Prediction and Classification Using Projection Pursuit Regression in SAS
Prediction and Classification Using
Projection Pursuit Regression in SAS

by

Seong-Yun Kim

A THESIS

Submitted to the faculty of
INHA UNIVERSITY
in partial fulfillment of the requirements
for the degree of
MASTER

Department of Statistics
February 2000
Prediction and Classification Using Projection Pursuit Regression in SAS

SAS 상에서 사영추적 회귀를 이용한 예측과 분류

2000年 2月

指導教授  朴 憲 鎖

이 論文을 碩士學位 論文으로 提出함

仁荷大學校 大學院

統計學科

金 性 潤
이 논문은 金性潤의 副士學位論文으로 認定함

2000年 2月

主審 全洪錫

副審 朴憲鎮

委員 金真卿
ABSTRACT

Projection pursuit regression and classification (PPR&C) is a tool in data-mining for prediction and classification proposed by Friedman and Stuetzle (1981). One of the most exciting features of PPR&C is to bypass the "curse of dimensionality" caused by the fact that it becomes difficult to collect enough samples for high-dimensional functions. And another exciting feature of that PPR&C is a universal approximator for any continuous function on a compact set when the number of terms is large.

PPR&C provides several criteria for selection of the number of terms. The criteria can help us to justify the choice of the number of terms and to provide stable results.

In this thesis, we developed a SAS macro for PPR&C based on the algorithm proposed by Friedman (1984). We also compared the macro for PPR&C with previous PPR&C programs. The algorithm is constructed with the least squares fit through the Gauss-Newton iteration. The algorithm uses supersmoothing technique for function smoothing. Also the algorithm uses the Hermite polynomials as a function smoother proposed by Hwang et al. (1994), which produces smoother functions than the supersmoothing technique.

We conclude that PPR&C is a new prediction and classification method for data mining and very flexible and useful method.
CONTENTS

1. Introduction ..................................................................................1

2. Projection pursuit regression and classification procedure ........................................3
   2.1 Regression ..............................................................................5
   2.2 Classification .........................................................................7
   2.3 Parameter estimation procedure .............................................9
   2.4 Criteria for term selection .....................................................13
   2.4 Model selection .....................................................................15
   2.6 Smoothing technique .............................................................18

3. Simulation and Comparison ..........................................................22
   3.1 Comparison with nonlinear Regression function .........................22
   3.2 Comparison with Classification data .....................................28

4. Concluding Remarks .....................................................................32

References .......................................................................................33

Abstracts (in Korean) .................................................................36
1. INTRODUCTION

Projection pursuit regression and classification (PPR&C) is a tool in data mining for prediction and classification proposed by Friedman and Stuetzle (1981). The PPR&C method is based on a generalized form of linear models as follows.

\[ P_M(\overline{X}_j) = \overline{y}_j = \beta_0 + \sum_{m=1}^{M} \beta_m f_m(a_m^T \overline{X}_j) + e_j, \quad j = 1, \ldots, n, \]  

(1.1)

where \( \overline{y}_j \) and \( \overline{X}_j \) are observation vectors and \( \{ e_j \} \) is a noise process. The model is a function of projections of \( \overline{X}_j, \ a_m^T \overline{X}_j, \) instead of predictor variables \( \overline{X}_j \) themselves. Therefore, the number of terms in the right hand side of the model in (1.1) can be reduced and simplified. As a result, overfitting problem can be avoided. The most exciting feature of PPR&C method is to bypass the "curse of dimensionality" caused by the fact that it becomes difficult to collect enough samples for high-dimensional functions.

For example, consider points taken from a 10-dimensional uniform distribution on the unit hypercube. If one wishes to enclose 10% of the samples, the edge length of the hypercube is 0.80 (\( x^{10} = 0.1, \text{ then } x = 0.1^{1/10} \approx 0.8 \)). This shows that very large neighborhoods are required to capture even small portion of data. See Friedman (1994) for details of this problem.

And another exciting feature of the PPR&C is a universal approximator for any continuous function on a compact set when the number of term \( M \) is large. That is, for any function \( g(\overline{X}) \)
and any positive $\varepsilon$, there exists a number of term, $M$, $P_M(X)$, such that $||g(X) - P_M(X)|| < \varepsilon$ for every $X$. See DeVore (1991) or Vladimir and Filip (1998) for details.

In Model (1.1), a function form of $f_m$ is not given and the function is predicted in PPR&C, which allows more flexible results than methods with fixed forms of functions such as neural network, which is frequently used in data mining. Also, PPR&C provides several criteria for selection of the number of terms, $M$. The criteria can help us to justify the choice of the number of terms and to provide stable results.

In this thesis, we developed a SAS macro for PPR&C based on the algorithm proposed by Friedman (1984). This algorithm is constructed with the least squares fit through the Gauss–Newton iteration. The algorithm uses supersmoothing technique for function smoothing. On the other hand, Hwang et al. (1994) proposed the Hermite polynomials as a function smoother, which produces smoother functions than the supersmoothing technique. The macro for PPR&C contain both smoothing techniques. And this macro can be shared with useful accomplishments in SAS since the macro is programmed with SAS/IML and SAS/Macro. Especially, for data mining process, this macro can be used together with other modeling methods in SAS Enterprise Miner, which is a popular data-mining tool. We also compared the macro for PPR&C with previous PPR&C programs for simulation.
2. PROJECTION PURSUIT REGRESSION AND CLASSIFICATION PROCEDURE

PPR&C is a statistical learning procedure for multivariate data analysis. This procedure is a tool for finding the most interesting lower dimensional feature of high dimensional data and optimization with respect to this projection direction. The projection part of the term projection pursuit indicates that p-dimensional predictor vector $X_k$ is projected onto the direction vectors $\{a_m, 1 \leq m \leq M\}$ to get the lengths $a_m^T X_k$ of the projections, for $1 \leq m \leq M$, and the pursuit part indicates that the optimization technique is used to find good direction vectors, $a_m$, for $1 \leq m \leq M$.

PPR&C performs prediction and classification based on the following model. For a q-dimensional response observed response vector $Y_k$ and a p-dimensional predictor observed predictor vector $X_k$,

$$Y_{ik} = \beta_0 + \sum_{m=1}^{M} \beta_m f_m(a_m^T X_k) + e_{ik}, \ i = 1, \ldots, q , \ k = 1, \ldots, n. \quad (2.1)$$

Here $Y_{ik}$ is an $i$-th attribute of $k$-th observed vector $Y_k$ and $(e_{ik})$ is a q-dimensional random process with zero expectation. Also, Model (2.1) has three sets of parameters $(\beta_{il}), (\alpha_m)$ and $(f_m)$. Here $\beta_0$ and $\beta_{im}$ are unknown parameters and projection directions, $a_m$, is an unknown p-dimensional parameter vector with $a_m^T a_m = 1$. For convenience, let $\beta_l$ for $l = 0, 1, \ldots, M$ be a
q-dimensional vector whose i-th elements are \( \beta_{ii} \). Also \( f_m \) is an unknown function with \( E(f_m(a_m^T X_k)) = 0 \) and \( E(f_m^2(a_m^T X_k)) = 1 \). In PPR&C procedure, \( a_m, \beta_{im} \) and \( f_m \) are estimated.

PPR&C can be considered as a generalization of neural network with one hidden layer. Usually neural network with one hidden layer can be represented as follows. Each neuron of input and output layer denotes each attribute of predictor and response vector respectively.

\[
Y_{ih} = g_i(\beta_0 + \sum_{m=1}^{M} \beta_{im} f_m(w_m^T X_k + w_{m0})) , \quad i = 1, \ldots, q, \quad k = 1, \ldots, n,
\]

where \( \beta_0 \) and \( w_{m0} \) are bias terms of the i-th output neuron and m-th neuron in the hidden layer, respectively. \( w_m \) is a p-dimensional weight vector linked between m-th hidden neuron and input vector. \( \beta_{im} \) is the weight linked between the i-th output neuron and the m-th hidden neuron. \( g_i \) and \( f_m \) are activation functions of response and hidden neuron. These functions can be taken to be linear or nonlinear functions, among which the popular functions are logistic function, \( f(x) = e^x/(1+e^x) \), or hyperbolic tangent function, \( f(x) = (e^x - 1)/(e^x + 1) \). Comparing with PPR&C, function \( f(x) \) in neural network is fixed. But PPR&C uses flexible nonlinear ridge functions as nonparametric function or linear combination of the Hermite functions. Even though both methods are difficult to interpret, PPR&C is known to give more flexible and stable result than neural network. (See Donoho and Johnstone (1989) or Zao and Atkeson (1992) for details.)

In this thesis, we developed a SAS macro for PPR&C based on
the algorithm proposed by Friedman (1984) and Hwang et al. (1994). This algorithm is constructed with the least squares fit through the Gauss–Newton iteration. The macro for PPR&C can be used supersmoothing technique for function smoothing and a smoothing with the Hermite functions. The feature of the supersmoothing technique is a powerful smoother for any kind of functions since it does not assume a function form. And the smoothing with the Hermite functions tends to be smoother than supersmoothing. Also, the smoothing with the Hermite functions is faster and need less memory space. In the macro, users can choose one of two smoothing techniques for the feature of data.

Another feature of this macro can be shared with useful accomplishments in SAS since the macro is programmed with SAS/IML and SAS/Macro. Especially, for data mining process, this macro can be used together with other modeling methods in SAS Enterprise Miner, which is a popular data-mining tool. For example, we can use a hybrid model that is from PPR&C after Tree based model in SAS Enterprise–Miner. Also, data handling is easy and we can assess performance with another models in SAS at the same time.

2.1 Regression

Usually regression is a method for modeling a set of response variables \( Y_i \) as a function of a set of predictor variables \( X_j \), where \( Y_i \) is \( i \)-th attribute of q-dimensional response vector and
$X_j$ is $j$-th attribute of $p$-dimensional predictor vector. This modeling aims to estimate the conditional expectation of each $Y_i$ given a set of values for the predictor variables $(x_1, x_2, \ldots x_p)$. Projection pursuit regression procedure is a model based this regression modeling method as follows:

$$
\hat{Y}_i(x_1, x_2, \ldots, x_p) = E[Y_i \mid X_1 = x_1, X_2 = x_2, \ldots, X_p = x_p] = \bar{Y}_i + \sum_{m=1}^{M} \beta_{im} f_m(a_m^T X),
$$

with $\bar{Y}_i = E[Y_i], E[f_m] = 0, E[f_m^2] = 1$, and $a_m^T a_m = 1$. Here each response variable is modeled as a different linear combination of functions $f_m$. Each of these functions is a nonlinear smooth function of a linear combination of the predictor variables. Projection pursuit regression uses different two smoothing techniques, which are nonparametric smoothing function called supersmoother and polynomial smoothing function to be discussed section 2.5.

Parameter estimates in projection pursuit regression are obtained by minimizing

$$
L_2 = \sum_{i=1}^{n} W_i \sum_{j=1}^{n} w_i (Y_{ij} - \bar{Y}_i - \sum_{m=1}^{M} \beta_{im} f_m(a_m^T X_j))^2, \quad (2.2)
$$

where $Y_{ij}$ is the $i$-th element of observation $Y$, and $\bar{Y}_i$ is the $i$-th element of intercept vector $\bar{Y}$. Also $w_j$ are weights for observations given by users and $W_i$ are weights for the response variables specified by users. In PPR&C, if $w_j$ and $W_i$ are not specified, both values are set to one. And the projection pursuit regression model is sensitive to the relative scales of the response
variables $Y_i$ such as linear regression model for using squared error loss criterion. As so, if the $Y_i$ have the different variances $\text{Var}(Y_i)$, then one can set $W_i = 1/\text{Var}(Y_i)$. The parameter estimation process is described as section 2.3.

2.2 Classification

Suppose that the response variable has $q$ possible categories, $c_1, \cdots, c_q$ and $l(c_i, c_k)$ is a loss of misclassifying an object from category $c_i$ into category $c_k$. Also suppose that $p(c_i|x_1, \cdots, x_p)$ is a probability to classify category $c_i$ given $x_1, \cdots, x_p$. The loss for classifying into $c_k$ given $x_1, \cdots, x_p$ is

$$\sum_{i=1}^{q} l(c_i, c_k) p(c_i|x_1, \cdots, x_p),$$

and, hence, the misclassification risk is

$$R = E[ \min_{1 \leq k \leq q} \sum_{i=1}^{q} l(c_i, c_k) p(c_i|x_1, \cdots, x_p) ]. \quad (2.3)$$

By defining

$$H_i = 1 \text{ if } Y \text{ belong to category } c_i$$

$$0 \text{ otherwise},$$

one has

$$p(c_i|x_1, \cdots, x_p) = \frac{S_{x_i}}{s_i} E(H_i|x_1, \cdots, x_p),$$

where $\pi_i$ is the unconditional probability or prior for category $c_i$. 

- 7 -
\[ s_i = \sum_{j=1}^{q} w_j \delta(Y_j, c_i), \]
\[ S = \sum_{i=1}^{q} s_i, \]

where \( w_j \) are weights for observations given by users and \( \delta \) is the Kronecker delta function
\[ \delta(a, b) = \begin{cases} 
1 & \text{if } a = b \\
0 & \text{otherwise} 
\end{cases} \]

Then, (2.3) is rewritten as
\[ R = E[ \min_{1 \leq k \leq q} \sum_{i=1}^{q} \frac{\pi_i k(c_i, c_k)}{s_i} E(H_i | x_1, \ldots, x_p)]. \]  
(2.4)

In projection pursuit classification, \( E(H_i | x_1, \ldots, x_p) \) is in the form of
\[ \beta_0 + \sum_{m=1}^{M} \beta_m f_m(a_m^T X). \]

Then, the optimal decision rule given \( x_1, \ldots, x_p \) is to classify into \( c_k^* \) where (2.4) is minimized. But, due to non-convexity of (2.4), it is not desirable to choose the decision rule to minimize (2.4) in practice. See Breiman, Friedman, Olshen and Stone (1983) for details. Instead, projection pursuit classification obtain estimates to minimize
\[ R^* = \sum_{i=1}^{q} W_i \sum_{j=1}^{q} w_j (H_{ij} - \beta_0 - \sum_{m=1}^{M} \beta_m f_m(a_m^T X))^2, \]
where
\[ W_i = \frac{S \pi_i}{s_i} \sum_{j=1}^{q} k(c_i, c_j), \]
\[ H_{ij} = \begin{cases} 
1 & \text{if } Y_j = c_i \\
0 & \text{otherwise} \end{cases} \]

Then the decision rule in projection pursuit classification is to classifying \( j \)-th observation into the category where
\[ \beta_0 + \sum_{m=1}^{M} \beta_{im} f_m(a_m^T X_i) \]

is maximized over \( i \).

2.3 Parameter Estimation Procedure

PPR&C has three sets of parameters, coefficient parameters of functions \( \{ \beta_{im}, 1 \leq i \leq q \} \), projection direction \( \{ a_{jm}, 1 \leq j \leq p \} \), and unknown functions \( \{ f_m, 1 \leq m \leq M \} \) for a given the number of terms \( M \). They are estimated through iterative optimization strategy to minimize equation (2.2). By fixing other two sets, the estimator for one set is obtained. For the \( m \)-th term, \( \{ \beta_{im}, 1 \leq i \leq q \} \) is estimated with \( \{ a_{jm}, 1 \leq j \leq p \} \) and \( f_m \) fixed. Next procedure, \( \{ a_{jm}, 1 \leq j \leq p \} \) is obtained with \( f_m \) fixed and \( \beta_{im} \) updated for \( 1 \leq i \leq q \). Also \( f_m \) is obtained through smoothing procedure with \( a_{jm} \) and \( \beta_{im} \) fixed. These procedures are repeated until convergence. The following is detailed estimation process.

Step 1. Obtaining estimates for the model with one term

Consider the model with one term,

\[ Y_{ij} = \beta_0 + \beta_{a_1} f_1(a_1^T X_{ij}) + e_{ij}, \ i=1, \ldots, q, \ j=1, \ldots, n. \]  

(2.5)

Given \( a_1 \), prediction of \( f_1 \), \( \hat{f}_1 \) is obtained through the super smoothing or smoothing with the Hermite functions. The smoothed value of \( f_1(a_1^T X_i) \) is standardized for the weighted mean to be
zero and for the weighted variance to be one. Let the standardized smoothing function be \( \hat{f}_1 \). For \( \beta_\alpha \), the estimate is obtained in regression of \( Y_{ij} \) on \( \hat{f}_1(a_1^TX_j) \). The loss, \( L_2 \) for the model in (2.5) can be written as

\[
L_2 = \sum_{i=1}^{k} W_i E[ (Y_i - \bar{Y}_i - \beta_\alpha f_1(a_1^TX_i))^2 ]
\]

\[
= \sum_{i=1}^{k} W_i \sum_{j=1}^{q} w_j (Y_{ij} - \bar{Y}_i - \beta_\alpha f_1(a_1^TX_j))^2, \quad (2.6)
\]

where \( Y_{ij} \) is the \( i \)-th element of observation \( Y_i \) and \( \bar{Y}_i \) is the \( i \)-th element of intercept vector \( \bar{Y} \). We solved least square estimates of the parameter \( \beta_\alpha \) by setting the derivatives of \( L_2 \) in (2.6) with respect to \( \beta_\alpha \) equal to zero. The estimates of \( \beta_\alpha \) are

\[
\hat{\beta}_\alpha = \frac{E[ (Y_i - \bar{Y}_i) f_1(a_1^TX_i) ]}{E[f_1(a_1^TX_i)]^2}
\]

\[
= \sum_{i=1}^{k} w_i (Y_{ij} - \bar{Y}_i) \hat{f}_1(a_1^TX_j), \quad i=1, \ldots, q,
\]

since the weighted variance of \( \hat{f}_1(a_1^TX_j) \) is 1.

When the derivatives of \( L_2 \) in (2.6) with respect to \( a_1 \) is set to zero, the solution is not given in a closed form. Therefore, an iterative optimization method must be performed. PPR&C uses the Gauss–Newton method to obtain the estimate of \( a_1 \) to minimize \( L_2 \) in (2.6) with \( \beta_\alpha \) and \( f_1 \) fixed. Let

\[
g_{\alpha}(a_1) = Y_{ij} - \bar{Y}_i - \beta_\alpha f_1(a_1^TX_i).
\]

Then, the estimate of \( a_1 \) to minimize \( L_2 \) in (2.6) is obtained by iteration as follows.

\[
\hat{a}_1^{(i+1)} = \hat{a}_1^{(i)} + d_1^{(i)},
\]
where $d^{(0)}$ is the solution of

$$
\sum_{i=1}^{a} W_i \sum_{j=1}^{b} w_j \left( \frac{\partial g_{ij}(\hat{a}_1^{(0)})}{\partial a_1} \right)^T \left( \frac{\partial g_{ij}(\hat{a}_1^{(0)})}{\partial a_1} \right) d^{(0)}
$$

\begin{equation}
= \sum_{i=1}^{a} W_i \sum_{j=1}^{b} w_j \left( \frac{\partial g_{ij}(\hat{a}_1^{(0)})}{\partial a_1} \right)^T g_{ij}(\hat{a}_1^{(0)})
\end{equation}

and $\frac{\partial g_{ij}(a_1)}{\partial a_1}$ is the gradient vector of $g_{ij}(a_1)$. The equation of $d^{(0)}$ is solved using the conjugate gradient method. See Golub and Van Loan (1996) for details of the conjugate gradient method. In iteration, $\beta_i$ and $f_i$ are replaced with their estimates and prediction. The iteration continues until relative and absolute decrease of error sum of square in (2.6) are smaller than their criteria specified by users. In PPR&C, two convergence criteria are used. One is the convergence criterion of the relative error sum of square and the other is convergence criterion for the absolute error sum of square. The former value is 1E-5 and the latter value is 1E-10. The initial values are given internally.

Step 2. Obtaining initial values by increasing number of terms

For general $M$, PPR&C starts with one term. Then, residuals, $R_{ij} = Y_{ij} - \bar{Y}_i - \beta_1 f_1(a_i^T X_j)$ are obtained and the estimates for the second terms, $\beta_2$, $f_2$, $a_2$, are obtained by replacing $Y_{ij}$ with $R_{ij}$ and applying Step 1 to it. In this manner, the number of terms increase up to $M$. The increment of the number of terms may stop before it reaches $M$ if the absolute and relative decrease of error
sum of square is small. Let $M_L$ be the number of terms where increment of the number of terms stops. The estimates and predicted values for each term in this stage are used as initial values.

Step 3. Fitting the model with more than one term

Using the estimates from Step 2, in the model with $M_L$ terms, estimates for the $k$-th term are obtained by computing residuals excluding the $k$-th term, $R_{i(\cdot-k)} = Y_i - \overline{Y}_i - \sum_{j \neq k} \beta_i \beta_j (\alpha_j^T X_i)$. That is, by performing Step 1 for $R_{i(\cdot-k)}$ and the $k$-th term, the estimates are recomputed for $k = 1, \ldots, M_L$. This process is iterated until convergence. After convergence, criteria for selection of the number of terms given in "Criteria for term selection" are provided.

Step 4. Decreasing number of terms

After fitting the model in Step 3, PPR&C reduces the number of terms by one and redo Step 3. The term with the smallest estimate of $\beta_m$ in the absolute value is eliminated. The number of terms is reduced one by one up to the number of terms, $M_U$, where the estimation process stops. With numbers of terms from $M_L$ to $M_U$, the model is fitted and criteria for selection of number of terms are provided.
2.4 Criteria for term selection

PPR&C provides the various criteria for selection of the number of terms and model selection method. They are fraction of variance unexplained for prediction, misclassification error rate for classification, term importance, and relative importance of predictor variables.

2.4.1 Fraction of Variance Unexplained (FVU)

FVU is a ratio of the weighted sum of square of residuals to the corrected sum of square,

\[
FVU = \frac{\sum_{i=1}^{n} W_i \sum_{j=1}^{p} w_j (y_{ij} - \hat{y}_{ij})^2}{\sum_{i=1}^{n} W_i \sum_{j=1}^{p} w_j (y_{ij} - \bar{y}_i)^2},
\]

where \( y_{ij} \) is the \( i \)-th element of observation \( Y_j \), \( \hat{y}_{ij} \) is a predicted value of \( y_{ij} \), and \( \bar{y}_i \) is the weighted mean of the \( i \)-th attribute of the response observations. Smaller value of FVU indicates better fit. This criterion is used for prediction.

2.4.2 Misclassification Error Rate (MER)

Suppose that the response variable has \( q \) possible categories, \( c_1, \ldots, c_q \), and \( \pi_1, \ldots, \pi_q \) are priors corresponding to \( c_1, \ldots, c_q \). Also, suppose that \( \hat{y}_j \) is a classified category for the observed response \( y_j \) and \( L(c, d) \) is a loss of classification of category \( c \) into category \( d \). Then the misclassification error for classification is
\[ R = \sum_{j=1}^{n} w_j \frac{\pi^{(y_j)}}{s^{(y_j)}} K(y_j, \hat{y}_j), \]

where

\[ s^{(c)} = \sum_{k=1}^{n} w_k \delta(y_k, c), \]

\( \pi^{(c)} \) is a prior corresponding to category \( c \) and \( \delta \) is the Kronecker delta function

\[ \delta(a, b) = \begin{cases} 
1 & \text{if } a = b \\
0 & \text{otherwise} 
\end{cases}. \]

### 2.4.3 Term Importance

The term importance of the \( m \)-th term \( (1 \leq m \leq M) \) is

\[ I_m = \sum_{i=1}^{q} W_i |\beta_{im}|, \]

where \( \beta_{im} \) is the \( i \)-th element of \( \beta_m \). Dividing by the largest value (Note that the variance of all the smoothing function is one) normalizes the term importance.

### 2.4.4 Relative Importance of Predictor Variables

PPR&C provides the relative importance of each predictor variable for the final model. The relative importance of the \( k \)-th predictor variable is

\[ I_k = \sigma_k \sum_{i=1}^{p} W_i \left| \sum_{m=1}^{M} \beta_{im} \alpha_{km} f' \left( a_{m}^{T} X \right) \right| \]

\[ = \sigma_k \sum_{i=1}^{p} W_i \left| \sum_{j=1}^{q} w_j \sum_{m=1}^{M} \beta_{im} \alpha_{km} f' \left( a_{m}^{T} X_j \right) / \sum_{j=1}^{q} w_j \right|, \]

for \( 1 \leq k \leq p \). Here, the term \( \beta_{im} \alpha_{km} f' \left( a_{m}^{T} X_j \right) \) is a slope at \( X_{ki} \) where \( X_{kj} \) is the \( k \)-th predictor variable \( j \)-the observation. Hence,
the relative term importance of predictor variable is the weighted sum of a weighted average of the absolute slopes at data points multiplied by the standard deviation $\sigma_h$, the scale measure, of the predictor variable.

2.5 Model selection

We consider modeling with the optimal number of term $M$. The bias–variance decomposition of the mean squared error is a useful principle for understanding the effect of different values of $M$. Mean squared error (MSE) can be decomposed as the sum of two terms that distinguish the error due to estimation from finite samples (variance) and error due to mismatch between true value and predicted value (bias squared). That is,

$$MSE = (bias)^2 + variance.$$  

For "bias–variance trade–off", increasing the number of terms decreases bias at the expense of increasing the variance of the estimates. We find a model for minimum mean squared error, which has an optimal value for $M$. We can estimate MSE through resampling methods, cross–validation or bagging (Bootstrap aggregating, Brieman, 1994). Basic resampling method and cross–validation are detailed later.

Since the variance tends to increase more or less with increasing number of term $M$ while the $(bias)^2$ tends to drop rapidly for small $M$, leveling off to a slow decrease for larger $M$, a good estimate for the optimal $M$ value can usually be made by simply inspecting MSE vs. $M$ for various values of $M$. We choose the
optimal $M$ value in the region near the MSE tends to vary slowly as a function of $M$.

Resampling method has the basic approach, which is first to estimate a model using a portion of data and then using the remaining data to estimate the prediction risk for the estimated model. The first portion of the data (a data set of size $n_i$ used for model estimation) is called training set, and the remaining portion of the data of size $n_v = n - n_i$ is called a validation set. The various implementations of the resampling differ according to strategies to divide data. The simple approach is to split the data randomly into two portions, based on the assumption that the training set and the validation set chosen in this manner reflect the population. This is usually true for large size data, but the strategy has an obvious disadvantage that only part of all data are used for training. With smaller number of data, the specific method of splitting data starts to have an impact on the accuracy of an estimate average loss on the validation set. Cross-validation, which is to perform this estimate for all $\binom{n}{n_i}$ possible partitioning and average these estimates, is to make this estimate invariant to a particular partitioning of the data sets. From a computational point of view, it is usually impractical, except in the case of $n_v = 1$ (called leave-one-out cross-validation). An practical approach, known as $k$-fold cross-validation, is to divide the data into $k$ (randomly selected) disjoint sub-data sets of roughly equal size $n_v = n/k$. Typical choice for $k$ is between 5 and 10. The following is an algorithm of $k$-fold cross-validation given training
data $Z = [X, Y]$, where $X = [x_1, \ldots, x_n]'$ and $Y = [y_1, \ldots, y_n]'$ of sample size $n$, and assuming the squared error loss function,

1. Divide data $Z$ into $k$ disjoint set of roughly equal size, $Z_1, Z_2, \ldots, Z_k$.

2. For each $Z_i$ of size $n_i$,
   (a) Use the remaining data, $Z_i = \bigcup_{j \neq i} Z_j$ for estimator or training.
   (b) For estimated prediction $f_i(x)$ from (a), the sum of the empirical risk for $Z_i$ "left out" is
      \[ r_i = \frac{1}{n_i} \sum_{(x,y) \in Z_i} (f_i(x) - y)^2. \]

3. Compute the estimate for the prediction risk by averaging the empirical risk sums for $Z_1, Z_2, \ldots, Z_k$,
   \[ R(w) \approx R_{cv}(w) = \frac{1}{k} \sum_{i=1}^{k} r_i. \]

There is an empirical evidence that $k$-fold cross-validation gives better result than leave-one-out (Brieman and Spector, 1992).
2.6 Smoothing technique

There are various kinds of smoothers, but we consider two smoothers in PPR&C, which are super smoothing and smoothing with the Hermite functions. The super smoothing is nonparametric approach. The super smoothing does not assume a function form and, hence, is a powerful smoother for any kind of functions. But, the function of smoothing is not explicitly given and only an estimated value of the function at each observation is given. Also it takes a lot of time and needs large size of memory.

On the other hand, the smoother with the Hermite functions provides a polynomial for the smoothing function as well as estimated values. The smoothing function in the Hermite functions tends to be smoother than that in the super smoothing. The smoothing with the Hermite function is faster and need less memory space than the super smoothing. None of two smoother is better overall, but the Hermite function is a parametric approach and obtains a function form.

The macro for PPR&C uses the super smoothing technique proposed by Friedman (1984) and the smoothing with the Hermite functions that uses dynamic polynomial order selection using backward elimination with F-test.

2.6.1 The super smoothing – A Nonparametric smoother

The supersmoothing technique proposed by Friedman (1984) is generalization of the running line smoother. Comparing with the
simple regression, which uses whole data, the running line smoother generates a least squares line with $2k+1$ data. Assume data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are sorted in the magnitude of $x$ variable. In the running line smoother, prediction of $y_i$ at $x_i$ is obtained by least squares fitting for $(x_{i-k}, y_{i-k}), \ldots, (x_i, y_i), \ldots, (x_{i+k}, y_{i+k})$. For $i$ less than $k+1$ (or larger than $n-k$), least squares fit for $(x_1, y_1), \ldots, (x_{2k+1}, y_{2k+1})$ (or $(x_{n-2k}, y_{n-2k}), \ldots, (x_n, y_n)$) is performed and prediction is made for the least squares line. The running line smoother also called the local line smoother. Here $2k+1$, the number of data in least squares fitting, divided by number of whole observations are called span. For details, see Hastie and Tibshirani (1990).

The super smoothing technique is performed as follows.

1. Perform the running line smoother with spans 0.02, 0.2 and 0.5 and obtain the predicted value and the value of cross-validated absolute residual for each data point and for each span.

2. Perform the running line smoother for the cross-validated absolute residuals with span 0.2. Then we have a set of smoothed residuals for each span.

3. Choose the span with minimum of smoothed residual in Step 2 for each observation.

4. Perform the running line smoother for the spans chosen in Step 3 with span 0.2. Then we obtain a smoothed span for each observation.

5. For the smoothed span for each observation in Step 4, perform interpolation with predicted values and spans in Step 1.

6. Finally, perform the running line smoother for the predicted
values in Step 5 with span 0.02.

In the supersmoothing, the function form is not obtained and only the predicted value for each observation is generated. It may be uncomfortable for users, but the supersmoother covers more general forms in the model than other smoothing technique. In PPR&C, the derivative of the smoothed function is needed to obtain estimates. The estimate derivative at $x_i$ is

$$f'(x_i) = \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}}.$$

At end point $i=1$ (or $n$), $i-1$ (or $i+1$) is replaced with 1 (or $n$).

2.6.2 The Hermite polynomials – Parametric smoothing function

The Hermite polynomials are constructed recursively as follows.

$$H_0(z) = 1,$$
$$H_1(z) = 2z,$$
$$H_r(z) = 2(zH_{r-1}(z) - (r-1)H_{r-2}(z)), \quad r = 2, 3, \ldots.$$

The Hermite polynomials are orthogonal on $(-\infty, \infty)$ with respect to the square of the standard normal density function. That is, for $i \neq j$

$$\int_{-\infty}^{\infty} H_i(z) H_j(z) \phi^2(z) \, dz = 0,$$

where

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}.$$

Since
\[ \int_{-\infty}^{\infty} H_r^2(z) \phi(z) \, dz = r! \pi^{-1/2} 2^{r-1}, \]

we can construct the orthogonal Hermite function

\[ h_r(z) = (r!)^{-1/2} \pi^{1/4} 2^{-(r-1)/2} H_r(z) \phi(z), \]

so that

\[ \int_{-\infty}^{\infty} h_i(z) h_j(z) \, dz = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}. \]

Then, usually the smoothing function based on the Hermite polynomial of order \( R \) is defined as

\[ f(z) = \sum_{r=0}^{R} c_r h_r(z) \]

and its derivative is

\[ f'(z) = \sum_{r=1}^{R} c_r h'_r(z) \\
= \sum_{r=1}^{R} c_r \left( \sqrt{2r} h_{r-1}(z) - z h_r(z) \right). \]

The coefficients \( c_r \) is estimated by the least squares method.

In Equation (2.1), all smoothing functions have the same order \( R \). But, the high order of these functions may not contribute to a specific term. That is, overfitting problem can exist. To avoid this problem, in the macro for PPR&C with the Hermite functions, each order of Hermite polynomial is determined using backward elimination with F-test. Then the smoothing function is defined as

\[ f_m(z) = \sum_{r=0}^{R} c_r h_r(z) , \quad m = 1, \ldots, M, \]

and, its derivative is

\[ f'_m(z) = \sum_{r=1}^{R} c_r h'_r(z) \\
= \sum_{r=1}^{R} c_r \left( \sqrt{2r} h_{r-1}(z) - z h_r(z) \right) , \quad m = 1, \ldots, M. \]
Here $m$ denotes a specific term of the model.

In comparison of the super smoothing method, smoothing with the Hermite functions gives a parametric model and provides smooth interpolation instead of piecewise interpolation. Also, an accurate derivative is obtained in a smoothing function with the Hermite functions.

3. SIMULATION AND COMPARISON

We compare PPR&C in the macro in SAS with PPR&C for the super smoothing technique proposed by Friedman (1984) and PPR&C for the smoothing with Hermite functions proposed by Hwang et al. (1994) in regression problem. These comparisons were made on the learning accuracy which is the fraction of variance unexplained (FVU) of independent test data. And, in classification, we compare the macro for PPR&C in SAS with PPR&C for super smoothing technique for classification and linear discriminant analysis in S-Plus.

3.1 Comparison with nonlinear regression function

In regression, we investigate performance in five nonlinear functions $f(x_1, x_2): [0, 1]^2 \rightarrow R$, which are used in Hwang et al. (1994). These functions are as follows.
1) Simple function
\[ f(x_1, x_2) = 10.391((x_1 - 0.4) \cdot (x_2 - 0.6) + 0.36) \]

2) Radial function
\[ f(x_1, x_2) = 24.234(r^2(0.75 - r^2)), \quad r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \]

3) Harmonic function
\[ f(x_1, x_2) = 42.659(0.1 + x_1^4(0.05 + x_1^2 \cdot x_2^2 + 5 x_2^4)) \]
with \( x_1^* = x_1 - 0.5, \quad x_2^* = x_2 - 0.5 \)

4) Additive function
\[ f(x_1, x_2) = 1.3356(1.5(1 - x_1) + e^{2x_1-1}\sin(3\pi(x_1 - 0.6)^2)) + e^{3(x_2 - 0.5)}\sin(4\pi(x_2 - 0.9)^2)) \]

5) Complicated function
\[ f(x_1, x_2) = 1.9(1.35 + e^{-x_1}\sin(13(x_1 - 0.6)^2) \times e^{-x_1}\sin(7x_2)) \]

Shape of these functions are illustrated in Figure 1.

We generate independent two data sets for each of five functions. The first data set is model fitting data set, training data set. Two independent predictor variables \((x_{1j}, x_{2j})\) that are generated from the uniform distribution \(U([0,1]^2)\) and the function values are generated with independent and identically distributed Gaussian noises.

\[ y_j = f(x_{1j}, x_{2j}) + 0.25\varepsilon_j. \]

Here \( \varepsilon_j \sim N(0,1) \). 225 observation are generated for training.

Another data set is for the performance assessment by comparing the fitted values with the true value, this data set is called a test data set. The test data set is generated on a regularly spaced grid on \([0,1]^2\) and the size of test data set is 10000, i.e. with
\[ x_{li} := \frac{(2l-1)}{200} , \quad \text{for } l = 1, \cdots, 100, \quad i = 1, 2. \]

Figure 1. (a) simple function, (b) radial function, (c) harmonic function, (d) additive function, (e) complicated function.
The training data sets were generated 50 times. For each training set, the model was fitted and assessed with the test data set. The same test data set was used for 50 training data set.

In these simulation, we used `ppreg()` in S-Plus to conduct PPR&C with super smoothing technique proposed by Friedman. Also, we used Hwang et al's C program for conducting PPR&C with the Hermite functions. We assess each algorithm with 3 and 5 terms. For the models with 3 terms, we increase the number of terms up to 5, and then, decrease the number of terms to 3. We set 7 to be the maximum number of terms for the model with 5 terms. Mean and standard deviation for FVU of 50 independent training data in Table 1 to 5 is used for comparison.

<table>
<thead>
<tr>
<th>simple function</th>
<th>3 terms FVU(S.D)</th>
<th>5 terms FVU(S.D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friedman’s program</td>
<td>0.007137 (0.012613)</td>
<td>0.005890 (0.001773)</td>
</tr>
<tr>
<td>Hwang’s program</td>
<td>0.011707 (0.008657)</td>
<td>0.019140 (0.028266)</td>
</tr>
<tr>
<td>(Order=7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro</td>
<td>0.151762 (0.988262)</td>
<td>0.022488 (0.039272)</td>
</tr>
<tr>
<td>supersmooother</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro</td>
<td>0.018986 (0.017709)</td>
<td>0.055560 (0.071443)</td>
</tr>
<tr>
<td>Hermite functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table 1 to 5, Friedman’s super smoothing more accurate for the simple, radial, and additive function. Also Friedman’s super smoothing gives better fitting for the model with more terms. In the harmonic and complicated function, method with
the Hermite functions is more accurate with 5 terms than with 3 terms. Especially, the macro with the Hermite functions is most accurate and stable for the model with 5 terms in the complicated function.

Table 2. The average FVU of the test data set for radial function

<table>
<thead>
<tr>
<th>radial function</th>
<th>3 terms FVU(S.D)</th>
<th>5 terms FVU(S.D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friedman’s program</td>
<td>0.015082(0.006261)</td>
<td>0.008651(0.002595)</td>
</tr>
<tr>
<td>Hwang’s program</td>
<td>0.028767(0.015994)</td>
<td>0.018220(0.029168)</td>
</tr>
<tr>
<td>(Order=7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro</td>
<td>0.039459(0.015287)</td>
<td>0.023813(0.008029)</td>
</tr>
<tr>
<td>supersmoother</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro</td>
<td>0.027361(0.093672)</td>
<td>0.037545(0.094102)</td>
</tr>
<tr>
<td>Hermite functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. The average FVU of the test data set for harmonic function

<table>
<thead>
<tr>
<th>harmonic function</th>
<th>3 terms FVU(S.D)</th>
<th>5 terms FVU(S.D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friedman’s program</td>
<td>0.085837(0.025316)</td>
<td>0.058937(0.018378)</td>
</tr>
<tr>
<td>Hwang’s program</td>
<td>0.106434(0.030988)</td>
<td>0.019383(0.008809)</td>
</tr>
<tr>
<td>(Order=7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro</td>
<td>0.258134(0.072476)</td>
<td>0.191715(0.063112)</td>
</tr>
<tr>
<td>supersmoother</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro</td>
<td>0.126477(0.050835)</td>
<td>0.037131(0.028032)</td>
</tr>
<tr>
<td>Hermite functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4. The average FVU of the test data set for additive function

<table>
<thead>
<tr>
<th>additive function</th>
<th>3 terms FVU(S.D)</th>
<th>5 terms FVU(S.D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friedman's program</td>
<td>0.016895(0.032121)</td>
<td>0.007672(0.002297)</td>
</tr>
<tr>
<td>Hwang’s program (Order=7)</td>
<td>0.024549(0.015856)</td>
<td>0.032621(0.021618)</td>
</tr>
<tr>
<td>macro supersmoother</td>
<td>0.018466(0.029768)</td>
<td>0.022246(0.020430)</td>
</tr>
<tr>
<td>macro Hermite functions</td>
<td>0.028350(0.021299)</td>
<td>0.044645(0.057123)</td>
</tr>
</tbody>
</table>

Table 5. The average FVU of the test data set for complicated function

<table>
<thead>
<tr>
<th>complicated function</th>
<th>3 terms FVU(S.D)</th>
<th>5 terms FVU(S.D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friedman's program</td>
<td>0.102228(0.051759)</td>
<td>0.040119(0.021892)</td>
</tr>
<tr>
<td>Hwang’s program (Order=7)</td>
<td>0.120885(0.063521)</td>
<td>0.058062(0.026960)</td>
</tr>
<tr>
<td>macro supersmoother</td>
<td>0.120383(0.063936)</td>
<td>0.063269(0.022143)</td>
</tr>
<tr>
<td>macro Hermite functions</td>
<td>0.105683(0.054944)</td>
<td>0.029846(0.014096)</td>
</tr>
</tbody>
</table>

For the number of terms, as the number of terms increase, FVU tends to decrease in supersmoothing. But method of the Hermite functions does not have this tendency. Also, for the model with 3 or 5 terms, in the macro for the Hermite functions, the number of terms does not increase the maximum number of terms. In some training data sets, for the model with 5 terms, macro for the
Hermite functions could not fit with 5 terms because the number of terms does not increase up to 5. So, this macro has less number of terms and order of Hermite functions.

3.2 Comparison with Classification data

We investigate these performances for German credit data, which was used in StatLog project. German credit data, provided by Professor Dr. Hans Hofmann at University Hamburg, contains some categorical and symbolic attributes. The attributes of the original dataset include the following.

- status of existing current account, marital status and sex,
- duration of current account, credit history,
- reason for loan request, credit amount,
- savings account/bonds, length of employment,
- installment rate in percentage of disposable income,
- length of time at present residence, age and job

For algorithms that require numerical attributes, we added several indicator variables for categorical attributes. This preprocessed data set had 51 numerical attributes, for predictor, and a categorical attribute for response variable with 2 possible categories. Response variable of German credit data is to show that credit is good or bad for each customer. And we called for concerned category of the response variable as "target class". Bad
category is the target class in German credit data. 10-fold cross-validation was used, and all algorithms used this preprocessed data.

We used classification routine using ppreg() in S-Plus to conduct Friedman’s PPR&C with super smoothing, lda() in S-Plus for conducting linear discriminant analysis, and the macro for PPR&C with Hermite funtions.

For the model selection, we set up the number of terms through 10-fold cross-validation. That is, we obtained training and validation average misclassification error rate for 10-fold cross-validation for the models with terms varying 1 to 20. Then the number of terms with minimum misclassification error rate in 10-fold cross-validation is the number of terms for the final model.

Figure 2. Average misclassification error rate for 10-fold cross-validation
From Figure 2, we choose 8 as the number of terms and obtain parameter estimates for the model with 8 terms.

In addition to average misclassification error rate, obtained by averaged over all cycles in 10-fold cross-validation for the selected model, sensitivity (with specificity) are used as a measure of performance. These values are concerned with target class and other class. For example, the fitted model classifies a credit. Given that the credit is a bad, the probability that the classifier predicts "bad" is called the sensitivity. Given that the credit is a good, the probability the classifier predicts "good" is called the specificity. That is, sensitivity is the ability to correctly identify whose credit is bad. Specificity is the ability to correctly identify whose credit is good. Ideally, a test should have 100% sensitivity and 100% specificity. In other words, the classifier always correctly identifies the credit.

When contingency table for predicted and actual category is given below.

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Target</td>
</tr>
<tr>
<td>Target</td>
<td>a</td>
</tr>
<tr>
<td>Another</td>
<td>c</td>
</tr>
</tbody>
</table>

Sensitivity is computed as following.

\[ Sensitivity = \frac{a}{a + c} \times 100 \text{ (\%)} \]

When the "false another" is a small number relative to the "true target", sensitivity approaches 100%. Also, specificity is
$Specificity = \frac{d}{b+d} \times 100 \, (\%),$

When the "false target" is a small number relative to the "true another", specificity approaches 100%.

Table 6. The average misclassification error rate
and Confusion Matrix for German credit data

<table>
<thead>
<tr>
<th></th>
<th>LDA</th>
<th>Friedman's PPR&amp;C with super smoothing</th>
<th>The macro for PPR&amp;C with Hermite</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Confusion Matrix</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Predict</strong></td>
<td>Actual</td>
<td>Actual</td>
<td>Actual</td>
</tr>
<tr>
<td>good</td>
<td>573</td>
<td>554</td>
<td>600</td>
</tr>
<tr>
<td>bad</td>
<td>127</td>
<td>146</td>
<td>100</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1000</td>
<td>700</td>
<td>700</td>
</tr>
</tbody>
</table>

| **Average of MER** | 0.348          | 0.302                                 | 0.241                           |
| **S.D of MER**     | 0.100          | 0.038                                 | 0.044                           |
| **Sensitivity**    | 26.33          | 48.00                                 | 53.00                           |
| **Specificity**    | 81.86          | 79.14                                 | 85.71                           |

Results for German credit data are given in Table 6. For German credit data, the macro for PPR&C is a quite good performance for misclassification error rate and sensitivity, and specificity. Also, this is more stable result by that standard deviation for misclassification error rate is less than other programs. Especially, sensitivity that is ability of detecting bad credit class is twice of LDA.
4. CONCLUDING REMARKS

We examined algorithms for projection pursuit regression and classification, and developed the macro program for this method in SAS. Also, we evaluated the performance for prediction and classification. The macro for PPR&C is very useful for high dimensional data analysis. And that is more accurate prediction and classification for complicated data.

For learning speed, the macro with supersmoothing takes a lot of time. But, the macro with the Hermite functions takes much less time. In addition, this offers automatic order determination.

The macro for PPR&C does not have the procedure of automatic term determination. We will study further for this procedure. Also, any other smoothing method can be applied and the macro for PPR&C can be adapted to real high dimensional data.
REFERENCE


국문 초록

사영추적회귀를 이용한 예측과 분류방법은 1981년 Friedman과 Stuetzle가 제안한 것으로 예측과 분류의 정확도를 높일 수 있는 새로운 방법으로 평가 받고있다. 이 방법은 두 가지의 큰 특징을 가지고 있는데, 첫 번째는 다차원 자료에서 발생되는 "다차원의 저주", 즉, 다차원 자료에서 충분한 표본 자료를 얻기 위해 발생하는 문제를 설명변수의 사영 함수를 사용함으로써 해결 할 수 있다. 또한 모형에서 항의 수가 충분히 많을 경우 연속적인 모든 함수를 설명해 줄 수 있는 범용 근사법의 특징 을 가지고 있다. 다른 특징으로, 모형의 크기를 결정해 줄 수 있는 여러 가지 선택 기준을 제시해 안정적인 모형 선택을 할 수 있도록 해준다.


결론 부분에서는 비선형 회귀 함수와 분류 자료를 사용한 실험을 통해 사영추적 회귀를 이용한 예측과 분류 SAS 매크로가 데이터마이닝 모형 화에 사용할 수 있는 새로운 방법이며 적응성 있고 유용한 방법임을 보였다.
감사의 글

본 논문이 완성되기까지 저성하고 세심하게 지도해 주시고, 항상 따뜻한 격려와 충고로 제 학업의 길을 바로 잡아 주신 박현진 교수님께 깊은 감사 드립니다. 또한 학부 시절부터 통계학과를 다니는 한 학생에서 통계학을 제대로 이해하고 배울 수 있도록 도와주신 최지훈 교수님, 구자홍 교수님, 자식들처럼 학생들에게 관심과 애정을 주신 진홍석 교수님, 이론뿐 아니라 실무적인 학문의 중요성을 일깨워 주신 이재준 교수님, 바쁘신 가운데도 꼭꼭 논문 심사를 해 주신 김진경 교수님, 1학년 때부터 많은 도움을 주신 황진수 교수님, 부족한 지식을 배우고자 힘전 많은 질문에 열심히 답해 주시던 박진호 교수님께 깊은 감사 드립니다. 그리 고 대학원 생활동안 지도 교수님과 터들여 통계학의 새로운 내용과 넓은 학문의 폭을 알도록 해주신 한림대학교 구자홍 교수님과 한국 외국어대 학교 최대우 교수님께 깊은 감사 드립니다.

학부에서 대학원 생활동안 많은 힘과 도움을 주신 김승환 선배님, 위로와 격정으로 보살피주신 전상표 선배님과 성혜형과 성용이형님께 감사의 말씀을 전하고 싶습니다. 함께 공부하며 많은 도움을 준 김현선생님, 이주영 선배님, 인석이형, 대호형, 천우형, 영민이형, 유나선배, 영숙선배와 해청누나에게도 고마운 마음을 전합니다. 특히 논문을 쓰는 동안 바쁘신 가운데도 많은 도움과 격정을 해 주신 중선이형, 지훈이형, 동혁이형, 찬범이형, 홍국이형, 후배 장순, 방실이에게 진심으로 고마운 마음을 전합니다.

고등학교 시절부터 멀리서 격정해주며 위로를 해준 친구 동주, 여러 가지 도움을 준 원미, 인기형, 경우형, 창인을 비롯한 과 동기 친구들에게도 고마운 마음을 전하고 싶습니다.
특히 배움의 열정과 올바른 생활을 살도록 이끌어 주시고 사랑과 신앙으로 키워주시며 학업을 계속할 수 있도록 도와주신 아버지, 어머니께 감사드리며 항상 앞으로 뒤인 여러 분에게 존경스러운 위로로 협이 되려 희생 성품, 미신에게도 고마움을 전달하고 함께 기쁨을 나누고 싶습니다.

마지막으로 이 한편의 논문으로 제 공부가 끝났다고 생각하지 않으며 많은 도움과 격려를 주신 분들에게 보답하는 마음으로 열심히 배움의 자세를 잃지 않고 살아가도록 노력하겠습니다.

1999년 겨울
김성윤