Sparsity and Learning

Bohan Chen
UID:805219216
UCLA Department of Mathematics
bhchenyz@g.ucla.edu

Xianxing Jiang
UID:604958018
UCLA Deparment of ECE
xjiang19@g.ucla.edu

Weibo Hong
UID: 405427378
UCLA Deparment of ECE
weiboh1@g.ucla.edu

Abstract

In machine learning, overfitting is a common problem that may happen while data fitting. Thus regression item is an important part we should consider during design. There are many kinds of regression design and LASSO is a popular one because it can produce sparse coefficients. This means it can help us to pick important features of data. In this paper, we are trying to discuss why LASSO can do features selection theoretically and experimentally and compare it with Ridge Regression. Also, we analyzed some additional benefits of combining LASSO with different loss functions and achieve them through experiments.

1 Instructions

According to (1), now large quantities of data are collected and mined in nearly every area of science, entertainment, business, and industry. Medical scientists study the genomes of patients to choose the best treatments, to learn the underlying causes of their disease. Online movie and book stores study customer ratings to recommend or sell them new movies or books. Social networks mine information about members and their friends to try to enhance their online experience. And yes, most major league baseball teams have statisticians who collect and analyze detailed information on batters and pitchers to help team managers and players make better decisions.

Thus the world is awash with data. But as Rutherford D. Roger (and others) has said: "We are drowning in information and starving for knowledge."

There is a crucial need to sort through this mass of information, and pare it down to its bare essentials. For this process to be successful, we need to hope that the world is not as complex as it might be. For example, we hope that not all of the 30, 000 or so genes in the human body are directly involved in the process that leads to the development of cancer. Or that the ratings by a customer on perhaps 50 or 100 different movies are enough to give us a good idea of their tastes. Or that the success of a left-handed pitcher against left-handed batters will be fairly consistent for different batters.

This points to an underlying assumption of simplicity. One form of simplicity is sparsity. Loosely speaking, a sparse statistical model is one in which only a relatively small number of parameters (or predictors) play an important role. Our project is to generate the sparse form of our familiar algorithms such as linear least square regression, logistic regression and support vector machine (SVM). And we will make experiments to compare non-sparse algorithms (L_2 norm regularized) and sparse algorithm (L_1 norm regularized).

2 Why regularization

The network are easy to be over fitting when the data has too many features or the training data sets are too small. In order to prevent over fitting, there are many ways. First, we can reduce numbers of features by ignoring some features and keep the features which we think are important. However, it's hard to achieve because we can not always find a definition of "import". Another way used quite often is using regularization term. Regularization term can represent the complexity of the network and it increases when the complexity of network increases. Thus, adding regularization term can give some restrictions on network's complexity during training so that it can help to reduce over-fit.
There are many kinds of regularization terms and usually they are varies because of different purpose. Ridge Regression and least absolute shrinkage and selection operator(LASSO) are used quite common during design and Ridge Regression can produce non-sparse coefficients while LASSO can produce sparse coefficients. These two Regression methods are also consider as L_2 norm and L_1 norm.

3 Showing LASSO can produce sparse coefficients theoretically

To have a clear view of why LASSO can produce sparse coefficients, we will do a comparison between Ridge Regression and LASSO. By doing that, we can prove this mathematically and geometrically. First, we will go through weight updata and results they can achieve

$$L_1 = |w_1| + |w_2| + ... + |w_n| \qquad \frac{\mathrm{d}L_1}{\mathrm{d}w_i} = sign(w_i) = \pm 1 \tag{1}$$

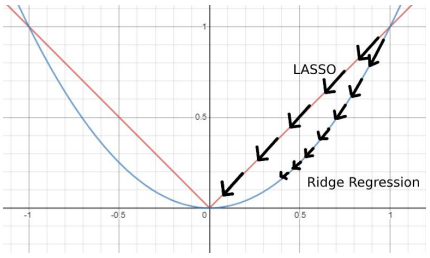
$$L_2 = \frac{1}{2}(w_1^2 + w_2^2 + ... + w_n^2) \qquad \frac{\mathrm{d}L_2}{\mathrm{d}w_i} = w_i \tag{2}$$

for simplicity, we assume learning rate = 0.5 and update the weight.

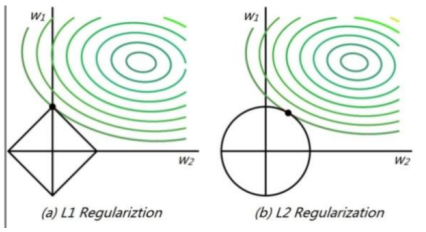
$$L_1 : w_i + 1 = w_i - \eta * 1 = w_i - 0.5$$

$$L_2 : w_i + 1 = w_i - \eta * w_i = 0.5 * w_i$$

we can see that although both weights are decreasing, L_1 norm can be zero after several times of iteration while L_2 is just approaching to zero. This was also shown on Figure 1a. Through that, we can conclude that LASSO is more likely to get zero coefficients which can result in having sparse coefficients.



(a) gradient descent show by graph



(b) L_1 norm and L_2 norm with object function

Figure 1: Why produce sparse coefficients

Second, we will show why LASSO can produce sparse coefficients geometrically. Thought Figure 1b, L_1 norm regression has an intersection with loss function at the coordinate axis. This means some of the weight coefficients will be zero and will result in sparse coefficients. However, for L_2 norm regression, the intersection is at a location where weight coefficients are close, Thus it can produce non-sparse coefficients. Thus, we can conclude that LASSO can produce sparse coefficients.

4 Showing LASSO can produce sparse coefficients numerically

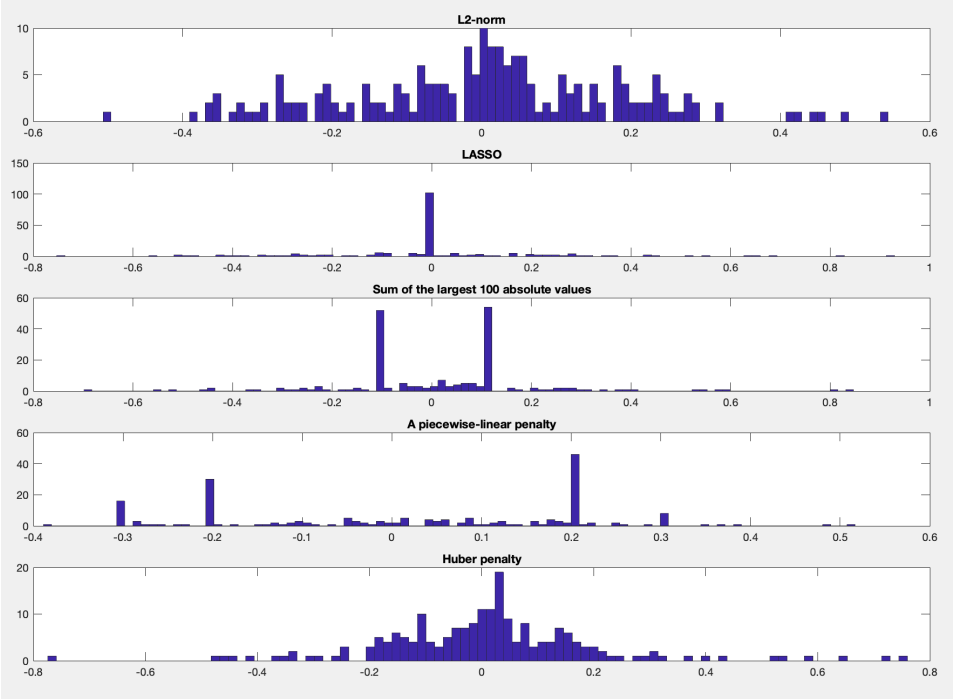
In this section, we found an example (3) which is good to show the LASSO can produce sparse coefficients. We generate a random matrix $A \in \mathbb{R}^{200*100}$ and $b \in \mathbb{R}^{200*1}$. Since matrix A has much

more rows than columns and they are generated randomly, it's almost impossible to have a solution x that can make $\phi(r)$ equals to zero. Thus, we solve this by minimization.

$$\begin{aligned} &\min \quad \phi(r) \\ s.t: \quad &r = Ax - b \end{aligned}$$

Ridge Regression	$\phi_1(r) = \|r\|_2$
LASSO	$\phi_2(r) = \|r\|_1$
Sum largest 100 absolute values	$\phi_3(r) = \sum_{i=1}^{100} \|r_i\|$
Piecewise-linear penalty	$\phi_4(r) = \sum_{i=1}^{200} h(r_i) \quad h(r_i) = \begin{cases} 0 & \|r_i\| < 0.2 \\ \|r_i\| - 0.2 & 0.2 < \|r_i\| < 0.3 \\ 2\|r_i\| - 0.5 & \|r_i\| > 0.3 \end{cases}$
Huber penalty	$\phi_5(r) = \sum_{i=1}^{200} h(r_i) \quad h(r_i) = \begin{cases} {r_i}^2 & 0.2 < \|r_i\| <= 0.2 \\ 0.4\|r_i\| - 0.04 & \|r_i\| > 0.2 \end{cases}$

We solve this through Matlab with the help of CVX and have the results
Through the second row of this histogram, we can see that LASSO has a very high chance to have $\phi(r)_i = 0$. Thus, we can conclude that LASSO can produce sparse coefficients.



5 Elastic Net

From the above, we have proved that LASSO can produce sparse coefficients. On the other hand, it does not handle highly correlated variables very well (1). There is a method call Elastic net. Elastic net compares differences between the ridge and lasso penalization of different models. Generally, the convex problem of solving optimal loss function problems can be shown as

$$\min \sum_{i=1}^{N} (LOSS) + \lambda(\frac{1}{2}(1-\alpha)\|\beta\|_2^2 + \alpha\|\beta\|_1)$$

where $\alpha \in$ [0,1] is the parameter to change when applying the penalization. The penalty applied to an individual coefficient can be described as $\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|$. When $\alpha = 1$, it reduces to Lasso

penalty and when $\alpha = 0$ it reduces to ridge penalty. Thus, we can say the strength of LASSO can be controlled by adding some component of the ridge regression to LASSO. By comparing different values of α, we then can evaluate the performance of lasso and ridge penalization and performance of sparsity. We will see more on experiment parts.
Besides that, From Figure 2, we can compare the constraint region for the Elastic net to the LASSO. Here is an example of three variables. From the left figure, we can see the elastic-net ball combines L_1 ball and L_2 ball's attributes.

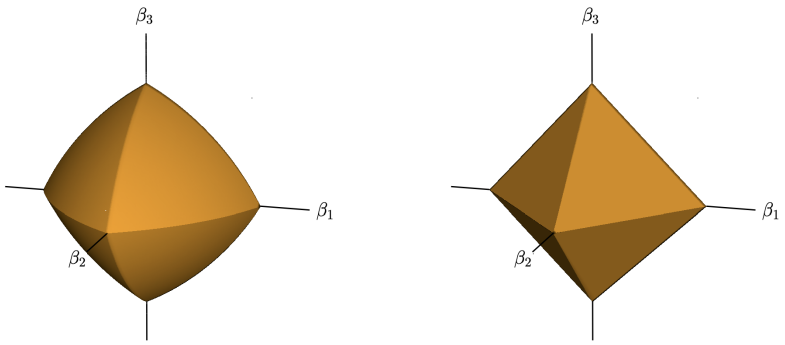


Figure 2: the elastic-net ball with $\alpha = 0.7 \in \mathbb{R}^3$, compared to the L_1 ball(right panel). The curved contour encourage strongly correlated variables to share coefficients

6 Coordinate Descent

Since there is absolute value involved in the penalization terms, we can not simply use gradient descent with the fact that the gradient is not differentiable in some points. As a result, coordinate descent is introduced. We choose one dimension each time to optimize until the function converges.

We can generalize any cost function as a RSS part and a penalization part. Firstly, we have the RSS part of the cost function as

$$RSS(w) = \sum_{i=1}^{N}(y_i - \sum_{j=0}^{p} x_{ij}w_j)^2$$

where $w = [\beta_0 \quad \beta_1 \quad ... \quad \beta_p]^T$ and $\beta = [\beta_1 \quad \beta_2 \quad ... \quad \beta_p]^T$ for cleanness. N is number of data points and p is dimension of the input data. We will keep the notation for algorithm below as well.

For different loss functions this form can vary as loss changes. This is just a generalized form. Then the gradient over w_k is then

$$\frac{\partial RSS(w)}{\partial w_k} = -2p_k + 2z_k w_k, \qquad p_k = \sum_{i=1}^{N} x_{ik}(y_i - \sum_{j=0, j\neq k}^{p} x_{ij}w_j), \qquad z_k = \sum_{i=1}^{N} x_{ik}^2.$$

compute the gradient for penalization term $$\lambda\frac{\partial \sum_{j=0}^{p}|w_j|}{\partial w_k} = \begin{cases} -\lambda & w_k < 0 \\ [-\lambda, \lambda] & w_k = 0 \\ \lambda & w_k > 0 \end{cases}$$

combine penalization term with RSS(w) $$\frac{\partial f(w)}{\partial w_k} = 2z_k w_k - 2p_k + \begin{cases} -\lambda & w_k < 0 \\ [-\lambda, \lambda] & w_k = 0 \\ \lambda & w_k > 0 \end{cases}$$

let $\frac{\partial f(w)}{\partial w_k} = 0$, we can get $$w_{k\,opt} = \begin{cases} -(p_k + \lambda/2)/z_k & p_k < -\lambda/2 \\ 0 & -\lambda/2 <= p_k <= \lambda/2 \\ (p_k - \lambda/2)z_k & \lambda/2 < p_k \end{cases}$$

With one dimensional optimization on each iteration going on and on, we can finally get the optimal value of the regression coefficient.

7 Algorithm

Algorithm 1 least square, binary logistic regression and SVM

1: **INPUT**: N training samples X_i and corresponding value y_i

2: Predictor for least square: $\min_{\beta_0,\beta} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$

Predictor for logistic regression: $\min_{\beta_0,\beta} -\frac{1}{N}\sum_{i=1}^{N}(y_i(\beta_0 + \beta^T x_i) - \log(1 + \exp(\beta_0 + \beta^T x_i)))$

Penalized SVM algorithm predictor: $\min_{\beta_0,\beta} \frac{1}{N}\sum_{i=1}^{N}(1 - y_i(\beta_0 + \beta^T x_i))_+$

3: Applying lasso will turn the objective respectively to

least square: $\min_{\beta_0,\beta} \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\|\beta\|_1$

logistic regression: $\min_{\beta_0,\beta} -\frac{1}{N}\sum_{i=1}^{N}(y_i(\beta_0 + \beta^T x_i) - \log(1 + \exp(\beta_0 + \beta^T x_i))) + \lambda\|\beta\|_1$

SVM: $\min_{\beta_0,\beta} \frac{1}{N}\sum_{i=1}^{N}(1 - y_i(\beta_0 + \beta^T x_i))_+ + \lambda\|\beta\|_1$

4: Train the model using coordinate descent

5: **Output**: optimal predictor variables β_0,β_j

7.1 Least Square Algorithm

The shown objective on the Algorithm will almost always give a predictor with nonzero coefficients in such linear regression models, making the diagonal of the final result very hard. And in general, the predictor result will not be unique if $p >> N$. This will further cause some interpretation difficulties. These predictor models will almost always overfit the data as well, which is a thing we usually try to avoid for data analysis. This is when we introduce the regularization term.

l_1 norm is specially useful in these cases because lasso combines the least-squares loss, bounding on the sum of the absolute values of the coefficients in least square. It generally has the effect of shrinking the coefficients and set some of them to be zero, which provides sparsity. For any l_q norm, when $q > 1$, sparsity will not occur the strongly and when $q < 1$, the problem will not be convex any more, leading to bigger problem for optimization.

There is another advantage emerging in the recent years, also known as "bet on sparsity" principle. It states that "*Use a procedure that does well in sparse problems, since no procedure does well in dense problems.*" For explanation, let us think about the amount of information N/p in Algorithm. If $p >> N$ and model is not sparse, then N is too small to allow for accurate estimation of the parameters. But if the true model is sparse, so that only $k < N$ parameters are actually nonzero in the true underlying model, then it turns out that we can estimate the parameters effectively, without knowing which k of the p parameters are actually nonzero.

Given the N predictor-response pairs, we use lasso to generate the problem

$$\min_{\beta_0,\beta} \frac{1}{2N}(\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2) \qquad \text{subject to} \sum_{j=0}^{p}|\beta_j| \leq t \tag{3}$$

when t is a user-specified parameter, a budget on the total l_1 norm of the parameter vector. This is usually rewrite to the Lagrangian form as shown in the Algorithm.

7.2 Logistic Regression

Logistic regression is a very popular model in biomedical field and is gaining more popularity in modeling a wider range of data. Consider the setting where the dimension of the input data, p, is very high-dimensional, which can not be used directly. When the dimension of data is even higher than the number of valid data points, we can clearly see that any model is over-parametrized and

regularization is needed in this case to get stable model. For example, document classification where we need to decide presence versus absence over a predefined dictionary of more than 20 thousand words and tokens as input.

For binary logistic regression, we usually classify the two cases of y as 0 or 1. We focus on calculating the probability of $Y = 1$ and $Y = 0$ given $X = x$ by replacing them with $\log(\beta_0 + \beta^T x_i)$ and $\log(1 + \exp(\beta_0 + \beta^T x_i))$.The log-likelihood estimation

$$\min_{\beta_0,\beta} -\frac{1}{N}\sum_{i=1}^{N}(y_i Pr(Y = 1|x_i) - (1 - y_1)\log Pr(Y = 0|x_i)) \tag{4}$$

generating the form we show on the Algorithm. Notice that in machine-learning community, there is a also a representation where response Y is represented with -1 and 1. This difference will certainly change the form of the penalized log-likelihood function to the form

$$\frac{1}{N}\sum_{i=1}^{N}\log(1 + \exp(-y_i(\beta_0 + \beta^T x_i))) + \lambda\|\beta\|_1 \tag{5}$$

and the product inside the exponential refers to the margin. Positive margin means a correct classification and negative margin means incorrect one. Maximizing the likelihood minimizes the loss function and then will optimize the predictor.

Lasso-penalized binary logistic models is convex and the likelihood part is differentiable generally. Meanwhile, it provides more sparsity compared to l_2 norm, proving it very suitable for regression application.

For mutliclass logistic regression, we will not prove how lasso generates sparsity. However, this concept is fairly popular too. We generally use multinomial likelihood and represent the probabilities using the log-linear representation

$$Pr(Y = k|X = x) = \frac{\exp(\beta_{0k} + \beta_k^T x)}{\sum_{l=1}^{K}\exp(\beta_{0l} + \beta_l^T x)} \tag{6}$$

for model generation. So, the penalized multiclass logistic regression model will be in different form as well.

7.3 Support Vector Machine

SVM is another very useful method in binary classification application. The idea is that the correctly classified points close to the margin will generate small loss and incorrectly classified points will generate big loss. Minimizing the algorithm will then give us the optimal predictor.

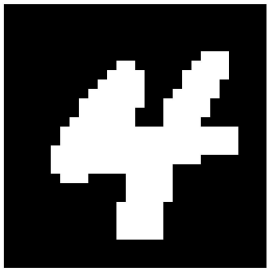
SVM Algorithm is popular in high-dimensional classification problems since the computation costs are same for both linear and nonlinear kernels. Computation cost can be even smaller if stochastic subgradient methods is used for optimization. However, this method will not generate sparse predictor.

In SVM, lasso method generally promotes sparsity. However, it's not recommend because the corresponding logistic regression problem gives very similar solutions when the penalty is active, and the algorithms are more stable. In Algorithm penalization, there are many constraints. The solution paths have many jumps and show many discontinuities (Zhu, Rosset, Hastie and Tibshirani 2004, Wang, Zhu and Zou 2006).

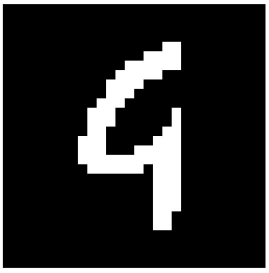
8 Experiments

In this section, we make some experiments around 3 algorithms, namely, least square regression, logistic regression and soft support vector machine. For each algorithm, we will compare the performance of different regularization terms, namely, L_1 norm (Lasso), L_2 norm (Ridge) and elastic net.

Our data set is a subset of MNIST, handwriting number images. We choose 2000 images of number 4 and 9 respectively from the training data set and choose 500 images of number 4 and 9 respectively from the test data set. Two randomly sampled images are shown in Figure 3.



(a) A sampled image of number 4



(b) A sampled image of number 9

Figure 3: Sampled images

8.1 Comparison of Lasso and Ridge regularization

This part we compare Lasso and Ridge regularization with 3 algorithms. For each algorithm, there are 3 metrics to evaluate the performance:

- Test error: The proportion of wrong predicted labels in test data set.
- 0 ratio: For the output weights, 0 ratio = (number of 0)/(number of weights).
- 10^{-8} ratio: For the output weights, 10^{-8} ratio = (number of elements between -10^{-8} and 10^{-8})/(number of weights).

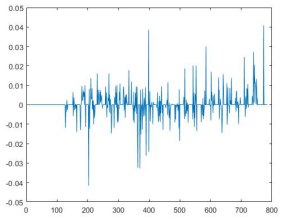
It can be seen that **Test error** reflects the learning performance while 0 **ratio** and 10^{-8} **ratio** reflect the sparsity of output weights, in which 0 ratio is strict and 10^{-8} ratio is softer. The result is shown in Table 1. We can see that for each type of algorithm, output weights of L_1 regularization is more sparse than those of L_2 regularization while the test error is similar. When comparing between algorithms, logistic and soft svm have similar performance and both of them perform better than linear regression. This is because that our data set is not linear separable. Least square (linear) regression can not handle such a data set well. But soft SVM and logistic regression can deal with such a data set.

Table 1: Performances of different algorithms

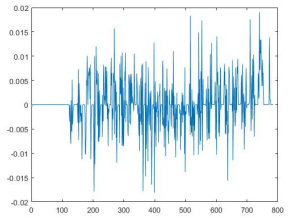
Algorithms	SVM		Least Square		Logistic	
	L_1	L_2	L_1	L_2	L_1	L_2
Test error	4.40%	4.20%	6.60%	6.00%	4.20%	4.10%
0 ratio	74.3%	31.7%	80.4%	61.7%	77.6%	39.2%
10^{-8} ratio	74.3%	38.0%	80.5%	61.7%	77.6%	39.2%

Remarks

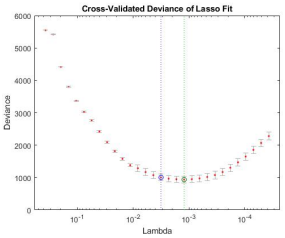
1. To make it more clear, we plot the output weights of L_1 and L_2 types of SVM model in Figure 4a and Figure 4b.
2. In our 3 models, we have a parameter λ. We apply cross validation method to find the best λ. The deviance - λ plot for L_1 regularized logistic regression is shown in Figure 4c, where $\lambda = 1$ is the parameter in soft SVM loss function.



(a) Output weights of L_1 regularized SVM when $\lambda = 1$



(b) Output weights of L_2 regularized SVM when $\lambda = 1$



(c) Deviance - λ plot L_1 regularized logistic model

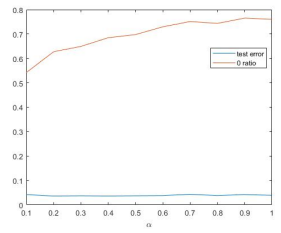
Figure 4: Figures for remark

8.2 Elastic Net

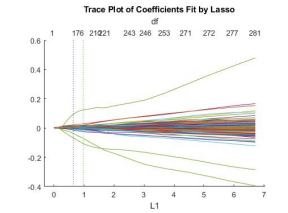
In this part, we focus on the elastic net regularization. We compare the performance of different parameter α in the regularization term in Logistic regression.

$$R_\alpha(w) = \alpha\|w\|_1 + (1-\alpha)\|w\|_2.$$

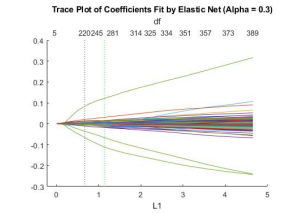
Figure 5a shows how the test error and 0 ratio change with parameter α. We can see that with the increasing of α, 0 ratio becomes higher and test error stays flat. This means that as L_1 norm counts more in the regularization term, the result weights tends to be more sparse. Figure 5b and Figure 5c are plots to trace the changes of weights fit by Elastic Net with different parameter α. In Figure 5b and Figure 5c, bottom x-axis means the L_1 norm, top x-axis means the degree of freedom, which is the number of nonzero weights, and y-axis means the value of weights. At the same value of L_1 norm, we can see that smaller α will leads to higher degree of freedom, which means less sparse of weights.



(a) Test error and 0 ratio - α plot



(b) Trace plot of weights fit by Lasso



(c) Trace plot of weights fit by Elastic Net $\alpha = 0.3$

Figure 5: Experiments for Elastic Net

In most of cases, according to Section 5, we need to choose α to combine the advantages of L_1 and L_2 norms.

9 Conclusion

After doing these, We know that the regularization can help us to prevent overfitting. Comparing between Ridge Regression with LASSO through theoretically and experimentally, we know LASSO can produce sparse coefficients because it can descent to zero more quickly than other regression terms during iteration. Besides that, we also analyzed Elastic Net which can give us both benefits of L_1 and L_2 norm.

Theoretically, LASSO has the effect of shrinking the coefficients and set some of them to be zero, which leads to sparsity. At the same time, l_1 norm is the smallest norm to provide a convex problem. Since the dense problem is much hard to deal with, LASSO is then generally the best way to achieve a simple and stable model.

At last, we use experiments to show that coordinate descent algorithm works and verify the connection of L_1 norm and sparsity. Besides, we can see how the results of model changes with parameter α in Elastic Net.

References

[1] Hastie, Trevor, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations.* Chapman and Hall/CRC, 2015.

[2] Sara Iris Garcia. *L0 Norm, L1 Norm, L2 Norm L-Infinity Norm* Medium, Apr 30, 2018.

[3] Stephen Boyd L, et al. *‚Additional Exercises for Convex Optimization.* Exercises 5.4(a-e), 2020.

[4] Zhu, J., Rosset, S., Hastie, T. and Tibshirani, R. *1-norm support vector machines, in Advances in Neural Information Processing Systems, Vol. 16, pp. 49–56.* 2004

[5] Wang, L., Zhu, J. and Zou, H. *The doubly regularized support vector machine, Statistica Sinica 16(2), 589.* 2006