Optimization Methods - Mid-Term Project
Implement of Linear Regression Logistic Regression with L1 Regularization

Jay Chiehen Liao (廖傑恩)†

Institute of Data Science
National Cheng Kung University
Tainan City, Taiwan
re6094028@gs.ncku.edu.tw

Abstract

This project aimed to construct the linear regression with L1 regularization and the logistic regression with L1 regularization. We also aim to utilize cross-validation to tune the best hyper-parameter λ in both tasks. Comparing results of numerical experiment, we found that the performance of linear regression was not stable against the change of λ. It is possible that the logistic form is more appropriate in the task of binary classification. The logistic regression with L1 regularization can achieve a reliable performance in the large number of predictors. All source codes and experiment procedures are available on GitHub: https://github.com/jayenliao/Optimization-L1-regularization.

Keywords

L1 regularization, LASSO, logistic regression, variable selection, cross-validation

1 Introduction

Consider the typical scenario of linear regression problem: given the dependent variable $\mathbf{y} \in \mathbb{R}^N$ and independent variables $\mathbf{X} \in \mathbb{R}^{N \times P}$, where N is the number of observations and P is the number of independent variables, we aim to find a function (i.e., model) f that best describes and predicts $\mathbf{y}$ by using $\mathbf{X}$ with a linear form:

$$\mathbf{y} = \beta_0 + \tilde{\beta}\mathbf{X}^T + \epsilon,\ \epsilon \sim N(0,\sigma^2),$$

where $i = 1,...,N$, β_0 is the intercept, $\tilde{\beta} \in \mathbb{R}^{1 \times P}$, and ϵ is the error term which follows the normal distribution with zero mean and variance σ^2. With the method of the ordinary least squared (OLS), the distance between the i^{th} true value y_i and the i^{th} fitted value $f(\mathbf{X}_i)$ is quantified by the squared error loss function:

$$L(y_i,\ f(\mathbf{X}_i)) = |\, y_i - \beta_0 - \tilde{\beta}\mathbf{X}_i^T \,|^2,$$

The OLS estimates of regression coefficients, denoted as $\hat{\beta} = [\hat{\beta}_j], \forall j = 0,...,P$, are obtained by minimizing the OLS estimation criterion (Hutcheson, 2011):

$$\mathfrak{D}_{OLS}(\beta) = \frac{1}{N}|\, y_i - \beta_0 - \tilde{\beta}\mathbf{X}_i^T \,|^2.$$

However, there are two reasons why the OLS estimates are not satisfying enough. The first one is *prediction accuracy*. The OLS estimates often have low bias but large variance. Prediction accuracy of regression model can be improved by shrinking or setting 0 on some coefficients (Endelman & Jannink, 2012). The second reason is *model interpretation*. With a large number of predictors (i.e., P), we often would like to determine a smaller subset that demonstrate strong effects and explained well (Helland, 1987).

We can utilize regularization to shrink regression coefficients. In regularized methods, the parameter is estimated via minimizing an estimation criterion with constraints:

$$\min_{\beta} \mathfrak{D}(\beta), \text{ subject to } \mathfrak{R}(\beta) \leq C,$$

where $\mathfrak{R}(\beta)$ is a regularizer (or penalty) measuring the "complexity" of β, and C is a positive constant representing some kind of "budget".

Ridge regression is a common-used shrinking approach that adopts with L2 regularization:

$$\min_{\beta}[SS_E + \lambda \sum_{j=1}^{P} \beta_j^2],$$

where $SS_E = \sum_{i=1}^{N} (y_i - f(\mathbf{X}_i))^2$ is the error sum of squares and λ is a hyper-parameter. Ridge regression is

a continuous process that shrinks coefficients and thus is more stable. However, it can not set any coefficients to 0 and hence is not able to achieve predictor selection, which often leads to difficulty in model interpretation.

As a result, Tibshirani (1996) tried to FIGure out another approach and proposed a new technique, least absolute shrinkage a selection operator (LASSO), which shrinks some coefficients to non-zero values and sets others to 0. Therefore, LASSO is able to retain useful predicates of both subset selection and ridge regression. The LASSO estimates are defined as:

$$\hat{\beta} = [\hat{\beta}_j] = \arg\min[SS_E], \forall j = 0,1,...,P,$$

subject to $\sum_{j=1}^{P} |\beta_j| \leq C$. Now LASSO approach is often called as L1 regularization, which can be denoted as:

$$\min_{\beta}[SS_E + \lambda \sum_{j=1}^{P} |\beta_j|],$$

where SS_E is the error sum of squares and λ is a hyper-parameter. Moreover, when treating linear regression as a kind of machine learning model, regressions with L1 and L2 regularization are common-used methods to avoid underfitting and overfitting.

In addition to the linear regression, regularization approach also can be utilized in logistic regression when we consider a binary classification problem: given the dependent variable $\mathbf{y} \in \mathbb{R}^N$ where $y_i = 0,1$, for $i = 1,...,N$ and independent variables $\mathbf{X} \in \mathbb{R}^{N \times P}$, we aim to find a function f that best describes and predicts $\mathbf{y}$ by using $\mathbf{X}$ with a probability form:

$$\Pr[y_i = 1] = \beta_0 + \tilde{\beta}\mathbf{X}_i + \epsilon_i, \forall i = 1,...,N.$$

Friedman et al. (2010) applied the idea of LASSO to generalized linear models (GLMS). They developed fast algorithms for estimation of generalized linear models with convex penalties. The models include linear regression, binary logistic regression, and multinomial regression problems while the penalties include L1 (i.e., the lasso regression), L2 (i.e., the ridge regression) and mixtures of the two (i.e., the elastic net).

In the presenting project, we aim to follow Tibshirani (1996) and Friedman et al. (2010) to apply LASSO approach to the linear regression and the logistic regression. We would like conduct numerical experiment on a large dataset to evaluate models. The followings are purposes of this project:

1. Use cross-validation (cv) to tune the best hyper-parameter λ in the task of linear regression with L1 regularization.
2. Use cv to tune the best hyper-parameter λ in the task of logistic regression with L1 regularization.
3. Compare results of numerical experiment of problem 1 and 2 to explore difference between linear regression and logistic regression.

2 Methods

2.1 Dataset

To demonstrate the performance of linear regression and logistic regression with LASSO approach, a well-known artificial wave dataset (Breiman et al., 1984) is used in this project. Originally, there were three classes with 21 variables in this wave dataset, and the variables of each class were generated based on a random convex combination of two of three wave forms with noise. Rakotomalala (2005) expanded the dataset by adding noise variables and generating more subjects and also modified this dataset as a binary response dataset by only keeping the subjects of the first two classes. We used the binary version, which consists of 10,000 observations and 121 predictors in the presenting project.

2.2 Algorithm of optimization

Following Friedman et al. (2010), we utilized cyclical coordinate descent to compute along a regularization path for both tasks of the linear regression and logistic regression.

2.2.1 Regularized linear regression

Consider a coordinate descent step for solving $\min_{\beta}[SS_E + \lambda \sum_{j=1}^{P} |\beta_j|]$. That is, suppose we have estimates $\hat{\beta}_0$ and $\hat{\beta}_k \forall j \neq k$, and we want to partially optimize with respect to β_j. We would like to compute the gradient at $\beta_j = \hat{\beta}_j$, which only exists if $\hat{\beta}_j \neq 0$. If $\hat{\beta}_j > 0$, then

$$\frac{\partial R_\lambda}{\partial \beta_j}|_{\beta=\hat{\beta}} = \frac{-1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \hat{\beta}_0 - \hat{\beta}\mathbf{X}_i^T) + \lambda|\beta|_j.$$

A similar expression exists if $\hat{\beta}_j < 0$, and $\hat{\beta}_j = 0$ is treated separately. Simple calculus shows (Donoho and Johnstone, 1994) that the coordinate-wise update has the form:

$$\hat{\beta}_j \leftarrow S(\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \hat{y}_i^{(j)}), \lambda),$$

where

(1) $\hat{y}_i^{(j)} = \hat{\beta}_0 + \sum_{j \neq k} x_{ik}\hat{\beta}_k$ is the fitted value excluding the contribution from x_{ij}, and thus $y_i - \hat{y}_i^{(j)}$ is the partial residual for fitting β_j. Due to the standardization, $\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \hat{y}_i^{(j)})$ is the simple LS coefficient when fitting this partial residual to x_{ij}.

(2) $S(z, \gamma)$ is the soft-thresholding operator with value

$$sign(z)(|z| - \gamma)_+ = \begin{cases} z - \gamma, \; if \; z > 0 \; \& \; \gamma < |z| \\ z + \gamma, \; if \; z < 0 \; \& \; \gamma < |z| \\ 0, \quad if \; \gamma \geq |z| \end{cases}$$

Thus we compute the simple least-squares coefficient on the partial residual, apply soft-thresholding to take care of the lasso contribution to the penalty, and then apply a proportional shrinkage for the ridge penalty. This algorithm was suggested by Van der Kooij (2007).

2.2.2 Regularized logistic regression

When the response variable is binary, the linear logistic regression model is often used. Denote by G the response variable, taking values in {1, 0} (the labeling of the elements is arbitrary). The logistic regression model represents the class-conditional probabilities through a linear function of the predictors:

$$\Pr[Y = 1 \,|\, x] = 1/[1 + e^{-(\beta_0 + \beta x^T)}]$$

$$\Pr[Y = 0 \,|\, x] = 1/[1 + e^{+(\beta_0 + \beta x^T)}] = 1 - \Pr[Y = 1 \,|\, x],$$

which implies that $\log \frac{\Pr[Y = 1 \,|\, x]}{\Pr[Y = 0 \,|\, x]} = \beta_0 + \beta x^T$. Here we fit this model by regularized maximum (binomial) likelihood. Let $p(x_i) = \Pr[Y = 1 \,|\, x_i]$ be the probability for the i^{th} observation at a particular value for the parameters $\beta = [\beta_j], \forall j = 0,...,P$, then we maximize the penalized log likelihood

$$\max_{\beta \in \mathbb{R}^{P+1}} [\frac{1}{N}\sum_{i=1}^{N} (I(y_i = 1)\log p(x_i) + I(y_i = 0)\log(1 - p(x_i))) - \lambda \sum_{j=1}^{P} |\beta_j|]$$

We denote $y_i^{(1)} = I(y_i = 1)$. he Newton algorithm for maximizing the (unpenalized) log-likelihood amounts to iteratively re-weighted LS. Therefore, if the current estimates of the parameters are $\hat{\beta} = [\hat{\beta}_0, \hat{\beta}]$, we form a quadratic approximation to the log-likelihood (Taylor expansion about current estimates), which is

$$l_Q(\beta_0, \beta) = \frac{-1}{2N}\sum_{i=1}^{N} w_i(z_i - \beta_0 - \beta \mathbf{X}_i^T)^2 + C(\hat{\beta}_0, \hat{\beta})^2,$$

where

$$w_i = \hat{p}(x_i)(1 - \hat{p}(x_i)), \quad z_i = \hat{\beta}_0 + \hat{\beta}\mathbf{X}_i^T + \frac{y_i^{(1)} - \hat{p}(x_i)}{w_i},$$

and $\hat{p}(x_i)$ is the estimated $p(x_i)$ evaluated at the current parameters. The last term C is a constant. The Newton update is obtained by minimizing l_Q. Friedman et al. (2010)'s approach is similar. For each value of λ, an outer loop which computes the quadratic approximation l_Q about the current parameters $[\hat{\beta}_0, \hat{\beta}]$ was created. Then the coordinate descent was used to solve the penalized weighted LS problem

$$\min_{[\beta_0, \beta] \in \mathbb{R}^{P+1}} [-l_Q([\beta_0, \beta) + \lambda \sum_{j=1}^{P} |\beta_j|].$$

2.3 Approach of evaluation and tuning of the hyper-parameter

We evaluated models by several common-used metrics, including mean squared error, accuracy score, precision score, recall score, and F1 score, which are described in TABLE 1. Note that in the task of linear regression, we also classified the output as 0 or 1 by setting the threshold of 0.5 on the original continuous output. That is,

$$y_i \leftarrow \begin{cases} 1 \; if \; y_i \geq 0.5 \\ 0, \; otherwise \end{cases}.$$

The running time of models were also recorded.

The hyper-parameter λ was tuned by k-fold cv for both tasks in purpose of getting reliable results. Take $k = 10$ as an example, the original training set was randomly spited into 10 folds. For the i^{th} iteration, fold i is treated as the validation set and remained $k - 1 = 9$ folds are used to train a model. The model is evaluated by the validation set (i.e., fold i). The procedure would be repeated for 10 iterations. Evaluation results of 10 iterations are collected to compute the mean value and standard deviation. λ that leads the model have the best mean performance of cv would be selected.

TABLE 1: Definitions of Evaluation Metrics

Metric	Definition
MSE	Mean squared error. Given the i^{th} true value y_i and the i^{th} fitted value $f(\mathbf{X}_i)$, $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{X}_i))^2$
TP	*True positive*. The number of observations that really belong to the target class and are correctly predicted as target cases by the model.
FP	*False positive*. The number of observations who actually do not belong to the target class but are predicted as target cases by the model.
TN	*True negative*. The number of observations who do not belong to the target class and are not predicted as target cases by the model.
FN	*False negative*. The number of observations who actually belong to the target class but are not correctly predicted as target cases by the model.
Accuracy	(TP + TN) / (TP + FP + TN + FN)
Precision	TP / (TP + FP)
Recall	TP / (TP + FN)
F1-score	The harmonic mean of precision and recall. F1 = 2 / (1 / Precision + 1 / Recall).

TABLE 2: Required Packages

Package	Version	Use
NumPy	1.16.3	Matrix operation
Pandas	1.1.5	Data loading and preprocessing
Spicy	1.4.1	Loading arff file
Scikit-Learn	0.24.1	Models construction, optimization, hyper-parameter tuning with cross-validation, and model evaluation
matplotlib	3.1.3	Visualization of experiments results
os	-	Create output folders
time	-	Timing
datetime	-	Get date and time info
pickle	-	Model saving

TABLE 3: Hyper-parameters Setting for Linear Regression and Logistic Regression with L1 Regularization

Hyper-parameter	Value
Number of folds in cross-validation (i.e., k)	10
Number of λ (i.e., n)	99
Max iteration	10,000
Tolerance	0.0001
Random state	4028

2.4 Implement tools and environment

This project was implemented with Python (version 3.6.9) on a Linux server (Linux Intel Xeon Gold 6138 @2.0GHZ RAM 450G CPU). All required packages are listed as TABLE 2.

3 Numerical experiments and results

The original dataset with 33,334 observations was randomly spitted into two subsets with the ratio of 4:1: 75% observations went to the training set and 25% went to the testing set. The hyper-parameter λ was tuned by 10-fold cv in the range of (0,1] with n candidates with the same bin cut for both tasks. That is, we fitted k folds for each of n candidates and thus had $n \times k$ fits in total in both tasks. The n candidates of λ contain $\frac{1}{n+1}, \frac{2}{n+1}, \ldots, \frac{n}{n+1}$, 1. We did not try the value of 0 for λ to avoid to fit a non-regularized model (i.e., $\lambda = 0$). All hyper-parameters were set as TABLE 3 shows. We set $k = 10$ and $n = 99$. For all parts with randomization, the random state were set as 4028, the last four digits of author's student ID number.

TABLE 4: Testing Performance of Regression Models With L1 Regularization With The Tuned Lambda

Model	Tuned λ	No. of non-zero coeffieents	MSE	Accuracy	Precision	Recall	F1-score
Linear regression	0.0100	10	0.0802	0.9189	0.9334	0.9042	0.9186
Linear regression	0.0012	15	0.0766	0.9234	0.9392	0.076	0.9231
Logistic regression	0.4800	68	0.0766	0.9234	0.9332	0.9147	0.9238

MSE=mean squared error

3.1 Hyper-parameter tuning on linear regression with L1 regularization

Numerical results for the linear regression with L1 regularization were presented on the first two rows of TABLE 4. The hyper-parameter λ was tuned as 0.01, which was the lowest one among 99 candidate values in the range of (0, 1]. Since it was possible that there was a better λ in the range of (0, 0.01], we conducted 10-cv again to tune λ in the range of (0, 0.01] with 99 candidate values. The tuned λ of the second round was 0.0012.

L1-regularized linear regression with $\lambda = 0.0012$ performed very well on the testing set in term of all evaluation metrics. This indicated that most of observations can be predicted correctly by the model. Plots of evaluation metrics except for MSE (Fig 1 & 2) and plots of MSE (Fig 3 & 4) in the λ-tuning process with 10-fold cv demonstrated that the model's excellent performance was very stable against the change of λ value when $\lambda < 0.4$. When λ value got closer to 0.5, model performance became worse intensely. Variances of both precision and F1-score among 10 cv folds were very large when $\lambda > 0.5$, indicating the instability of the model. Plots of mean fitting time (Fig 5 & 6) showed that the fitting time decreases as λ value increases generally. It is possible that larger λ value shrinkage more coefficients to 0 and thus leads to shorter fitting time cost.

Fig 7 is the lollipop plot of coefficients of the linear regression with $\lambda = 0.0012$. It showed that coefficients of most predictors were shrunk to 0 and only some of them were not under such λ value, indicating that LASSO approach really achieved subset selection to remain useful predictors.

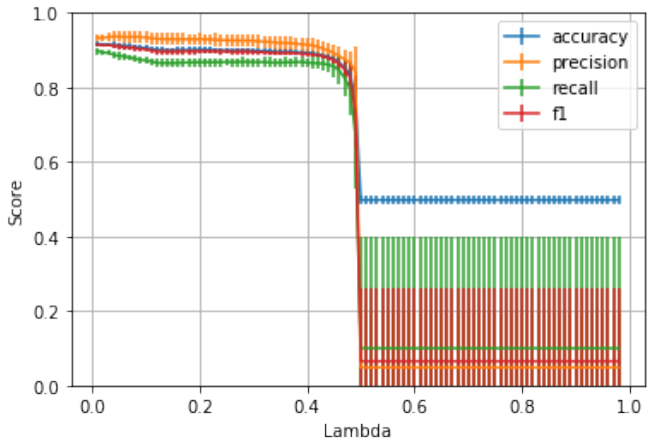

Fig 1: Plot Non-MSE Testing Performance and λ of The Linear Regression with Error Bars in The Range of (0, 1]

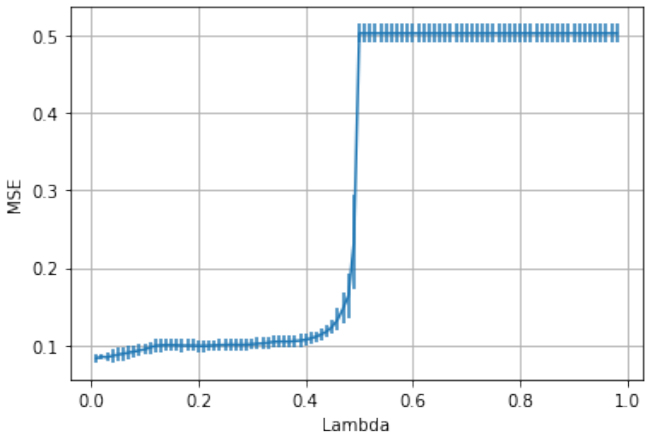

Fig 3: Plot of Testing MSE and λ of The Linear rRegression with Error Bars in The Range of (0, 1]

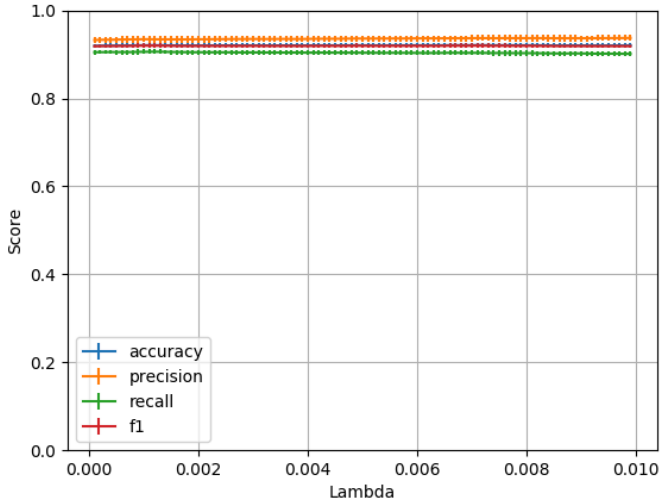

Fig 2: Plot Non-MSE Testing Performance and λ of The Linear Regression with Error Bars in The Range of (0, 0.01]

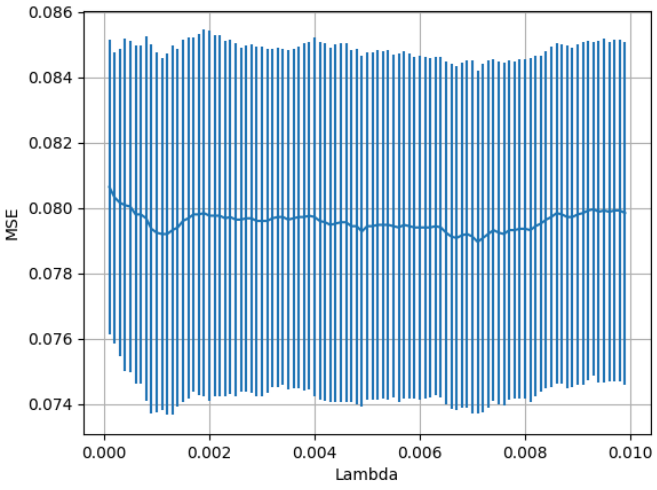

Fig 4: Plot of Testing MSE and λ of The Linear rRegression with Error Bars in The Range of (0, 0.01]

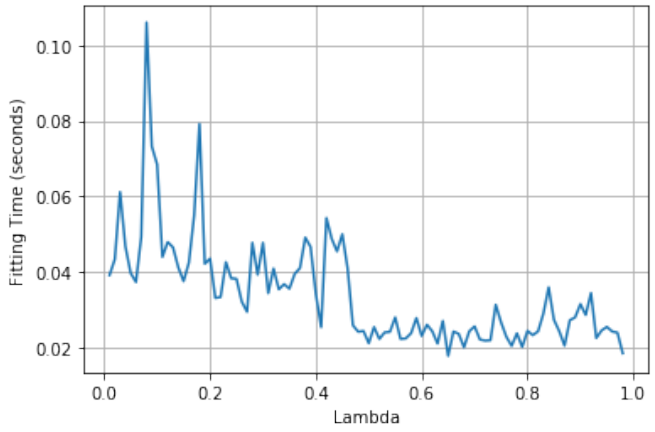

Fig 5: Plot of Mean Fitting Time and λ of The Linear regression in The Range of (0, 1]

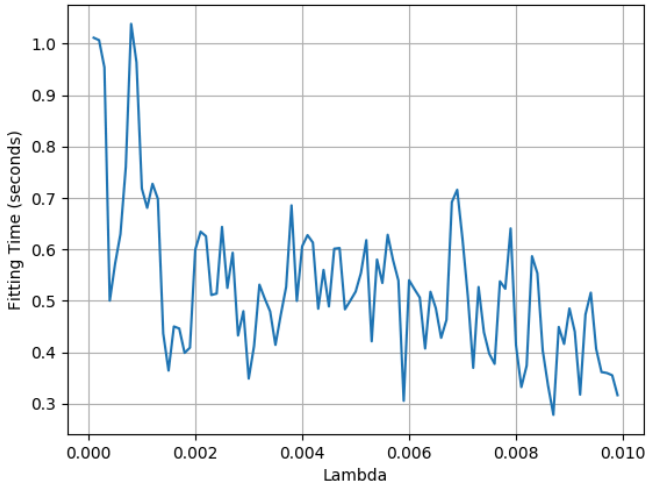

Fig 6: Plot of Mean Fitting Time and λ of The Linear regression in The Range of (0, 0.01]

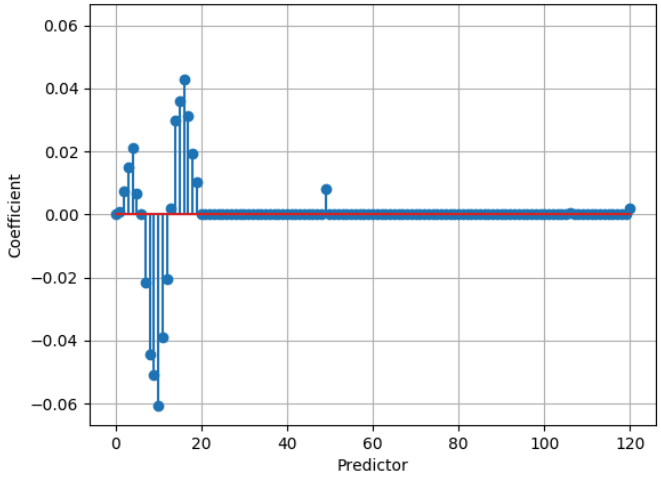

Fig 7: Lollipop Plot of Coefficients of The Logistic regression with $\lambda = 0.0012$

3.2 Hyper-parameter tuning on logistic regression with L1 regularization

Numerical results for the logistic regression with L1 regularization were presented on the last row of TABLE 4. The hyper-parameter λ was tuned as 0.48, which was about the median among 99 candidate values. L1-regularized logistic regression with $\lambda = 0.48$ performed very well on the testing set in term of all evaluation metrics.

Both plot of evaluation metrics except for MSE (Fig 8) and plot of MSE (Fig 9) demonstrated that the model's excellent performance was very stable against the change of λ in the range of candidate values among 10 cv folds. Plot of fitting time (Fig 10) showed that the fitting time decreases as λ value increases. It is possible that larger λ value shrinkage more coefficients to 0 and thus leads to shorter fitting time cost.

Fig 11 is the lollipop plot of coefficients of the logistic regression with $\lambda = 0.48$. It showed that coefficients of some useless predictors were shrunk to 0 but most of them were not under such λ value, indicating that we might really have inputted many influential predictors.

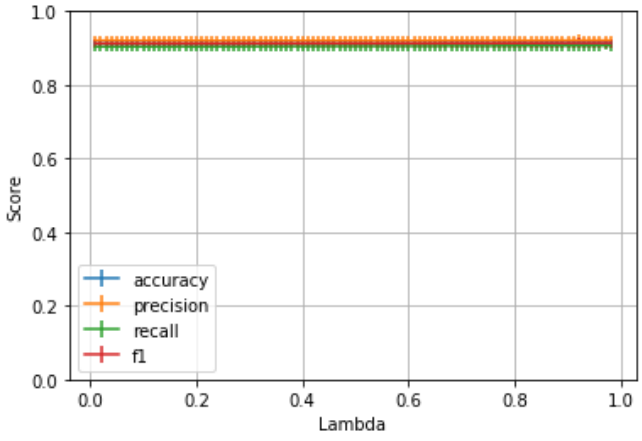

Fig 8: Plot of Non-MSE Testing Performance and λ of The Logistic Regression with Error Bars in The Range of (0, 1]

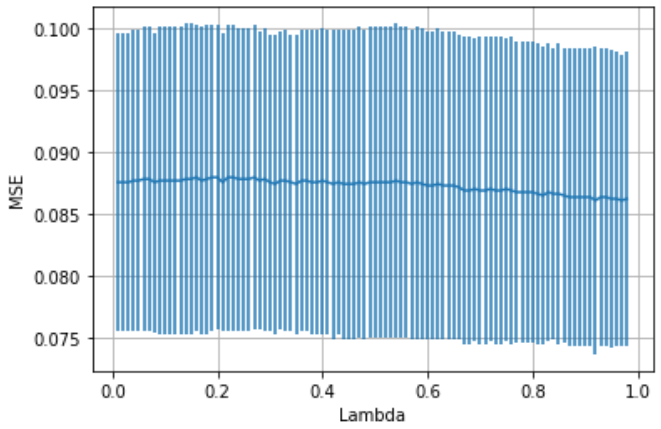

Fig 9: Plot of MSE and λ of The Logistic regression with Error Bars

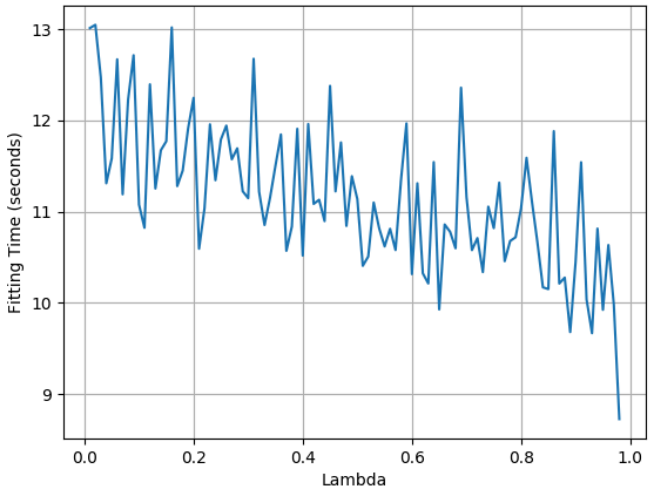

Fig 10: Plot of Mean Fitting Time and λ of The Logistic Regression

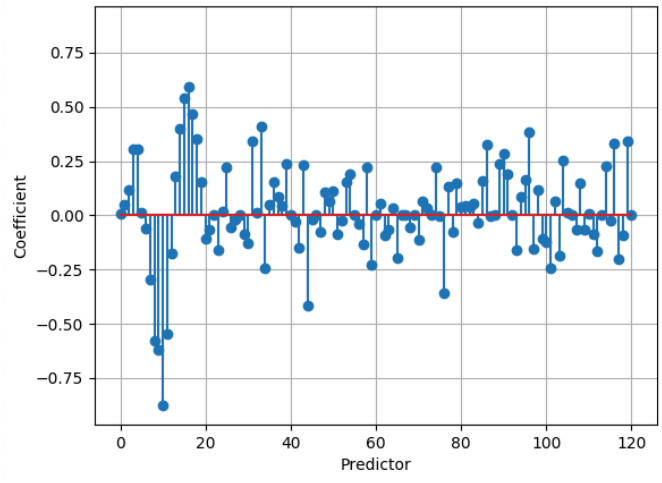

Fig 11: Lollipop Plot of Coefficients of Logistic regression with $\lambda = 0.48$

3.3 Comparison

There was only an extremely small difference between the linear regression with the tuned $\lambda = 0.0012$ and the logistic regression with the tuned $\lambda = 0.48$ when evaluating on the testing set. However, in the λ-tuning processes with 10-fold cv, linear regression models had relatively unstable performance compared to logistic regression models in this dataset. When λ was small enough (i.e., with merely slight L1 regularization), L1-regularized linear regressions gave a promising performance while models performed badly as λ went to 1. On the other hand, L1-regularized logistic regressions had extremely stable and robust performances in term of all evaluation metrics in the range of 99 candidate values of λ in (0,1].

In addition, even if the tuned λ is lower than the lowest one of candidate value of the first tuning round (i.e., with very slight L1 regularization), coefficients of many predictors of the linear regression were shrunk to 0, indicating that these predictors were not useful for predicting the response in the linear form. However, coefficients of only some predictors were shrunk to 0 but most of them were not under the same λ value, indicating that many input predictors were influential to the response in the logistic form.

Moreover, the decreasing pattern of fitting time cost as λ went to 1 was more obvious in logistic regressions than that in linear regressions. It was possible that time cost for each linear regression was lower than that for the logistic regression, making the decreasing pattern not so obvious.

4 Conclusion

We implemented the linear regression and the logistic regression with L1 regularization and tuned hyper-parameter of L1 regularization (i.e., λ) with 10-fold cv. We found that both the linear regression and the logistic regression with L1 regularization provided excellent testing performance when the best λ was tuned and chosen. However, we also found that the performance of linear regression was not stable against the change of λ. It is possible that the logistic form is more appropriate in the task of binary classification. The logistic regression with L1 regularization can achieve a reliable performance in the large number of predictors.

REFERENCE

1. Hutcheson, G. D. (2011). Ordinary least-squares regression. *L. Moutinho and GD Hutcheson, The SAGE dictionary of quantitative management research*, 224-228.
2. Endelman, J. B., & Jannink, J. L. (2012). Shrinkage estimation of the realized relationship matrix. *G3: Genes| genomes| genetics*, *2*(11), 1405-1413.
3. Helland, I. S. (1987). On the interpretation and use of R2 in regression analysis. *Biometrics*, 61-69.
4. Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267-288.
5. Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1.
6. Breiman, L., & Ihaka, R. (1984). *Nonlinear discriminant analysis via scaling and ACE*. Department of Statistics, University of California.
7. Rakotomalala, R. (2005). TANAGRA: une plate-forme d'expérimentation pour la fouille de données. *Revue MODULAD*, *32*, 70-85.