May 15, 2018

Abstract

In machine learning and statistics, classification is one of the most common supervised learning methods considered. In this dissertation, we analyze the Singapore Eye Dataset which contains around 2700 samples of 300 predictors each. These predictors include blood data, eye data, body index, and a binary categorical variable heart disease. We expect to get a simple and practical model to predict whether a person suffers heart disease with the help of some predictors.

Due to the efficiency in clinical medicine, we use Logisitic Model as the classifier. In order to achieve feature selection, some Regularization Methods have been combined to the original model: LASSO (least absolute shrinkage and selection operator), Group LASSO (which select important factors rather than variables in lasso) and Sparse Group LASSO (SGL, which is sparse at both the group and individual feature levels).

Next, we tune the hyperparameter in these penalized models. Some well-known tuning criteria are implemented: Cross Validation (CV) with deviance or misclassification error as the loss function, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) and Extended Bayesian Information Criterion (EBIC, which is more parsimonious).

Finally, we try to measure goodness of fit of the models with some index: Correct Classification Rate (CCR), Area Under receiver operating characteristic Curve (AUC) and Polytomous Discrimination Index (PDI) for single model and Net Reclassification Improvement (NRI) for comparison.

After applying all these methods to the dataset, we assert that SGL with EBIC parameter tuning gives the best model containing 8 important predictors and achieves AUC 0.84 on test data.

Contents

1 Backgroud **3**
- 1.1 Classification 3
- 1.2 Feature Selection 4
- 1.3 Hyper Parameter Selection 5
- 1.4 Model Evaluation 6

2 Regularization Family **6**
- 2.1 Statistical Property 6

2.1.1 LASSO . . . 6
2.1.2 Group LASSO . . . 10
2.1.3 Sparse Group LASSO . . . 11
2.2 Solution with Normal Distribution . . . 12
2.2.1 LASSO . . . 13
2.2.2 Group LASSO . . . 14
2.2.3 Sparse Group LASSO . . . 14
2.3 Solution with Binomial Distribution . . . 15
2.3.1 LASSO . . . 16
2.3.2 Group LASSO . . . 17
2.3.3 Sparse Group LASSO . . . 17

3 Selecting Tuning Parameter 18
3.1 CV . . . 19
3.2 IC family . . . 19
3.2.1 Akaike Information Criterion . . . 20
3.2.2 Bayesian Information Criterion . . . 20
3.2.3 Extended Bayesian Information Criterion . . . 20
3.2.4 Degree of Freedom . . . 21

4 Other Models 23
4.1 Multiply Layer Perceptron . . . 23

5 Final Model Evaluation 26
5.1 Single Model Evaluation . . . 26
5.1.1 Correct Classification Rate . . . 26
5.1.2 Area Under ROC Curve . . . 27
5.1.3 Polytomous Discrimination Index . . . 27
5.2 Model Comparison . . . 28
5.2.1 Net Reclassification Improvement . . . 29

6 Software 29
6.1 Models . . . 29
6.2 Hyper Parameter Tuning . . . 30
6.3 Model Evaluation . . . 30

7 Singapore Eye Dataset 30
7.1 Preprosessing . . . 31
7.2 Training and Test Set . . . 31
7.3 LASSO . . . 32
7.3.1 Best lambda through CV: dev . . . 33
7.3.2 Best lambda through CV: ME . . . 34
7.3.3 Best lambda through BIC . . . 35
7.3.4 Best lambda through EBIC . . . 36
7.4 Group LASSO . . . 37
7.4.1 Best lambda through CV: dev . . . 38

7.4.2 Best lambda through CV: ME 39
7.4.3 Best lambda through BIC 40
7.4.4 Best lambda through EBIC 41
7.5 Sparse Group LASSO 42
7.5.1 Best lambda through CV: dev 43
7.5.2 Best lambda through CV: ME 44
7.5.3 Best lambda through BIC 45
7.5.4 Best lambda through EBIC 46
7.6 Comparison 48
7.6.1 LASSO Based Models 48
7.6.2 Other Models 50
7.6.3 Important Predictors 52

8 Further Discussion **54**

1 Backgroud

1.1 Classification

In machine learning and statistics, classification is one of the most common supervised learning methods taken into account. In general, we have many samples of data whose category memberships are known and want to identify which category a new observation belongs to. Each sample is more than a single number and, for instance, a multi-dimensional entry, which is said to have several predictors or features. There are many commonly-used classifiers such as Logistic Regression [1], Naive Bayes [2], Linear Discriminant Analysis (LDA) [3], K-Nearest Neighbours (KNN) [4], Support Vector Machine (SVM) [5], Classification and Decision Tree (CADT) [6] and newly deep learning methods [7].

All these classifiers have both pros and cons, and it is impossible to assert that one of them always works superior to the others. There is no supreme method but a most suitable one for a specific question, which I assert, to some extent, is the charm of machine learning. Since we explore on the Singapore Eye Dataset, we expect other properties of the final classifier:

- Writable model. We desire to get a writable equation of the predictors and dependent variable.
- Simple model. The model should be simple enough for practical and clinical use, which means the predictors should be as sparse as possible.
- Time efficient. Due to so many observations, we hope our classifier is fast.

After we consider these properties, some classifiers such as deep learning method may not be adopted since the hardness to interpret. On that state, Logistic Regression gives a clear relationship between predictors and output.

1.2 Feature Selection

There is a "sparsity hypothesis" [8] goes, in a huge amount of predictors, barely a few of them are relevant to dependent variable. Actually it resembles the renowned Occam's razor rule [9]: given candidate models of similar predictive or explanatory power, the simplest model is most likely to be the best choice also implies. This hypothesis has many advantages. First, it reduce predictors which are actually noises and thus increase the precision. Second, the model is simpler and medical workers can make decision based on it more proficient. Even if the true model is not sparse, we may still be inspired by the mechanism behind it. Third, it is time-saving, which is a precious property in large data analysis. Fourth, it can prevent overfitting, that is the production of an analysis that corresponds too closely to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably. An overfitting model has more predictors than can be justified by the data and consequently has higher variance.

We have several wonderful measures to achieve feature selection.

- Filter Methods. For univariate feature selection, we examine each feature individually to determine the strength of the relationship of the feature with the response variable. The strength can be balanced by Pearson Correlation Coefficient [10] or Distance Correlation.
- Wrapper Methods. They generate different subsets of features, each subset is subsequently used to build a model and train the learning algorithm. The best subset is selected by testing the algorithm. For instance, Forward and Backward selection [11] is one of the representatives.
- Embedded Methods. They utilize machine learning models bound with feature selection methods. For example, get most relevant variables with the help of tree-based bagging algorithm such as random forest [12]. It provides two straightforward methods for feature selection: mean decrease impurity and mean decrease accuracy [13].

In this dissertation, we focus on the third approach by using Regularization Method [14] to select features. The general form is the convex optimization problem:

$$\min_{\beta} \sum_{i=1}^{n} L\left(y_i, f(\mathbf{x_i}, \beta)\right) + \lambda\phi(\beta) \tag{1}$$

Here,

- $D = \{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i=1}}^{\mathbf{n}}, \mathbf{x_i} = (\mathbf{x_{i1}}, ..., \mathbf{x_{ip}})^{\mathbf{T}}$ is the dataset.
- β is the coefficients in the model.
- $f(\mathbf{x_i}, \beta)$ is the estimator of y_i.
- L is the loss (cost) function, which measures the difference between y_i and its estimator. It is defined by specific machine learning models.

- ϕ is the penalty function (regularization term). Its value is proportional to the complexity of the model.
- $\lambda \geq 0$ is the tuning parameter. It is a hyper parameter, which should be determined before fitting the model.

Notice that regularization method is a trade-off between goodness of fit and model complexity. If $\lambda = 0$, we don't penalize at all, the model will turn out to be too sophisticated; however, if $\lambda = \infty$, no predictors will be chosen.

Tibshirani has proposed LASSO (least absolute shrinkage and selection operator) [15] which is a Regularization Methods. LASSO can shrink some of the regression coefficients to zero by imposing a penalty on their size. However, variable selection typically amounts to the selection of important factors (groups of variables) rather than individual derived variables, M. Yuan proposed Group LASSO [16] to make the model more explainable. Friedman has improved Group LASSO into a more generalized one: Sparse Group LASSO (SGL) [17], thus variables selected are sparse at both the group and individual feature levels. M. Vincent applied Sparse Group LASSO into Multinomial Logistic Regression [18] and made a R package msgl (High Dimensional Multiclass Classification Using Sparse Group Lasso) to realize it. Krishnapuram has developed fast Algorithms to Sparse Multinomial Logistic Regression [19]. To summarize:

- Ridge [20]: $\phi(\beta) = |\beta|_2$
- LASSO: $\phi(\beta) = |\beta|_1$
- Elastic net [21]: $\phi(\beta) = |\beta|_1 + \lambda_1|\beta|_2$
- Group LASSO (GL): $\phi(\beta) = \sum_{j=1}^{J} \sqrt{p_j}|\beta_{G_j}|_2$

  Here we divide {1...m} into J group $\{G_1, ..., G_J\}$. $p_j \triangleq \#\{G_j\}$. $\beta_{G_j} = (\beta_k)_{k \in G_j}$
- Sparse Group LASSO (SGL): $\phi(\beta) = \{(1-\alpha)\sum_{j=1}^{J} \sqrt{p_j}|\beta_{G_j}|_2 + \alpha|\beta|_1\}$

  Here $0 \leq \alpha \leq 1$

1.3 Hyper Parameter Selection

Next, we select best model, that is, we try to choose the best λ based on some criteria. It is indeed a sort of art for hyper parameter tuning. Different λ may result in totally divergent model, predictors selected and final precision. Some criteria:

- Cross validation (CV) [22]
- Akaike Information Criterion (AIC) [23]
- Bayesian Information Criterion (BIC) [24]
- Extended Bayesian Information Criterion (EBIC) [25]

1.4 Model Evaluation

Finally, we try to measure goodness of fit of the model. Since this is a classifier, we may simply use Correct Classification Rate (CCR) or samely, Accuracy (ACC) and Correct Classification Probability (CCP) [26] to the test samples after we split the whole dataset into training and test part and fit the classifier to the training subset. Nevertheless, because our dataset is highly unbalanced, CCR is meaningless. We use Area Under receiver operating characteristic Curve (AUC) [27] or samely, Polytomous Discrimination Index(PDI) [28] and Hypervolume Under Manifold (HUM) [29] in two class classifier instead to represent precision.

We use Net Reclassification Improvement (NRI) [30] as the improvement of nested models.

2 Regularization Family

2.1 Statistical Property

2.1.1 LASSO

Ordinary least square (OLS) estimators just minimize the Loss function, without any penalization at all. Hence, we are likely to run into a stone wall. OLS estimators may overfit the result, and may fit badly provided that the predictors have multicollinearity.

Two common solutions are ridge regression and step-wise best subset selection. However, step-wise best subset selection is a discrete process, which is too extreme, and can thus be turbulent and sensible to the dataset. Ridge regression can only set coefficients to nearly zero, but not directly zero.

LASSO can solve both of the disadvantages. It tends to produce some coefficients to zero so as to realize feature selection and it is a continuous method.

Definition LASSO:

$$\beta^L = \arg\min_{\beta} \{\text{LOSS} + \lambda|\beta|_1\} \tag{2}$$

Standardization We should standardize dataset before running the LASSO algorithm: $\sum_i x_{ij}/n = 0$; $\sum_i x_{ij}^2/n = 1$.

Since the interception is not penalized at all, we can just omit it for explicit notation by centering output y: $y_i = y_i - \overline{y}$.

Comparison with Ridge The reason why LASSO can cause some coefficients to zero while ridge can't is that one-norm penalty has non-differentiable point $\beta_j = 0$. For detail, we suppose the design matrix X is orthonormal,$X^T X =$

I and we use least square loss function. LASSO tries to minimize:

$$\beta^L = \arg\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \mathbf{x_i}^\mathbf{T} \beta)^\mathbf{2} + \lambda |\beta|_\mathbf{1} \right\} \tag{3}$$

while Ridge tries to minimize:

$$\beta^R = \arg\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \mathbf{x_i}^\mathbf{T} \beta)^\mathbf{2} + \lambda |\beta|_\mathbf{2} \right\} \tag{4}$$

The explicit expression for LASSO:

$$\beta_j^L = s[\hat{\beta}_j, \lambda] \triangleq sign(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+ \tag{5}$$

here, s is called soft-thresholding operator, which means that $\beta_j^L = 0, \quad if\ -\lambda \leq \hat{\beta}_j \leq \lambda$. $\hat{\beta}_j$ is the OLS of residual, fixing $\beta_k,\ k \neq j$.

The derivation of the equation will be stated as follow:

1. Separate β into β_j and others and we just focus on β_j. Hence, this is why we get the OLS of residual fixing others.

2. Use subgradient [31] method. Note that there is a non-differentiable point $\beta_j = 0$, thus ordinary gradient method becomes invalid. Subgradient method returns the same result at the differentiable point, and returns a interval of left and right derivatives at the non-differentiable point. If $\beta_j = 0$ falls in the interval, we regard that it is the extreme point. Hence, this is why there is a soft-thresholding operator.

If we regard this equation as a constraint optimization problem, and write down the KarushKuhnTucker (KKT) [32] optimality conditions, we may receive same expression. Pay attention to see that we get the solution only when the design matrix is orthonormal.

The explicit expression for Ridge:

$$\beta_j^R = \frac{1}{1+\lambda} \hat{\beta}_j \tag{6}$$

Ridge estimator actually scales the OLS estimators by a constant factor which is inverse ratio to λ. Hence it may never be zero at all. At the other hand, LASSO estimator has a period of zero and is a soft shrinkage method. LASSO is somewhat indifferent to very correlated predictors, and will tend to pick one and ignore the rest, which expects many coefficients to be close to zero, and a small subset to be larger and nonzero. Figure1 is a famous insight for the case $p = 2$.

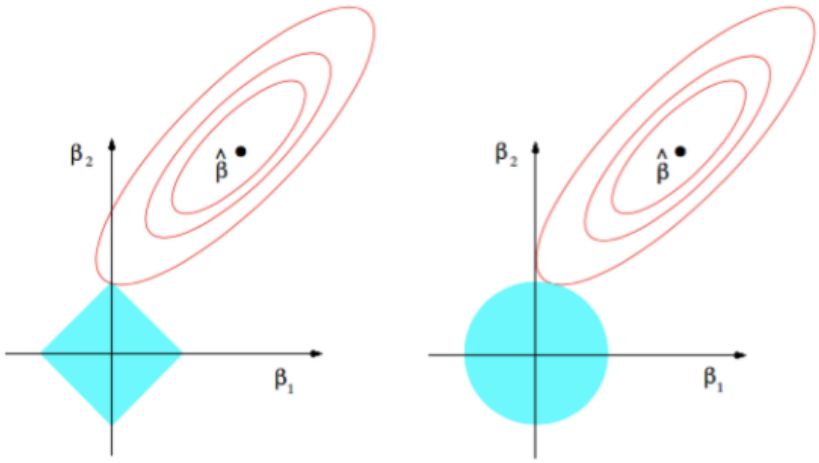


Figure 1: Estimation picture for LASSO and Ridge when dimension is 2. Black point is the OLS estimator. Ellipses are the contours of LOSS functions which are centered at the OLS estimates. The constraint regions are the feasible regions. Note that in LASSO, the optimum parameter β_2 is zero.

In Figure 1, These ellipses are the contours of LOSS functions which are centered at the OLS estimates. The constraint regions are the feasible regions if we regard both algorithms as convex optimization problems and use Lagrange duality [33]. Thus, LASSO can be equivalent to a optimization problem and there is a one-to-one match between t and λ in equation 7 .

$$\beta^L = \arg\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - \mathbf{x_i}^{\mathbf{T}} \beta)^{\mathbf{2}} \right\} \quad subject\ to |\beta|_1 \leq t \tag{7}$$

Due to the rotated square in LASSO, sometimes the contours touch the corner of it, corresponding to a zero. However, there is no probability for a circle.

Standard error Tibshirani discussed about the statistical inference of the LASSO estimators. Since they are non-linear and non-differentiable, we are difficult to calculate the accurate standard error. There are two approaches. First is via the bootstrap. However, it may be difficult since selected variables would change from one sample to the other. An alternative is to regard one-norm penalty as a weighted two-norm: $\sum_j \frac{1}{|\beta_j|} \beta_j^2$. The ridge estimators β^R is easy to approximate with a diagonal matrix with elements $|\beta_j^L|$. Hence, we may get the standard error by assuming output y is normal. If σ^2 is unknown, we can replace it with its estimates from the full model. Note that standard error of zero-coefficient is zero.

In the numerical experiment, we won't calculate the standard error. The reason for this is that standard errors are not very meaningful for strongly biased estimates such as LASSO, arised from penalized estimation methods.

Hyper parameter λ λ balances the trade-off between goodness of fit and model complexity. If $\lambda = 0$, we don't penalize at all, the model will turn out to be too sophisticated; however, if $\lambda = \infty$, no predictors will be chosen. The bigger the value is, the higher bias and the lower variance the model fits. We leave sophisticated discussion about how to choose it in the next section.

Bayes approach If we regard LASSO as a one-norm penalty term to the residual sum of squares, we can consider the formula as Bayes posterior mode with a Laplace prior [34] while Ridge as a Normal prior.

Maximum Likelihood Estimate (MLE) often fails in ill-conditioning data ("large p small n" for example), resulting a lack of convergence, large coefficient variance and overfitting.

Maximum a posteriori (MAP) is proportional to the likelihood times the prior. If the prior arises from a Laplace (double exponential) distribution [35], we could prove that the posteriori distribution is equal to the residual sum of square plus one-norm. More precisely, let $\beta_j \sim \text{Laplace}(0, \frac{1}{\lambda})$, i.i.d, the probability density function is $f(\beta_j|\lambda) = \frac{\lambda}{2}e^{-\lambda|\beta_j|}$.

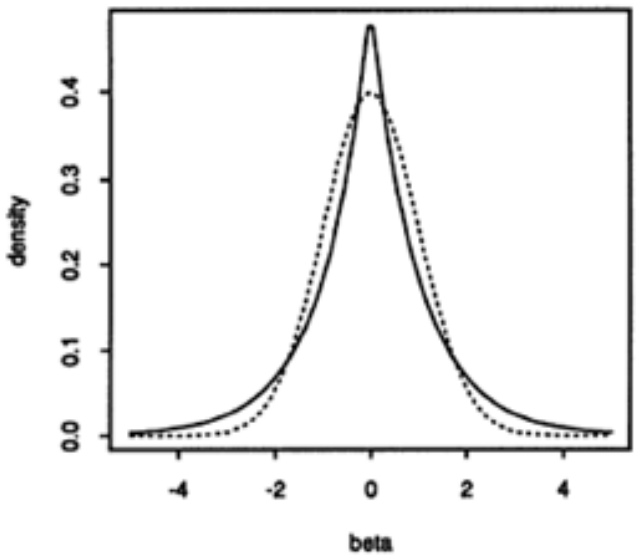


Figure 2: Laplace density in solid line and Normal density in slash.

Figure 2 shows these two distributions. Note Laplace distribution puts more mass near zero and in the tails, which reflects the greater tendency of LASSO

to produce coefficients either zero or large.

Model evaluation Ordinary, after we fit a model, we manipulate some statistics for goodness of fit (R^2 for example) and residual analysis to check whether the model accord with assumptions ($\epsilon \sim Normal$ in OLR for example). However, in LASSO, we do not implement these analysis. There are no distributional assumptions on ϵ ,thus it does not make any sense to analysis them the way we analysis in OLS (normality tests, heteroscedasticity, Durbin-Watson, etc). Furthermore, nothing is postulate on conditional distribution $(y|X)$. We calculate CCR and AUC value on the test data as goodness of fit which will be discussed in the following section.

Application to other models LASSO can be used in numerous other models. In this dissertation, I will apply LASSO to binary logistic regression model. Nearly all the Generalized Linear Model (GLM) [36] can be used similarly.

More general, any machine learning model which has a LOSS function could combine LASSO. However, usually we consider a quadratic approximation of LOSS function to accelerate speed. Although convergence may not be satisfied, in real experiment, it does quite a great job.

Also, we could apply LASSO to tree-based model. Note that ordinary Classification And Regression Tree (CART) is a two step algorithm. First we grow a total large tree by maximize Gini Impurity [37] each time, which is greedy. Then we prune it with penalty of the number of leaves. Now, we use LASSO idea to shrink the large tree. Parameter is the mean contrasts at each node [38].

Shortcoming Predictors should not be too correlated, or else, LASSO tends to choose one of them and transform the others to zeros. Hence, the result and predictors chosen are not stable. There are several ways to solve this shortcoming.

- Elastic Net penalty. Elastic net is the combination of LASSO and Ridge. Since as we all know, Ridge regression can handle multicollinearity.
- Use principal components regression [39]. However, it can not actually realize variable shrinkage.
- Group LASSO, which will be discussed as follow.

2.1.2 Group LASSO

Usually, we have both continuous and categorical variables in our dataset and we use one-hot-encode to turn categorical variables into dummy variables. However, LASSO tends to select the important individual predictors rather than important factors, which is a group of variables. To state it in another way, it tends to make selection based on the strength of individual variable rather than the strength of groups of input variables, often resulting in selecting more

factors than necessary. Yuan and Lin has proposed Group LASSO, which is a extension of LASSO in factor selection problems.

Definition We divide {1...p} into J group $\{G_1, ..., G_J\}$. $\#\{G_j\} \triangleq p_j$. $\beta_{G_j} = (\beta_k)_{k \in G_j}$. Group LASSO tries to minimize:

$$\beta^{GL} = \arg\min_{\beta} \left\{ \text{LOSS} + \lambda \sum_{\text{j=1}}^{\text{J}} \sqrt{\text{p}_\text{j}} |\beta_{\text{G}_\text{j}}|_2 \right\} \tag{8}$$

The penalized weight p_j is just for a historical reason and is not a must. It penalizes a group with more predictors more heavily.

Comparison with LASSO Group LASSO is an intermediate between LASSO and Ridge. Note that if $p_1 = ... = p_J = 1$, it is LASSO; if $J = 1$, it is Ridge. Consider a case where there are two factors: a bi-dimension vector $\beta_1 = (\beta_{11}, \beta_{12})^T$ and a scalar β_2. We combine β_{11}, β_{12} in a group and β_2 another group. Samely as before, we can draw the feasible region when regarding group LASSO as a optimization problem. The group LASSO encourages sparsity at the factor level.

Suppose the design matrix X is orthonormal, we can get explicit solution for Group LASSO almost the same as LASSO.

$$\beta_{G_j}^{GL} = s[\hat{\beta}_{G_j}, \lambda\sqrt{p_j}] \tag{9}$$

s is soft-thresholding operator defined before. $\hat{\beta}_{G_j}$ is the OLS of residual, fixing $\beta_{G_k},\ k \neq j$.

shortcoming The group lasso does not, however, yield sparsity within a group. If a group of parameters is non-zero, they will all be non-zero, which may select too many predictors.

2.1.3 Sparse Group LASSO

Hastie and Tibshirani consider a more general penalty blends LASSO and Group LASSO, which yields solutions that are sparse at both the group and individual feature levels. It is called Sparse Group LASSO (SGL).

Definition We divide {1...p} into J group $\{G_1, ..., G_J\}$. $\#\{G_j\} \triangleq p_j$. $\beta_{G_j} = (\beta_k)_{k \in G_j}$. Sparse Group LASSO tries to minimize:

$$\beta^{SGL} = \arg\min_{\beta} \left\{ \text{LOSS} + \lambda \left[(1-\alpha) \sum_{\text{j=1}}^{\text{J}} \sqrt{\text{p}_\text{j}} |\beta_{\text{G}_\text{j}}|_2 + \alpha|\beta|_1 \right] \right\} \tag{10}$$

This expression is the sum of convex functions and is therefore convex. $0 \leq \alpha \leq 1$.

Comparison with GL & LASSO Sparse Group LASSO is an intermediate between LASSO and GL. Note that if $\alpha = 0$, it is GL; if $\alpha = 1$, it is LASSO. Consider two predictors β_1 and β_2, suppose we relegate them to a single group. Figure 3 draws contour lines of the penalty for GL: a circle, which has the biggest area; LASSO: a rotated square with the smallest; and SGL: an area between them, that contains LASSO and be contained by GL.

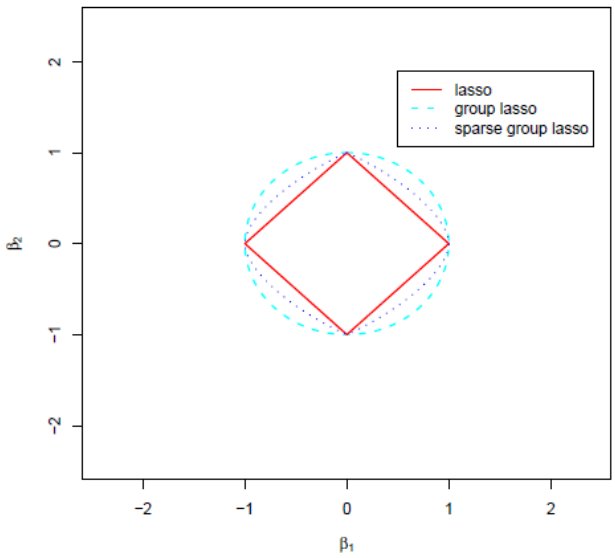


Figure 3: Feasible regions of LASSO, GL and SGL penalties when dimension is two.

If design matrix X is orthogonal, we may get explicit result.

α is a hyper parameter to balance LASSO and GL. Models with a low value have a larger number of non-zero parameters (tends to GL) than models with a high value. The author suggests $\alpha = 0.25, 0.5$.

LASSO makes the best precision and GL the worst for the sparse configuration, while SGL is the compromise of them.

2.2 Solution with Normal Distribution

In this subsection, we give the algorithms of these LASSO family on the occasion that the dependent variable y is of normal distribution. Note that time efficiency is of top emphasis.

Given a data set $\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^n$, suppose $y_i \sim N(\mathbf{x}_i^\top \beta, \sigma^2)$, then the model takes the form:

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \beta + \varepsilon_i, \qquad i = 1, \ldots, n, \tag{11}$$

Often these n equations are stacked together and written in vector form as $\mathbf{y} = X\beta + \varepsilon$.

We can calculate the likelihood function:

$$\sum_{i=1}^{n} f(y_i|\beta, \sigma^2) = (\frac{1}{\sqrt{2\pi}})^n \sigma^{-n} e^{-\frac{(y-X\beta)^T(y-X\beta)}{2\sigma^2}} \tag{12}$$

By minimize the loss function, we obtain Maximum Likelihood Estimate (MLE):

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{13}$$

$$\hat{\sigma}^2 = \frac{1}{n}(y - X\hat{\beta})^T(y - X\hat{\beta}) \tag{14}$$

When we look it up in the matrix space, it is the same to minimize (when fixing σ^2):

$$\text{LOSS}(\beta) = (\text{y} - \text{X}\beta)^{\text{T}}(\text{y} - \text{X}\beta) \tag{15}$$

Thus, we use this Least Square LOSS function in normal distribution.

2.2.1 LASSO

It is a quadratic programming problem with linear inequality constraints. We use Cyclical Coordinate Descent [40] to iterate in order to get LASSO estimator. It is an optimization algorithm that successively minimizes along coordinate directions to find the minimum of a function. At each iteration, we minimize the objective function in line search while fixing all other coordinates.

Algorithm 1 LASSO

1: **repeat**
2: Choose next index j.
3: Update β_j.
4:

$$\beta_j = arg\min_{\beta_j} \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - \mathbf{x_i}^{\mathbf{T}}\beta)^{\mathbf{2}} + \lambda|\beta|_{\mathbf{1}} \right\}$$

5: **until** stopping condition is met.

The reason why Coordinate Descent works well in LASSO family is that:

- Each coordinate minimization can be done quickly, it is a quadratic optimization programming. We have given the explicit form of orthogonal cases.
- Quite a lot of β_j stay zero while iterating, thus the speed is very fast.

Convergence Tseng (2001) has established that coordinate descent converges to the minimizer of objective function [41]. The key to this result is the separability of the penalty function, that is each β_j can be separated from others. Hence, the coordinate-wise algorithms for the LASSO and Group LASSO converge to their optimal solutions.

Comparison with Other Algorithms Many other algorithms have been proposed to solve LASSO, such as Least Angle Regression (LARS) [42], which is a compromise between Forward Selection and Forward Stagewise. Several detailed comparisons have been experimented [43], which shows that coordinate-wise descent is very competitive with others, probably the fastest procedure for that problem to-date. In terms of numerical errors, for heavily correlated variables, Lars will accumulate more errors, while the coordinate descent algorithm will only sample the path on a grid. Many **R** packages such as **glmnet** have enclosed this algorithm.

2.2.2 Group LASSO

We use Cyclical Block Coordinate Descent [44] to iterate in order to get Group LASSO estimator. It quite resembles Coordinate Descent except we iterate one group at a time.

Algorithm 2 Group LASSO

1: **repeat**
2: Choose next block index G_j.
3: Update β_{G_j}.
4:

$$\beta_{G_j} = arg\min_{\beta_{G_j}} \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - \mathbf{x_i}^{\mathbf{T}} \beta)^{\mathbf{2}} + \lambda \sum_{\mathbf{j=1}}^{\mathbf{J}} \sqrt{\mathbf{p_j}} |\beta_{\mathbf{G_j}}|_{\mathbf{2}} \right\}$$

5: **until** stopping condition is met.

The authors of Group LASSO establish an **R** package **grplasso** for numerical experiment. Also, **grpreg** is an elegant choice.

Comparison with LASSO As shown in Efron(2004) [42], the solution paths of LASSO is piecewise linear, and thus can be computed very efficiently. However, group LASSO is generally not piecewise linear [16] and the time consumed will be much longer.

2.2.3 Sparse Group LASSO

We use Cyclical Block Coordinate Descent to iterate in order to get Sparse Group LASSO estimator. We utilize two loops in one iteration, the outer one is for group index and the inner for item index.

Algorithm 3 Sparse Group LASSO

repeat
  Choose next block index G_j.
  repeat
    Choose next index k in G_j.
    Update β_k.

$$\beta_k = arg\min_{\beta_k}\left\{\frac{1}{2}\sum_{i=1}^{n}(y_i - \mathbf{x_i}^{\mathbf{T}}\beta)^{\mathbf{2}} + \lambda(\mathbf{1}-\alpha)\sum_{\mathbf{j=1}}^{\mathbf{J}}\sqrt{\mathbf{p_j}}|\beta_{\mathbf{G_j}}|_{\mathbf{2}} + \lambda\alpha|\beta|_{\mathbf{1}}\right\}$$

  until stopping condition is met.
until stopping condition is met.

The authors of Sparse Group LASSO establish an **R** package **msgl** for numerical experiment.

Convergence The Sparse Group LASSO penalty is not completely separable, which is problematic using the standard method to prove convergence. Hence, we modify coordinate descent method a little. After setting several assumptions, we can prove that logit and normal SGL is indeed convergent with some sophisticated mathematical computing in Vincent (2014) [45].

2.3 Solution with Binomial Distribution

In this subsection, we give the algorithms of these LASSO family on the occasion that the dependent variable y is of binomial distribution.

Suppose $y_i \sim \mathrm{Bin}(1, \mathrm{p_i})$, then $\mu_i \triangleq E(y_i) = p_i$. Let $\mathbf{x_i}^{\mathbf{T}}\beta = \mathbf{g}(\mu_{\mathbf{i}}) \triangleq \ln\frac{\mathbf{p_i}}{\mathbf{1-p_i}}$. Hence,

$$p_i = P(y_i = 1|\mathbf{x_i}) = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{e}^{-\mathbf{x_i}^{\mathbf{T}}\beta}} \tag{16}$$

We use minus log likelihood as the LOSS function. We often use log function to prevent number underflow, especially in the continuously multiplication of probability, which is called Laplace Correction.

$$\begin{aligned}\mathrm{LOSS}(\beta) &= \frac{1}{n}\sum_{i=1}^{n} y_i \ln p_i + (1-y_i)\ln(1-p_i) \\ &= \frac{1}{n}\sum_{i=1}^{n} y_i(\mathbf{x_i}^{\mathbf{T}}\beta) - \ln(\mathbf{1} + \mathbf{e}^{\mathbf{x_i}^{\mathbf{T}}\beta})\end{aligned}$$

When we minimize LOSS function without any penalty, the explicit result is impossible due to the exponential term. Hence, usually we manipulate a quadratic approximation.

Algorithm 4 Newton Method

repeat
  Evaluate $g = \nabla Q(\beta) = X^T(\mu - y)$
  Evaluate $H = \nabla g = X^T S X$, here $S = \text{diag}\{\mu_1(1-\mu_1), ..., \mu_n(1-\mu_n)\}$
  Solve $d = -H^{-1}g$
  Update β: $\beta = \beta + d$
  $\beta^{t+1} = (X^T S^t X)^{-1} X^T S^t z^t$, here $z^t = X\beta^t + S^{-1t}(y - \mu^t)$ at t iteration
until stopping condition is met.

Note that newton method is actually a quadratic approximation, a taylor expansion about current estimates β. In a more statistical type, we gives Iteratively Reweighted Least Squares (IRLS) [46] method to get the estimator, which is actually equivalent. We can regard output y as normal distribution, with a diagonal covariance matrix. Then, it amounts to weighted least square.

Algorithm 5 Iteratively Reweighted Least Squares

repeat *at* t^{th} iteration

$$Q(\beta^t) = -\frac{1}{2n}\sum_{i=1}^{n} s_i^t(z_i^t - \mathbf{x_i}^\mathbf{T}\beta^\mathbf{t})^\mathbf{2} + \mathbf{CONSTANT}$$

  Update $\beta^{t+1} = \arg\min_\beta Q(\beta^t)$
  From weighted LS, $\beta^{t+1} = (X^T S^t X)^{-1} X^T S^t z^t$
until stopping condition is met.

2.3.1 LASSO

We combine IRLS and coordinate descent algorithm to compute it. The outer loop is the approximation of LOSS function and the inner loop is the coordinate index.

Algorithm 6 Logistic LASSO

repeat *at* t^{th} iterate
  Update the quadratic approximation $LOSS(\beta^t)$ at β^t
  repeat
    Choose next index j
    Update $\beta_j^t = arg\min_{\beta_j^t}\{Q(\beta^t) + \lambda|\beta^t|_1\}$.
  until stopping condition is met.
until stopping condition is met.

Convergence Just follow Tseng (2001), we may prove that coordinate descent converges to the minimizer of objective function similarly. Hence, the

coordinate-wise logistic algorithms for the LASSO and Group LASSO converge to their optimal solutions.

2.3.2 Group LASSO

Since it is time-consuming for us to compute the Hessian matrix for block terms, we make some approximations.

- Use a diagonal matrix $h^t_{G_j} I$ to approximate Hessian matrix, where $h^t_{G_j}$ is a scalar term and is fixed in the t^{th} iteration.
- Use Armijo rule [47] to get inexact optimum. Armijo Goldstein condition involves starting with a relatively large estimate of the step size for movement along the search direction and iteratively shrinking the step size until a decrease of the objective function is observed.

Block Coordinate Gradient Descent (BCGD) [48] is used here.

Algorithm 7 Logistic Group LASSO

repeat *at* t^{th} iterate
  Update the quadratic approximation $Q(\beta^t)$ at β^t
  repeat
    Choose next block index G_j
    Approximate $H^t_{G_j G_j} = h^t_{G_j} I$
    Find direction $d^t_{G_j}$

$$d^t_{G_j} = arg \min_{d^t_{G_j}} \left\{ Q(\beta^t) + \lambda \sum_{j=1}^{J} \sqrt{p^t_{G_j}} |\beta^t_{G_j}|_2 \right\}$$

    Line search τ using the Armijo rule.
    Update $\beta^t_{G_j} = \beta^t_{G_j} + \tau d^t_{G_j}$
  until stopping condition is met.
until stopping condition is met.

Comparison with Others Meier has made thorough computational speed experiments, and BCGD for sparse models is efficient for high dimensional data [18].

2.3.3 Sparse Group LASSO

We make a hybrid of BCGD and IRLS.

Remark 1 *All the above algorithms need to minimize a function of β_j with a non differentiable point $\beta_j = 0$. Applying subgradient method, we should inspect whether $\beta_j = 0$ is the minimal point before using Newton method. That is, if*

Algorithm 8 Logistic Sparse Group LASSO

repeat *at* t^{th} iterate
  Update the quadratic approximation $Q(\beta^t)$ at β^t
  repeat
    Choose next block index G_j
    Approximate $H^t_{G_jG_j} = h^t_{G_j} I$
    repeat
      Choose next index k in G_j
      Find direction d^t_k

$$d^t_k = arg\min_{d^t_k} \left\{ Q(\beta^t) + \lambda(1-\alpha) \sum_{j=1}^{J} \sqrt{p_{G_j}} |\beta^t_{G_j}|_2 + \lambda\alpha|\beta^t|_1 \right\}$$

      Line search τ using the Armijo rule.
      Update $\beta^t_k = \beta^t_k + \tau d^t_k$
    until stopping condition is met.
  until stopping condition is met.
until stopping condition is met.

0 belongs to the interval of left and right derivative of the function, we should replace $\beta_j = 0$ *instead.*

Remark 2 *All the above algorithms are done after hyper parameter* λ *is fixed. In practice, we use a decreasing sequence of values for* λ*, starting at the largest value* λ_{max} *for which the entire vector* $\beta = 0$*. Then we exploit* ***warm starts*** *[40], i.e., we use* β^{λ_k} *as a starting value for* $\beta^{\lambda_{k+1}}$*. It may accelerate since lots of* β *remain zero in the previous step.*

Remark 3 *In practice, after several complete cycles through all the variables, we just only iterate on the active set (those with nonzero coefficients) till convergence in order to accelerate speed.*

3 Selecting Tuning Parameter

Every λ gives a unique model and results in different β. We try to get the best model based on some criteria.

We split the whole dataset into test set and training set. The test set is used only to measure the goodness of fit of the final model in the last step. We never contaminate the test data. This thought comes from Nested Cross Validation [49].

There are some common criteria.

3.1 CV

In K-fold cross-validation, the original sample is randomly partitioned into K equal sized subsamples. Of the K subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $K-1$ subsamples are used as training data.

For each $k = 1, ..., K$, we fit the model with parameter λ to the other $K-1$ parts, giving $\tilde{\beta}^{-k}(\lambda)$ and compute its loss $\mathrm{LOSS_k}(\lambda)$ in predicting the k^{th} part. Here, $\tilde{\beta}$ can be any estimator. This gives the cross-validation error:

$$\mathrm{CV}(\lambda) = \frac{1}{\mathrm{K}} \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{LOSS_k}(\lambda) \tag{17}$$

$$\lambda^* = \arg\min_{\beta} \mathrm{CV}(\lambda) \tag{18}$$

LOSS Function After the model is fitted, we need to define LOSS function for CV. For normal distribution, we use Residual Sum of Square as the LOSS function. For binomial distribution, we have some other choices:

- Deviance.

$$\mathrm{Dev_k}(\lambda) = \frac{-2}{\mathrm{n/k}} \mathrm{Loglik}(\tilde{\beta}^{\mathrm{k}}(\lambda)) \tag{19}$$

  Deviance is inverse ratio to Log likelihood function, which is a measure of goodness of fit. Usually, deviance is obtained by log-likelihood ratio which contains the saturated model. However, since the principal use is in the form of the difference of the deviances of two models, this confusion in definition is unimportant. We use deviance on the left-out data with size n/k.

- Misclassification Error (ME).

$$\mathrm{ME} = \frac{1}{\mathrm{n/k}} \{ \#_{\mathrm{i}}(\mathrm{p_i} > 0.5 \ \& \ \mathrm{y_i} = 0) + \#_{\mathrm{i}}(\mathrm{p_i} < 0.5 \ \& \ \mathrm{y_i} = 1) \} \tag{20}$$

  All the p_i and y_i are calculated on the validation set. ME is directly perceived through the sense. We use ME on the left-out data. ME can be treat as discrete type of Deviance.

Shortcoming CV is time-consuming because we need to fit K models. Also, it is too generous, that it, it tends to select more predictors. As a consequence, it may not achieve the goal of sparsity.

3.2 IC family

In this section, we only consider MLE $\hat{\beta}$ rather than LASSO estimator β^L. The lower the IC value, the better the model is.

3.2.1 Akaike Information Criterion

$$\mathrm{AIC}(\lambda) = -2\ln \mathrm{Loglik}(\hat{\beta}(\lambda)) + 2\nu(\lambda) \tag{21}$$

Here, $\nu(\lambda) = \mathrm{df}(\lambda)$

Motivation Theoretical derivation of AIC is a little complex. First, we use Kullback-Leibler distance $-K$ + CONSTANT to measure how large is the difference between MLE estimator and true density. $K = \int p(y) \ln p(y|\hat{\beta}) dy$. Then, we apply Central Limit Theorem (CLT) to the score function (first derivative). We use taylor expansion twice and some approximations to prove that AIC/n = −K.

AIC can also be regarded as the compromise between goodness of fit and model complexity.

Shortcoming We can see that there are a lot of approximations and assumptions being used. So AIC is a very crude tool and is generous too. The model selected by AIC asymptotically achieves the smallest error among the candidates.

3.2.2 Bayesian Information Criterion

BIC is the same as AIC but the penalty is harsher. Thus, BIC tends to choose simpler models.

$$\mathrm{BIC}(\lambda) = -2\ln \mathrm{Loglik}(\hat{\beta}(\lambda)) + \nu(\lambda)\ln \mathrm{n} \tag{22}$$

Here, $\nu(\lambda) = \mathrm{df}(\lambda)$

Motivation Theoretical derivation of BIC is quite much complex. We gives prior distribution of estimators (uniform prior) and try to maximize a posterior. We do a taylor expansion at MLE point. Then we do Eigen Value Decomposition of the Hessian Matrix. In the integral part, we use Laplace method to estimate. After using weak law of large numbers and set n to infinity, we get BIC value.

Comparison with AIC BIC is consistent [50], which is shown below. Also, Yang (2003) [51] proved that there is no selection criterion that can both achieve consistency and optional in error.

3.2.3 Extended Bayesian Information Criterion

BIC is sometimes generous too. Chen and Chen has proposed EBIC.

$$\mathrm{EBIC}_{\gamma}(\lambda) = -2\ln \mathrm{Loglik}(\hat{\beta}(\lambda)) + \nu(\lambda)\ln \mathrm{n} + 2\gamma\nu(\lambda)\ln \mathrm{p} \tag{23}$$

Motivation BIC prior is the same to every estimator, which is unreasonable. Suppose a set S_j is a subset of whole predictors containing j of them. Instead of assigning probabilities $p(S_j)$ proportional to $\binom{p}{j}$, as in the ordinary BIC (due to uniform prior distribution), we assign $p(S_j)$ proportional to $\binom{p}{j}^{-\gamma}$ for some γ between 0 and 1. Thus, we derive EBIC value.

Consistency The most important property of EBIC and BIC is consistency.

Theorem 1 *Suppose λ_0 is the true model. Under some mild conditions with $n \to \infty$, we have*

$$P\{\min \mathrm{EBIC}_\gamma(\lambda) \leq \mathrm{EBIC}_\gamma(\lambda_0)\} \to 0 \tag{24}$$

The proof of consistency can be seen in the research paper of Chen and Chen [25].

Comparison with BIC We compare the set s^* selected by the extended BIC with the real model s. We define the positive selection rate (PSR) as the ratio $\#(s^* \cap s)/\#(s)$, and the false discovery rate (FDR) as the ratio $\#(s^* - s)/\#(s^*)$. If γ travels from 0 to 1, both the positive selection rate and the false discovery rate decrease. However, the decline of PSR is inconspicuous but the decline of FDR is significant. We may choose $\gamma = 0.5, 1$ as recommended.

Remark 4 *All the definitions of IC family is based on MLE. However, we just replace it as the Loglike funtion of LASSO family instead when using them as criteria in LASSO family. Indeed, the perfect consistent property doesn't hold in this situation. There exists some attempts to adjust the model a little to persist the property, for example, Adaptive LASSO [52] with Extended Regularized Information Criterion (ERIC) [53]. However, these attempts are too specific, limit and unable to generalize.*

3.2.4 Degree of Freedom

Degree of freedom clearly implies to the model complexity and can be instilled into IC value for comparing two models with different number of parameters. For example, the df in multiple linear regression exactly equals the number of predictors. What is the df in LASSO family needs consideration. The Stein's unbiased risk estimation (SURE) theory (Stein 1981) [54] gives a insightful generalization to LASSO.

Definition of df Suppose $\hat{y} = f(y)$ represent its fit and we assume $y \sim (\mu, \sigma^2 I)$, where μ is the true mean and σ^2 is the common variance. We define

$$\mathrm{df}(\hat{\mathrm{y}}) = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{cov}(\hat{\mathrm{y}}_\mathrm{i}, \mathrm{y}_\mathrm{i})/\sigma^2 \tag{25}$$

If $\hat{y}$ is a linear smoother, i.e., $\hat{y} = Sy$, we simply obtain $\mathrm{df}(\hat{\mathrm{y}}) = \mathrm{tr}(\mathrm{S})$, which relegate to the effective degrees of freedom. Effective degrees of freedom [55] is quite common in calculating Generalized Cross Validation (GCV) [56], which is a method in selecting best bandwidth h of non parametric regression.

Theorem 2 *(Stein's Lemma) Suppose $\hat{y}$ is almost differentiable and $\nabla \bullet \hat{y} = \sum_{i=1}^{n} \partial \hat{y}_i / \partial y_i$. If $y \sim \mathrm{N}(\mu, \sigma^2 \mathrm{I})$, then*

$$\sum_{i=1}^{n} \mathrm{cov}(\hat{\mathrm{y}}_\mathrm{i}, \mathrm{y}_\mathrm{i})/\sigma^2 = \mathrm{E}[\nabla \bullet \hat{\mathrm{y}}] \tag{26}$$

Even if $\nabla \bullet \hat{y}$ depends on y, we can get an unbiased estimate for df: $\hat{\mathrm{df}}(\hat{y}) \triangleq \nabla \bullet \hat{y} = tr(\frac{\partial \hat{y}}{\partial y})$.

Df of LASSO The LASSO fit is not a linear smoother, which may cause quite difficulty. The first line of thinking is to turn LASSO fit into a linear one, i.e., regard LASSO as a weighted Ridge, which we have already discussed before. Hence, we may get $\mathrm{df}(\beta^{\mathrm{L}}) = \mathrm{tr}(\mathrm{S})$, where S is the smooth matrix.

Alternative way is derived from Stein's Lemma:

Theorem 3 *Denote the set of non-zero elements (support, or active set) of β_λ^L as $A(\lambda)$, then*

$$\nabla \bullet \hat{y} = tr(H_\lambda) = \#A(\lambda) \tag{27}$$

where $H_\lambda = X_A(X_A^T X_A)^{-1} X_A^T$ is the projection matrix.

The proof is shown by Zou, Hastie and Tibshirani [57].

From the explicit form of β^L before, we may find that the LASSO fit y^L is the residual from projecting y onto a polyhedron, thus is a function of y and hence continuous and almost differentiable. By the theorem, the number of non-zero coefficients is an unbiased estimate for the degrees of freedom of the Lasso.

Df of Group LASSO By the chain rule, we have $tr(\frac{\partial \hat{y}}{\partial y}) = tr(\frac{\partial \beta^{GL}}{\hat{\beta}})$. Recall that ${\beta_{G_j}}^{GL} = s[\hat{\beta}_{G_j}, \lambda\sqrt{p_j}] \triangleq \mathrm{sign}(\hat{\beta}_{\mathrm{G_j}})(|\hat{\beta}_{\mathrm{G_j}}| - \lambda\sqrt{\mathrm{p_j}})_+$, where $\hat{\beta}_{G_j}$ is the OLS of residual. We thus can calculate the partial. Hence we get the unbiased estimator of DF:

$$\hat{\mathrm{df}}^{GL} = \sum_{j=1}^{J} I(|\hat{\beta}_{G_j}| > \lambda\sqrt{p_j}) + \sum_{j=1}^{J} \{1 - \frac{\lambda\sqrt{p_j}}{|\hat{\beta}_{G_j}|}\}_+ (p_j - 1) \tag{28}$$

$$= \sum_{j=1}^{J} I(|\beta_{G_j}^{GL}| > 0) + \sum_{j=1}^{J} \frac{\beta_{G_j}^{GL}}{\hat{\beta}_{G_j}} (p_j - 1) \tag{29}$$

Remark 5 *In order to use SURE theory, we assume the response variable is Normal distributed. We simply neglect this restriction in practice.*

Remark 6 *We can get the unbiased estimator of Sparse Group LASSO too, however, both the GL and SGL estimators are too cumbersome. In numerical experiment, we just omit the minor term and use non-zero predictors size as the df. The reason why it works is that the impact on df is quite mimic.*

4 Other Models

Apart from LASSO based logistic model, we have many other choices. Although as we have discussed in the backgroud that we only apply logistic model to achieve sparsity, we still state some other model for comparison.

4.1 Multiply Layer Perceptron

A multilayer perceptron [58] (MLP) is a class of feed-forward artificial neural network where connections between the units do not form a cycle. It can be viewed as a logistic regression classifier where the input is first transformed using a learned non-linear transformation f. This transformation projects the input data into a space where it becomes linearly separable. Figure 4 provides the flow sheet of this approach.

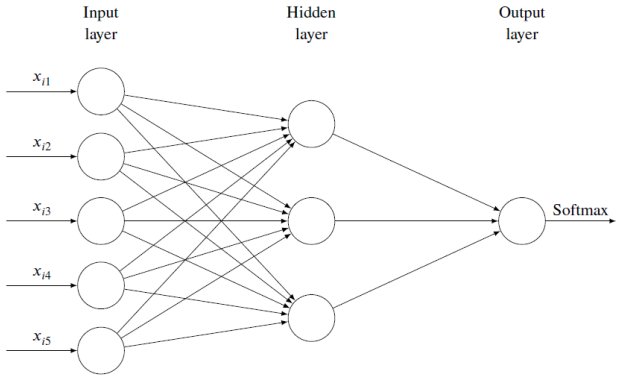


Figure 4: Flowsheet of multilayer perceptron. Suppose the input is $x_i = (x_{i1}, \cdots, x_{i5})$. Each circle represents a neuron and each arrow is a connection between layers. The output of each neuron is calculated by a non-linear function f of the combination of the input. In this hypothetic example, there is just one hidden layer. We use a softmax function at last to turn output value into a probability.

To take a closer look at the structure, we may ignore the hidden layer and set the dimension of input x_i to be $p = 3$ and show the single layer computation in Figure 5. Mathematically, a neuron in the $(t+1)$th layer $x(t+1)$ is adjusted from $x(t)$ via

$$x(t+1) = f[b(t) + w(t)^T x(t)], \tag{30}$$

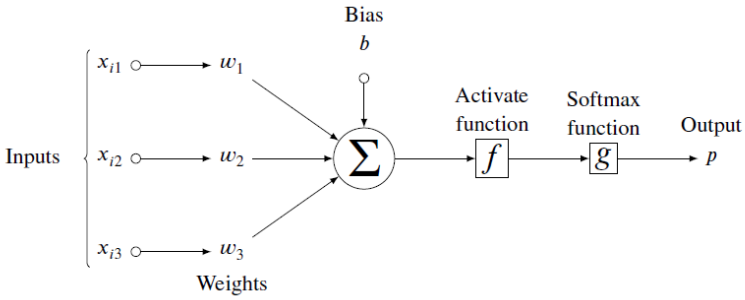


Figure 5: Flowsheet of a simplified multilayer perceptron with no hidden layers. Suppose the input is $\mathbf{x_i} = (\mathbf{x_{i1}}, \mathbf{x_{i2}}, \mathbf{x_{i3}})$. The output of each neuron is calculated by a non-linear function f of the sum of its input plus a bias b as in equation (30).

where f is the activation function, $b(t)$ is the so-called bias term at the tth layer, and $w(t)$ is the weight vector of the tth layer.

An MLP algorithm repeats the above calculation for a number of layers and stops at the Tth layer. In the output layer or the Tth layer, class-membership probabilities can be obtained from the softmax function.

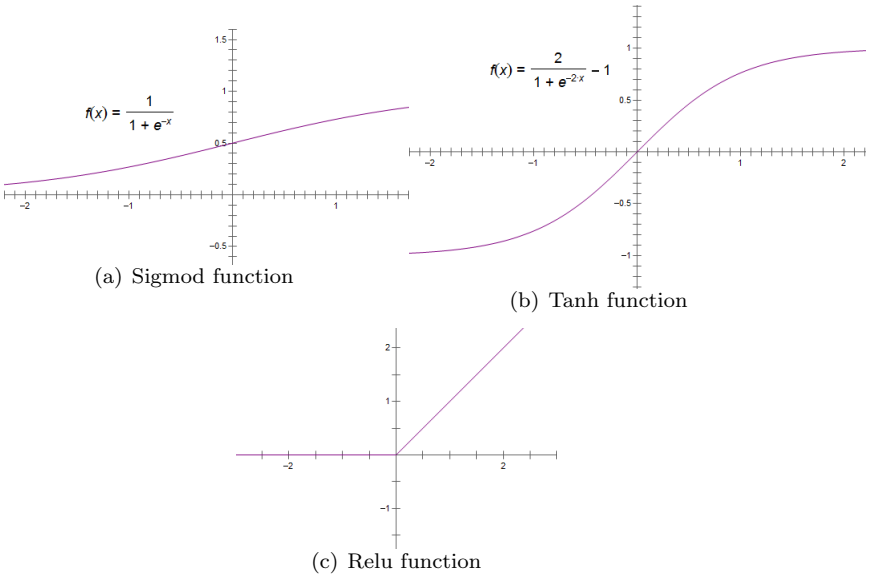
To train an MLP, we need to learn all the parameters of the model including the weight w and the bias b. The estimation is usually carried out under the stochastic gradient descent algorithm with minibatches [59]. Evaluating the gradients can be achieved through the backpropagation algorithm [**?**] (a special case of the chain-rule of derivation). Software development for deep learning has been abundant and is rapidly evolving in the recent decade. In particular, open source libraries such as **mxnet** enable non-experts to easily design, train and implement deep neural networks. We will carry out all the computation in this paper using **mxnet** since it supports languages such as **Python** or **R** and can train quickly on multiple GPUs. For more information about the software visit http://mxnet.incubator.apache.org/index.html.

In practice, the following operational parameters for the MLP need to be tuned by the user:

Layers Number of layers and number of neurons in each layer. Usually the number of neurons can be very large while the number of layers may be relatively moderate or small. The number of neurons in different layers may also differ.

Activation functions Activation functions in each layer. All three functions can be used while the Relu function is usually preferred for the simplicity of computation.

Loss function Loss function in the output layer. Usually we choose the soft-



(a) Sigmod function

(b) Tanh function

(c) Relu function

Figure 6: Activation functions.

max function.

$$f_k(x) = \text{softmax}_k(x) = \frac{\exp(\beta_k^T x)}{\sum_{m=1}^{M} \exp(\beta_m^T x)}, \qquad k = 1, \cdots, M. \tag{31}$$

Number of round or epoch The number of iterations over the sample data to train the model parameters. Often we need very large number of round to achieve satisfactory numeric accuracy, similar to other nonlinear programming problems. One may specify the option `num.round` in the software.

Learning rate The step size in gradient descent method. This tuning parameter can be optimized via the cross validation. Alternatively many practitioners recommended to use a small value such as 0.1 or 0.01. One may specify the option `learning.rate` in the software.

Initializer The initialization scheme for parameters which specifies the unknown weight at the beginning, usually drawn from a uniform design. One may specify the option `initializer` in the software.

Array batch size The batch size used for array training. The whole training data is usually divided into batches to facilitate the computing.

There is still limited discussion on how to set all these options to optimize the classification performance. Depending on the scale of the problem and

complexity of the data, settings of real world examples using MLP need to be addressed case by case.

After the model architectures are constructed, we begin to update the parameters. Samely as logistic regression, we use gradient decent method to minimize the LOSS of true output and the predicted output. Also, the LOSS is minus log likelihood, which is equivalent to cross entropy.

$$L = \sum_{i=1}^{n} L(y_i, f(x_i)) = \sum_{i=1}^{n} -\ln f_{y_i}(x_i) \tag{32}$$

After all parameters are learned, every input $\mathbf{x_i}$ can lead to a probability vector $f(x_i)$ of length M through the softmax loss.

5 Final Model Evaluation

After we select the best hyper parameter λ in section 3, the total model is fixed. Thus, we may get the coefficients $\tilde{\beta}$ of it using algorithms in section 2. Following that, it is easy for us to make prediction:

$$P(\hat{y}_i = 1|\mathbf{x_i}) = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{e}^{-\mathbf{x_i}^{\mathbf{T}}\tilde{\beta}}} \tag{33}$$

where $\mathbf{x_i}$ is a sample in the test dataset, $\hat{y}_i$ is the prediction of y_i. Thus, we may get the probability matrix $\mathbf{P} \in \mathbf{R^{n\times 2}}$, where $\mathbf{P_{ij}} = \mathbf{P}(\hat{\mathbf{y}}_\mathbf{i} = \mathbf{j})$.

We need to select a threshold c, $0 \leq c \leq 1$. Then we forcast $\hat{y}_i = \mathrm{I}(\mathbf{P_{i2}} > \mathbf{c})$.

5.1 Single Model Evaluation

5.1.1 Correct Classification Rate

Correct Classification Rate (CCR) or Accuracy (ACC) or Correct Classification Probability (CCP) is the intuition to evaluate a model. We set threshold $c = 0.5$ as default.

$$\mathrm{CCR} = \mathrm{P(y = \hat{y})} \tag{34}$$

Using the law of total probability, the equation above can be re-written as

$$CCR = \sum_{m=1}^{M} P(y = m)P(\hat{y} = m|y = m) \triangleq \sum_{m=1}^{M} \rho_m CCR_m \tag{35}$$

where the probability $P(\hat{y} = m|y = m)$ can be regarded as a class-specific CCR for the mth category. When $M = 2$, the two class-specific CCR values are commonly referred to as the sensitivity and the specificity. CCR directly assesses whether the mode-based classification for a subject is identical to his true class status. Its empirical version is simply the proportion of correctly classified subjects in the sample.

$$\widehat{\mathrm{CCR}} = \frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{y}_i). \tag{36}$$

Such a simple formula facilitates the application of CCR in many real problems, especially when M is large. In fact, the computation time for CCR does not increase as the number of categories increases. CCR and its complement, misclassification rate, are commonly used in machine learning literature. In practice, we replace probability as frequency in test dataset.

Though it provides a straightforward assessment on the performance of a fixed sample, the estimated overall CCR value is quite sensitive to the distribution of different classes in the particular data sample and thus cannot lend support to external validity for some studies where prevalence information is unavailable. For example, CCR values obtained for low-risk disease groups may be dominated by the specificity.

The implementation of CCR is quite easy after a classification is done. We will use the function **CCR** in **R** package **mcca** in the following illustration.

5.1.2 Area Under ROC Curve

It is too assertive to set a single threshold, hence we introdue AUC. Receiver Operating Characteristic (ROC) curve summarizes the models performance by evaluating the tradeoffs between true positive rate (TPR, sensitivity) and false positive rate (FPR, 1-specificity), where $\mathrm{FPR} = \mathrm{P}(\hat{\mathrm{y}} > \mathrm{c}|\mathrm{y} = 0)$ and $\mathrm{TPR} = \mathrm{P}(\hat{\mathrm{y}} > \mathrm{c}|\mathrm{y} = 1)$.

The reason why we introduce TPR and FPR is that in unbalanced y, i.e. $\frac{\#\{y=1\}}{\#\{y=0\}} = 100$, it is meaningless to compute CCR since a classifier which turns all precitions as $\hat{y} = 1$ can achieve high value. We consider conditional probability instead.

$$P(y = \hat{y}) = P(\hat{y} = 1|y = 1) \cdot P(y = 1) + P(\hat{y} = 0|y = 0) \cdot P(y = 0) \tag{37}$$

The ROC of a perfect predictive model has TPR equals 1 and FPR equals 0. This curve will touch the top left corner of the graph and the area under it is 1.

AUC is the area under ROC curve. It is equivalent to the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative example [60].

$$\mathrm{AUC} = \int \mathrm{TPR(c)}\ \mathrm{dFPR(c)} = \mathrm{P}(\hat{\mathrm{y}}|_{\mathrm{y}=1} > \hat{\mathrm{y}}|_{\mathrm{y}=0}) \tag{38}$$

In practice, we replace probability as frequency in test dataset.

Note that HUM (hyper volumn under manifold) [29] is the generalization of AUC. Thus, in the following sections, we equate both of these proper noun.

5.1.3 Polytomous Discrimination Index

Polytomous discrimination index (PDI) [28] is a recently proposed diagnostic accuracy measure for multi-category classification. Similar to HUM, PDI is also evaluating the probability of an event related to simultaneously classifying M

subjects from M categories. While HUM is pertaining to the event that all M subjects are correctly identified to their corresponding categories, PDI is pertaining to the number of subjects in the set of M subjects that are correctly identified to his/her category.

To define the PDI, we consider a random sample that involves M subjects and each subject is chosen from one of the M distinct categories. Without loss of generality, we assume that the ith subject is from the ith category. The classification decision is achieved via a joint comparison of the M subjects. Using earlier notations, for a classification model, we may denote the probability of placing a subject from category i into category j by $\mathbf{P_{ij}}$. A class i subject can be correctly classified if $\mathbf{P_{ii}} > \mathbf{P_{ji}}$ for all $j \neq i$. For a fixed category i, we may define the class-specific PDI to be

$$\mathrm{PDI_i} = \mathrm{P}(\mathbf{P_{ii}} > \mathbf{P_{ji}} \;\; \mathbf{j} \neq \mathbf{i} | \mathbf{y_i} = \mathbf{i}) \tag{39}$$

and the overall PDI to be

$$\mathrm{PDI} = \frac{1}{\mathrm{M}} \sum_{\mathrm{m}=1}^{\mathrm{M}} \mathrm{PDI_m}. \tag{40}$$

In practice, we replace probability as frequency in test dataset and we consider each situation that involves M subjects and each subject is chosen from one of the M distinct categories.

When $M = 2$, PDI also reduces to AUC and thus can be viewed as a generalization of AUC for the multi-class problem. However, when $M \geq 3$, PDI value is always greater than HUM for a classifier. Models or diagnostic biomarkers with poor PDI values usually also have poor HUM values. The lower bound for PDI is $1/M$, corresponding to random guess.

PDI is not as widely applied as HUM and CCP to assess the diagnostic and classification accuracy. We suggest its value should also be reported along with other major accuracy measures. Specifically, we would recommend computing PDI for big data studies and screen out unnecessary biomarkers whose PDI values are below a satisfactory level. The computation of PDI is implemented in function **PDI** in **R** package **mcca**.

5.2 Model Comparison

When comparing two models used to make prediction or classification for the same data, it is usually not advisable to only check the difference of a single accuracy measure. For example, it is noted by many authors that the difference of HUM between two models may not be informative when the baseline model is already quite accurate. We next consider two more appropriate measures for comparing two models.

We need more notations. Now suppose that in addition to model $\mathcal{M}_1$ introduced earlier, more variable(s) are included and we construct a model$\mathcal{M}_2$ that is based on a set of predictors $\Omega_2 \supset \Omega_1$. We use the nested-structure notations as

they are widely discussed in the literature. We note that there are studies where the accuracy improvement occurs among non-nested models as well. Our proposed methods can apply with slight modification. The newly constructed model $\mathcal{M}_2$ generates another probability vector $p_i(\mathcal{M}_2) = (p_{i1}(\mathcal{M}_2), ..., p_{iM}(\mathcal{M}_2))$ for the ith subject.

5.2.1 Net Reclassification Improvement

Net reclassification improvement (NRI) [30] is a numeric characterizations of accuracy improvement for diagnostic tests or classification models and were shown to have certain advantage over analyses based on ROC curves.

While ROC-based measures have been widely adopted, it has been argued by many authors that such measures may not be good criteria to quantify improvements in diagnostic or classification accuracy when the added value of a new predictor to an existing model is of interest. NRI can address these limitations since it essentially calculates the increase in correct classification between models.

The multi-category NRI from a baseline model $\mathcal{M}_1$ to a more complicated model $\mathcal{M}_2$ is

$$NRI = \sum_{m=1}^{M} \frac{1}{M} \{CCP_m(\mathcal{M}_2) - CCP_m(\mathcal{M}_1)\}. \tag{41}$$

NRI directly reflects how often the new model corrects the incorrectly classified cases in the old model and is therefore very appealing to practitioners. We note another interpretation for NRI is the difference of Youden's index of the two models.

For additional discussion of these recent developments, Hilden [61] pointed out that NRI sometimes may inflate the prognostic performance of added biomarkers and Kerr [62] argued that NRI may perform poorly under some nonlinear data generating mechanisms. Thus users of these popular metrics should also exercise caution in practice.

6 Software

R is a programming language and free software environment for statistical computing and graphics. Due to its simplicity to handle, in this dissertation, we implement **R** as our main software. In the deep learning approach, we will also use **python** to achieve more structures. All the following packages can be downloaded from The Comprehensive R Archive Network (CRAN).

6.1 Models

For different model, we adopt different library. Every of them is well established and has been tuned many times, hence, we may feel free to use for sure. Note

that since there are more than one libraries to achieve LASSO for example, we just select the prominent one.

- LASSO. Package **glmnet**.
- Group LASSO. Package **grpreg**.
- Sparse Group LASSO. Package **msgl**.
- Multiple Layer Perceptron. Package **mxnet** or **Python** framework **TensorFlow**.

Actually, I try to read the source codes of these LASSO family packages. Sadly, quite amount of them are written in Fortran and C, which are encapsulated and hence impossible for me to read. However, what rejoice me is that both the authors have given their algorithms and the simply versions is discussed in section 3.

6.2 Hyper Parameter Tuning

For CV, we just manually fit models K times and calculate on the validation set. For information criteria, we first calculate the Log likelihood and then the IC value from their definitions.

6.3 Model Evaluation

Here, we use package **mcca** which is donated myself. The function **hum** relegate to AUC for binary output. To compute standard error, we use function **ests** for single model evaluation and **estp** for model comparison.

7 Singapore Eye Dataset

During my summer research in Singapore, I got a database about Singapore Eye Study (SES). It contains around 2700 samples of 300 instances each, where each sample represents a single person and each instance is a body index. There are some indexes I concern most, i.e. heart attack, stroke, hypertension, diabetes (each of them is a categorical variable). Since these indexes are vital for people and people wont be aware of them until they go to see a doctor, it is quite meaningful for us to forecast these diseases. In this research, I will try to get whether these mortal diseases can be forecasted by some common and easily-accessed eye indexes such as cataract, myopia, eye trauma, and so on. If there exists high correlation or I can draw a stable equation between these indexes, it will be a gospel to these potential patients.

7.1 Preprosessing

The Singapore Eye Study (SES) database conduct around 3000 peoples 300 indexes, mainly separated into 6 parts:

- Basic information (age, height,...),
- Blood data (glucose, cholesterol,...),
- Eye data (myopia, blindness, sphere,...),
- Eye disease (cataract,...),
- Self-information (education, job, smoke, income,...),
- Main disease (heart attack, stroke, hypertension, diabetes,...).

First, we clean the data by removing or imputing missing entries. We inspect the dataset and delete columns and rows which have too many NA values until there exists less than 2% NAs. Then we assort variables less than 6 different values as factors and others as continuous predictors. Finally, we fill in the missing values with their mode and median separately.

By putting main disease variable into the output term and other variables into input terms, we may establish a logistic regression model. After adding a penalty term, we deduce the refined model.

In this dissertation, we only focus on heart attack (variable "mi") as the output and it is a binary factor.

7.2 Training and Test Set

First, we divide the whole dataset randomly into two parts: a training data and a test data. We use the training data to build the statistical model and the test data to assess the out-of-sample performance. The ratio between $y = 0/1$ of the training and test data is the same, and we use approximate $1/10$ observations as the test. Here $y = 1$ means that this person is not suffered from heart attack.

Then, we use One Hot Encoding to turn all the factors into dummy variables. Figure 7 shows first several rows and columns. Dimension of training input:(2949, 339); dimension of test input:(327, 339); length of training output:(2949); length of test output:(327).

	czmi1	gender2	age	agegp2	agegp3	agegp4	agegp22	agegp23	agegp24	agegp25	agegp3	bpsys_f
887	1	1	58.55441478	1	0	0	1	0	0	0	4	99.5
1248	0	0	75.71800137	0	0	1	0	0	1	0	8	106
1918	0	0	54.00136893	1	0	0	1	0	0	0	3	107
3047	0	1	73.10335387	0	0	1	0	0	1	0	7	139.5
673	0	1	50.41204654	1	0	0	1	0	0	0	3	115.5
3013	0	0	78.65023956	0	0	1	0	0	1	0	8	180
3166	0	1	47.75633128	0	0	0	0	0	0	0	2	136.5
2211	1	0	66.69130732	0	1	0	0	1	0	0	6	136
2103	0	0	53.60711841	1	0	0	1	0	0	0	3	142
208	1	1	49.10882957	0	0	0	0	0	0	0	2	105
686	0	1	78.74606434	0	0	1	0	0	1	0	8	184
588	1	0	47.13483915	0	0	0	0	0	0	0	2	108
2296	1	1	58.88843258	1	0	0	1	0	0	0	4	135
1283	0	0	63.88227242	0	1	0	0	1	0	0	5	153

Figure 7: First several rows and columns of dataset.

7.3 LASSO

We use a series of λ to fit the logistic LASSO.

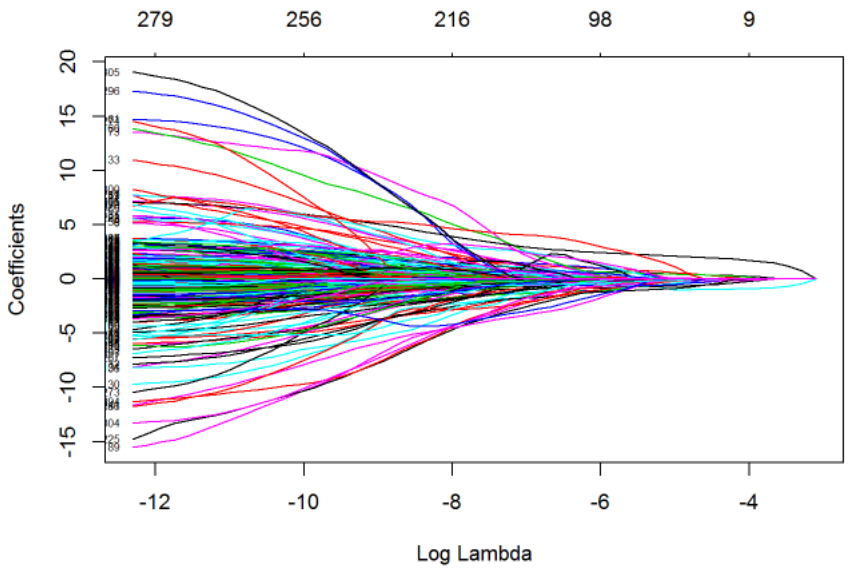


Figure 8: Coefficient path of λ. Each curve represents a coefficient (labelled on the left). The top panel is the number of non-zero variables.

Figure 8 shows the coefficient path of λ where lasso estimates as a function of λ. If λ grows bigger, the non-zero variables become smaller. All the coefficients are standardized for comparison.

Now we determine the best λ and the best model from this series of λ.

7.3.1 Best lambda through CV: dev

We choose the best model through 10 fold CV: dev.

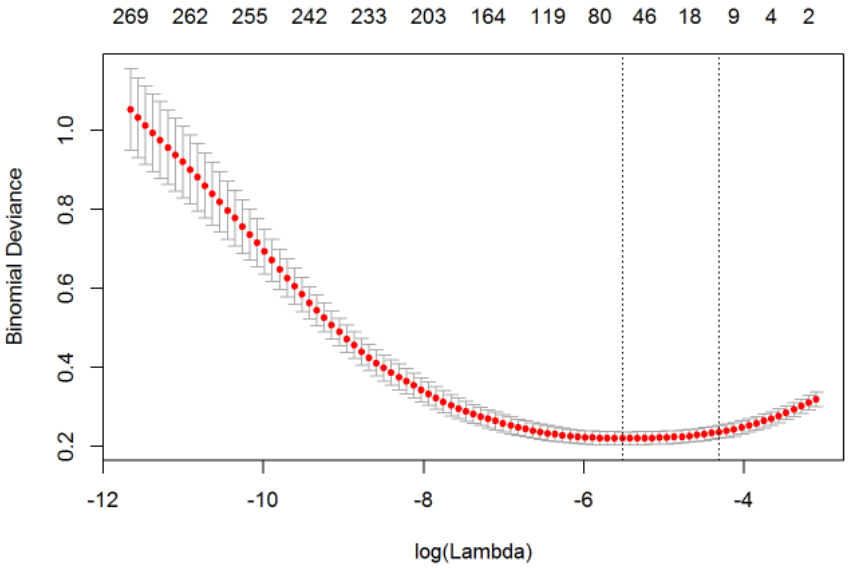


Figure 9: The red dot is the average of 10 result, and the upper and lower bound is the one standard error. The left vertical line corresponds to the minimum error. The top panel is the number of non-zero variables.

After choosing the best model, we may get these results:

- The best lambda is 0.0040135.
- There are 63 no-zero variables.
- Correct classification rate of training data: 0.9677857
- Area under curve of training data: 0.9367191
- Correct classification rate of test data: 0.9602446
- Area under curve of test data: 0.8455043

Also, we may get β^L and the final model. Since there are too many non-zero variables selected, we omit them here. You may take a look at the link in the appendix.

7.3.2 Best lambda through CV: ME

We choose the best model through 10 fold CV: ME.

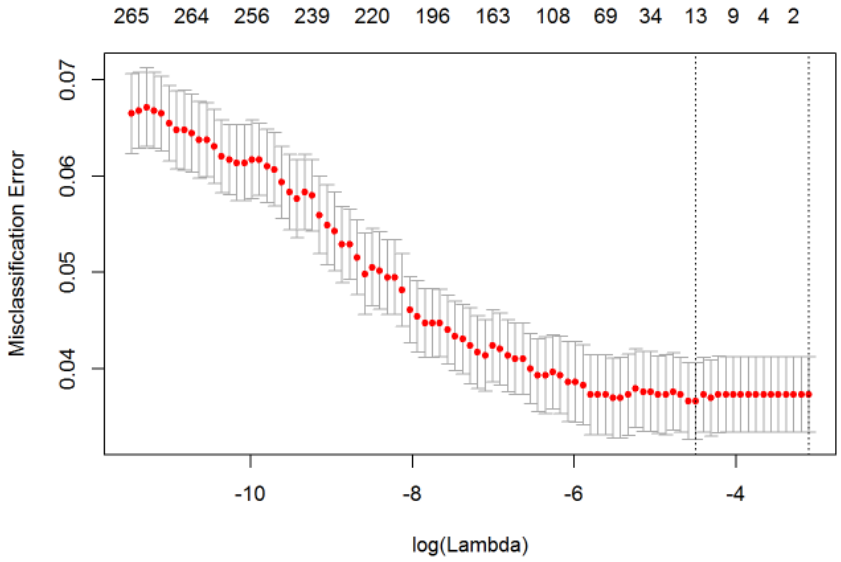


Figure 10: The red dot is the average of 10 result, and the upper and lower bound is the one standard error. The left vertical line corresponds to the minimum error. The top panel is the number of non-zero variables.

After choosing the best model, we may get these results:

- The best lambda is 0.0111678.
- There are 14 no-zero variables.
- Correct classification rate of training data: 0.9637165
- Area under curve of training data: 0.9105655
- Correct classification rate of test data: 0.9633028
- Area under curve of test data: 0.8523505

Also, we may get β^L and the final model.

$$\begin{aligned}\ln \frac{p}{1-p} = &(-0.87) + (0.32) * gender2 + (-0.23) * agegp25 \\ &+ (-0.19) * anti\_ht1 + (-0.97) * anti\_chol1 + (-0.49) * drugs\_others1 \\ &+ (-0.07) * hypertension1 + (0.35) * chol + (0.01) * GFR\_EPI \\ &+ (-0.16) * bvalogr\_USA2 + (0.33) * smkyn2 + (-0.19) * smk\_cat3 \\ &+ (1.94) * ang2 + (-0.47) * R\_retino\_cat2\end{aligned}$$

The definition of all these predictors will be explained in the next section.

7.3.3 Best lambda through BIC

We choose the best model through BIC. Here the definition of BIC is:

$$\mathrm{BIC}(\lambda) = -2 \ln \mathrm{Loglik}(\beta^{\mathrm{L}}(\lambda)) + \mathrm{df}(\lambda) \ln \mathrm{n} \tag{42}$$

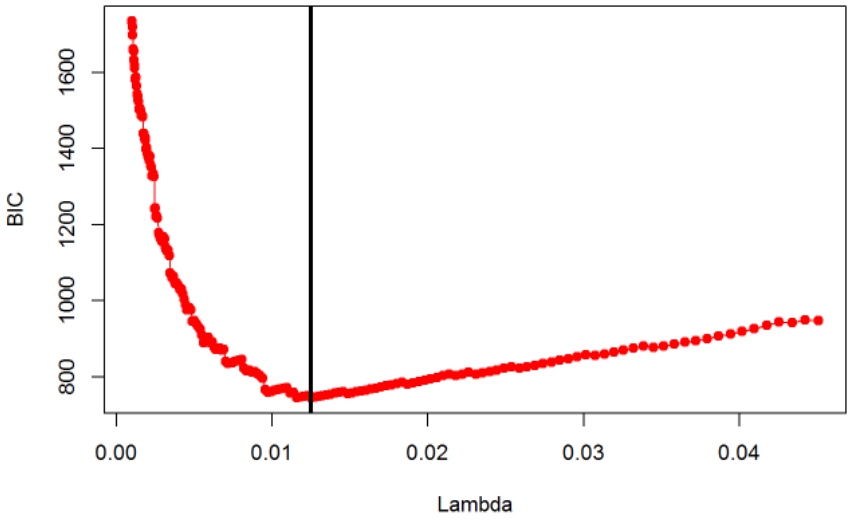


Figure 11: The red dot is bic value of a specific lambda. The vertical line corresponds to the minimum error.

After choosing the best model, we may get these results:

- The best lambda is 0.0101757.
- There are 15 no-zero variables.
- Correct classification rate of training data: 0.9637165
- Area under curve of training data: 0.9126357
- Correct classification rate of test data: 0.9633028
- Area under curve of test data: 0.8505249

Also, we may get β^L and the final model.

$$\begin{aligned}\ln \frac{p}{1-p} &= (-0.90) + (0.38) * gender2 + (-0.29) * agegp25 \\ &+ (-0.19) * anti\_ht1 + (-0.99) * anti\_chol1 + (-0.52) * drugs\_others1 \\ &+ (-0.14) * hypertension1 + (0.37) * chol + (0.01) * GFR\_EPI \\ &+ (0.02) * srepcylr + (0.35) * smkyn2 + (-0.20) * smk\_cat3 \\ &+ (1.98) * ang2 + (-0.56) * R\_retino\_cat2 + (-0.19) * bvalogr\_USA2\end{aligned}$$

7.3.4 Best lambda through EBIC

We choose the best model through EBIC with $\gamma = 0.5$.

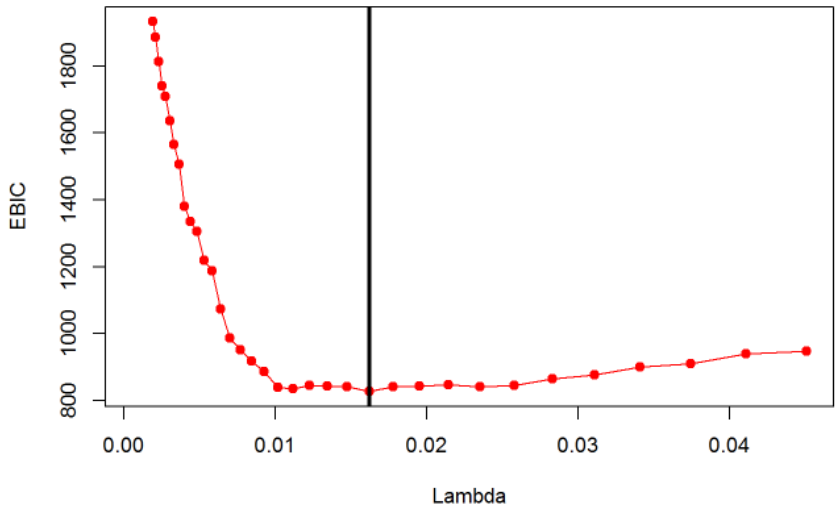


Figure 12: The red dot is bic value of a specific lambda. The vertical line corresponds to the minimum error.

After choosing the best model, we may get these results:

- The best lambda is 0.0162026.
- There are 10 no-zero variables.
- Correct classification rate of training data: 0.9633774
- Area under curve of training data: 0.9001441
- Correct classification rate of test data: 0.9633028
- Area under curve of test data: 0.8461890

Also, we may get β^L and the final model.

$$\begin{aligned}\ln \frac{p}{1-p} = &(-0.48) + (0.08) * gender2 + (-0.11) * anti\_ht1 \\ &+ (-0.92) * anti\_chol1 + (-0.32) * drugs\_others1 + (0.27) * chol \\ &+ (0.01) * GFR\_EPI + (0.21) * smkyn2 + (-0.15) * smk\_cat3 \\ &+ (1.77) * ang2\end{aligned}$$

7.4 Group LASSO

Before fitting, group should be defined. Here we bind all the dummy variables of a single variable in a group and use an index vector which describes the grouping of the coefficients. It is a vector of consecutive integers and if two variables share the same value, we take them in a group.

```
##   [1]   1   2   3   4   4   4   5   5   5   5   6   7   8   9  10  11  12
##  [18]  13  14  15  15  15  16  17  18  19  20  21  22  23  24  24  24  25
##  [35]  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42
##  [52]  43  43  44  44  45  45  46  46  47  47  48  48  49  49  50  50  51
##  [69]  51  52  53  54  55  55  55  55  55  56  56  56  56  56  57  57  57
##  [86]  57  57  58  58  58  58  58  59  60  61  62  63  64  65  66  67  68
## [103]  69  69  70  70  71  71  72  73  74  75  76  77  78  79  80  81  82
## [120]  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  97  97
## [137]  98  98  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112
## [154] 113 114 115 116 116 117 117 118 119 120 121 122 123 124 125 126 126
## [171] 126 127 127 127 128 128 128 128 129 130 131 132 132 132 132 133 133
## [188] 133 134 135 136 137 138 138 139 140 141 142 143 144 145 146 147 148
## [205] 149 150 151 151 151 152 153 154 155 156 157 158 159 160 161 162 163
## [222] 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
## [239] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 196
## [256] 197 197 198 198 199 200 201 202 203 204 205 205 205 205 206 207 208
## [273] 209 210 211 212 213 214 214 214 214 215 215 215 215 216 217 218 219
## [290] 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236
## [307] 237 238 239 240 241 242 243 244 244 245 246 247 247 248 249 250 251
## [324] 251 252 253 254 254 255 256 257 258 259 260 261 262 263 264 265
```

Figure 13: Index vector. Predictors with same number are in the same group.

We use a series of λ to fit the logistic Group LASSO.

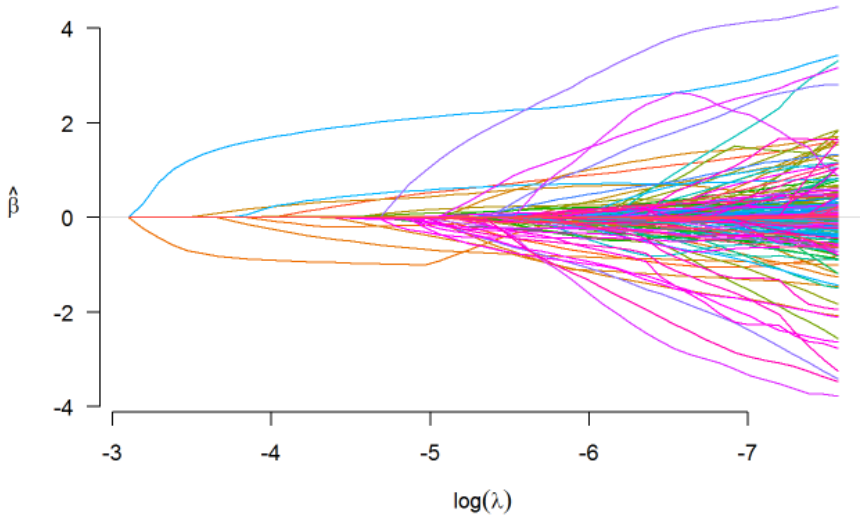


Figure 14: Coefficient path of lambda. Each curve represents a coefficient.

Figure 14 shows the coefficient path of lambda where lasso estimates as a function of λ. If λ grows bigger, the non-zero variables become smaller.

Now we determine the best λ and the best model from this series of λ.

7.4.1 Best lambda through CV: dev

We choose the best model through 10 fold CV: dev.

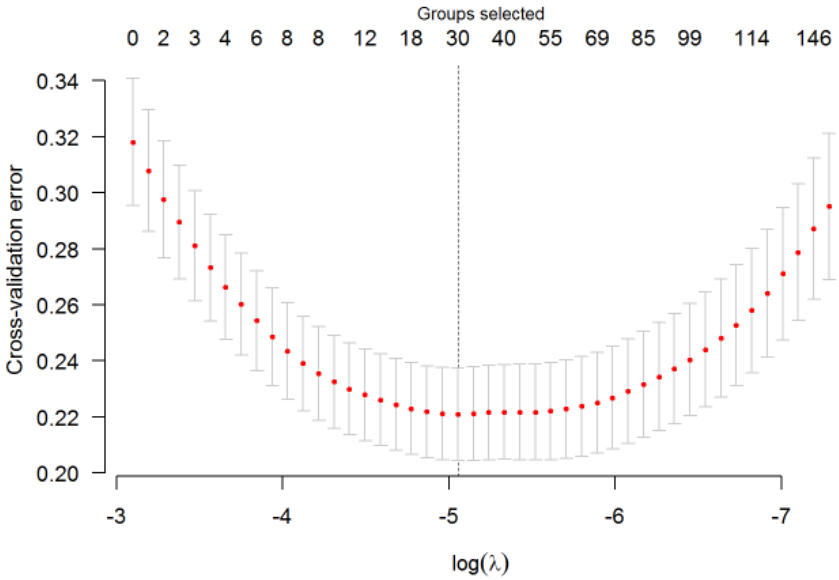


Figure 15: The red dot is the average of 10 result, and the upper and lower bound is the one standard error. The vertical line corresponds to the minimum error. The top panel is the group number of non-zero variables.

After choosing the best model, we may get these results:

- The best lambda is 0.0063906.
- There are 37 no-zero variables.
- Correct classification rate of training data: 0.9647338
- Area under curve of training data: 0.9200487
- Correct classification rate of test data: 0.9602446
- Area under curve of test data: 0.8386581

Also, we may get β^{GL} and the final model. Since there are too many non-zero variable selected, we omit it here. You may take a look at the link in the appendix.

7.4.2 Best lambda through CV: ME

We choose the best model through 10 fold CV: ME.

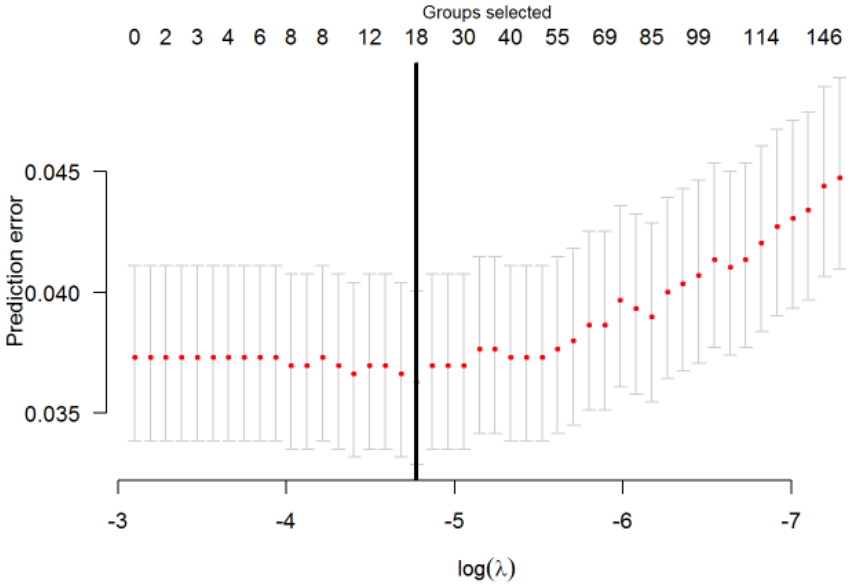


Figure 16: The red dot is the average of 10 result, and the upper and lower bound is the one standard error. The vertical line corresponds to the minimum error. The top panel is the group number of non-zero variables.

After choosing the best model, we may get these results:

- The best lambda is 0.008448.
- There are 19 no-zero variables.
- Correct classification rate of training data: 0.9640556
- Area under curve of training data: 0.9149669
- Correct classification rate of test data: 0.9633028
- Area under curve of test data: 0.8322684

Also, we may get β^{GL} and the final model. Since there are too many non-zero variable selected, we omit it here. You may take a look at the link in the appendix.

7.4.3 Best lambda through BIC

We choose the best model through BIC. Here the definition of BIC is:

$$\mathrm{BIC}(\lambda) = -2\ln \mathrm{Loglik}(\beta^{\mathrm{GL}}(\lambda)) + \mathrm{df}(\lambda)\ln \mathrm{n} \tag{43}$$

As discussed before, the unbiased estimator of DF is quite complex and is not practical in use. Thus, we throw the complicated minor term.

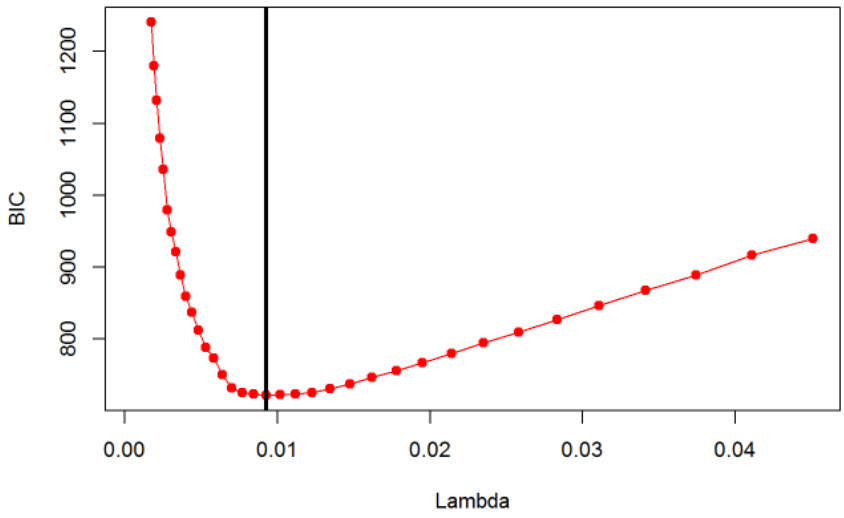


Figure 17: The red dot is bic value of a specific lambda. The vertical line corresponds to the minimum error.

After choosing the best model, we may get these results:

- The best lambda is 0.0092717.
- There are 17 no-zero variables.
- Correct classification rate of training data: 0.9640556
- Area under curve of training data: 0.9133049
- Correct classification rate of test data: 0.9633028
- Area under curve of test data: 0.8343222

Also, we may get β^{GL} and the final model. Since there are too many non-zero variable selected, we omit it here. You may take a look at the link in the appendix.

7.4.4 Best lambda through EBIC

We choose the best model through EBIC with $\gamma = 0.5$.

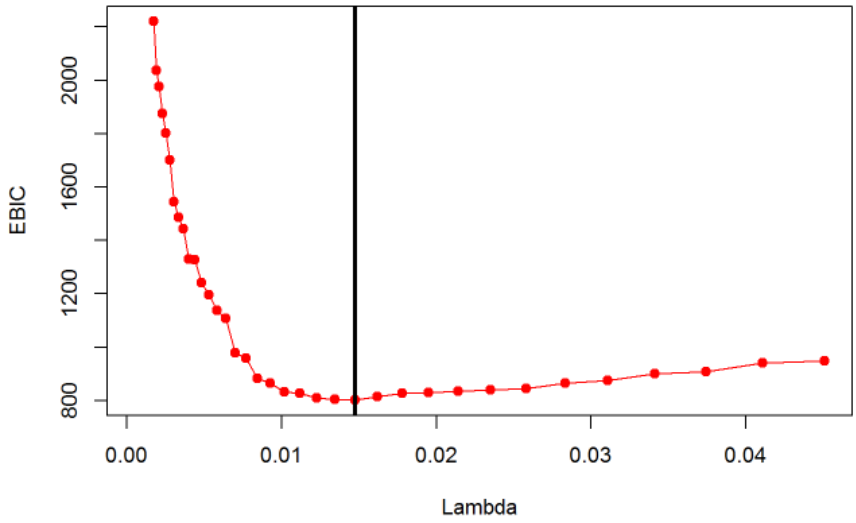


Figure 18: The red dot is ebic value of a specific lambda. The vertical line corresponds to the minimum error.

After choosing the best model, we may get these results:

- The best lambda is 0.0152026.
- There are 9 no-zero variables.
- Correct classification rate of training data: 0.9630383
- Area under curve of training data: 0.9036424
- Correct classification rate of test data: 0.9633028
- Area under curve of test data: 0.8409402

Also, we may get β^{GL} and the final model.

$$\begin{aligned}\ln\frac{p}{1-p} &= (-0.75) + (0.13) * gender2 + (-0.15) * anti\_ht1 \\ &\quad + (-0.93) * anti\_chol1 + (-0.37) * drugs\_others1 + (0.30) * chol \\ &\quad + (0.01) * GFR\_EPI + (0.35) * smkyn2 + (1.81) * ang2\end{aligned}$$

7.5 Sparse Group LASSO

As we have discussed before, $\alpha = 0.5, 1$ is two recommended choices to balance positive selection rate (PSR) and false discovery rate (FDR). In this paper, we just use $\alpha = 0.5$

We use a series of λ to fit the logistic Sparse Group LASSO.

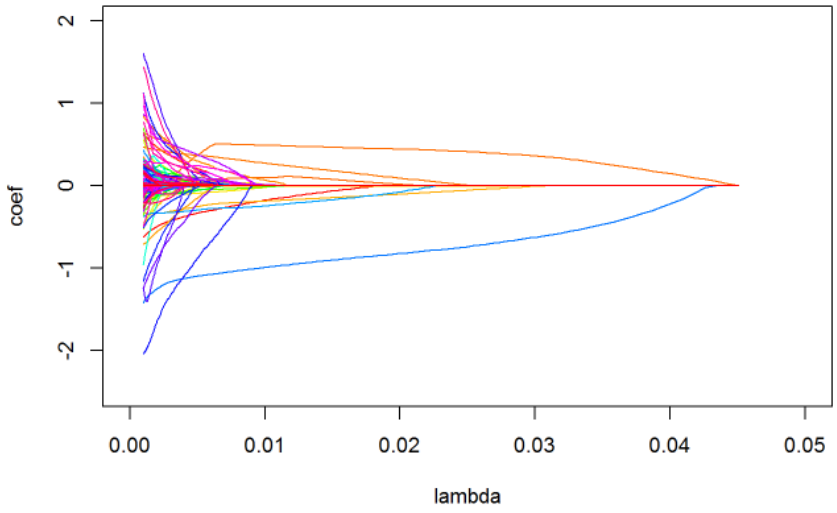


Figure 19: Coefficient path of lambda. Each curve represents a coefficient.

Figure 19 shows the coefficient path of lambda where lasso estimates as a function of λ. If λ grows bigger, the non-zero variables become smaller.

Now we determine the best λ and the best model from this series of λ.

7.5.1 Best lambda through CV: dev

We choose the best model through 10 fold CV: dev.

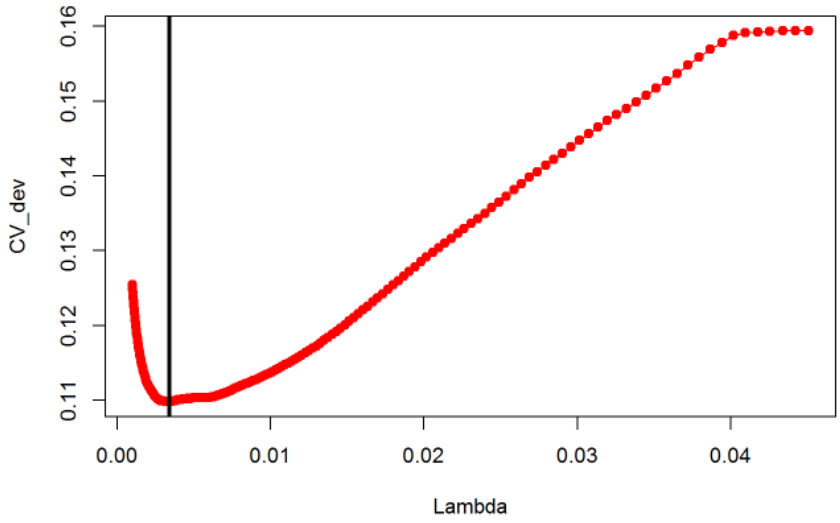


Figure 20: The red dot is the average of 10 result The vertical line corresponds to the minimum error.

After choosing the best model, we may get these results:

- The best lambda is 0.003403494
- There are 74 no-zero variables.
- Correct classification rate of training data: 0.9671075
- Area under curve of training data: 0.9402350
- Correct classification rate of test data: 0.9602446
- Area under curve of test data: 0.8404838

Also, we may get β^{SGL} and the final model. Since there are too many non-zero variable selected, we omit it here. You may take a look at the link in the appendix.

7.5.2 Best lambda through CV: ME

We choose the best model through 10 fold CV: ME.

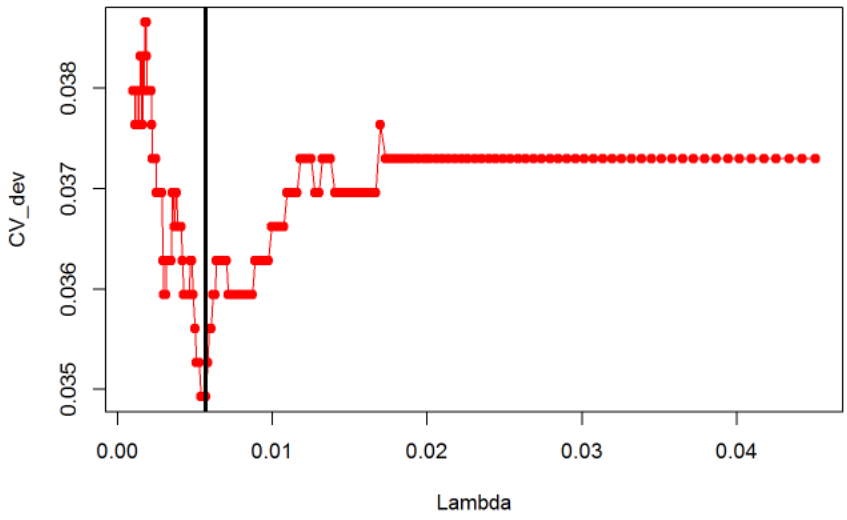


Figure 21: The red dot is the average of 10 result. The vertical line corresponds to the minimum error.

After choosing the best model, we may get these results:

- The best lambda is 0.005705989
- There are 38 no-zero variables.
- Correct classification rate of training data: 0.9657511
- Area under curve of training data: 0.9245413
- Correct classification rate of test data: 0.9602446
- Area under curve of test data: 0.8457325

Also, we may get β^{SGL} and the final model. Since there are too many non-zero variable selected, we omit it here. You may take a look at the link in the appendix.

7.5.3 Best lambda through BIC

We choose the best model through BIC. Here the definition of BIC is:

$$\mathrm{BIC}(\lambda) = -2\ln \mathrm{Loglik}(\beta^{\mathrm{SGL}}(\lambda)) + \mathrm{df}(\lambda)\ln \mathrm{n} \tag{44}$$

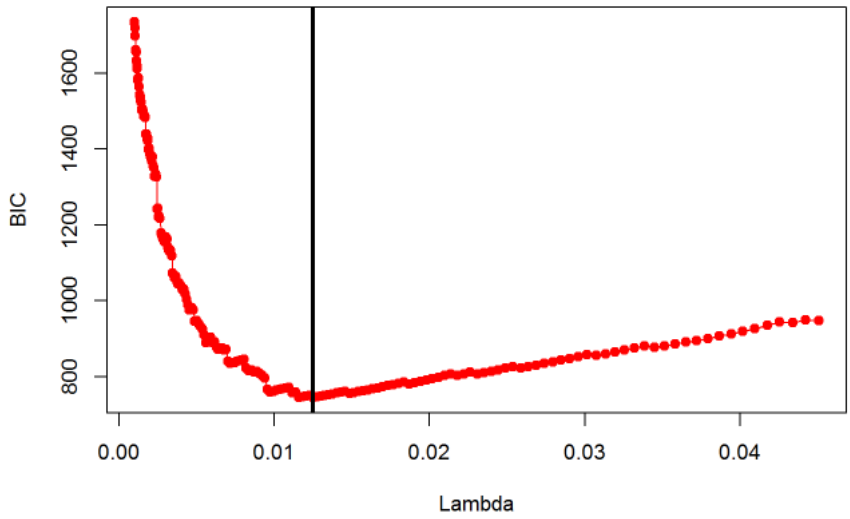


Figure 22: The red dot is bic value of a specific lambda. The vertical line corresponds to the minimum error.

After choosing the best model, we may get these results:

- The best lambda is 0.012505337
- There are 10 no-zero variables.
- Correct classification rate of training data: 0.9640556
- Area under curve of training data: 0.9070127
- Correct classification rate of test data: 0.9633028
- Area under curve of test data: 0.8400274

Also, we may get β^{SGL} and the final model.

$$\begin{aligned}\ln \frac{p}{1-p} &= (-2.265) + (0.118) * gender2 + (-0.002) * age \\ &+ (-0.104) * anti\_ht1 + (-0.477) * anti\_chol1 + (-0.226) * drugs\_others1 \\ &+ (0.166) * chol + (0.006) * GFR\_EPI + (0.207) * smkyn2 \\ &+ (0.947) * ang2\end{aligned}$$

7.5.4 Best lambda through EBIC

We choose the best model through EBIC with $\gamma = 0.5$.

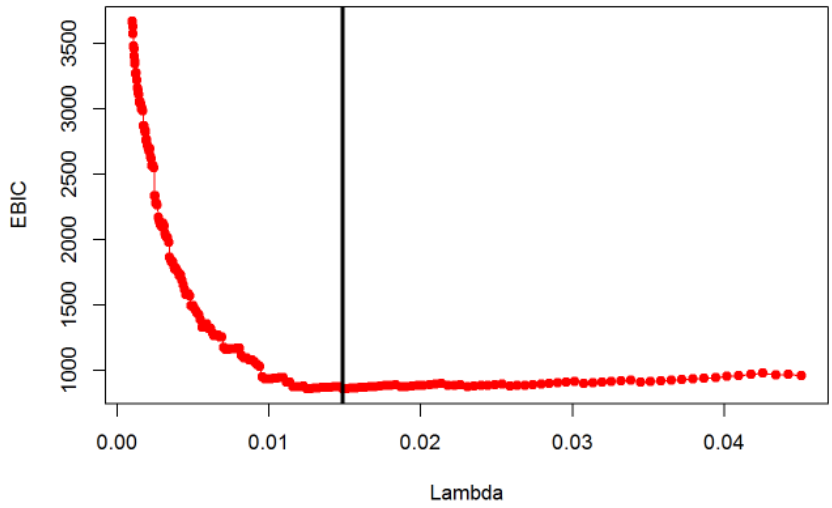


Figure 23: The red dot is ebic value of a specific lambda. The vertical line corresponds to the minimum error.

After choosing the best model, we may get these results:

- The best lambda is 0.014855845
- There are 9 no-zero variables.
- Correct classification rate of training data: 0.9630383
- Area under curve of training data: 0.9036008
- Correct classification rate of test data: 0.9633028
- Area under curve of test data: 0.8411684

Also, we may get β^{SGL} and the final model.

$$\begin{aligned}\ln \frac{p}{1-p} &= (-2.192) + (0.067) * gender2 + (-0.078) * anti\_ht1 \\ &+ (-0.467) * anti\_chol1 + (-0.185) * drugs\_others1 + (0.146) * chol \\ &+ (0.006) * GFR\_EPI + (0.172) * smkyn2 + (0.904) * ang2\end{aligned}$$

Remark 7 *We standardize all the predictors when fitting these methods, and the coefficients are un-standardized to the original scale. The purpose of standardization is to create a common scale for features because there is a penalization term. Since we also standardize the dummy variables, it indeed destroy the sparsity, however it is worthwhile.*

7.6 Comparison

7.6.1 LASSO Based Models

To sum up all the statistics in the previous section, we get this table:

	lambda	num_non_zero	CCR_train	AUC_train	CCR_test	AUC_test
CV_dev_L	0.004	63	0.968	0.937	0.960	0.846
CV_ME_L	0.011	14	0.964	0.911	0.963	0.852
BIC_L	0.010	15	0.964	0.913	0.963	0.851
EBIC_L	0.016	10	0.963	0.900	0.963	0.846
CV_dev_GL	0.006	37	0.965	0.920	0.960	0.839
CV_ME_GL	0.008	19	0.964	0.915	0.963	0.832
BIC_GL	0.009	17	0.964	0.913	0.963	0.834
EBIC_GL	0.015	9	0.963	0.904	0.963	0.841
CV_dev_SGL	0.003	74	0.967	0.940	0.960	0.840
CV_ME_SGL	0.006	38	0.966	0.925	0.960	0.846
BIC_SGL	0.013	10	0.964	0.907	0.963	0.840
EBIC_SGL	0.015	9	0.963	0.904	0.963	0.841

Also, we plot figures of four evaluation values.

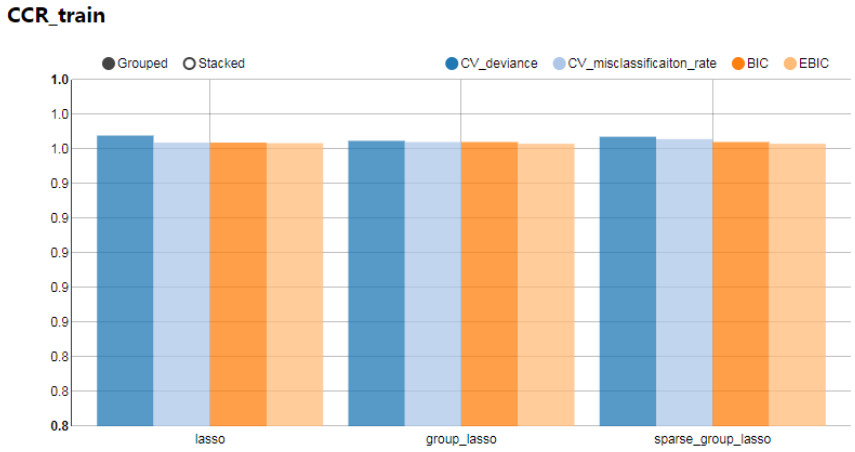


Figure 24: CCR value of training dataset of 12 models.

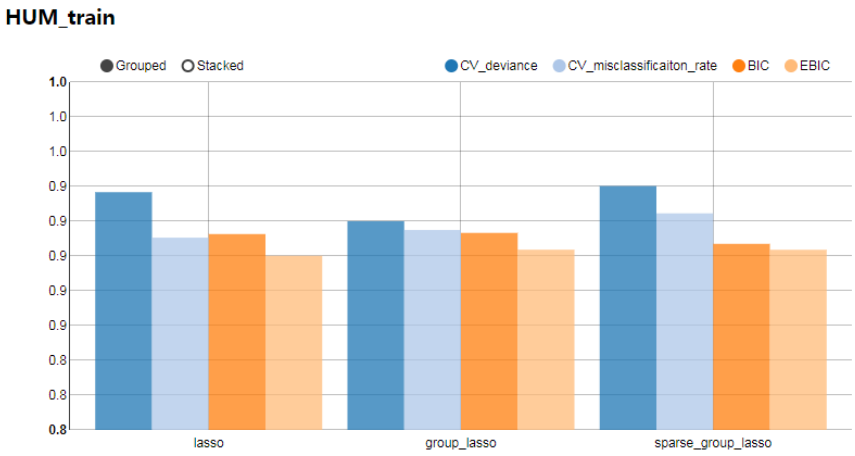


Figure 25: AUC value of training dataset of 12 models.

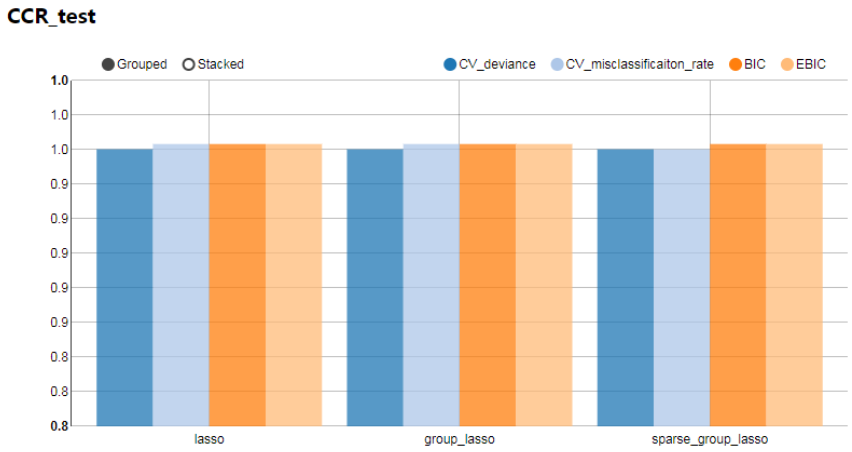


Figure 26: CCR value of test dataset of 12 models.

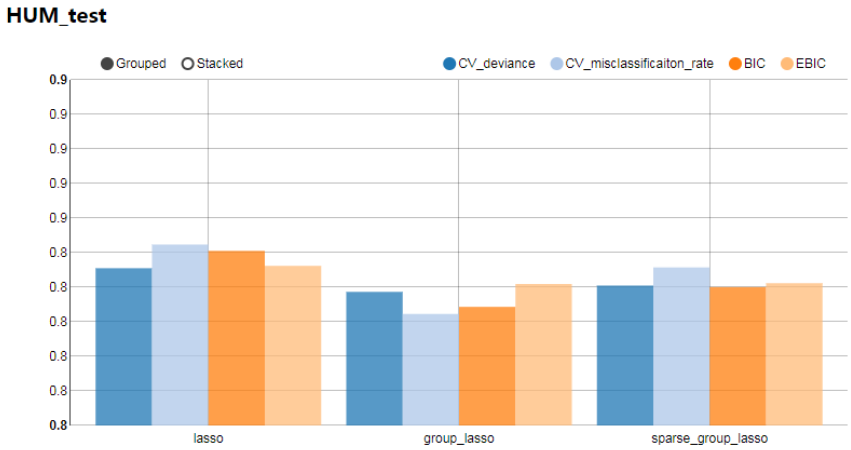


Figure 27: AUC value of test dataset of 12 models.

Several observations can be made from this table. First, CV choose the most predictors and EBIC choose the least, just the same as we discussed. Second, the smallest number of non zero is 9 including the interception. Third, we focus on the HUM (AUC) value on the test data. The smallest value is 0.832 containing 19 non-zero variables, and the largest value is 0.852 containing 14 variables. The EBIC criterion is of top favourite since it both achieve small number of variable and median AUC value.

Next, we evaluate paired models. SGL with EBIC and GL with EBIC achieve the same predictors selected. Thus, we only consider SGL with EBIC as our benchmark model. After thoroughly going through all these 12 models, we find that all the other 11 models select these 9 predictors of the benchmark model. Hence, it is a nest model comparison problem, which means that we could use NRI as the index. We compare all the other 11 models with the default one and calculate the improvement.

The table shows some statistics. The first column is the NRI improvement of the selected model versus SGL with EBIC, and the second column is the bootstrap standard error of NRI. We can find that the top NRI improvement is the largest model containing more than 30 predictors, however this model has big standard error. Since the difference of NRI is quite small, we may assert that SGL with EBIC is the best with 9 predictors and 0.841 AUC value.

7.6.2 Other Models

As a comprehensive comparison, we give other available models.

Logistic Model First, we do not penalize at all and fit a binary logistic model in the same training dataset. The CCR value on the test dataset is 0.927 and

	NRI	Standard Error
CV_dev_L	0.11	0.06
CV_ME_L	0.03	0.01
BIC_L	0.04	0.02
EBIC_L	-0.01	0.00
CV_dev_GL	0.07	0.03
CV_ME_GL	0.05	0.03
BIC_GL	0.05	0.03
EBIC_GL	0.00	0.00
CV_dev_SGL	0.12	0.06
CV_ME_SGL	0.09	0.03
BIC_SGL	0.02	0.01
EBIC_SGL	0.00	0.00

the AUC value on the test dataset is 0.728, which is much worse. It implies that feature selection is helpful.

Multiple Layer Perceptron Second, we implement the simplest deep learning approach: Multiple Layer Perceptron (MLP). Actually MLP is the deep version of logistic regression. If there is no hidden layer, MLP degrades to ordinary logistic model. We construct two hidden layers, first contains 500 neurons and second 100 neurons. Both of them are activated by Relu function. Also, we make some tricks: a dropout layer is built between these two hidden layers, and we use SGD (stochastic gradient decent) with momentum 0.9 to update weights. We use 40 batch size, 0.0004 learning rate and 60 iterations. The graph is shown below.

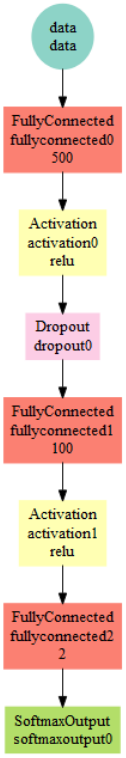


Figure 28: The graph of MLP structure.

The AUC value on the test dataset is 0.86 which is the top of all the models. I have to admit that I am just a beginner of deep learning and thus I am not sure whether this result could be refined (which I believe is of high probability). Hence, it is no doubt that deep learning method can obtain state of art goodness of fit, especially when numbers of samples and predictors exceed to infinity. However, it is quite hard for us to interpret the model and give clinic instructions.

7.6.3 Important Predictors

We choose SGL with EBIC as the final model. Here, p is the predicted probability that one doesn't get heart attack. Hence, we get the equation:

$$\begin{aligned}\ln \frac{p}{1-p} &= (-1.473) + (0.067) * gender2 + (-0.078) * anti\_ht1 \\ &+ (-0.467) * anti\_chol1 + (-0.185) * drugs\_others1 + (0.146) * chol \\ &+ (0.006) * GFR\_EPI + (0.172) * smkyn2 + (0.904) * ang2\end{aligned}$$

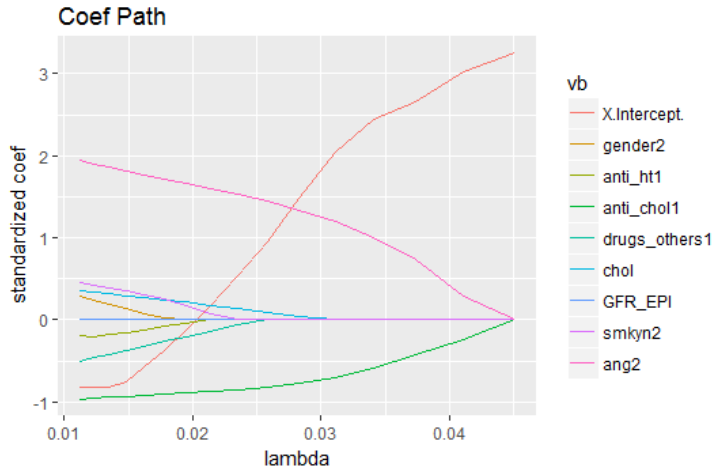


Figure 29: Coef path of these nine predictors. Note that we do not penalize the interception.

The coefficient path and meanings of these predictors are stated in the table.

	Variable's code	Meaning	Type	Range
1	gender2	gender	binary	1:female
2	anti_ht1	Anti-hyperstensive drugs	binary	1:yes
3	anti_chol1	Anti-cholesterol drugs	binary	1:yes
4	drugs_others1	Drugs - Others	binary	1:yes
5	chol	Blood Total Cholesterol	continuous	/
6	GFR_EPI	Glomerular Filtration Rate (EPI)	continuous	/
7	smkyn2	Have you ever smoked?	binary	1:no
8	ang2	Angina (self-reported history)	binary	1:no

When we standardize these predictors, we can conclude that women who don't intake any drugs, don't suffer from Angina and don't smoke have the least probability to get heart attack.

Remark 8 *It may seems strange that Blood Total Cholesterol is proportional to p. However, when I look deep into some medical references [63], there are two types of cholesterol: LDL (bad) and HDL (good). It is believed that high levels of HDL type of cholesterol removes excess plaque from your arteries, slows its buildup and helps to protect against a heart attack.*

Although no eye data are chosen in this smallest model, when we investigate all the 12 models, several of them are quite important. Here, we regard a variable to be important if in more than 6 out of 12 model, its coefficient is non zero.

	Variable's code	Meaning	Type	Range
1	bvalogr_USA2	Best-corrected visual acuity (BCVA)	three category	1:virtually impa
2	R_retino_cat2	Retinopathy category (Right)	six category	1:mild
3	anisometropia1	Anisometropia indicator	binary	1:yes
4	CSME_any1	Clinically Significant Macular Edema (CSME)	binary	1:yes
5	srepcylr	Cylinder (Right) ,	continuous	/

8 Further Discussion

References

[1] F. E. Harrell, "Ordinal logistic regression," in *Regression modeling strategies*. Springer, 2001, pp. 331–343.

[2] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Advances in neural information processing systems*, 2002, pp. 841–848.

[3] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.* Ieee, 1999, pp. 41–48.

[4] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[5] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[6] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[7] D. B. Fogel, L. J. Fogel, and V. Porto, "Evolving neural networks," *Biological cybernetics*, vol. 63, no. 6, pp. 487–493, 1990.

[8] Y. Ning, H. Liu *et al.*, "A general theory of hypothesis tests and confidence regions for sparse high dimensional models," *The Annals of Statistics*, vol. 45, no. 1, pp. 158–195, 2017.

[9] K. R. Gregg, "Krashen's monitor and occam's razor," *Applied linguistics*, vol. 5, no. 2, pp. 79–100, 1984.

[10] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[11] G. Morpurgo, "Forward and backward" selection rule" in a class of inelastic collisions," *Physical Review*, vol. 131, no. 5, p. 2205, 1963.

[12] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[13] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Advances in neural information processing systems*, 2013, pp. 431–439.

[14] A. N. Tikhonov, "On the solution of ill-posed problems and the method of regularization," in *Doklady Akademii Nauk*, vol. 151, no. 3. Russian Academy of Sciences, 1963, pp. 501–504.

[15] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[16] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[17] J. Friedman, T. Hastie, and R. Tibshirani, "A note on the group lasso and a sparse group lasso," *arXiv preprint arXiv:1001.0736*, 2010.

[18] L. Meier, S. Van De Geer, and P. Bühlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 53–71, 2008.

[19] B. Krishnapuram, L. Carin, M. A. Figueiredo, and A. J. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 6, pp. 957–968, 2005.

[20] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[21] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[22] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[23] H. Akaike, "Akaikes information criterion," in *International encyclopedia of statistical science*. Springer, 2011, pp. 25–25.

[24] D. Posada and T. R. Buckley, "Model selection and model averaging in phylogenetics: advantages of akaike information criterion and bayesian approaches over likelihood ratio tests," *Systematic biology*, vol. 53, no. 5, pp. 793–808, 2004.

[25] J. Chen and Z. Chen, "Extended bayesian information criteria for model selection with large model spaces," *Biometrika*, vol. 95, no. 3, pp. 759–771, 2008.

[26] J. Li, B. Jiang, and J. P. Fine, "Multicategory reclassification statistics for assessing improvements in diagnostic accuracy," *Biostatistics*, vol. 14, no. 2, pp. 382–394, 2012.

[27] J. Huang and C. X. Ling, "Using auc and accuracy in evaluating learning algorithms," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[28] B. Van Calster, Y. Vergouwe, C. W. Looman, V. Van Belle, D. Timmerman, and E. W. Steyerberg, "Assessing the discriminative ability of risk models for more than two outcome categories," *European journal of epidemiology*, vol. 27, no. 10, pp. 761–770, 2012.

[29] J. Li and J. P. Fine, "Roc analysis with multiple classes and multiple tests: methodology and its application in microarray studies," *Biostatistics*, vol. 9, no. 3, pp. 566–576, 2008.

[30] M. J. Pencina, R. B. D'Agostino, and E. W. Steyerberg, "Extensions of net reclassification improvement calculations to measure usefulness of new biomarkers," *Statistics in medicine*, vol. 30, no. 1, pp. 11–21, 2011.

[31] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical programming*, vol. 6, no. 1, pp. 62–88, 1974.

[32] H.-C. Wu, "The karush–kuhn–tucker optimality conditions in an optimization problem with interval-valued objective function," *European Journal of Operational Research*, vol. 176, no. 1, pp. 46–59, 2007.

[33] W. W. Hager and S. K. Mitter, "Lagrange duality theory for convex control problems," *SIAM Journal on Control and Optimization*, vol. 14, no. 5, pp. 843–856, 1976.

[34] P. M. Williams, "Bayesian regularization and pruning using a laplace prior," *Neural computation*, vol. 7, no. 1, pp. 117–143, 1995.

[35] B. Efron, "Double exponential families and their use in generalized linear regression," *Journal of the American Statistical Association*, vol. 81, no. 395, pp. 709–721, 1986.

[36] P. McCullagh, "Generalized linear models," *European Journal of Operational Research*, vol. 16, no. 3, pp. 285–292, 1984.

[37] I. Kurt, M. Ture, and A. T. Kurum, "Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease," *Expert systems with applications*, vol. 34, no. 1, pp. 366–374, 2008.

[38] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-task regression with structured sparsity," 2010.

[39] C.-W. Chang, D. A. Laird, M. J. Mausbach, and C. R. Hurburgh, "Near-infrared reflectance spectroscopy–principal components regression analyses of soil properties," *Soil Science Society of America Journal*, vol. 65, no. 2, pp. 480–490, 2001.

[40] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, vol. 33, no. 1, p. 1, 2010.

[41] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of optimization theory and applications*, vol. 109, no. 3, pp. 475–494, 2001.

[42] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[43] J. Friedman, T. Hastie, H. Höfling, R. Tibshirani *et al.*, "Pathwise coordinate optimization," *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[44] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[45] M. Vincent and N. R. Hansen, "Sparse group lasso and high dimensional multinomial classification," *Computational Statistics & Data Analysis*, vol. 71, pp. 771–786, 2014.

[46] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics-theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977.

[47] P.-L. Liu and A. Der Kiureghian, "Optimization algorithms for structural reliability," *Structural safety*, vol. 9, no. 3, pp. 161–177, 1991.

[48] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Mathematical Programming*, vol. 117, no. 1-2, pp. 387–423, 2009.

[49] M. W. Browne, "Cross-validation methods," *Journal of mathematical psychology*, vol. 44, no. 1, pp. 108–132, 2000.

[50] K. P. Burnham and D. R. Anderson, "Multimodel inference: understanding aic and bic in model selection," *Sociological methods & research*, vol. 33, no. 2, pp. 261–304, 2004.

[51] Y. Yang, "Can the strengths of aic and bic be shared? a conflict between model indentification and regression estimation," *Biometrika*, vol. 92, no. 4, pp. 937–950, 2005.

[52] H. Zou, "The adaptive lasso and its oracle properties," *Journal of the American statistical association*, vol. 101, no. 476, pp. 1418–1429, 2006.

[53] F. K. Hui, D. I. Warton, and S. D. Foster, "Tuning parameter selection for the adaptive lasso using eric," *Journal of the American Statistical Association*, vol. 110, no. 509, pp. 262–269, 2015.

[54] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *The annals of Statistics*, pp. 1135–1151, 1981.

[55] H. Huynh and L. S. Feldt, "Estimation of the box correction for degrees of freedom from sample data in randomized block and split-plot designs," *Journal of educational statistics*, vol. 1, no. 1, pp. 69–82, 1976.

[56] G. H. Golub, M. Heath, and G. Wahba, "Generalized cross-validation as a method for choosing a good ridge parameter," *Technometrics*, vol. 21, no. 2, pp. 215–223, 1979.

[57] H. Zou, T. Hastie, R. Tibshirani *et al.*, "On the degrees of freedom of the lasso," *The Annals of Statistics*, vol. 35, no. 5, pp. 2173–2192, 2007.

[58] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[59] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[60] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[61] J. Hilden and T. A. Gerds, "A note on the evaluation of novel biomarkers: do not rely on integrated discrimination improvement and net reclassification index," *Statistics in medicine*, vol. 33, no. 19, pp. 3405–3414, 2014.

[62] K. F. Kerr, Z. Wang, H. Janes, R. L. McClelland, B. M. Psaty, and M. S. Pepe, "Net reclassification indices for evaluating risk-prediction instruments: a critical review," *Epidemiology (Cambridge, Mass.)*, vol. 25, no. 1, p. 114, 2014.

[63] B. H. Olson, S. M. Anderson, M. P. Becker, J. W. Anderson, D. B. Hunninghake, D. J. Jenkins, J. C. LaRosa, J. M. Rippe, D. C. Roberts, D. B. Stoy *et al.*, "Psyllium-enriched cereals lower blood total cholesterol and ldl cholesterol, but not hdl cholesterol, in hypercholesterolemic adults: results of a meta-analysis," *The Journal of nutrition*, vol. 127, no. 10, pp. 1973–1980, 1997.