Project Report: Loan Default Prediction

Xingshuo Chen, Chuyang Ke, Muyuan Li
Purdue University
{chen2240,cke,li2430}@purdue.edu

April 23, 2020

1 Introduction

In this section we introduce the problem settings in our project. We are interested in the Loan Default Prediction dataset [London, 2014]. The dataset consists of the financial transactions records associated with loans. Each piece of record comes with a label, which measures the loss in the case of a loan default.

In this project we train a supervised learning model to predict the amount of loss from loan records.

1.1 Dataset Overview

Our dataset includes $105,471$ pieces of loan records, and each records contains 769 features. The label of each record is a normalized value between 0 and 100, where 0 corresponds to the case that the loan did not default. Before we train our model, we first normalize the values of each feature to the range of $[0,1]$. We also remove categorical features, and the features in which every record shares exactly the same value. It is worth mentioning that there exists missing entries in the dataset.

2 Algorithms

In this section we introduce the machine learning algorithms we use in the project.

We use $X = [0,1]^{n\times d}$ to denote the dataset, where $x_i \in [0,1]^d$ is the i-th sample's feature vector. We use $y_i \in \{0\} \cup \mathbb{R}^+$ to denote the label associated with the sample.

In this project compare the performance of four regression models: one-sided linear regression, ridge regression, LASSO regression, and Support Vector Regression (SVR). For the first three models we implement the algorithms from scratch. For SVR we use the existing solver `LIBLINEAR` [Fan et al., 2008].

2.1 One-sided Linear Regression

Our base model is the one-sided linear regression in the following form

$$\underset{w\in\mathbb{R}^d}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n}\left(\max(w^\top x_i, 0) - y_i\right)^2 . \tag{1}$$

Note that we introduce the max function, because if a loan did not default (i.e., $y_i = 0$) and the model prediction is a negative quantity, we do not want to penalize it.

One can rewrite (1) in the matrix form

$$\underset{w\in\mathbb{R}^d}{\text{minimize}} \quad \frac{1}{2}\|\max(Xw, \mathbf{0}) - y\|^2 . \tag{2}$$

Note that unlike regular linear regression, problem (2) has no closed-form solutions. To solve for w we use a *gradient descent* approach. The gradient of (2) is

$$\nabla_w = X^\top \left(\max(Xw, \mathbf{0}) - y\right) .$$

The gradient descent algorithm can be found in 1. Note that η is a chosen step size. We initialize w to be the solution to the regular linear regression, because it is closer to the optimal solution compared to a random starting point.

Algorithm 1 Gradient Descent

```
1: initialize $w = (X^\top X)^{-1} X^\top y$
2: while not converge do
3:     $w \leftarrow w - \eta \nabla_w$
4: end while
```

2.2 Ridge Regression

Our next model is ridge regression. The objective function is in the following form

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad \frac{1}{2} \|\max(Xw, \mathbf{0}) - y\|^2 + \frac{\lambda}{2} \|w\|^2 . \tag{3}$$

The motivation behind introducing the l_2-norm is to shrink all coefficients in w by a factor. This makes the coefficients less extreme and in some cases, avoids overfitting. To solve for w we use the gradient descent algorithm in 1. The gradient of (3) is

$$\nabla_w = X^\top \left(\max(Xw, \mathbf{0}) - y\right) + \lambda w .$$

2.3 LASSO Regression

The next model is LASSO regression. Here we use the following objective function

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad \frac{1}{2} \|\max(Xw, \mathbf{0}) - y\|^2 + \frac{\lambda}{2} \|w\|_1 . \tag{4}$$

The motivation behind introducing the l_1-norm is to shrink all coefficients in w by a fixed quantity towards 0. As a result some coefficients in the final w will be zero. This selects important features and potentially avoids overfitting. However the l_1-norm is not differentiable at zero, and vanilla gradient descent will not work. Instead we use a *proximal gradient descent* algorithm (see 2).

Algorithm 2 Proximal Gradient Descent

```
1: initialize $w = (X^\top X)^{-1} X^\top y$
2: while not converge do
3:     $\nabla_w \leftarrow X^\top \left(\max(Xw, \mathbf{0}) - y\right) + \frac{\lambda}{2} \cdot \text{sign}(w)$
4:     $w \leftarrow w - \eta \nabla_w$
5:     for $i = 1 : d$ do
6:         if $-\frac{\lambda}{2} \le w_i \le \frac{\lambda}{2}$ then
7:             $w_i \leftarrow 0$
8:         end if
9:     end for
10: end while
```

Compared to the vanilla gradient descent, Algorithm 2 includes a *soft thresholding* step [Liu and Barber, 2018].

2.4 Missing Value Imputation

To address the missing entries in the dataset, we employ two separate approaches. For every missing entry we fill its value using the minimum (or the average, respectively) of the corresponding feature column.

3 Experiment Setup

In this section we introduce our experiment settings.

By default we divide our dataset into two parts: the training set with 80,000 samples, and the test set with 25,471 samples. In some experiments (e.g., measuring bias variance tradeoff) we only use part of the training set. We always use the whole test set.

In every experiment we use a 5-fold cross-validation for model selection and parameter tuning. More specifically, we partition the training set into 5 folds. For each training iteration, we use 4 folds as the inputs of the model, and test the model's performance on the remaining fold. Every training iteration reports a training error and a validation error. After all 5 iterations, we calculate the average 5-fold training and validation error. Finally we evaluate the model's performance on the test set and report the test error. In this project we use the root mean square error (RMSE). Mathematically, the RMSE is defined as

$$\text{RMSE}(w) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\max(w^\top x_i, 0) - y_i)^2}\,. \tag{5}$$

All algorithms in this section are implemented from scratch.

3.1 Tuning Hyperparameters

Due to missing entries, we train all models mentioned above twice. One uses average imputation, and the other one uses minimum imputation. In linear regression there are no additional hyperparameters. In ridge and LASSO regression, the hyperparameter is λ, which controls the strength of the regularization. Similarly in SVR, the hyperparameter is C, which controls the tolerance for points outside of the margin.

4 Experiment Results

In this section we present our experiment results.

4.1 Imputation Method

We first compare how different missing data imputation approaches will affect the performance of our models. To do so, we train all four models twice. The first time we use the minimum imputation, and the second time we use the average imputation. For each model, the best hyperparameter (in terms of the minimum test error) is picked.

We report the test results in Table 1.

	Minimum Imputation			Average Imputation		
	Traning	Validation	Test	Traning	Validation	Test
One-sided LR	4.1557	4.2090	**4.2524**	4.1586	4.2126	4.2543
Ridge ($\lambda = 2500$)	4.1621	4.2102	**4.2512**	4.1678	4.2133	4.2531
LASSO ($\lambda = 0.3$)	4.1571	4.2104	**4.2517**	4.1600	4.2136	4.2536
SVR ($C = 0.001$)	4.3085	4.3159	4.3295	4.3040	4.3160	**4.3291**

Table 1: Minimum imputation vs. average imputation in four models.

From our experiments, the minimum value imputation achieves a better performance than the average value imputation in 3 out of 4 models. For clarity of presentation, we use minimum value imputation for all experiments in the following sections.

4.2 Choice of Hyperparameters

We now investigate how the choice of hyperparameters will affect the performance of ridge regression, LASSO regression, and SVR. For each model, we plot the training, validation, and test errors against the choice of λ (C respectively).

The results are presented in Figure 1, 2, and 3.

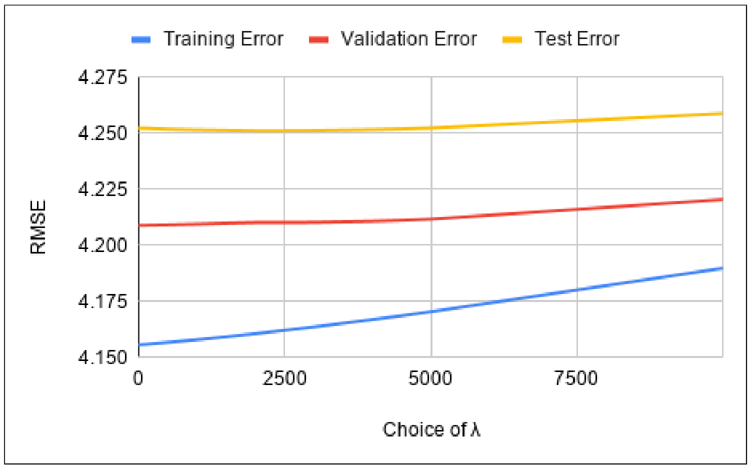

Figure 1: Test results of ridge regression versus the choice of λ.

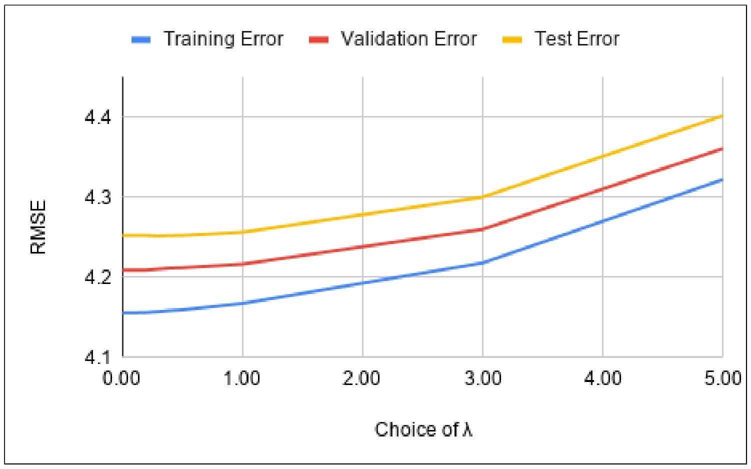

Figure 2: Test results of LASSO versus the choice of λ.

Our experiments show that as λ (C respectively) increases, the training error of each model increases monotonically. This is not true for the test error, however. The test error first decreases and then increases. For ridge regression, the minimum point is $\lambda = 2500$. For LASSO, it is $\lambda = 0.3$. For SVR, it is $C = 0.001$. As we discussed in previous sections, the hyperparameter λ (C respectively) controls the strength of regularization. This suggests that each model without regularization is overfitting on our dataset. Our experiments confirm that by setting the hyperparameter properly, overfitting can be mitigated.

It is well-known that unlike ridge regression, LASSO shrinks some features' weights to zero. We are also interested in the feature selection property of LASSO under different settings of λ. The test results are plotted in Figure 4.

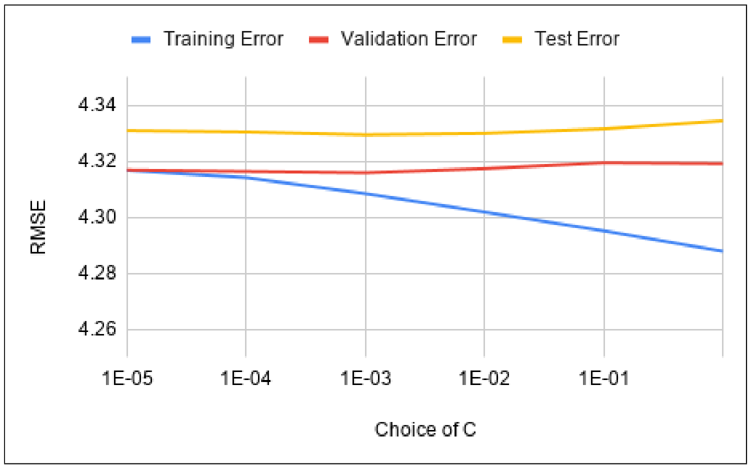

Figure 3: Test results of SVR versus the choice of C.

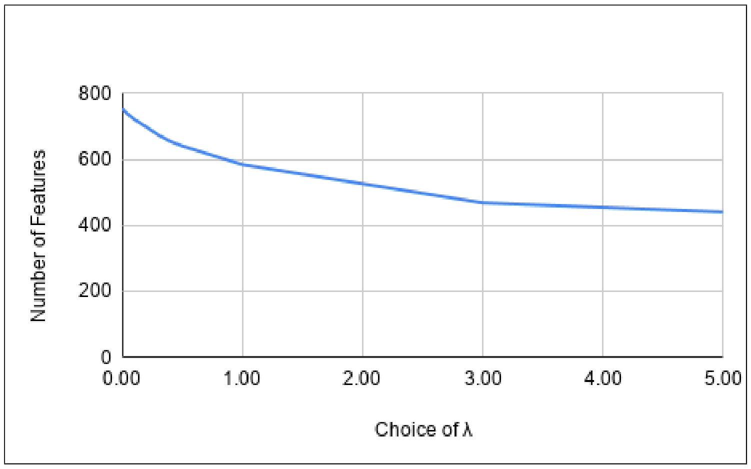

Figure 4: Number of features selected in LASSO versus the choice of λ.

The experiment result shows that as λ increases, the number of features selected becomes smaller and the model becomes sparser.

4.3 Bias-variance Tradeoff

We also want to understand bias-variance tradeoff in the context of our model. In order to measure bias-variance tradeoff, we plot training, validation, and test errors versus the size of the training set. We want to highlight that the size of the test set is always fixed. Here we use two models: one-sided linear regression and ridge regression ($\lambda = 2500$). The experiment results can be found in Figure 5 and 6. The patterns of other models or parameters are highly similar.

The experiment results show that when the size of the training set is small, there exists a large gap between the training error and the test error. An explanation is that the model is overfitting when the dataset is small. As the number of samples increases, the training-test error gap becomes smaller.

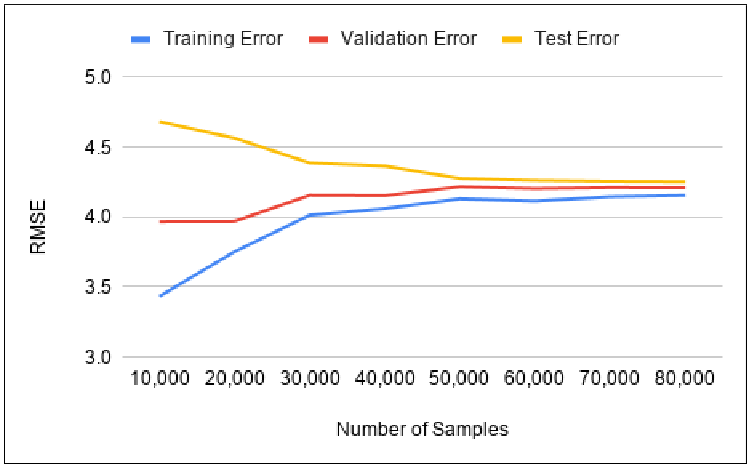

Figure 5: Test results of one-sided linear regression versus the number of samples in the training set.

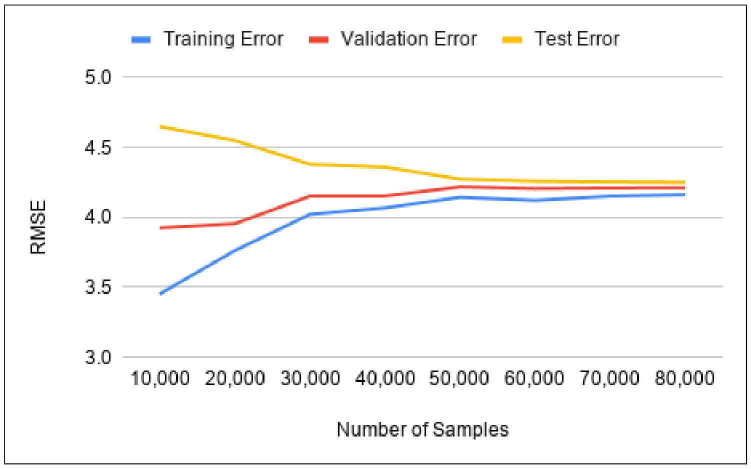

Figure 6: Test results of ridge regression versus the number of samples in the training set.

References

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

Haoyang Liu and Rina Foygel Barber. Between hard and soft thresholding: optimal iterative thresholding algorithms. *arXiv preprint arXiv:1804.08841*, 2018.

Imperial College London. Loan default prediction - imperial college london. `https://www.kaggle.com/c/loan-default-prediction/`, 2014.