國 立 交 通 大 學

統計學研究所

碩 士 論 文

廣義線性模型使用懲罰概似函數之模型選取

Model Selection for Generalized Linear Models Using Penalized Likelihood

研 究 生 ： 宋佩芸

指導教授 ： 黃信誠 教授

中華民國一百年六月

廣義線性模型使用懲罰概似函數之模型選取

Model Selection for Generalized Linear Models Using Penalized Likelihood

研　究　生　：宋佩芸　　Student：Pei-Yun Sung

指 導 教 授：黃信誠　　Advisor：Dr. Hsin-Cheng Huang

國 立 交 通 大 學

統計學研究所

碩 士 論 文

A Thesis
Submitted to Institute of Statistics
College of Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master
in
Statistics
June 2011
Hsinchu, Taiwan, Republic of China

中華民國一百年六月

廣義線性模型使用懲罰概似函數之模型選取

研究生：宋佩芸　　指導教授：黃信誠 教授

國立交通大學統計學研究所

摘要

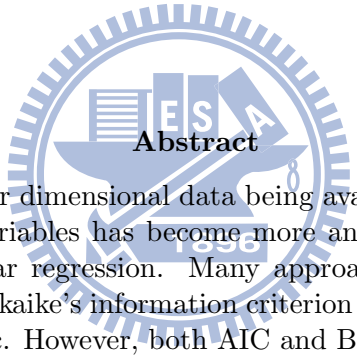
隨著越來越多高維度資料需要被了解，在一般線性模型迴歸和廣益線性模型迴歸問題下找出資料中的重要變數已成為越來越重要的問題。許多方法已在文獻中被提出，包括 Akaike's information criterion (AIC)、Bayesian information criterion (BIC)、 Lasso 等等，但AIC和BIC在變數量很大時都難以實行，因此通常是透過使用一些逐步的選取程序。在這篇論文中，我們延伸 Shen *et al.* (2010)的方法，將有懲罰項的最小平方法和L1懲罰的變形應用在廣義線性迴歸上，這樣的方法在變數量大時可以逼近AIC和BIC。我們介紹一個有效的演算方法，其利用Difference convex programming (DCP), iteratively reweighted penalized least squares以及coordinate descent演算法解決問題。在文章中，我們探討此方法具有之性質並提供數值模擬呈現我們方法的優勢。最後，將該方法使用於分析體重過輕嬰兒的數據，並指出影響出生嬰兒體重過輕之因素。

Model Selection for Generalized Linear Models Using Penalized Likelihood

Student: Pei-Yun Sung Advisor: Dr. Hsin-Cheng Huang

Institute of Statistics

National Chiao Tung University



Abstract

With higher and higher dimensional data being available, identifying important variables among many variables has become more and more important in regression and generalized linear regression. Many approaches have been proposed in the literature, including Akaike's information criterion (AIC), Bayesian information criterion (BIC), Lasso, etc. However, both AIC and BIC are difficult to implement when the number of variables is large, and hence are usually done using some stepwise procedures. In this thesis, we extend an approach of Shen *et al.* (2010), who considered a penalized least squares method with a truncated L_1 penalty for linear regression, to generalized linear regression. This approach enables us to well approximate AIC and BIC even when the number of variables is large. A computational efficient algorithm is introduced, which utilizes difference convex programming, iteratively reweighted penalized least squares, and the coordinate descent algorithm. Some oracle property of the proposed method is established, and some numerical examples are provided to demonstrate the superiority of the proposed method over AIC and BIC. Finally, the proposed method is applied to analyze a low birth weight dataset, in which we identify important variables associated with a low birth weight baby.

Key words: Akaike information criterion, Bayesian information criterion, coordinate descent, difference convex programming, Lasso, L_0 penalty, oracle property.

致謝

身為交大統計所碩士生兩年，我在學校裡學到了很多事，不管是老師們在課堂上教的專業知識，還有課間閒聊的人生道理，我覺得我真的獲益良多。除了老師們的指導之外，也結交很多知心的朋友，讓我這兩年的生活，充實且快樂，我真的很滿足。

此篇論文的完成，我要感謝我的指導教授黃信誠老師，每次的討論都指點我許多，作研究遇到困難時，也都幫助我一起解決，讓我能夠順利完成這篇論文。

在論文口試時，感謝清大統計所許文郁老師、鄭少為老師以及本所的王維菁老師，對於很多文章內容的細節，不吝提出指導與修正，並提供很多寶貴的意見和建議，協助我完成論文的修正。

最後要感謝的是我的家人及朋友，在我面對壓力的時候陪我聊天，幫助我排解困難，讓我順利完成階段性的人生目標。

宋 佩 芸　　謹誌于

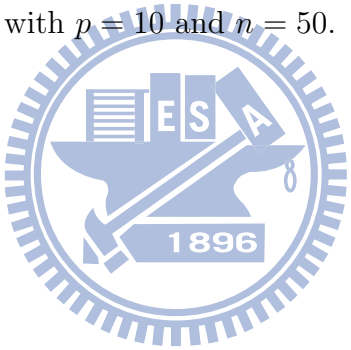
國立交通大學統計研究所

中 華 民 國 一百年六月

Contents

List of Tables

List of Figures



1 Introduction

Suppose that we observe data, $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$, where y_i and $\boldsymbol{x}_i = (1, x_{i1}, \ldots, x_{ip})'$ represent the response and a $(p+1)$-dimensional vector of explanatory variables, respectively. If $y_1, \ldots, y_n$ are continuous variables that are roughly normally distributed, then the linear regression model is usually applied:

$$y_i = \boldsymbol{x}_i'\boldsymbol{\beta} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2); \quad i = 1, \ldots, n,$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)'$ is a vector of regression parameters. In some situations, $y_1, \ldots, y_n$ are deviated from normal distributions, or take only discrete values, the usual linear regression model is no longer valid. Generalized linear models provide a flexible framework for these types of data.

For a generalized linear model, we assume that y_i has the following probability density function (pdf):

$$f(y_i; \theta_i) = \exp\left\{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)\right\}, \tag{1.1}$$

for some known functions $a(\cdot)$, $b(\cdot)$, and $c(\cdot)$, where ϕ is a dispersion parameter. If ϕ is known, (1.1) forms an exponential family model with canonical parameter θ_i. For example, the pdf of a normal distribution with mean μ_i and variance σ_i^2 can be written as the form of (1.1) with $\theta_i = \mu_i$, $\phi = \sigma_i^2$, $a(\phi) = \phi$, $b(\theta_i) = \theta_i^2/2$, and $c(y_i, \phi) = -(y_i^2/\sigma_i^2 + \log(2\pi\sigma_i^2))/2$. In general, the mean and the variance of y_i can be expressed as:

$$\mu_i = \mathrm{E}(y_i) = b'(\theta_i),$$
$$\sigma_i^2 = \mathrm{var}(y_i) = b''(\theta_i)a(\phi).$$

In a generalized linear model, there are two parts involved: the random component (response), y_i, and the systematic component (linear predictor), $\eta_i = \boldsymbol{x}_i'\boldsymbol{\beta}$, for $i = 1, \ldots, n$. The two parts are related through a link function $g(\cdot)$ by the followings:

$$\mu_i = E(y_i|\boldsymbol{x}_i), \tag{1.2}$$
$$g(\mu_i) = g(b'(\theta_i)) = \boldsymbol{x}_i'\boldsymbol{\beta}. \tag{1.3}$$

A link function $g(\cdot)$ is called the canonical link if it satisfies $g(\mu_i) = \theta_i = \boldsymbol{x}_i'\boldsymbol{\beta}$. For examples, the identity function is the canonical link for normal regression, the log function is that for Poisson regression, and the logit function is that for logistic regression.

The log-likelihood function under the canonical link can be written as:

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left\{ \frac{y_i \boldsymbol{x}_i'\boldsymbol{\beta} - b(\boldsymbol{x}_i'\boldsymbol{\beta})}{a(\phi)} + c(y_i, \phi) \right\}, \tag{1.4}$$

with its first and second derivatives given by

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\beta}} \ell(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \boldsymbol{x}_i (y_i - \mu_i(\boldsymbol{\beta}))/a(\phi), \\ \frac{\partial^2}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}'} \ell(\boldsymbol{\beta}) &= -\sum_{i=1}^{n} \sigma_i^2(\boldsymbol{\beta}) \boldsymbol{x}_i \boldsymbol{x}_i'/a(\phi). \end{aligned} \tag{1.5}$$

The parameter vector $\boldsymbol{\beta}$ is usually estimated by maximum likelihood (ML). The ML estimate of $\boldsymbol{\beta}$, given by $\arg\max_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta})$, can be obtained by solving the score equations, $\partial \ell / \partial \boldsymbol{\beta} = \mathbf{0}$ with $\partial \ell / \partial \boldsymbol{\beta}$ being the score function.

The rest of this thesis is organized as follows. In Chapter 2, we introduce some commonly used model selection methods for generalized linear models. In Chapter 3, we introduce a penalized likelihood method with a truncated L_1 (TL) penalty, which similar to Lasso (Tibshirani 1996), allows simultaneous variable selection and parameter estimation. But unlike Lasso, which tends to produce bias estimates, the proposed method shrinks only small coefficients to zero but not large coefficients. In Chapter 4, we give two examples showing some more details about the proposed method. The oracle property is established In Chapter 5, and some numerical examples are provided in Chapter 6. Some brief discussion is given in Chapter 7. Finally, the appendix contains R code used in the numerical examples.

2 Variable Selection

Consider the log-likelihood function given in (1.4) with p variables. Then there are 2^p candidate models to choose. Basically, a large model tends to produce low bias but high variance, and *vice versa*. Some commonly used methods are given in the following subsections.

2.1 Sequential Hypothesis Testing

One approach to identify important variables is to perform a sequence of tests using a forward selection procedure, backward selection procedure, or a stepwise procedure

that mixes between the two. Forward selection starts from the null model with only the intercept term and adds one variable at a time until some criterion is met. At each step, the variable with the smallest p-value is added until no p-value is smaller than a pre-specified threshold. Backward selection performs similarly but in the opposite direction by starting from the full model. At each step, the variable corresponding to the largest p-value is removed until all variables have p-values smaller than a pre-specified threshold. These procedures involve multiple tests, which make theoretical justification of this method difficult.

2.2 Information Criteria

Another approach for variable selection is to apply an information criterion, which selects the model by minimizing a cost function given in the form of

$$-\frac{1}{n}\ell(\boldsymbol{\beta}) + J(\boldsymbol{\beta}), \tag{2.1}$$

where $J(\boldsymbol{\beta})$ is a penalty term, which is typically larger for a larger model. By choosing a proper penalty term that suitably controls between goodness-of-fit and model parsimony, a good estimate of $\boldsymbol{\beta}$ balancing between bias and variance can be obtained. For example, the generalized information criterion (GIC; Nishii 1984) has the following L_0 penalty:

$$-\frac{1}{n}\ell(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} I(|\beta_j| \neq 0), \tag{2.2}$$

which penalizes the number of parameters. It includes Akaike's information criteria (AIC; Akaike 1974) with $\lambda = 1/n$ and Bayesian information criteria (BIC; Schwarz 1978) with $\lambda = \log(n)/(2n)$ as special cases. Although asymptotic justification of GIC can be found in Shao (1997), it is difficult to minimize GIC directly, because GIC is nonconvex and discontinuous, which generally requires computing GIC values for 2^p candidate models separately, and hence is computationally infeasible when p is large. Consequently, a stepwise procedure similar to those described in Section 2.1 is often applied, resulting in less satisfactory estimates.

2.3 Lasso

Instead of considering the penalized likelihood with an L_0 penalty as in (2.2), a penalized likelihood with an L_1 penalty was proposed by Tibshirani (1996), leading to a criterion,

called the least absolute shrinkage and selection operator (Lasso), which is very popular in recent years. The Lasso estimate minimizes

$$-\frac{1}{n}\ell(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{2.3}$$

where λ is a tuning parameter controlling the shrinkage of $\boldsymbol{\beta}$ toward zeros with a large λ corresponding to a higher degree of shrinkage. Different from the ridge regression, which substitutes an L_2 penalty $\sum_{j=1}^{p} |\beta_j|^2$ for $\sum_{j=1}^{p} |\beta_j|$ in (2.3), Lasso not only does shrinkage, but also estimates some $\boldsymbol{\beta}$ as exactly zeros. Consequently, it provides parameter estimation and model selection simultaneously. Originally, Lasso was solved using quadratic programming (Tibshirani 1996), which is not particularly fast. It was not until the introduction of the least angle regression (LARS) algorithm (Efron *et al.* 2004) that the Lasso becomes very popular, because the entire Lasso solutions along a path of λ can be solved in the order equivalent to solving the ordinary least squares estimate based on the full model. Interestingly, the LARS algorithm is equivalent to a homotopy algorithm introduced earlier by Osborne *et al.* (2000) for solving Lasso. Recently, Friedman *et al.* (2007) demonstrate that the simple coordinate descent algorithm can do even faster than the LARS algorithm.

However, the Lasso is known to produce a bias estimate of $\boldsymbol{\beta}$, and hence can only achieve selection consistency under some restricted conditions (Zhao and Yu 2006). Some nonconvex penalties have been proposed to remedy the bias problem of Lasso in the context of linear regression, including the SCAD penalty of Fan and Li (2001), the MCP penalty of Zhang, and the TL penalty of Shen and Huang (2010) and Shen *et al.* (2010).

3 The Proposed Method

Consider the generalized linear model given by (1.4). Following the TL penalty introduced by Shen and Huang (2010) and Shen *et al.* (2010), we propose to estimate $\boldsymbol{\beta}$ by minimizing the following penalized log-likelihood function:

$$S(\boldsymbol{\beta}) = -\frac{1}{n}\ell(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \min\left\{\frac{|\beta_j|}{\tau}, 1\right\}, \tag{3.1}$$

where $\lambda \geq 0$ is a tuning parameter controlling the degree of shrinkage with a larger λ corresponding to a higher degree of shrinkage, and $\tau > 0$ is a thresholding parameter.

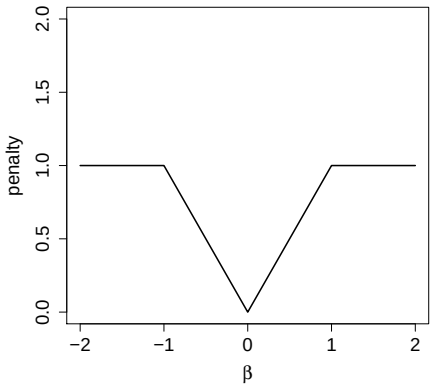


Figure 1: Truncated L_1 penalty with $\tau = 1$.

Since $\min\{|\beta|/\tau, 1\} \to I(|\beta| \neq 0)$ as $\tau \to 0$, the L_0 penalized log-likelihood of (2.2) can be well approximated by $S(\boldsymbol{\beta})$ if τ is sufficiently small. By choosing a small τ, we obtain approximated AIC and BIC with $\lambda = 1/n$ and $\log(n)/(2n)$, respectively. On the other hand, when τ is large, $S(\boldsymbol{\beta})$ can be rewritten as $-\frac{1}{n}\ell(\boldsymbol{\beta}) + \frac{\lambda}{\tau}\sum_{j=1}^{p}|\beta_j|$, leading to the usual Lasso with tuning parameter λ/τ. The penalty function, $\min\{|\beta|/\tau, 1\}$, with $\tau = 1$ is shown in Figure 1.

Although from (1.5), the loglikelihood function $\ell(\boldsymbol{\beta})$ is concave, the objective function $S(\boldsymbol{\beta})$ of (3.1) is not convex unless $\tau = \infty$, which in general is difficult to minimize directly. We propose to solve the nonconvex optimization problem of (3.1) using difference convex programming (DCP) (An and Tau 1997), iteratively reweighted penalized least squares and the coordinate descent algorithm, which will be introduced in the next subsection.

3.1 Computation Algorithm

Since $S(\cdot)$ is not a convex function, we adopt the approach of Shen and Huang (2010) and Shen *et al.* (2010) by finding a minimizer of (3.1) through DCP, which decomposes $S(\cdot)$ into a difference of two convex functions:

$$S(\boldsymbol{\beta}) = S_1(\boldsymbol{\beta}) - S_2(\boldsymbol{\beta}), \tag{3.2}$$

where $S_1(\boldsymbol{\beta}) = -\frac{1}{n}\ell(\boldsymbol{\beta}) + \lambda\sum_{j=1}^{p}|\beta_j|/\tau$ and $S_2(\boldsymbol{\beta}) = \lambda\sum_{j=1}^{p}\max\{|\beta_j|/\tau - 1, 0\}$.

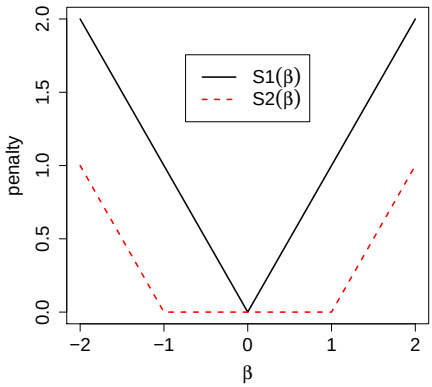


Figure 2: Separate the penalty term of the objective function into two convex functions with $\tau = 1$.

Figure 2 shows the decomposition of $\min\{|\beta|/\tau, 1\}$ into the difference of two convex functions, $|\beta|/\tau - \max(|\beta| - \tau, 0)/\tau$. The main idea of DCP is to approximate a minimum of $S(\boldsymbol{\beta})$ iteratively by the minimizers from a sequence convex functions. Specifically, we successively replace $S_2(\boldsymbol{\beta})$ at the m-th step iteration with its first order approximation:

$$\begin{aligned} S_2^{(m)}(\boldsymbol{\beta}) &= S_2\big(|\hat{\boldsymbol{\beta}}^{(m-1)}|\big) + \big(|\boldsymbol{\beta}| - |\hat{\boldsymbol{\beta}}^{(m-1)}|\big)^T \nabla S_2\big(|\hat{\boldsymbol{\beta}}^{(m-1)}|\big), \\ &= \lambda \sum_{j=1}^{p} \Big(\frac{|\beta_j|}{\tau} - 1\Big) I\big(|\hat{\beta}_j^{(m-1)}| > \tau\big), \end{aligned}$$

evaluated at the solution $\hat{\boldsymbol{\beta}}^{(m-1)} = \big(\hat{\beta}_1^{(m-1)}, \ldots, \hat{\beta}_p^{(m-1)}\big)'$ obtained in the previous step, where $\nabla S_2(\boldsymbol{\beta}) = \lambda \sum_{j=1}^{p} \frac{1}{\tau} I(|\beta_j| > \tau)\text{sign}(\beta_j)$, $|\boldsymbol{\beta}| \equiv (|\beta_1|, \ldots, |\beta_p|)'$, and $\big|\hat{\boldsymbol{\beta}}_{\lambda,\tau}^{(m-1)}\big|$ is defined similarly. Thus we obtain a sequence of upper approximation functions given by:

$$\begin{aligned} S^{(m)}(\boldsymbol{\beta}) &= S_1(\boldsymbol{\beta}) - S_2^{(m)}(\boldsymbol{\beta}) \\ &= -\frac{1}{n}\ell(\boldsymbol{\beta}) + \frac{\lambda}{\tau} \sum_{j=1}^{p} |\beta_j| I\big(|\hat{\beta}_j^{(m-1)}| \le \tau\big) - \lambda \sum_{j=1}^{p} I\big(|\hat{\beta}_j^{(m-1)}| > \tau\big), \end{aligned} \tag{3.3}$$

where the last term of (3.3), involving no $\boldsymbol{\beta}$, can be removed when solving for $\hat{\boldsymbol{\beta}}^{(m)}$. Note that $S^{(m)}(\cdot)$ is a convex function and has a form similar to Lasso in (2.3), except that λ is replaced by λ/τ and the L_1 penalty is only on those β_j's for which $\big|\hat{\beta}_j^{(m-1)}\big| < \tau$. Clearly, $\text{sign}\big(|\hat{\beta}_j^{(m)}| - \tau\big) = \text{sign}\big(|\hat{\beta}_j^{(m-1)}| - \tau\big)$ for all $j = 1, \ldots, p$, implies that $\hat{\boldsymbol{\beta}}^{(m+1)} = \hat{\boldsymbol{\beta}}^{(m)}$, which happens when the components of $\hat{\boldsymbol{\beta}}^{(m+1)}$ show no crossing over τ from $\hat{\boldsymbol{\beta}}^{(m)}$. Since there are only a finite number of possible combinations for $\big(\text{sign}\big(|\hat{\beta}_1^{(m)}| - \tau\big), \ldots, \text{sign}\big(|\hat{\beta}_p^{(m)}| - \tau\big)\big)'$, the DCP algorithm has a quite unique feature that it converges in a finite number of

steps. We denote the solution after convergence based on tuning parameters λ and τ by $\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda, \tau)$.

The penalized likelihood $S^{(m)}(\cdot)$ of (3.3) can be minimized efficiently using the iterated reweighted penalized least squares (IRPLS) method. First, we consider a second order approximation $\ell_Q(\boldsymbol{\beta})$ to the log-likelihood $\ell(\boldsymbol{\beta})$ at $\tilde{\boldsymbol{\beta}}$:

$$\begin{aligned} \ell(\boldsymbol{\beta}) \approx \ell_Q(\boldsymbol{\beta}) &= \ell(\tilde{\boldsymbol{\beta}}) + (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})'\nabla\ell(\tilde{\boldsymbol{\beta}}) + \frac{1}{2}(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})'\nabla^2\ell(\tilde{\boldsymbol{\beta}})(\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) \\ &= \frac{-1}{2a(\phi)}\sum_{i=1}^{n}\sigma_i^2(\tilde{\boldsymbol{\beta}})\left\{\boldsymbol{x}_i'\tilde{\boldsymbol{\beta}} + \frac{y_i - \mu_i(\tilde{\boldsymbol{\beta}})}{\sigma_i^2(\tilde{\boldsymbol{\beta}})} - \boldsymbol{x}_i'\boldsymbol{\beta}\right\}^2 + C(\tilde{\boldsymbol{\beta}}), \end{aligned} \tag{3.4}$$

where $C(\tilde{\boldsymbol{\beta}})$ is some constant independent of $\boldsymbol{\beta}$. Then (3.3) can be solved by iteratively minimizing

$$\frac{1}{2n}\sum_{i=1}^{n} w_i(\tilde{\boldsymbol{\beta}})\big(z_i(\tilde{\boldsymbol{\beta}}) - \boldsymbol{x}_i'\boldsymbol{\beta}\big)^2 + \frac{\lambda}{\tau}\sum_{i=1}^{p}|\beta_j| I(|\hat{\beta}_j^{(m-1)}| \le \tau), \tag{3.5}$$

with $\tilde{\boldsymbol{\beta}}$ being the current estimate of $\boldsymbol{\beta}$ in IRPLS, where $w_i(\tilde{\boldsymbol{\beta}}) = \sigma_i^2(\tilde{\boldsymbol{\beta}})/a(\phi)$ and $z_i(\tilde{\boldsymbol{\beta}}) = \boldsymbol{x}_i'\tilde{\boldsymbol{\beta}} + (y_i - \mu_i(\tilde{\boldsymbol{\beta}}))/\sigma_i^2(\tilde{\boldsymbol{\beta}})$.

As in Friedman *et al.* (2007) and Friedman *et al.* (2010), we apply the coordinate decent algorithm to solve the Lasso problem of (3.5), which iteratively minimizes one parameter at a time while holding the others fixed until convergence. The coordinate descent algorithm iteratively update $\beta_0, \beta_1, \ldots, \beta_p$ by:

$$\tilde{\tilde{\beta}}_0^{(new)} = \frac{\sum_{i=1}^{n} w_i(\tilde{\boldsymbol{\beta}})\Big(z_i(\tilde{\boldsymbol{\beta}}) - \sum_{j=1}^{p} x_{ij}\tilde{\tilde{\beta}}_j^{(old)}\Big)}{\sum_{i=1}^{n} w_i(\tilde{\boldsymbol{\beta}})}, \tag{3.6}$$

and

$$\tilde{\tilde{\beta}}_j^{(new)} = \frac{S\left(\frac{1}{n}\sum_{i=1}^{n} w_i(\tilde{\boldsymbol{\beta}})x_{ij}\Big(z_i(\tilde{\boldsymbol{\beta}}) - \big(\boldsymbol{x}_i'\tilde{\tilde{\boldsymbol{\beta}}}_0^{(old)} - x_{ij}\tilde{\tilde{\beta}}_j^{(old)}\big)\Big), \frac{\lambda}{\tau}I\big(\big|\hat{\beta}_j{}^{(m-1)}\big| \le \tau\big)\right)}{\frac{1}{n}\sum_{i=1}^{n} w_i(\tilde{\boldsymbol{\beta}})x_{ij}^2} \tag{3.7}$$

for $j = 1, \ldots, p$, where $S(x, \alpha) \equiv \text{sign}(x)(|x| - \alpha)_+$ is the soft-thresholding operator.

Note that the solution $\hat{\boldsymbol{\beta}}^{(m)}$ of (3.3) is continuous and piecewise linear in λ. Therefore, for each τ, we can obtain $\hat{\boldsymbol{\beta}}^{(m)}$ by starting from a large value of λ, and successively

solve for $\hat{\boldsymbol{\beta}}^{(m)}$ along a decreasing sequence of λ values with each solution being used as a warm start for the next. Given λ and τ, the proposed algorithm in solving (3.1) can be summarized below:

- Outer loop : Solve the DCP solution $\hat{\boldsymbol{\beta}}^{(m)}(\lambda, \tau)$ of (3.3) until $\text{sign}(|\hat{\beta}_j^{(m)}| - \tau) = \text{sign}(|\hat{\beta}_j^{(m-1)}| - \tau)$ for $j = 1, \ldots, p$.
- Middle loop : Update the quadratic approximation expressed in (3.5).
- Inner loop : Apply the coordinate descent algorithm with the updating formulae given by (3.6) and (3.7).

In practice, we compute the penalized ML (PML) estimate $\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda, \tau)$ of $\boldsymbol{\beta}$ at some evaluation points: $(\lambda, \tau) \in \{\lambda_1, \ldots, \lambda_U\} \times \{\tau_1, \ldots, \tau_V\}$, where $\lambda_1 > \lambda_2 > \cdots > \lambda_U$ and $\tau_1 > \tau_2 > \cdots > \tau_V$. Since the solution $\hat{\boldsymbol{\beta}}^{(m)}(\lambda, \tau)$ of (3.3) is continuous in λ for each m and τ, to ensure fast convergence, the proposed algorithm is applied in the following order:

1. Start by computing $\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda_1, \tau_1)$ with initial $\hat{\boldsymbol{\beta}}^{(0)}(\lambda_1, \tau_1) = \mathbf{0}$.
2. Successively compute $\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda_1, \tau_v)$ with initial $\hat{\boldsymbol{\beta}}^{(0)}(\lambda_1, \tau_v) = \hat{\boldsymbol{\beta}}^{(\infty)}(\lambda_1, \tau_{v-1})$, for $v = 2, \ldots, V$.
3. For each $v = 1, \ldots, V$, successively compute $\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda_u, \tau_v)$ with initial $\hat{\boldsymbol{\beta}}^{(0)}(\lambda_u, \tau_v) = \hat{\boldsymbol{\beta}}^{(\infty)}(\lambda_{u-1}, \tau_v)$, for $u = 2, \ldots, U$.

3.2 Selection of tuning parameters

In the previous subsection, we focus on how to obtain the PML solution of (3.1) for a fixed pair of (λ, τ). In this subsection, we introduce two methods, K-fold cross-validation (CV) and BIC, to select the tuning parameters λ and τ. Let $\boldsymbol{Y} = (y_1, \ldots, y_n)'$ denote the response and $\boldsymbol{X} = (\boldsymbol{x}_i, \ldots, \boldsymbol{x}_n)'$ denote the design matrix. For K-fold CV, the index set $\{1, \ldots, n\}$ and the data $(\boldsymbol{Y}, \boldsymbol{X})$ are randomly divided into K sections of roughly the same size. Denote the k-th section of data by $(\boldsymbol{Y}_k, \boldsymbol{X}_k)$; $k = 1, \ldots, K$. For each k, we obtain the PML estimate of $\hat{\boldsymbol{\beta}}_{-k}^{(\infty)}(\lambda, \tau)$ based on data $(\boldsymbol{Y}_{-k}, \boldsymbol{X}_{-k})$ of all the other $K-1$ sections. To assess the prediction performance of $\hat{\boldsymbol{\mu}}_{-k}^{(\infty)}(\lambda, \tau)$ on $\boldsymbol{Y}_k$, we consider the following deviance criterion:

$$\text{Dev}\big(\hat{\boldsymbol{\beta}}_{-k}^{(\infty)}(\lambda, \tau)\big) \equiv -2 \log \frac{L_k\big(\hat{\boldsymbol{\mu}}_{-k}^{(\infty)}(\lambda, \tau), \boldsymbol{Y}_k\big)}{L_k(\boldsymbol{Y}_k, \boldsymbol{Y}_k)}, \tag{3.8}$$

where $\hat{\boldsymbol{\mu}}_{-k}^{(\infty)}(\lambda,\tau)$ is the estimate of $\boldsymbol{\mu}_k$ based on $\hat{\boldsymbol{\beta}}_{-k}^{(\infty)}(\lambda,\tau)$, and $L_k\big(\hat{\boldsymbol{\mu}}_{-k}^{(\infty)}(\lambda,\tau),\boldsymbol{Y}_k\big)$ is the likelihood function of $\boldsymbol{Y}_k$ with mean $\boldsymbol{\mu}_{-k}^{(\infty)}(\lambda,\tau)$. Then our CV criterion is defined by the total deviance:

$$\mathrm{CV}(\lambda,\tau) \equiv \sum_{k=1}^{K} \mathrm{Dev}(\hat{\boldsymbol{\beta}}_{-k}^{(\infty)}(\lambda,\tau))). \tag{3.9}$$

The K-fold CV then selects the pair (λ,τ) with the smallest CV value.

The second criterion we consider for tuning parameter selection is similar to the usual BIC criterion for model selection:

$$BIC(\lambda,\tau) \equiv -2\ell(\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda,\tau)) + \log(n)\sum_{j=1}^{p} I\big(\hat{\beta}_j^{(\infty)}(\lambda,\tau)\neq 0\big).$$

The BIC criterion selects the tuning parameters, $(\hat{\lambda},\hat{\tau}) = \arg\min_{(\lambda,\tau)} BIC(\lambda,\tau)$. The resulting estimate of $\boldsymbol{\beta}$ is given by $\hat{\boldsymbol{\beta}}^{(\infty)}(\hat{\lambda},\hat{\tau})$. We shall apply this criterion for all of our numerical examples, because from our limited experience, it appears to perform better is computationally more efficient than CV.

4 Examples

To understand how the proposed PML method works, we provide two examples: Poisson regression and logistic regression, with detailed computation steps in this section.

4.1 Poisson Regression

For Poisson regression with the canonical link function: $g(\mu_i) = \log\mu_i = \boldsymbol{x}_i'\boldsymbol{\beta}$, the log-likelihood can be expressed as:

$$\ell(\boldsymbol{\beta}) = -\sum_{i=1}^{n}\exp(\boldsymbol{x}_i'\boldsymbol{\beta}) + \sum_{i=1}^{n} y_i(\boldsymbol{x}_i'\boldsymbol{\beta}) - \sum_{i=1}^{n}\log y_i!\,. \tag{4.1}$$

The first and second derivatives of the log-likelihood functions are:

$$\begin{aligned}
\nabla\ell(\boldsymbol{\beta}) &= -\sum_{i=1}^{n}\exp(\boldsymbol{x}_i'\boldsymbol{\beta})\boldsymbol{x}_i + \sum_{i=1}^{n} y_i\boldsymbol{x}_i = \sum_{i=1}^{n}\boldsymbol{x}_i(y_i-\mu_i),\\
\nabla^2\ell(\boldsymbol{\beta}) &= -\sum_{i=1}^{n}\exp(\boldsymbol{x}_i'\boldsymbol{\beta})\boldsymbol{x}_i\boldsymbol{x}_i' = -\sum_{i=1}^{n}\mu_i\boldsymbol{x}_i\boldsymbol{x}_i'\,.
\end{aligned}$$

Thus from (3.4), the quadratic approximation of the log-likelihood function expanded at $\tilde{\boldsymbol{\beta}}$ is:

$$-\frac{1}{2n}\sum_{i=1}^{n}\mu_i(\tilde{\boldsymbol{\beta}})\left\{\boldsymbol{x}_i'\tilde{\boldsymbol{\beta}}+\frac{y_i-\mu_i(\tilde{\boldsymbol{\beta}})}{\mu_i(\tilde{\boldsymbol{\beta}})}-\boldsymbol{x}_i'\boldsymbol{\beta}\right\}^2+C(\tilde{\boldsymbol{\beta}}). \tag{4.2}$$

Consequently, the m-th step DCP solution $\hat{\boldsymbol{\beta}}^{(m)}$ of (3.3) can be obtained by iteratively solving

$$\arg\min_{\boldsymbol{\beta}}\left\{\frac{1}{2n}\sum_{i=1}^{n}\mu_i(\tilde{\boldsymbol{\beta}})\left\{\boldsymbol{x}_i'\tilde{\boldsymbol{\beta}}+\frac{y_i-\mu_i(\tilde{\boldsymbol{\beta}})}{\mu_i(\tilde{\boldsymbol{\beta}})}-\boldsymbol{x}_i'\boldsymbol{\beta}\right\}^2+\frac{\lambda}{\tau}\sum_{i=1}^{n}|\beta_j|I(|\hat{\beta}_j^{(m-1)}|\le\tau)\right\},$$

using the coordinate decent algorithm given in (3.6) and (3.7). From (3.9), the tuning parameters selected by CV satisfies

$$(\hat{\lambda},\hat{\tau})=\arg\min_{(\lambda,\tau)}2\sum_{i=1}^{n}\Big\{y_i(\log y_i-\log\hat{\mu}_i(\lambda,\tau))-(y_i-\hat{\mu}_i(\lambda,\tau))\Big\} \tag{4.3}$$

and the tuning parameters selected by BIC satisfies

$$\begin{aligned}(\hat{\lambda},\hat{\tau}) \;=\; \arg\min_{(\lambda,\tau)}\Bigg\{&2\sum_{i=1}^{n}\Big(\exp(\boldsymbol{x}_i'\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda,\tau)))-y_i(\boldsymbol{x}_i'\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda,\tau))\Big)\\&+\log(n)\sum_{j=1}^{p}I\big(\hat{\beta}_j^{(\infty)}(\lambda,\tau)\neq 0\big)\Bigg\}.\end{aligned}$$

4.2 Logistic Regression

The canonical link function for logistic regression is the logit function: $g(p_i)=\log\dfrac{p_i}{1-p_i}=\boldsymbol{x}_i'\boldsymbol{\beta}$, where p_i is the Bernoulli probability of y_i, which is equal to $\exp(\boldsymbol{x}_i'\boldsymbol{\beta})/(1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta}))$ for $i=1\ldots n$. With the canonical link function, the log-likelihood can be represented by

$$\ell(\boldsymbol{\beta})=\sum_{i=1}^{n}\left\{y_i(\boldsymbol{x}_i'\boldsymbol{\beta})-\log(1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta}))\right\}. \tag{4.4}$$

The first and second derivatives of the log-likelihood functions are:

$$\begin{aligned}\nabla\ell(\boldsymbol{\beta}) \;&=\; \sum_{i=1}^{n}y_i\boldsymbol{x_i}-\sum_{i=1}^{n}\frac{\exp\{\boldsymbol{x}_i'\boldsymbol{\beta}\}\boldsymbol{x_i}}{1+\exp\{\boldsymbol{x}_i'\boldsymbol{\beta}\}}=\sum_{i=1}^{n}\boldsymbol{x_i}(y_i-p_i),\\ \nabla^2\ell(\boldsymbol{\beta}) \;&=\; -\sum_{i=1}^{n}\frac{\exp\{\boldsymbol{x}_i'\boldsymbol{\beta}\}\boldsymbol{x_i}\boldsymbol{x_i}'}{1+\exp\{\boldsymbol{x}_i'\boldsymbol{\beta}\}}\left\{1-\frac{\exp\{\boldsymbol{x}_i'\boldsymbol{\beta}\}}{1+\exp\{\boldsymbol{x}_i'\boldsymbol{\beta}\}}\right\}=-\sum_{i=1}^{n}p_i(1-p_i)\boldsymbol{x_i}\boldsymbol{x_i}'.\end{aligned}$$

Thus from (3.4), the quadratic approximation of the log-likelihood function expanded at $\tilde{\boldsymbol{\beta}}$ is:

$$-\frac{1}{2n}\sum_{i=1}^{n}\tilde{p}_i(1-\tilde{p}_i)\left\{\boldsymbol{x}_i'\tilde{\boldsymbol{\beta}}+\frac{y_i-\tilde{p}_i}{\tilde{p}_i(1-\tilde{p}_i)}-\boldsymbol{x}_i'\boldsymbol{\beta}\right\}^2+C(\tilde{\boldsymbol{\beta}}), \tag{4.5}$$

where $\tilde{p}_i=\exp\{\boldsymbol{x}_i'\tilde{\boldsymbol{\beta}}\}/(1+\exp\{\boldsymbol{x}_i'\tilde{\boldsymbol{\beta}}\})$. Consequently, the m-th step DCP solution $\hat{\boldsymbol{\beta}}^{(m)}$ of (3.3) can be obtained by iteratively solving

$$\arg\min_{\boldsymbol{\beta}}\left\{\frac{1}{2n}\sum_{i=1}^{n}\tilde{p}_i(1-\tilde{p}_i)\left\{\boldsymbol{x}_i'\tilde{\boldsymbol{\beta}}+\frac{y_i-\tilde{p}_i}{\tilde{p}_i(1-\tilde{p}_i)}-\boldsymbol{x}_i'\boldsymbol{\beta}\right\}^2+\frac{\lambda}{\tau}\sum_{i=1}^{n}|\beta_j|I(|\hat{\beta}_j^{(m-1)}|\le\tau)\right\}$$

using the coordinate decent algorithm given in (3.6) and (3.7). From (3.9), the tuning parameters selected by CV satisfies

$$(\hat{\lambda},\hat{\tau})=\arg\min_{(\lambda,\tau)}2\sum_{i=1}^{n}\left\{y_i\log\frac{y_i}{\hat{p}_i(\lambda,\tau)}+(1-y_i)\log\frac{1-y_i}{1-\hat{p}_i(\lambda,\tau)}\right\}, \tag{4.6}$$

and the tuning parameters selected by BIC satisfies

$$(\hat{\lambda},\hat{\tau})=\arg\min_{(\lambda,\tau)}\left\{2\sum_{i=1}^{n}\{\log(1+\exp(\boldsymbol{x}_i'\boldsymbol{\beta}))-y_i(\boldsymbol{x}_i'\boldsymbol{\beta})\}+\log(n)\sum_{j=1}^{p}I\big(\hat{\beta}_j^{(\infty)}(\lambda,\tau)\neq 0\big)\right\}.$$

5 Oracle Property

Let $\boldsymbol{\beta}_0\equiv(\beta_{00},\beta_{10},\ldots,\beta_{p0})'$ be the true regression coefficients and let $A=\{j:|\beta_{j0}|\neq 0,\ j=1,\ldots,p\}$ be the index set of the true model. For notational simplicity, the proposed estimate $\hat{\boldsymbol{\beta}}^{(\infty)}(\lambda,\tau)$ is written as $\hat{\boldsymbol{\beta}}^{(\infty)}$. Let $\hat{A}\equiv\big\{j=1,\ldots,p:\big|\hat{\beta}_j^{(\infty)}\big|>0\big\}$ be the index set selected by the proposed PML method, and let $\hat{\boldsymbol{\beta}}^{(ml)}=\big(\hat{\beta}_0^{(ml)},\hat{\beta}_1^{(ml)},\ldots,\hat{\beta}_p^{(ml)}\big)'$ be the maximum likelihood (ML) estimate of $\boldsymbol{\beta}$ based on the true model, where $\hat{\beta}_j^{(ml)}=0$ if $j\notin A$. We would like to establish $P\big(\hat{\boldsymbol{\beta}}^{(\infty)}=\hat{\boldsymbol{\beta}}^{(ml)}\big)\to 1$ as $n\to\infty$ under some regularity conditions.

We first show that $\hat{\beta}_j^{(\infty)}\neq\tau$ for $j=1,\ldots,p$.

Lemma 5.1. *Let $h(\cdot)$ be any differentiable function in $\mathcal{R}^p$ and $\boldsymbol{\beta}^*=(\beta_1^*,\ldots,\beta_p^*)'$ be a local minimizer of $f(\boldsymbol{\beta})=h(\boldsymbol{\beta})+\lambda\sum_{j=1}^{p}\min\{|\beta_j|/\tau,1\}$. Then $\beta_j^*\neq\tau$ for $j=1,\ldots,p$.*

Proof. Prove by contradiction. Suppose $\beta_k^*=\tau$ for some $k\in\{1,\ldots,p\}$ and define $J_j(|\beta_j|)=\min\left\{\frac{|\beta_j|}{\tau},1\right\}$. Let $f_k(\beta_k)=f(\beta_1^*,\ldots,\beta_k,\ldots,\beta_p^*)$ and $h_k(\beta_k)=h(\beta_1^*,\ldots,\beta_k,\ldots,\beta_p^*)$

with $\beta_j = \beta_j^*$ for $j \neq k$. Then $f_k(\beta_k) = h_k(\beta_k) + \lambda \sum_{j \neq k} J_j(|\beta_j^*|) + \lambda J_k(|\beta_k|)$. The right derivative of $J_k(|\beta_k|)$ at β_k^* is 0 and the left derivative of $J_k(|\beta_k|)$ at β_k^* is $\frac{1}{\tau}$. Let D denote the derivative of $\sum_{j \neq k} J_j(|\beta_j^*|)$. Since β_k^* is a local minimizer of $f_k(\beta_k)$, the right and left derivatives of $f_k(\beta_k)$ at β_k^* are larger than 0 and smaller than 0. Then the right and left derivatives of $h_k(\beta_k)$ at β_k^* are larger than $-\lambda D$ and smaller than $-\lambda(D + \frac{1}{\tau})$, which contradicts to the fact that $h(\cdot)$ is a differentiable function in $\mathcal{R}^p$. Thus we obtain the conclusion that $\beta_j^* \neq \tau$ for $j = 1, \ldots, p$. □

Differentiating the object function $S(\cdot)$ in (3.1) with respect to $\boldsymbol{\beta}$ through the concept of subdifferentials, we obtain the local optimality condition:

$$-\frac{1}{n}\frac{\partial}{\partial \beta_j}\ell(\boldsymbol{\beta}) + b_j\frac{\lambda}{\tau} = 0, \quad \text{where } b_j \begin{cases} = \text{sign}(\beta_j) & \text{if } 0 < |\beta_j| < \tau, \\ = 0 & \text{if } |\beta_j| > \tau, \\ \in [-1, 1] & \text{if } |\beta_j| = 0. \end{cases} \tag{5.1}$$

for $j = 1, \ldots, p$. This local optimality condition is known as the Karush-Kuhn-Tucker (KKT) condition (see Lange 2004). Note that when $\beta_j = 0$, the local optimality condition can be represented by $\left|\frac{\partial}{\partial \beta_j}\ell(\boldsymbol{\beta})\right| \leq n\lambda/\tau$.

We shall establish the oracle property of $\hat{\boldsymbol{\beta}}^{(\infty)}$ in the theorem below by showing first that both $\hat{\boldsymbol{\beta}}^{(\infty)}$ and $\hat{\boldsymbol{\beta}}^{(ml)}$ satisfy the KKT condition of (5.1), and then by showing that the solution of (3.1) is unique with probability tending to one under some conditions.

Theorem 5.2. *Consider the generalized linear model of (1.4) with $\boldsymbol{\beta} \in \Theta \subset \mathbb{R}^{p+1}$, where Θ is compact. Let $\hat{\boldsymbol{\beta}}^{(\infty)}$ be the PML estimate of (3.1). Suppose that (i) $\max\{|A|, |\hat{A}|\} < q$ for some constant $0 < q < \infty$, (ii) $\inf_{j \in A} |\beta_{j0}| > 2\tau$, (iii) $n^{1/2}\lambda/\tau \to \infty$, and (iv) $\tau^2 c_{\min} - 4\lambda\sqrt{q} > 0$, where $c_{\min}$ denotes the minimum eigenvalue of $-\frac{1}{n}\min_{\boldsymbol{\beta} \in \Theta} \nabla^2 \ell(\boldsymbol{\beta})$. Then $P\big(\hat{\boldsymbol{\beta}}^{(\infty)} = \hat{\boldsymbol{\beta}}_A^{(ml)}\big) \to 1$ as $n \to \infty$.*

Proof. Let $F = F_1 \bigcap F_2$, where

$$F_1 = \left\{\min_{j \in A} \left|\hat{\beta}_j^{(ml)}\right| > \frac{3}{2}\tau\right\} \text{ and } F_2 = \left\{\max_{j \notin A} \left| -\frac{1}{n}\frac{\partial}{\partial \beta_j}\ell(\hat{\boldsymbol{\beta}}^{(ml)})\right| \leq \frac{\lambda}{\tau}\right\}.$$

First, we show that both $\hat{\boldsymbol{\beta}}^{(\infty)}$ and $\hat{\boldsymbol{\beta}}^{(ml)}$ satisfy the KKT condition of (5.1) on F. Since $\hat{\boldsymbol{\beta}}^{(m)}$ minimizes $S^{(m)}(\boldsymbol{\beta}) = -\frac{1}{n}\ell(\boldsymbol{\beta}) + \frac{\lambda}{\tau}\sum_{j=1}^{p} |\beta_j| I(|\beta_j^{(m-1)}| \leq \tau)$, it follows that $\hat{\boldsymbol{\beta}}^{(\infty)}$ is a

minimizer of $S^{(\infty)}(\boldsymbol{\beta}) = -\frac{1}{n}\ell(\boldsymbol{\beta}) + \frac{\lambda}{\tau}\sum_{j=1}^{p}|\beta_j| I(|\beta_j^{(\infty)}| \le \tau)$. This implies $\hat{\boldsymbol{\beta}}^{(\infty)}$ satisfies the KKT condition. Regarding $\hat{\boldsymbol{\beta}}_A^{(ml)}$, clearly it satisfies the KKT condition for $j \in A$ under F_1. In addition, it satisfies $\left| -\frac{1}{n}\frac{\partial}{\partial\beta_j}\ell(\hat{\boldsymbol{\beta}}^{(ml)}) \right| \le \frac{\lambda}{\tau}$, and hence the KKT condition, for $j \notin A$ under F_2. Therefore, $\hat{\boldsymbol{\beta}}^{(ml)}$ also satisfies the KKT condition under F.

Next, we show $\hat{\boldsymbol{\beta}}^{(\infty)} = \hat{\boldsymbol{\beta}}^{(ml)}$ on F. We prove by contradiction. Suppose that $\hat{\boldsymbol{\beta}}^{(\infty)} \neq \hat{\boldsymbol{\beta}}^{(ml)}$ on F. Consider two cases below, where we use $\|\cdot\|$ to denote the L_2 norm.

(i) $\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\| \le \tau/2$: For $j \in A$, since $|\hat{\beta}_j^{(ml)}| > 3\tau/2$, it implies $|\hat{\beta}_j^{(\infty)}| > \tau$. For $j \notin A$, since $\hat{\beta}_j^{(ml)} = 0$, it implies $|\hat{\beta}_j^{(\infty)}| < \tau$. Therefore, both estimates are solutions of a strictly convex function, $-\frac{1}{n}\ell(\boldsymbol{\beta}) + \frac{\lambda}{\tau}\sum_{j\notin A}|\beta_j|$, on F, and hence has the unique minimum on F. Thus $\hat{\boldsymbol{\beta}}^{(\infty)} = \hat{\boldsymbol{\beta}}^{(ml)}$ on F.

(ii) $\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\| > \tau/2$: We have

$$\left| \Big(\frac{\partial}{\partial\boldsymbol{\beta}_A} S\big(\hat{\boldsymbol{\beta}}^{(\infty)}\big) - \frac{\partial}{\partial\boldsymbol{\beta}_A} S\big(\hat{\boldsymbol{\beta}}^{(ml)}\big)\Big)^T \frac{\big(\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\big)}{\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\|} \right| \ge f_1 - f_2,$$

where

$$f_1 = \left| \Big(\frac{1}{n}\nabla\ell(\hat{\boldsymbol{\beta}}^{(\infty)}) - \frac{1}{n}\nabla\ell\big(\hat{\boldsymbol{\beta}}^{(ml)}\big)\Big)^T \frac{(\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)})}{\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\|} \right|,$$

$$f_2 = \frac{\lambda}{\tau}\left| \Big(\mathrm{sign}\big(\hat{\boldsymbol{\beta}}^{(\infty)}\big) I\big(|\hat{\boldsymbol{\beta}}^{(\infty)}| < \tau\big) - \mathrm{sign}\big(\hat{\boldsymbol{\beta}}^{(ml)}\big) I\big(|\hat{\boldsymbol{\beta}}^{(ml)}| < \tau\big)\Big)^T \frac{(\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)})}{\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\|} \right|.$$

For F_1, by the mean value theorem, there exists $\hat{\boldsymbol{\beta}}^*$ on the line segment between $\hat{\boldsymbol{\beta}}^{(\infty)}$ and $\hat{\boldsymbol{\beta}}^{(ml)}$ such that

$$f_1 = \frac{1}{n}\left| \Big(\nabla^2\ell\big(\hat{\boldsymbol{\beta}}^*\big)\big(\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\big)\Big)^T \frac{\big(\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\big)}{\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\|} \right| \ge c_{\min}\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\| \ge c_{\min}\frac{\tau}{2}.$$

For f_2, by Cauchy-Schwartz inequality,

$$\begin{aligned} f_2 &= \frac{\lambda}{\tau}\left| \Big(\mathrm{sign}\big(\hat{\boldsymbol{\beta}}^{(\infty)}\big) I\big(|\hat{\boldsymbol{\beta}}^{(\infty)}| < \tau\big) - \mathrm{sign}\big(\hat{\boldsymbol{\beta}}^{(ml)}\big) I\big(|\hat{\boldsymbol{\beta}}^{(ml)}| < \tau\big)\Big)^T \frac{\big(\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\big)}{\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\|} \right| \\ &\le \frac{\lambda}{\tau}\left\| \mathrm{sign}\big(\hat{\boldsymbol{\beta}}^{(\infty)}\big) I\big(|\hat{\boldsymbol{\beta}}^{(\infty)}| < \tau\big) - \mathrm{sign}\big(\hat{\boldsymbol{\beta}}^{(ml)}\big) I\big(|\hat{\boldsymbol{\beta}}^{(ml)}| < \tau\big) \right\| \frac{\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}_{A_0}^{(ml)}\|}{\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}_{A_0}^{(ml)}\|} \\ &\le 2\sqrt{q}\lambda/\tau\,. \end{aligned}$$

Consequently, $\left|\left(\frac{\partial}{\partial \boldsymbol{\beta}_A} S(\hat{\boldsymbol{\beta}}^{(\infty)}) - \frac{\partial}{\partial \boldsymbol{\beta}_A} S(\hat{\boldsymbol{\beta}}^{(ml)})\right)^T \frac{(\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)})}{\|\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)}\|}\right| \geq c_{\min}\tau/2 - 2\sqrt{q}\lambda/\tau > 0$, which contradicts to that $0 \in \left(\frac{\partial}{\partial \boldsymbol{\beta}_A} S(\hat{\boldsymbol{\beta}}^{(\infty)}) - \frac{\partial}{\partial \boldsymbol{\beta}_A} S(\hat{\boldsymbol{\beta}}^{(ml)})\right)^T (\hat{\boldsymbol{\beta}}^{(\infty)} - \hat{\boldsymbol{\beta}}^{(ml)})$. Thus $\hat{\boldsymbol{\beta}}^{(\infty)} = \hat{\boldsymbol{\beta}}^{(ml)}$ on F.

Finally, it is straightforward to show that $P(F) \to 1$ as $n \to \infty$. This completes the proof.

Note that under some regularity conditions,

$$n^{1/2}(\hat{\boldsymbol{\beta}}^{(ml)} - \boldsymbol{\beta}_0) \xrightarrow{d} N(\mathbf{0}, \boldsymbol{I}^{-1}), \tag{5.2}$$

where $\boldsymbol{I}$ is the Fisher information. It follows that

$$n^{1/2}(\hat{\boldsymbol{\beta}}^{(\infty)} - \boldsymbol{\beta}_0) \xrightarrow{d} N(\mathbf{0}, \boldsymbol{I}^{-1}),$$

under (5.2) and the conditions given by the above theorem. □

6 Numerical Examples

In this chapter, we consider two simulation experiments with one for Poisson regression and the other for logistic regression cases. We are interested in knowing the performance of the proposed PML method in comparison with AIC, BIC, and their stepwise versions in terms of the Kullback-Leibler (KL) risk and the probability of identifying the true model.

6.1 Poisson Regression

We generate data $y_1, \ldots, y_n$ independently from Poisson distributions with means $\mu_1, \ldots, \mu_n$ according to the model:

$$\log \mu_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} = \boldsymbol{x}_i' \boldsymbol{\beta}; \quad i = 1, \ldots, n,$$

where $\boldsymbol{x}_i$'s are generated independently from $N(\mathbf{0}, \boldsymbol{\Sigma})$ with $\rho^{|i-j|}$ being the (i,j)th entry of $\boldsymbol{\Sigma}$, and $\boldsymbol{\beta} = (0, 2, -1, 0, \ldots, 0)'$. We apply the proposed PML method for $\tau = \tau_1, \ldots, \tau_5$ and $\lambda = \lambda_1, \ldots, \lambda_{100}$, which are equally spaced in the log scale. Throughout the simulation, we choose $\tau_1 = 5$ and $\tau_5 = 0.1$. In addition, we choose λ_1 to be the smallest value such that $\hat{\beta}_1^{(\infty)}(\lambda, 5) = \cdots = \hat{\beta}_p^{(\infty)}(\lambda, 5) = 0$, which can be obtained from the R package "glmnet" developed by Friedman *et al.* (2010) only for the Lasso penalty, and choose $\lambda_{100} = 0.001$.

Table 1: Performance of various methods for Poisson regression with $p = 6$ and $n = 50$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	3.822 (0.273)	1.00	3.582 (0.319)	1.00
True model MLE	1.610 (0.173)	0.00	1.603 (0.191)	0.00
AIC	2.978 (0.298)	0.53	2.717 (0.325)	0.42
BIC	2.135 (0.244)	0.17	2.042 (0.258)	0.14
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	2.045 (0.245)	0.13	1.983 (0.258)	0.12
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	1.728 (0.181)	0.08	1.771 (0.212)	0.09
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	1.909 (0.216)	0.14	2.428 (0.338)	0.23
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	3.393 (0.282)	0.65	3.386 (0.347)	0.70
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	3.396 (0.278)	0.74	3.439 (0.342)	0.73
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	3.396 (0.278)	0.74	3.439 (0.342)	0.73
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	2.030 (0.243)	0.12	1.975 (0.258)	0.11

For different methods, we estimate the Kullback-Leibler risk (KL risk) which take expectation of the KL loss function. KL loss measures the difference between the estimates and the true parameters. The KL loss for Poisson cases is defined by

$$\sum_{i=1}^{n} \left\{ \mu_i (\log \mu_i - \log \hat{\mu}_i) - (\mu_i - \hat{\mu}_i) \right\}.$$

From the KL risk, we can see whether our method overcomes the other competitors. Table 1 and 2 present the results of repeating 100 times simulations and show the KL risk together with its standard error. The error rate is estimated by the proportion of how many times the methods chose the wrong model during the 100 simulations. When determining the tuning parameters, we utilize the BIC score method, which choose the estimates corresponding to the smallest BIC score.

From the results of Table 1 and 2, we notice that when there are more zero coefficients than non-zero coefficients, our method performs better than the others. We can see that the ML estimate of $\boldsymbol{\beta}$ based on the full model are more unstable. The method PML chooses a suitable tuning parameters pair (λ, τ) can do the model selection and estimation as good as AIC/BIC does even sometimes overcomes them. When the number of interested variables increases, AIC and BIC tend to consume much of time on computation. Therefore, PML offers an efficient way to select a suitable model and estimates the coefficient simultaneously, which not only saves the energy on examine each possible

Table 2: Performance of various methods for Poisson regression with $p = 10$ and $n = 50$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	5.731 (0.320)	1.00	5.830 (0.370)	1.00
True model MLE	1.610 (0.173)	0.00	1.603 (0.191)	0.00
AIC	3.931 (0.346)	0.75	3.955 (0.373)	0.73
BIC	2.529 (0.286)	0.27	2.522 (0.311)	0.26
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	2.050 (0.243)	0.14	2.317 (0.306)	0.20
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	1.689 (0.174)	0.08	1.915 (0.251)	0.15
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	1.957 (0.235)	0.19	2.812 (0.390)	0.36
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	4.012 (0.327)	0.80	4.421 (0.366)	0.86
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	3.902 (0.299)	0.89	4.405 (0.361)	0.90
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	3.902 (0.299)	0.89	4.405 (0.361)	0.90
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	2.045 (0.242)	0.14	2.309 (0.306)	0.19

models but also provides better estimates with smaller risk and standard error.

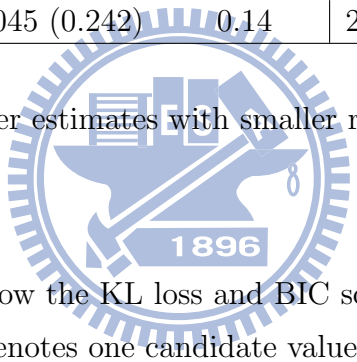


Figure 3 and 4 show the how the KL loss and BIC scores behave with the path of λ^* under log scale. Each curve denotes one candidate value of τ. It is easy to see that these curves for each τ has a minimum value. Obviously, the two figures present almost the same patterns. It is intuitive to use BIC scores as a criterion of determining the tuning parameters. Even though we have different values of τ, their BIC scores can help us find a good λ^* for each τ and give not bad estimates. Note that in the two tuning parameters, λ plays a more important role in model selection than τ does.

From small p cases, we can see that the PML method can provide better estimates which are more precise and accurate than the others. However, high dimension problems become more and more important recently. We are interested in whether PML is available when p increases. We try larger $p = 20$ and $p = 40$ for Poisson regression. Note that p is to large to use AIC/BIC method since there are 2^p models to compare. We utilize stepwise AIC/BIC instead. The results are shown in Table 3 to Table 5.

Under high dimensional conditions, the benefits of PML method are easier to see. Compared to the other methods, our method provides little risk and smaller standard error. The proportion of selecting the wrong model is smaller than its competitors.

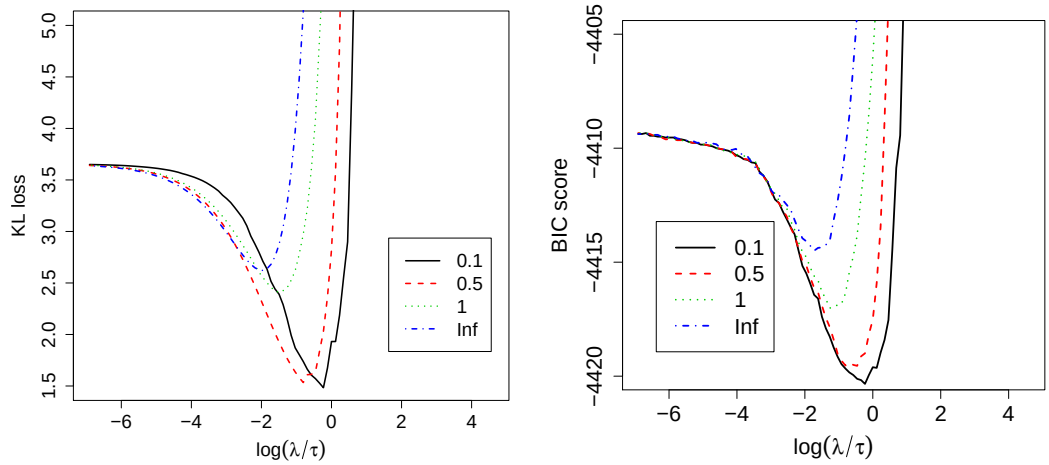


Figure 3: KL losses (left) and the corresponding BIC scores based on PML (right) for poisson regression with $p = 6$ and $n = 50$.

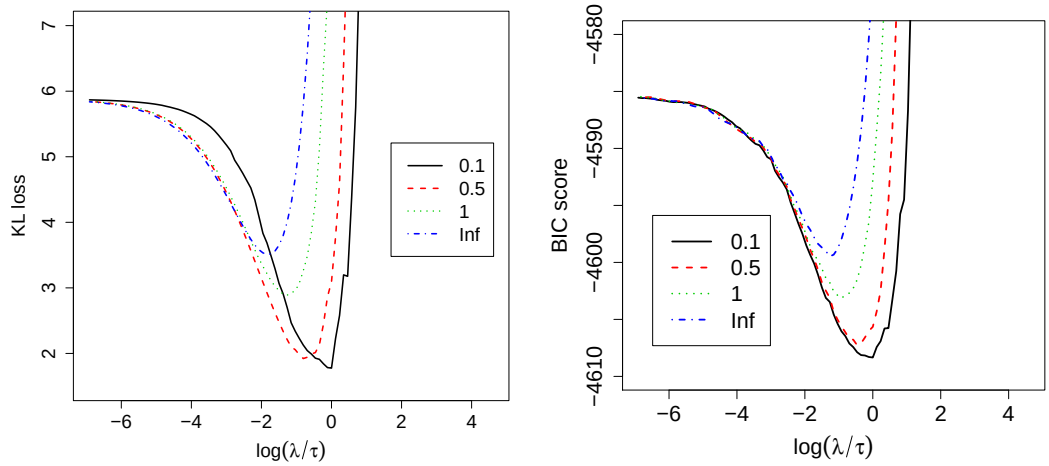


Figure 4: KL losses (left) and the corresponding BIC scores based on PML (right) for poisson regression with $p = 10$ and $n = 50$.

Table 3: Performance of various methods for Poisson regression with $p = 20$ and $n = 50$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	12.016 (0.610)	1.00	12.216 (0.528)	1.00
True model MLE	1.610 (0.173)	0.00	1.603 (0.191)	0.00
Stepwise AIC	8.223 (0.571)	0.94	8.306 (0.537)	0.97
Stepwise BIC	4.628 (0.502)	0.52	4.500 (0.505)	0.54
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	2.367 (0.300)	0.20	2.928 (0.430)	0.29
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	1.746 (0.181)	0.10	1.846 (0.256)	0.15
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	2.107 (0.251)	0.25	3.324 (0.425)	0.54
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	5.104 (0.340)	0.93	6.203 (0.456)	0.98
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	4.963 (0.325)	0.95	6.234 (0.455)	1.00
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	4.963 (0.325)	0.95	6.234 (0.455)	1.00
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	2.310 (0.295)	0.18	2.858 (0.429)	0.26

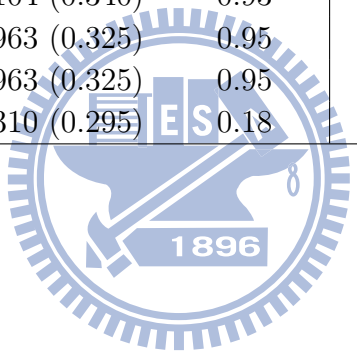


Table 4: Performance of various methods for Poisson regression with $p = 20$ and $n = 100$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	10.555 (0.317)	1.00	10.786 (0.310)	1.00
True model MLE	1.652 (0.145)	0.00	1.725 (0.145)	0.00
Stepwise AIC	7.057 (0.354)	0.92	7.185 (0.332)	0.94
Stepwise BIC	3.411 (0.268)	0.39	3.076 (0.257)	0.31
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	1.927 (0.162)	0.11	1.954 (0.163)	0.08
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	1.758 (0.151)	0.09	1.766 (0.145)	0.04
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	1.760 (0.151)	0.11	2.310 (0.281)	0.08
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	5.278 (0.399)	0.89	6.222 (0.396)	0.88
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	5.378 (0.389)	0.87	6.442 (0.390)	0.90
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	5.378 (0.389)	0.97	6.442 (0.390)	0.90
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	1.927 (0.162)	0.11	1.954 (0.163)	0.08

Table 5: Performance of various methods for Poisson regression with $p = 40$ and $n = 100$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	22.497 (0.554)	1.00	22.585 (0.519)	1.00
True model MLE	1.652 (0.145)	0.00	1.725 (0.145)	0.00
Stepwise AIC	14.538 (0.585)	1.00	14.246 (0.501)	1.00
Stepwise BIC	5.364 (0.435)	0.58	5.011 (0.400)	0.55
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	1.968 (0.184)	0.13	2.001 (0.157)	0.11
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	1.864 (0.166)	0.12	1.750 (0.145)	0.03
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	1.938 (0.187)	0.15	2.905 (0.375)	0.22
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	6.430 (0.409)	0.92	8.435 (0.451)	0.98
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	7.098 (0.399)	1.00	8.481 (0.444)	1.00
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	7.098 (0.399)	1.00	8.481 (0.444)	1.00
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	1.968 (0.184)	0.13	1.982 (0.158)	0.10

Note that our objective function (3.1) can approximate AIC/BIC when τ is close to zero. We are interested whether our L_0 approach works for the approximation of exhausted AIC and BIC. We want to know whether AIC/BIC and our approach select the same model and whether their AIC/BIC score are close under small p cases. Table 6 and Table 7 shows the proportion that our approach well approximates the results of AIC and BIC.

From the results, we can see that larger sample size leads to larger well-approximation proportion.

6.2 Logistic Regression

For logistic regression simulation settings, the observation $y_1, y_2, \ldots, y_n$ are generated from bernoulli distribution with probability p_i satisfying the model

$$\frac{p_i}{1 - p_i} = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{iq} = \boldsymbol{x}_i' \boldsymbol{\beta},$$

where $\boldsymbol{x}_i = (x_{i0}, x_{i1}, \ldots, x_{ip})'$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)'$ are $(p+1)-$dimension vectors. The predictors x_i are generated form i.i.d. normal $N(0, \Sigma_{p\times p})$ distribution where $\Sigma_{p\times p}$ denotes the correlation matrix of time series AR(1) model with each component expressed by $\rho^{|i-j|}$. We assign $\boldsymbol{\beta} = (0, 3, -2, 0, \ldots, 0)'$ We do the same procedure as we do in Poisson

Table 6: Approximations to AIC and BIC for Poisson regression with $p = 6$.

		proportion of selecting the same model		proportion of having the same estimates		proportion of both the events occur	
		AIC	BIC	AIC	BIC	AIC	BIC
n=50	$\tau = 0.5$	0.02	0	0	0	0	0
	$\tau = 0.1$	0.26	0.63	0.10	0.55	0.10	0.55
	$\tau = 0.01$	0.41	0.46	0.41	0.46	0.41	0.46
	$\tau = 0.001$	0.05	0.04	0.05	0.04	0.05	0.04
n=100	$\tau = 0.5$	0	0	0	0	0	0
	$\tau = 0.1$	0.05	0.27	0	0.26	0	0.26
	$\tau = 0.01$	0.45	0.78	0.44	0.78	0.44	0.78
	$\tau = 0.001$	0.14	0.05	0.14	0.05	0.14	0.05
n=200	$\tau = 0.5$	0	0	0	0	0	0
	$\tau = 0.1$	0.03	0.10	0	0.08	0	0.08
	$\tau = 0.01$	0.67	0.85	0.64	0.85	0.64	0.85
	$\tau = 0.001$	0.43	0.26	0.43	0.26	0.43	0.26
n=500	$\tau = 0.5$	0	0	0	0	0	0
	$\tau = 0.1$	0.02	0	0	0	0	0
	$\tau = 0.01$	0.76	0.91	0.59	0.91	0.59	0.91
	$\tau = 0.001$	0.55	0.89	0.55	0.89	0.55	0.89

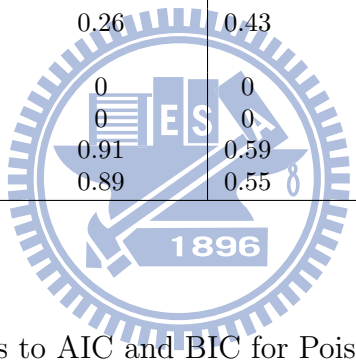


Table 7: Approximations to AIC and BIC for Poisson regression with $p = 10$.

		proportion of selecting the same model		proportion of having the same estimates		proportion of both the events occur	
		AIC	BIC	AIC	BIC	AIC	BIC
n=50	$\tau = 0.5$	0	0	0	0	0	0
	$\tau = 0.1$	0.08	0.40	0.04	0.34	0.04	0.34
	$\tau = 0.01$	0.16	0.38	0.16	0.38	0.16	0.38
	$\tau = 0.001$	0.04	0.01	0.03	0	0.03	0
n=100	$\tau = 0.5$	0	0	0	0	0	0
	$\tau = 0.1$	0	0.10	0	0.05	0	0.05
	$\tau = 0.01$	0.21	0.62	0.21	0.62	0.21	0.62
	$\tau = 0.001$	0.09	0.05	0.09	0.05	0.09	0.05
n=200	$\tau = 0.5$	0	0	0	0	0	0
	$\tau = 0.1$	0	0.02	0	0.01	0	0.01
	$\tau = 0.01$	0.27	0.71	0.26	0.71	0.26	0.71
	$\tau = 0.001$	0.30	0.14	0.30	0.14	0.30	0.14
n=500	$\tau = 0.5$	0	0	0	0	0	0
	$\tau = 0.1$	0	0	0	0	0	0
	$\tau = 0.01$	0.52	0.81	0.31	0.81	0.31	0.81
	$\tau = 0.001$	0.24	0.71	0.24	0.71	0.24	0.71

Table 8: Performance of various methods for logistic regression with $p = 6$ and $n = 100$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	4.906 (0.399)	1.00	4.133 (0.242)	1.00
True model MLE	2.004 (0.204)	0.00	1.591 (0.128)	0.00
AIC	3.727 (0.352)	0.47	3.074 (0.234)	0.47
BIC	2.558 (0.255)	0.12	2.017 (0.187)	0.10
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	2.558 (0.255)	0.12	2.017 (0.187)	0.10
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	2.561 (0.200)	0.11	2.017 (0.187)	0.10
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	2.317 (0.236)	0.09	1.875 (0.188)	0.09
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	3.693 (0.256)	0.56	3.045 (0.183)	0.74
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	4.095 (0.312)	0.60	3.154 (0.166)	0.74
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	4.108 (0.313)	0.60	3.154 (0.166)	0.74
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	2.558 (0.255)	0.12	2.017 (0.187)	0.10

regression. The KL loss for logistic regression is defined by

$$\sum_{i=1}^{n} \left\{ p_i \log \frac{p_i}{\hat{p}_i} + (1 - p_i) \log \frac{1 - p_i}{1 - \hat{p}_i} \right\}.$$

First we consider smaller dimension cases with $p = 6$, $p = 10$ and $\rho = 0$, $\rho = 0.5$, respectively. Table 8 and 9 presents the results of repeating 100 times of simulations and show the KL risk together with its standard error. The error rate is estimated by the proportion of how many times during the 100 simulations the methods chose the wrong model. When determining the tuning parameters, we utilize the BIC score method and choose the estimates with the smallest BIC score.

Compare to the results of Poisson simulations, the results of logistic regression are very unstable. The reason can be seen from Figure 5 and Figure 6. We can see the patterns in KL loss and BIC scores do not match. We choose our estimates with smallest BIC value, but the estimates we choose does not achieve the point with the smallest loss. The most possible reason might be the sample size is too small to estimate the coefficient precisely and accurately. The response we have is binary data and it tells little information. Although we can not choose the optimize solution, PML method still competes the other methods. We can see the KL risk of PML is smaller than the others and the proportion

Table 9: Performance of various methods for logistic regression with $p = 10$ and $n = 100$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	9.528 (0.741)	1.00	8.442 (0.674)	1.00
True model MLE	1.804 (0.210)	0.00	2.001 (0.310)	0.00
AIC	6.405 (0.505)	0.85	5.899 (0.574)	0.89
BIC	2.949 (0.288)	0.26	3.133 (0.388)	0.24
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	2.955 (0.307)	0.25	3.092 (0.393)	0.22
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	2.680 (0.289)	0.18	2.849 (0.390)	0.14
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	2.139 (0.237)	0.12	2.359 (0.340)	0.18
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	4.449 (0.267)	0.82	4.706 (0.309)	0.82
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	4.408 (0.210)	0.75	4.831 (0.254)	0.80
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	4.408 (0.210)	0.75	4.985 (0.299)	0.80
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	2.898 (0.286)	0.25	3.092 (0.393)	0.22

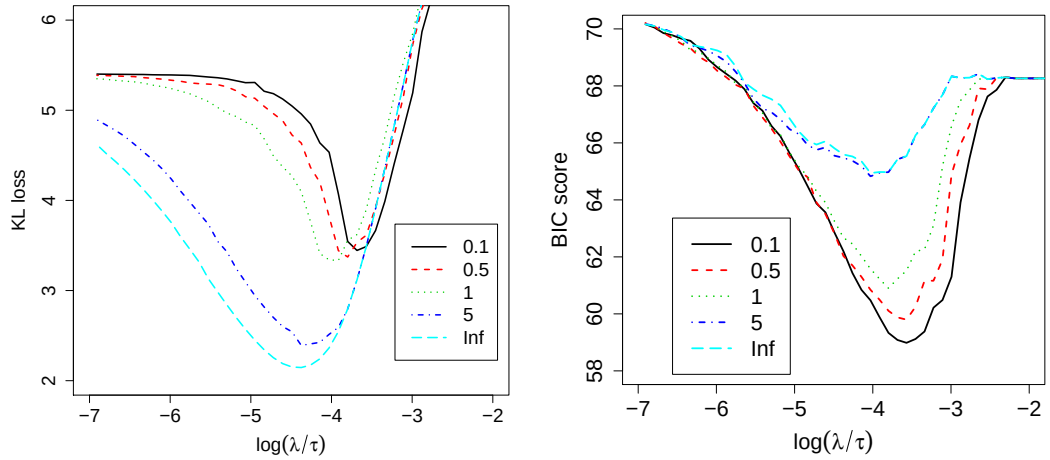


Figure 5: KL losses (left) and the corresponding BIC scores based on PML (right) for logistic regression with $p = 6$ and $n = 50$.

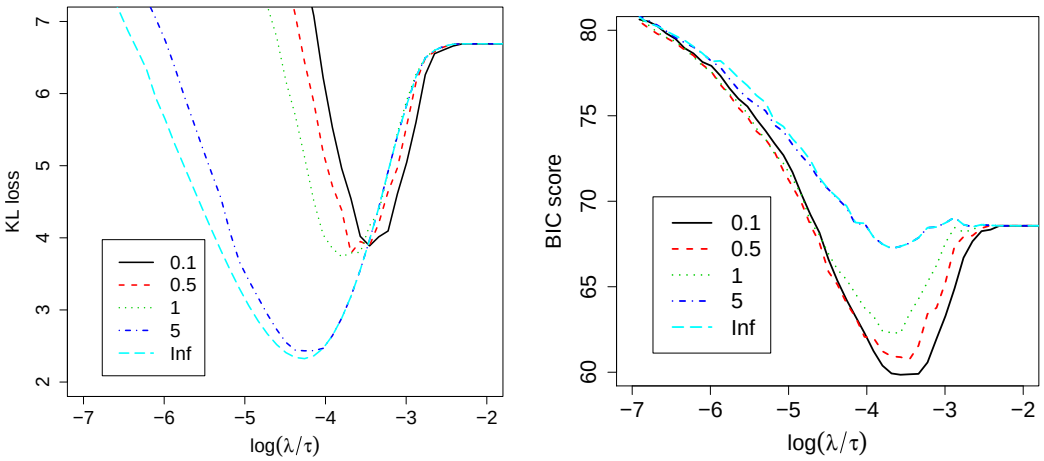


Figure 6: KL losses (left) and the corresponding BIC scores based on PML (right) for logistic regression with $p = 10$ and $n = 50$.

of selecting the wrong model is smaller than its competitors. So we can still conclude our method does better.

We are also interested in high dimensional logistic regression, then we try $p = 20$ and $p = 40$. The method of exact AIC and BIC are replaced by stepwise AIC/BIC procedure. Table 10 to 12 shows the simulation results.

We can see that the results in Table 10 to 12 our method can still give not bad estimates with smaller risk and error rate than the others. Then we say our method is effective.

Similarly, we are interested whether our L_0 approach works for approximating the exact AIC and BIC methods. Table 13 and Table 14 shows the proportion that our approach well approximates the results of AIC and BIC.

6.3 Data Analysis : Low Birth Weight Data

We apply our method to a low birth weight dataset of Hosmer and Lemeshow (1989). The data on 189 new born babies were collected at Baystate Medical Center, Springfield, MA during 1986. The data contains a binary response that indicates whether a new born baby has a low birth weight. The dataset also includes several risk factors associated with low birth weights, which are used as explanatory variables. We apply the following

Table 10: Performance of various methods for logistic regression with $p = 20$ and $n = 100$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	83.310 (8.767)	1.00	42.549 (6.512)	1.00
True model MLE	1.839 (0.209)	0.00	2.001 (0.309)	0.00
Stepwise AIC	75.063 (9.184)	0.98	33.391 (6.673)	0.98
Stepwise BIC	67.630 (9.637)	0.70	24.598 (6.801)	0.55
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	8.111 (1.527)	0.52	5.304 (0.595)	0.38
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	6.947 (1.757)	0.27	4.033 (0.519)	0.22
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	4.777 (1.390)	0.25	2.690 (0.389)	0.34
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	10.523 (1.966)	0.73	6.690 (0.403)	0.88
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	11.642 (2.470)	0.65	8.335 (1.611)	0.84
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	6.624 (0.400)	0.64	6.733 (0.305)	0.84
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	9.297 (1.976)	0.52	5.341 (0.594)	0.39

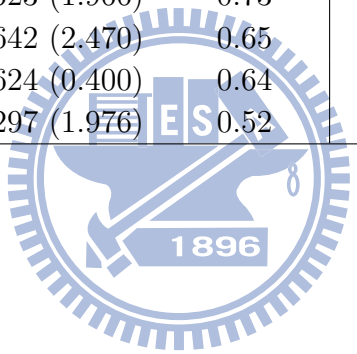


Table 11: Performance of various methods for logistic regression with $p = 20$ and $n = 200$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$		$\rho = 0.5$	
	KL risk	Error rate	KL risk	Error rate
Full model MLE	15.659 (0.801)	1.00	13.712 (0.334)	1.00
True model MLE	1.767 (0.127)	0.00	1.576 (0.090)	0.00
Stepwise AIC	9.683 (0.591)	0.93	8.632 (0.316)	0.96
Stepwise BIC	3.618 (0.243)	0.36	3.556 (0.217)	0.38
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	3.456 (0.239)	0.32	3.286 (0.202)	0.35
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	2.111 (0.164)	0.05	2.231 (0.169)	0.10
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	1.852 (0.128)	0.05	1.739 (0.093)	0.11
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	4.835 (0.224)	0.75	6.290 (0.319)	0.81
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	6.759 (0.182)	0.63	7.904 (0.275)	0.81
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	6.759 (0.182)	0.63	7.904 (0.275)	0.81
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	3.456 (0.239)	0.32	3.286 (0.202)	0.35

Table 12: Performance of various methods for logistic regression with $p = 40$ and $n = 200$, where values given in parentheses are the corresponding standard errors.

Method	$\rho = 0$			$\rho = 0.5$		
	KL risk		Error rate	KL risk		Error rate
Full model MLE	124.839	(9.943)	1.00	48.942	(2.483)	1.00
True model MLE	1.767	(0.127)	0.00	1.576	(0.090)	0.00
Stepwise AIC	93.931	(10.402)	1.00	26.552	(1.370)	1.00
Stepwise BIC	67.207	(10.908)	0.61	6.841	(0.472)	0.63
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.1)$	6.228	(0.695)	0.43	5.472	(0.402)	0.51
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.27)$	3.266	(0.600)	0.08	2.042	(0.157)	0.07
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 0.71)$	1.819	(0.127)	0.06	1.696	(0.094)	0.13
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 1.88)$	7.179	(0.298)	0.77	8.579	(0.382)	0.87
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, 5)$	9.340	(0.198)	0.55	10.947	(0.291)	0.78
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \infty)$	9.340	(0.198)	0.55	10.947	(0.291)	0.78
$\hat{\boldsymbol{\beta}}^{\infty}(\hat{\lambda}, \hat{\tau})$	6.228	(0.695)	0.43	5.472	(0.402)	0.51

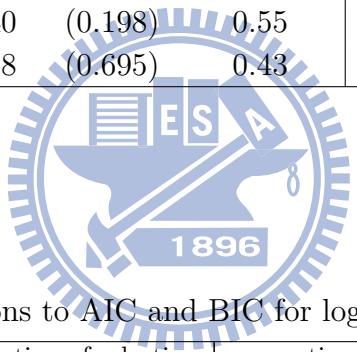


Table 13: Approximations to AIC and BIC for logistic regression with $p = 6$

		proportion of selecting the same model		proportion of having the same estimates		proportion of both the events occur	
		AIC	BIC	AIC	BIC	AIC	BIC
n=10	$\tau = 0.5$	0.23	0.67	0.11	0.58	0.11	0.58
	$\tau = 0.1$	0.50	0.17	0.50	0.09	0.50	0.09
	$\tau = 0.01$	0	0	0	0	0	0
	$\tau = 0.001$	0	0	0	0	0	0
n=200	$\tau = 0.5$	0.09	0.43	0.02	0.37	0.02	0.37
	$\tau = 0.1$	0.53	0.87	0.53	0.87	0.53	0.87
	$\tau = 0.01$	0	0	0	0	0	0
	$\tau = 0.001$	0	0	0	0	0	0
n=500	$\tau = 0.5$	0.01	0.14	0	0.12	0	0.12
	$\tau = 0.1$	0.93	0.99	0.56	0.99	0.56	0.99
	$\tau = 0.01$	0.26	0	0.24	0	0.24	0
	$\tau = 0.001$	0	0	0	0	0	0
n=1000	$\tau = 0.5$	0.01	0.09	0	0.09	0	0.09
	$\tau = 0.1$	0.54	0.97	0.29	0.97	0.29	0.97
	$\tau = 0.01$	0.44	0	0.44	0	0.44	0
	$\tau = 0.001$	0	0	0	0	0	0

Table 14: Approximations to AIC and BIC for logistic regression with $p = 10$

		proportion of selecting the same model		proportion of having the same estimates		proportion of both the events occur	
		AIC	BIC	AIC	BIC	AIC	BIC
n=10	$\tau = 0.5$	0.09	0.43	0	0.33	0	0.33
	$\tau = 0.1$	0.27	0.18	0.27	0.09	0.17	0.09
	$\tau = 0.01$	0	0	0	0	0	0
	$\tau = 0.001$	0	0	0	0	0	0
n=200	$\tau = 0.5$	0.01	0.17	0	0.16	0	0.16
	$\tau = 0.1$	0.25	0.86	0.25	0.86	0.25	0.86
	$\tau = 0.01$	0	0	0	0	0	0
	$\tau = 0.001$	0	0	0	0	0	0
n=500	$\tau = 0.5$	0	0	0	0	0	0
	$\tau = 0.1$	0.82	0.91	0.33	0.91	0.33	0.91
	$\tau = 0.01$	0.13	0	0.12	0	0.12	0
	$\tau = 0.001$	0	0	0	0	0	0
n=1000	$\tau = 0.5$	0	0.01	0	0.01	0	0.01
	$\tau = 0.1$	0.32	0.93	0.12	0.93	0.12	0.93
	$\tau = 0.01$	0.26	0	0.26	0	0.26	0
	$\tau = 0.001$	0	0	0	0	0	0

model:

$$low = \beta_0 + \beta_1 \times age + \beta_2 \times lwt + \beta_3 \times race : white + \beta_4 \times race : black$$
$$+ \beta_5 \times smoke + \beta_6 \times ht + \beta_7 \times ui + \beta_8 \times ftv + \beta_9 \times ptl,$$

where *age* and *lwt* are standardized to have mean 0 and variance 1. The details of these predictors are shown in Table 15 and the results of the selected model and the estimates are presented in Table 16. The standard deviations of the estimated coefficients are shown as "boot.std" in Table 16 using a parametric bootstrap method. We also show the standard deviations "glm.std" obtained based on the selected model using the R function "glm".

From the results shown in Table 16, we can see that AIC selected more variables than the others. Obviously, the variables *lwt* and *ptl* are important risk factors of this research. It is intuitive to see that the weight of the mothers and whether the mothers were in premature labors directly influence the results of having a low birth weight baby. One worth mentioning thing is that both AIC and BIC select the variable *ht* but PML does not. Note that there are only 12 people over the 189 observations having history of hypertension, so it is not easy to tell whether this variable is important. Another reason we can see from the "glm.std" and "boot.std". In contrast to the standard deviations

Table 15: Variables in the low birth weight dataset.

Name	Description
low	indicator of birth weight less than 2.5kg
age	mother's age in years
lwt	mother's weight in pounds at last menstrual period
race	mothers race ("white", "black", "other")
smoke	smoking status during pregnancy
ht	history of hypertension
ui	presence of uterine irritability
ftv	number of physician visits during the first trimester
ptl	number of previous premature labors

Table 16: Estimated parameters obtained from various methods.

Variable	PML		AIC			BIC		
$\hat{\lambda}^*$	0.0355							
$\hat{\tau}$	0.01							
	coeff	(boot.std)	coeff	(glm.std)	(boot.std)	coeff	(glm.std)	(boot.std)
intercept	-1.116	(0.260)	-1.211	(0.279)	(0.399)	-1.225	(0.200)	(0.257)
age	-0.322	(0.274)	0.000	(0.000)	(0.189)	0.000	(0.000)	(0.083)
lwt	-0.321	(0.278)	-0.431	(0.201)	(0.294)	-0.523	(0.207)	(0.341)
race:white	0.000	(0.069)	-1.011	(0.396)	(0.562)	0.000	(0.000)	(0.089)
race:black	0.000	(0.171)	0.000	(0.000)	(0.478)	0.000	(0.000)	(0.223)
smoke	0.000	(0.127)	0.931	(0.399)	(0.530)	0.000	(0.000)	(0.122)
ht	0.000	(0.254)	1.848	(0.705)	(1.123)	1.888	(0.720)	(2.200)
ui	0.000	(0.274)	0.739	(0.461)	(0.691)	0.000	(0.000)	(0.250)
ftv	0.000	(0.110)	0.000	(0.000)	(0.332)	0.000	(0.000)	(0.192)
ptl	1.523	(0.733)	1.119	(0.451)	(0.607)	1.407	(0.428)	(0.679)
Size of model	3		6			3		
Log-likelihood	-107.181		-99.309			-105.131		
BIC score	230.087		230.068			225.987		

obtained from parametric bootstrap, the standard deviations obtain by "glm" are much smaller, which are somewhat expected, since they do not take model selection into account and are expected to be underestimated.

7 Discussion

In the context of model selection under generalized linear models, we propose a PML method that enables simultaneous model selection and parameter estimation. Despite using a nonconvex penalty, the proposed estimates can be efficiently computed for high dimensional data thanks to difference convex programming and the coordinate descent algorithm. In addition, some numerical and theoretical results are provided to demonstrate the effectiveness of the proposed method.

References

[1] Akaike, H. (1973). Information theory and the maximum likelihood principle. In *International Symposium on Information Theory*. Edited by Petrov, V. and Csáki, F. Akademiai Kiádo, Budapest.

[2] Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, **19**, 716-713.

[3] An, H. L. T. and Tao, P. D. (1997). Solving a class of linearly constrained indefinite quadratic problems by D.C. algorithms. *Journal of Global Optimization*, **11**, 253-285.

[4] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression (with discussion). *The Annals of Statistics*, **32**, 407-499.

[5] Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties, *Journal of the American Statistical Association*, **96**, 1348-1360.

[6] Friedman, J., Haste, T., Hofling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, **1**, 302-332.

[7] Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Decent. *Journal of Statistic Software*, **33**.

[8] Hosmer, D.W., Lemeshow, S., (1989). Applied Logistic Regression. Wiley, New York

[9] Lange, K. (2004). *Optimization.* Springer, New York.

[10] Nishii, R. (1984). Asymptotic properties of criteria for selection of variables in multiple regression. *The Annals of Statistics*, **12**, 758-765.

[11] Osborne, M., Presnell, B., and Turlach, B. (2000). On the LASSO and its dual. *Journal of Computational and Graphical Statistics*, **9**, 319-337.

[12] Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annuals of Statistics*, **19**, 461-464.

[13] Shao, J. (1997). An asymptotic theory for linear model selection. *The Annals of Statistics*, **7**, 221-242.

[14] Shen, X. and Huang, H.-C. (2010). Grouping pursuit through a regularization solution surface. *Journal of the American Statistical Association*, **490**, 727-739.

[15] Shen, X., Pan, W., Zhu, Y., and Zhou, H. (2010). On L_0 regularization in high-dimensional regression, manuscript.

[16] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society, B*, **58**, 267-288.

[17] Zhao, P. and Yu, B. (2006). On model selection consistency of LASSO. *Journal of Machine Learning Research*, **7**, 2541-2567.

[18] Zhang, C. H. (2010). Nearly unbiased variable selection under minimax concave penalty, *The Annals of Statistics*, **38**, 894-942.

Appendices

A Data analysis code

Description

Fit a generalized linear model with truncated L_1 penalty.

Usage

glmTLP(y,x,lambda.min,maxtau,mintau,tol)

TLP-logistic(y,x,lambda.min,maxtau,mintau,tol)

Arguments

y	Response with binary output 0 and 1.
x	Input matrix with dimension $n \times p$ for n observations and p predictors.
lambda.min	Smallest value for lambda path.
maxtau	Largest thresholding parameter.
mintau	Smallest thresholding parameter.
tol	Convergence tolerance for the coordinate decent and iterated reweighted penalized least square.

```
# ============================================================================================================

TLP-logistic <- function(y,x,lambda.min,maxtau,mintau,tol,return=False)
{
    n <- nrow(x);   p <- ncol(x)
    up <- glmnet(x,y,family="binomial",thresh=1e-06,standardize=F,maxit=1e+6)$lambda[1]
    lambda <- 10^(c(0:100)*((log(up,10)-log(lambda.min,10))/100)+log(lambda.min,10))
    lambda <- sort(c(0,lambda),decreasing=T);   nlambda <- length(lambda)
    lasso <- glmnet(x,y,family="binomial",lambda=lambda,thresh=1e-06,standardize=F,maxit=1e+6)
    est.lasso <- cbind(as.vector(lasso$a0),t(as.matrix(lasso$beta)))
    tau <- c(10^(c(0:4)*((log(maxtau,10)-log(mintau,10))/4)+ log(mintau,10)),100);   q <- length(tau)
    est.table <- array(0,c(nlambda,(p+1),q));   est.ch <- array(0,c(q,(p+1)))
    case.lambda <- BIC.lambda <- rep(0,q);   score <- array(0,c(nlambda,q))

    for(j in 1:q)
    {
        est.table[,,j] <- lambda_path(y,x,est.lasso,lambda,tau[j],tol)
        CH <- lambda_choose_BIC(y,x,est.table[,,j])
        case <- CH$case; case.lambda[j] <- case;   score[,j] <- CH$BIC
        est.ch[j,] <- est.table[case,,j];   BIC.lambda[j] <- CH$min.BIC
    }
    case.tau <- which.min(BIC.lambda)
    est.tau <- est.ch[case.tau,];   BIC.tau <- BIC.lambda[case.tau]

    z <- list(est.tau=est.tau,BIC.tau=BIC.tau,est.lambda=est.ch,BIC.lambda=BIC.lambda,
            lambda=lambda[case.lambda[case.tau]],tau=tau[case.tau])
}

# ============================================================================================================
# Estimate the standard deviation of the coefficients using parametric bootstrap resampling

glmTLP <- function(y,x,lambda.min,maxtau,mintau,tol)
{
```

```
    p <- ncol(x);   n <- nrow(x);   k <- 100;   table <- array(0,c(k,p+1))
    temp <- TLP-logistic(y,x,lambda.min,maxtau,mintau,tol);   est <- temp$est.tau
    for(i in 1:k)
    {
        xx <- cbind(1,x)
        prob <- exp(xx %*% est)/(1+exp(xx %*% est))
        new.y <- rbinom(n,1,prob)
        table[i,] <- TLP-logistic(new.y,x,lambda.min,maxtau,mintau,tol)$est.tau
    }
    std <- apply(table,2,sd)
    list(estimates=est,std=std,BIC.score=temp$BIC.tau)
}

# ============================================================================================================
# Computing the estimates along lambda path for a fixed tau (logistic)

lambda_path <- function(Y,X,lasso,lambda.star,tau,tol)
{
    nlambda <- length(lambda.star);   p <- ncol(X);   table <- rep(0,(p+1));   initial <- lasso[1,]
    for(i in 1:nlambda)
    {
        est <- DiffConvProg(lasso[i,],initial,Y,X,lambda.star[i],tau,tol)$estimates
        table <- rbind(table,est);   initial <- est
    }
    table[(-1),]
}

# ============================================================================================================
# Choosing a suitable tuning parameter lambda with minimun BIC score (logistic)

lambda_choose_BIC <- function(Y,X,table)
{
    nlambda <- nrow(table);   n <- length(Y);   total <- rep(0,nlambda)
    for(i in 1:nlambda)
    {
        total[i] <- -2*Lfun(table[i,],Y,X) + log(n)*sum(table[i,-1]!=0)
    }
    case <- which.min(total)
    list(estimates=table[case,],case=case,min.BIC=total[case],BIC=total)
}

# ============================================================================================================
# Difference convex programing (logistic)

DiffConvProg <- function(estD,estQ,Y,X,lambda,tau,tol)
{
    p <- ncol(X);   ans <- estD;   check <- (-1);   k <- 0
    N.old <- (abs(ans[-1]) <= tau)
    while(check != p)
    {
        est <- Quapprox(ans,estQ,Y,X,lambda,tau,tol)$estimates
        N.new <- (abs(est[-1]) <= tau);   check <- sum(N.old == N.new)
        N.old <- N.new;   ans <- est;   k <- k+1
    }
    list(estimates=ans,times=k)
}

# ============================================================================================================
# Quadratic approximation of the log-likelihood function (logistic)

Quapprox <- function(estD,estQ,Y,X,lambda,tau,tol)
{
    check <- T;   k <- 0;   ans <- estQ
    while(check==T && k<250)
    {
        est <- CoorDecent(estD,ans,Y,X,lambda,tau,tol)$estimates
        check <- max(abs(ans-est))> tol ;   ans <- est;   k <- k+1
    }
```

```
    list(estimates=ans,times=k)
}

# ================================================================================
# Coordinate decent algorithm (logistic)

CoorDecent <- function(estD,estQ,Y,X,lambda,tau,tol)
{
    A <- cbind(1,X);   Xans <- XestQ <- A%*%estQ;   ph <- exp(XestQ)/(1+exp(XestQ))
    ph[XestQ >=100] <- 1;   ph[ph <= (1e-6)] <- 1e-6;   ph[ph>=(1-(1e-6))] <- 1-(1e-6)
    w <- as.vector(ph*(1-ph));   Z <- XestQ + (Y-ph)/w;   temp <- w*(A^2)
    check <- T;   k <- 0;   ans <- est <- estQ
    while(check==T && k<250)
    {
        Z0 <- Xans-ans[1];   est[1] <- sum(w*(Z-Z0))/sum(w)
        for(j in 2:(p+1))
        {
            Xans <- Xans-A[,j-1]*(ans[j-1]-est[j-1]);   Zj <- Xans-A[,j]*ans[j]
            est[j] <- soft((1/n)*sum(w*A[,j]*(Z-Zj)),lambda*(abs(estD[j])<=tau))/((1/n)*sum(temp[,j]))
        }
        Xans <- Xans-A[,p+1]*(ans[p+1]-est[p+1])
        check <- max(abs(ans-est))> tol;   k <- k+1;   ans <- est
    }
    list(estimates=ans,times=k)
}

# ================================================================================
# Soft-thresholding operator

soft <- function(z,r)
{
    ans <- 0
    if(z>0 && abs(z)>r) ans <- z-r
    if(z<0 && abs(z)>r) ans <- z+r
    ans
}

# ================================================================================
# Log-likelihood function (logsitic)

Lfun <- function(beta,Y,X)
{
    X <- cbind(1,X);   Xbeta <- X %*% beta
    sum(Y*(Xbeta)-log(1+exp(Xbeta)))
}

# ================================================================================
# Exac AIC & BIC : Exausted computation (logistic)

verify=function(Y,X)
{
    n <- length(Y);   p <- ncol(X);   index <- rep(c(T,F),each=2^(p-1))
    for(k in 1 :(p-1))
    {
        temp <- rep(rep(c(T,F),each=2^(p-1-k)),2^k)
        index <- rbind(index,temp)
    }
    index <- t(index);   total <- 2^p;   score.aic <- score.bic <- rep(0,total)
    for(i in 1:total)
    {
        subdata <- data.frame(Y)
        for(j in 1:p)
        {
            if(index[i,j]==T)   subdata <- data.frame(subdata,X[,j])
        }
        para <- glm(Y~.,subdata,family=binomial)$coeff
        subdata <- as.matrix(subdata)
        L <- Lfun(para,subdata[,1],subdata[,-1]);   check <- sum(para[-1]!=0)
```

```
        score.aic[i] <- -2*L+2*check;   score.bic[i] <- -2*L+log(n)*check
    }
    case.aic <- which.min(score.aic);   case.bic <- which.min(score.bic)
    subdata <- data.frame(Y)
    for(j in 1:p)
    {
        if(index[case.aic,j]==T)    subdata <- data.frame(subdata,X[,j])
    }
    tempA <- glm(Y~.,subdata,family=binomial);   temp.aic <- tempA$coeff;   est.aic <- temp.aic[1]
    i <- 1
    for(j in 1:p)
    {
        if(index[case.aic,j]==F)    {est.aic <- c(est.aic,0)}
        if(index[case.aic,j]==T)    {i=i+1; est.aic <- c(est.aic,temp.aic[i])}
    }
    subdata <- data.frame(Y)
    for(j in 1:p)
    {
        if(index[case.bic,j]==T)    subdata <- data.frame(subdata,X[,j])
    }
    tempB <- glm(Y~.,subdata,family=binomial);   temp.bic <- tempB$coeff;   est.bic <- temp.bic[1]
    i <- 1
    for(j in 1:p)
    {
        if(index[case.bic,j]==F)    {est.bic <- c(est.bic,0)}
        if(index[case.bic,j]==T)    {i <- i+1;  est.bic <- c(est.bic,temp.bic[i])}
    }
    list(aic.model=index[case.aic,],est.AIC=est.aic,bic.model=index[case.bic,],est.BIC=est.bic,
        coefficientA=(summary(tempA))$coeff,coefficientB=(summary(tempB))$coeff)
}
```

B Poisson simulation code

```
m <- 100;   n <- 50;   beta <- c(0,2,-1,rep(0,4))
tau.max <- 5;   tau.min <- 0.1;   tol <- 1e-6
tau <- c(10^(c(0:4)*((log(tau.max,10)-log(tau.min,10))/4)-1),100)
p <- length(beta)-1;   q <- length(tau);   nlambda <- 1022

time1 <- Sys.time()
num <- 1
est.mle <- est.true <- estchtau <- est.aic <- est.bic <- array(0,c(m,p+1))
estimate <- array(0,c(nlambda,(p+1),q,m))
estch <- array(0,c(q,p+1,m))
KL <- BIC.value <- array(0,c(nlambda,q,m))
KL.ch <- array(0,c(m,q))
KL.mle <- KL.true <- KL.aic <- KL.bic <- rep(0,m)
KL.chtau <- KL.aic <- KL.bic <- est.model <- rep(0,m)
case.tau <- value.tau <- rep(0,q)
while(num <= m)
{
    x <- matrix(rnorm(p*n,0,1),ncol=p);   mean <- y <- rep(0,n)
    xx <- cbind(1,x);   mean <- exp(xx %*% beta);   y <- rpois(n,mean)

    # MLE of the full model :
    data <- data.frame(y,x)
    fit <- glm(y~.,data,family=poisson);   est.mle[num,] <- fit$coeff
    trueMLE <- glm(y~X1+X2,data,family=poisson)$coeff;   est.true[num,] <- c(trueMLE,rep(0,p-2))
    KL.mle[num] <- KLloss(beta,est.mle[num,],y,x);   KL.true[num] <- KLloss(beta,est.true[num,],y,x)

    up <- glmnet(x,y,family="poisson",thresh=1e-06,standardize=F,maxit=1e+6)$lambda[1]
    lambda.star <- 10^(c(0:100)*((log(up,10)+3)/100)-3);   lambda.star <- sort(c(0,lambda.star),decreasing=T)
    lasso <- glmnet(x,y,family="poisson",lambda=lambda.star,thresh=1e-06,standardize=F,maxit=1e+6)
    est.lasso <- cbind(as.vector(lasso$a0),t(as.matrix(lasso$beta)))

    for(k in 1:q)
    {
```

```
        # Construct estimates and KL loss tables :
        estimate[,,k,num] <- lambda_path(y,x,est.lasso,lambda.star,tau[k],tol)
        KL[,k,num] <- loss(beta,estimate[,,k,num],y,x)

        # Choose lambda :
        CH <- lambda_choose_BIC(y,x,estimate[,,k,num])
        case <- CH$case;   case.tau[k] <- case
        BIC.value[,k,num] <- CH$BIC;   estch[k,,num] <- estimate[case,,k,num]
        value.tau[k] <- CH$min.BIC;   KL.ch[num,k] <- KL[case,k,num]
    }

    # Choose lambda and tau simultaneously
    h <- which.min(value.tau)
    estchtau[num,] <- estimate[case.tau[h],,h,num]
    KL.chtau[num] <- KL[case.tau[h],h,num]

    # Exact AIC and BIC :
    check <- verify(y,x)
    est.aic[num,] <- check$est.AIC;   KL.aic[num] <- KLloss(beta,check$est.AIC,y,x)
    est.bic[num,] <- check$est.BIC;   KL.bic[num] <- KLloss(beta,check$est.BIC,y,x)
    num <- num + 1
}
mle <- apply(est.mle,2,mean);   std.mle <- apply(est.mle,2,sd)
loss.mle <- mean(KL.mle);   sdloss.mle <- sd(KL.mle)
true.mle <- apply(est.mle,2,mean);  std.true <- apply(est.mle,2,sd)
loss.true <- mean(KL.true); sdloss.true <- sd(KL.true)
est <- apply(estimate,c(1,2,3),mean);   std.est <- apply(estimate,c(1,2,3),sd)
loss.est <- apply(KL,c(1,2),mean);  sdloss.est <- apply(KL,c(1,2),sd)
bic.value <- apply(BIC.value,c(1,2),mean)
est.ch <- apply(estch,c(1,2),mean); std.estch <- apply(estch,c(1,2),sd)
loss.estch <- apply(KL.ch,2,mean);  sdloss.estch <- apply(KL.ch,2,sd)
est.chtau <- apply(estchtau,2,mean);    std.chtau <- apply(estchtau,2,sd)
loss.chtau <- mean(KL.chtau)    ;sdloss.chtau <- sd(KL.chtau)
aic <- apply(est.aic,2,mean);   std.aic <- apply(est.aic,2,sd)
loss.aic <- mean(KL.aic);   sdloss.aic <- sd(KL.aic)
bic <- apply(est.bic,2,mean);   std.bic <- apply(est.bic,2,sd)
loss.bic <- mean(KL.bic);   sdloss.bic <- sd(KL.bic)
time2 <- Sys.time()
difftime(time2,time1,units="mins")

# ============================================================================================================
KL loss for a vector of estimates (poisson)

KLloss <- function(beta,est,Y,X)
{
    X <- cbind(1,X);   Xbeta <- X %*% beta;   Xest <- X %*% est
    mu <- exp(Xbeta);   muhat <- exp(Xest)
    sum(mu*(Xbeta-Xest)-(mu-muhat))
}

# ============================================================================================================
# KL loss for a matrix of estimates (poisson)

loss <- function(beta,table,Y,X)
{
    X <- cbind(1,X);   Xbeta <- X%*%beta;   Xtable <- X%*%t(table)
    mu <- exp(Xbeta);   muhat <- exp(Xtable);   D <- nrow(table);   loss <- rep(0,D)
    for(i in 1:D)
    {
        loss[i] <- sum(mu*(Xbeta-Xtable[,i])-(mu-muhat[,i]))
    }
    loss
}

# ============================================================================================================
# Computing the estimates along lambda path for a fixed tau (poisson)

lambda_path <- function(Y,X,lasso,lambda.star,tau,tol)
```

```
{
    nlambda <- length(lambda.star);   p <- length(beta)-1;   table <- rep(0,(p+1));   initial <- lasso[1,]
    for(i in 1:nlambda)
    {
        est <- DiffConvProg(lasso[i,],initial,Y,X,lambda.star[i],tau,tol)$estimates
        table <- rbind(table,est);   initial <- est
    }
    table[(-1),]
}

# ============================================================================================================
# Choosing a suitable tuning parameter lambda with minimun BIC score (poisson)

lambda_choose_BIC <- function(Y,X,table)
{
    nlambda <- nrow(table);   n <- length(Y);   total <- rep(0,nlambda)
    for(i in 1:nlambda)
    {
        total[i] <- -2*Lfun(table[i,],Y,X) + log(n)*sum(table[i,-1]!=0)
    }
    case <- which.min(total)
    list(estimates=table[case,],case=case,lambda.star=lambda.star[case],min.BIC=total[case],BIC=total)
}

# ============================================================================================================
# Difference convex programing (poisson)

DiffConvProg <- function(estD,estQ,Y,X,lambda,tau,tol)
{
    p <- ncol(X);   ans <- estD;   check <- (-1);   k <- 0
    N.old <- (abs(ans[-1]) <= tau)
    while(check != p)
    {
        est <- Quapprox(ans,estQ,Y,X,lambda,tau,tol)$estimates
        N.new <- (abs(est[-1]) <= tau);   check <- sum(N.old == N.new)
        N.old <- N.new;   ans <- est;   k <- k+1
    }
    list(estimates=ans,times=k)
}

# ============================================================================================================
# Quadratic approximation of the log-likelihood function (poisson)

Quapprox <- function(estD,estQ,Y,X,lambda,tau,tol)
{
    check <- T;   k <- 0;   ans <- estQ
    while(check==T)
    {
        est <- CoorDecent(estD,ans,Y,X,lambda,tau,tol)$estimates
        check <- max(abs(ans-est))> tol;   ans <- est;   k <- k+1
    }
    list(estimates=ans,times=k)
}

# ============================================================================================================
# Coordinate decent algorithm (poisson)

CoorDecent <- function(estD,estQ,Y,X,lambda,tau,tol)
{
    A <- cbind(1,X);   Xans <- XestQ <- A%*%estQ
    w <- muQ <- exp(XestQ);   w <- as.vector(w);   Z <- XestQ + (Y-muQ)/muQ;   temp <- w*(A^2)
    check <- T;   k <- 0;   ans <- est <- estQ
    while(check==T)
    {
        Z0 <- Xans-ans[1];   est[1] <- sum(w*(Z-Z0))/sum(w)
        for(j in 2:(p+1))
        {
            Xans <- Xans-A[,j-1]*(ans[j-1]-est[j-1]);   Zj <- Xans-A[,j]*ans[j]
```

```
            est[j] <- soft((1/n)*sum(w*A[,j]*(Z-Zj)),lambda*(abs(estD[j])<=tau))/((1/n)*sum(temp[,j]))
        }
        Xans <- Xans-A[,p+1]*(ans[p+1]-est[p+1])
        check <- max(abs(ans-est))> tol;   k <- k+1;   ans <- est
    }
    list(estimates=ans,times=k)
}

# ============================================================================================================
# Log-likelihood function (poisson)

Lfun <- function(beta,Y,X)
{
    X <- cbind(1,X);   Xbeta <- X %*% beta
    sum(-exp(Xbeta)+Y*(Xbeta))
}

# ============================================================================================================
# Exac AIC & BIC : Exausted computation (poisson)

verify=function(Y,X)
{
    n <- length(Y);   p <- ncol(X);   index <- rep(c(T,F),each=2^(p-1))
    for(k in 1 :(p-1))
    {
        temp <- rep(rep(c(T,F),each=2^(p-1-k)),2^k)
        index <- rbind(index,temp)
    }
    index <- t(index);   total <- 2^p;   score.aic <- score.bic <- rep(0,total)
    for(i in 1:total)
    {
        subdata <- data.frame(Y)
        for(j in 1:p)
        {
            if(index[i,j]==T)   subdata <- data.frame(subdata,X[,j])
        }
        para <- glm(Y~.,subdata,family=poisson)$coeff
        subdata <- as.matrix(subdata)
        L <- Lfun(para,subdata[,1],subdata[,-1]);   check <- sum(para[-1]!=0)
        score.aic[i] <- -2*L+2*check;   score.bic[i] <- -2*L+log(n)*check
    }
    case.aic <- which.min(score.aic);   case.bic <- which.min(score.bic)
    subdata <- data.frame(Y)
    for(j in 1:p)
    {
        if(index[case.aic,j]==T)    subdata <- data.frame(subdata,X[,j])
    }
    temp.aic <- glm(Y~.,subdata,family=poisson)$coeff;   est.aic <- temp.aic[1]
    i <- 1
    for(j in 1:p)
    {
        if(index[case.aic,j]==F)    {est.aic <- c(est.aic,0)}
        if(index[case.aic,j]==T)    {i=i+1; est.aic <- c(est.aic,temp.aic[i])}
    }
    subdata <- data.frame(Y)
    for(j in 1:p)
    {
        if(index[case.bic,j]==T)    subdata <- data.frame(subdata,X[,j])
    }
    temp.bic <- glm(Y~.,subdata,family=poisson)$coeff;   est.bic <- temp.bic[1]
    i <- 1
    for(j in 1:p)
    {
        if(index[case.bic,j]==F)    {est.bic <- c(est.bic,0)}
        if(index[case.bic,j]==T)    {i <- i+1;  est.bic <- c(est.bic,temp.bic[i])}
    }
    list(aic.model=index[case.aic,],est.AIC=est.aic,bic.model=index[case.bic,],est.BIC=est.bic)
}
```

C Logistic simulation code

```
m <- 100;   n <- 100;   beta <- c(0,3,-2,rep(0,4))
tau.max <- 5;   tau.min <- 0.1;   tol <- 1e-6
tau <- c(10^(c(0:4)*((log(tau.max,10)-log(tau.min,10))/4)-1),100)
p <- length(beta)-1;   q <- length(tau);   nlambda <- 102

time1 <- Sys.time()
num <- 1
est.mle <- est.true <- estchtau <- est.aic <- est.bic <- array(0,c(m,p+1))
estimate <- array(0,c(nlambda,(p+1),q,m))
estch <- array(0,c(q,p+1,m))
KL <- BIC.value <- array(0,c(nlambda,q,m))
KL.ch <- array(0,c(m,q))
KL.mle <- KL.true <- KL.aic <- KL.bic <- rep(0,m)
KL.chtau <- KL.aic <- KL.bic <- est.model <- rep(0,m)
case.tau <- value.tau <- rep(0,q)
while(num <= m)
{
    x <- matrix(rnorm(p*n,0,1),ncol=p);   prob <- y <- rep(0,n)
    xx <- cbind(1,x);   prob <- exp(xx %*% beta)/(1+exp(xx %*% beta))
    y <- rbinom(n,1,prob)

    # MLE of the full model :
    data <- data.frame(y,x)
    fit <- glm(y~.,data,family=binomial);   est.mle[num,] <- fit$coeff
    trueMLE <- glm(y~X1+X2,data,family=binomial)$coeff;   est.true[num,] <- c(trueMLE,rep(0,p-2))
    KL.mle[num] <- KLloss(beta,est.mle[num,],y,x);   KL.true[num] <- KLloss(beta,est.true[num,],y,x)

    up <- glmnet(x,y,family="binomial",thresh=1e-06,standardize=F,maxit=1e+6)$lambda[1]
    lambda.star <- 10^(c(0:100)*((log(up,10)+3)/100)-3);   lambda.star <- sort(c(0,lambda.star),decreasing=T)
    lasso <- glmnet(x,y,family="binomial",lambda=lambda.star,thresh=1e-06,standardize=F,maxit=1e+6)
    est.lasso <- cbind(as.vector(lasso$a0),t(as.matrix(lasso$beta)))

    for(k in 1:q)
    {
        # Construct estimates and KL loss tables :
        estimate[,,k,num] <- lambda_path(y,x,est.lasso,lambda.star,tau[k],tol)
        KL[,k,num] <- loss(beta,estimate[,,k,num],y,x)

        # Choose lambda :
        CH <- lambda_choose_BIC(y,x,estimate[,,k,num])
        case <- CH$case;   case.tau[k] <- case
        BIC.value[,k,num] <- CH$BIC;   estch[k,,num] <- estimate[case,,k,num]
        value.tau[k] <- CH$min.BIC;   KL.ch[num,k] <- KL[case,k,num]
    }

    # Choose lambda and tau simultaneously
    h <- which.min(value.tau)
    estchtau[num,] <- estimate[case.tau[h],,h,num]
    KL.chtau[num] <- KL[case.tau[h],h,num]
    est.model[num] <- sum((estchtau[num,-1]!=0)==(beta[-1]!=0))==p

    # Exact AIC and BIC :
    check <- verify(y,x)
    est.aic[num,] <- check$est.AIC;   KL.aic[num] <- KLloss(beta,check$est.AIC,y,x)
    est.bic[num,] <- check$est.BIC;   KL.bic[num] <- KLloss(beta,check$est.BIC,y,x)
    num <- num + 1
}
mle <- apply(est.mle,2,mean);   std.mle <- apply(est.mle,2,sd)
loss.mle <- mean(KL.mle);   sdloss.mle <- sd(KL.mle)
true.mle <- apply(est.mle,2,mean);  std.true <- apply(est.mle,2,sd)
loss.true <- mean(KL.true); sdloss.true <- sd(KL.true)
est <- apply(estimate,c(1,2,3),mean);   std.est <- apply(estimate,c(1,2,3),sd)
loss.est <- apply(KL,c(1,2),mean);  sdloss.est <- apply(KL,c(1,2),sd)
bic.value <- apply(BIC.value,c(1,2),mean)
est.ch <- apply(estch,c(1,2),mean); std.estch <- apply(estch,c(1,2),sd)
loss.estch <- apply(KL.ch,2,mean);  sdloss.estch <- apply(KL.ch,2,sd)
```

```
est.chtau <- apply(estchtau,2,mean);    std.chtau <- apply(estchtau,2,sd)
loss.chtau <- mean(KL.chtau)    ;sdloss.chtau <- sd(KL.chtau)
aic <- apply(est.aic,2,mean);   std.aic <- apply(est.aic,2,sd)
loss.aic <- mean(KL.aic);   sdloss.aic <- sd(KL.aic)
bic <- apply(est.bic,2,mean);   std.bic <- apply(est.bic,2,sd)
loss.bic <- mean(KL.bic);   sdloss.bic <- sd(KL.bic)
time2 <- Sys.time()
difftime(time2,time1,units="mins")

# ============================================================================================================
# KL loss for a vector of estimates (logistic)

KLloss <- function(beta,est,Y,X)
{
    X <- cbind(1,X);   Xbeta <- X %*% beta;   Xest <- X %*% est
    P <- exp(Xbeta)/(1+exp(Xbeta));   Ph <- exp(Xest)/(1+exp(Xest))
    Ph[Xest >=100] <- 1;   Ph[Ph <= (1e-6)] <- 1e-6;   Ph[Ph>=(1-(1e-6))] <- 1-(1e-6)
    sum(P*log(P/Ph)+(1-P)*log((1-P)/(1-Ph)))
}

# ============================================================================================================
# KL loss for a matrix of estimates (logistic)

loss <- function(beta,table,Y,X)
{
    nlambda <- nrow(table);   loss <- rep(0,nlambda)
    X <- cbind(1,X);   Xbeta <- X %*% beta;   Xest <- table %*% t(X)
    P <- exp(Xbeta)/(1+exp(Xbeta))
    for(i in 1:nlambda)
    {
        Ph <- exp(Xest[i,])/(1+exp(Xest[i,]))
        Ph[Xest >=100] <- 1;   Ph[Ph <= (1e-6)] <- 1e-6;   Ph[Ph>=(1-(1e-6))] <- 1-(1e-6)
        loss[i] <- sum(P*log(P/Ph)+(1-P)*log((1-P)/(1-Ph)))
    }
    loss
}
```