 \geq E(x\beta - x_p\tilde{\beta}_p)^2$$

$$If \quad \mathrm{cov}(\hat{\beta}_q) - \beta_q \beta_q{}' < 0, \quad E(x\beta - x\hat{\beta})^2 < E(x\beta - x_p \widetilde{\beta}_p)^2$$

Proof of Theorem 2:

The true linkage is $Y = X_p \beta_p + \varepsilon$.

1) $E(\widetilde{y}_p) = E(x_p \widetilde{\beta}_p) = x_p E(\widetilde{\beta}_p) = x_p E\left((X_p{}' X_p)^{-1} X_p{}' Y\right) = x_p (X_p{}' X_p)^{-1} X_p{}' X_p \beta_p = x_p \beta_p$

$$E(\hat{y}) = E(x\hat{\beta}) = xE(\hat{\beta}) = xE\left((X'X)^{-1} X'Y\right) = x(X'X)^{-1} X'X_p \beta_p$$
$$= x \begin{pmatrix} B_1 & C_1 \\ C_1{}' & D_1 \end{pmatrix} \begin{pmatrix} X_p{}' \\ X_q{}' \end{pmatrix} X_p \beta_p = x \begin{pmatrix} B_1 & C_1 \\ C_1{}' & D_1 \end{pmatrix} \begin{pmatrix} B \\ C' \end{pmatrix} \beta_p = x \begin{pmatrix} B_1 B + C_1 C' \\ C_1{}' B + D_1 C' \end{pmatrix} \beta_p$$
$$= x \begin{pmatrix} I_p + B^{-1} C D_1 C' - B^{-1} C D_1 C' \\ - D_1 C' + D_1 C' \end{pmatrix} \beta_p = x \begin{pmatrix} I_p \\ 0 \end{pmatrix} \beta_p = x_p \beta_p$$

2) Because both $x\hat{\beta}$ and $x_p \widetilde{\beta}_p$ are unbiased estimate of $x_p \beta_p$

$$E(x_p \beta_p - x\hat{\beta})^2 - E(x_p \beta_p - x_p \widetilde{\beta}_p)^2 = Var(x\hat{\beta}) - Var(x_p \widetilde{\beta}_p)$$
$$= x(X'X)^{-1} x' \sigma^2 - x_p (X_p{}' X_p)^{-1} x_p{}' \sigma^2$$
$$= (x_p B_1 x_p{}' + x_p C_1 x_q{}' + x_q C_1 x_p{}' + x_q D_1 x_q{}') \sigma^2 - x_p B^{-1} x_p{}' \sigma^2$$
$$= (x_p (B_1 - B^{-1}) x_p{}' + x_p C_1 x_q{}' + x_q C_1 x_p{}' + x_q D_1 x_q{}') \sigma^2$$
$$= (x_p B^{-1} C - x_q) D_1 \sigma^2 (x_p B^{-1} C - x_q)' \geq 0$$

The last step is because of $D_1 \sigma^2 = \mathrm{cov}(\hat{\beta}_q) \geq 0$.

The proofs are motivated by a similar theorem about MSE of linear regression models (Theorem 4.1 from Hocking (2003)).

A.2 Notes for Study 2

Following is the procedure of doing study 2

1. Generate an observation of predictors $z = (z_1, z_2, ..., z_{20})$, which will be used to calculate MSEP throughout the study.

2. For a value of α.

A. Simulate a data set ($\{x_{i,k}, \varepsilon_i\}, k = 1,2,...,20; \quad i = 1,2,...,n$) and calculate the observations of the responses $y_i = \alpha x_{i,5} + 2x_{i,10} + 3x_{i,15} + \varepsilon_i, i = 1,2,...,n$

B. Applying an algorithm to the data set, suppose we choose predictors $\{x_{c_1}, x_{c_2}, ..., x_{c_p}\}$.

a) Check whether the variable x_5 is included in the selected model or not.

b) Compute the MSEP values. Let $X_p : n \times (p+1)$ denote the matrix combined by the column vectors of the sample values corresponding to $\{1, x_{c_1}, x_{c_2}, ..., x_{c_p}\}$ and $X_T : n \times 3$ denote the matrix combined by the column vectors of the sample values corresponding to $\{x_5, x_{10}, x_{15}\}$. Let β_T denote the true coefficients $(\alpha, \ 2, \ 3)'$.

Thus, $\hat{f}(z) = z\hat{\beta} = z_p \hat{\beta}_p, \quad where \quad z_p = (1, z_{c_1}, z_{c_2}, ..., z_{c_p}), \quad \hat{\beta}_p = (X_p{}' X_p)^{-1} X_p{}' Y$

The MSEP can be calculated by

$$
\begin{aligned}
& E(f(z) - \hat{f}(z))^2 \\
&= E\left[f(z) - E(\hat{f}(z)) + E(\hat{f}(z)) - \hat{f}(z)\right]^2 \\
&= [f(z) - E(\hat{f}(z))]^2 + Var(\hat{f}(z)) \\
&= (z_p \beta_T - z_p (X_p{}' X_p)^{-1} X_p{}' X_T \beta_T) + z_p (X_p{}' X_p)^{-1} z_p{}' \sigma^2
\end{aligned}
$$

C. Repeat B using other algorithms.

D. Repeat steps A-C 100 times. Record the average MSEP value and the times x_5 being selected for each algorithm.

3. Change the value of α and do step 2.

Bibliography

Akaike, H. (1973), Information Theory and an Extension of the Maximum Likelihood Principle, in *Second International Symposium on Information Theory*, pp. 267-281.

Berk, R. (2008), *Statistical Learning from a Regression Perspective*, Springer.

Breiman, L. (1996), Bagging Predictors, *Machine Learning*, **24**, 123-140.

Breiman, L. (2001), Random Forests, *Machine Learning*, **45**, 5-32.

Freund, Y., and Schapire, R. (1996), Experiments with a New Boosting Algorithm, *Machine Learning: Proceedings for the Thirteenth International Conference,* Morgan Kauffman, San Francisco, pp. 148-156.

Friedman, J., Hastie, T. and Tibshirani, R. (2000), Additive Logistic Regression: A Statistical View of Boosting, *Annals of Statistics* **28**, 337-407.

George, E., and McCulloch, R. (1993), Variable Selection via Gibbs Sampling, *Journal of the American Statistical Association*, **88**, 881-889.

Goldberg, D. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley.

Hastie, T., Tibshirani, R. and Friedman, J. (2001), *The Elements of Statistical Learning*, Springer.

Hocking, R., (2003), *Methods and Applications of Linear Models*, 2nd Edition, Wiley-Interscience.

Kuncheva, L., (1993), Genetic Algorithm for Feature Selection for Parallel Classifiers, *Information Processing Letters*, **46**, 163-168.

Kymn, K., (1968), The Distribution of the Sample Correlation Coefficient Under the Null Hypothesis, *Econometrica*, **Vol. 36**, **No. 1**, pp. 187-189.

Miller, A., (2002), *Subset Selection in Regression*, 2nd Edition, Chapman & Hall/CRC.

Pudil, P., Novovicova, J. and Kittler, J. (1994), Floating Search Methods in Feature Selection, *Pattern Recognition Letters*, **15**, 1119-1125.

Schwarz, G. (1978), Estimation the Dimension of a Model, *The Annals of Statistics*, **6**, 461-464.

Stearns, S.D. (1976), On Selecting Features for Pattern Classifiers, *Third International Conference on Pattern Recognition*, Coronado, Ca, USA, 71-75

Tukey, J. (1977), *Exploratory Data Analysis*, Addison-Wesley.

Yang, Y., (2005), Can the Strengths of AIC and BIC be Shared? A Conflict Between Model Identification and Regression Estimation, *Biometrika* **92**, 937-950.

Yang, Y., (2007), Prediction/Estimation with Simple Linear Models: Is It Really That Simple, *Econometric Theory*, **23**, 1-36.

Zhu, M., and Chipman, H., (2006), Darwinian Evolution in Parallel Universes: A Parallel Genetic Algorithm for Variable Selection, *Technometrics*, **48**, 491-502.

Zhu, M., (2008), Kernels and Ensembles: Perspectives on Statistical Learning, *The American Statistician*, **62**, 97-109.