华南理工大学

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Panan Wu

Supervisor:
Mingkui Tan

Student ID:
201630665908

Grade:
Undergraduate

October 27, 2018

Linear Regression and Stochastic Gradient Descent

***Abstract*—In this experiment, I conduct a linear regressor and update parameters with stochastic gradient descent and closed-form solution using scaled Housing dataset in LIBSVM Data. I evaluated the two methods mentioned above by plotting the loss value and root mean square error (rmse) during the parameters update process. The experiment result shows that the process of parameters updates with closed-form solution is stable and parameters reach the optimal solution within one iteration, and parameters update with stochastic gradient descent needs multiple iterations to reach a good solution and unstable.**

I. Introduction

LINEAR regression is a classical way to solve regression problems. We have mutiple ways to update parameters like closed-form solution and stochastic gradient descent. Closed-form solution is very accurate but time consuming while stochastic gradient descent is efficient but not so accurate.

In this experiment, I implement two versions of linear regression: one with closed-form solution and another with stochastic gradient descent. By plotting the loss in training, we can intuitively know how these two methods update parameters.

The motivations of this experiment are: (1) further understand of linear regression, closed-form solution and Stochastic gradient descent, (2) conduct some experiments under small scale dataset and (3) realize the process of optimization and adjusting parameters.

II. Methods and Theory

A. Linear Regression

Linear regression describes the linear relationship between a predictor variable and a response variable.

The input of linear regression $\mathbf{X}$ is within the input space $\mathbb{R}^m$ (m is corresponding to the number of features), and it's also called feature, covariants, predictors, etc. The groundtruth of linear regression $\mathbf{Y}$ is the value that we use X to predict. And the goal of linear regression is to learn a hypothesis or model $f : \mathbf{X} \to \mathbf{Y}$.

In order to learn a model, we first need to construct a dataset. Given set of input, output pairs:

$$D = \{(x_i, y_i)\}_{i=1}^{n} \tag{1}$$

In the equation mentioned above, x_i is a single sample of input, y_i is a groundtruth of this sample and n is the number of training samples. To learn the best model on dataset D, we use parameters $\mathbf{w}$ to help us make predict. And the model funtion is:

$$\begin{aligned} f(\mathbf{x}; w_0; \mathbf{w}) &= w_0 + w_1 x_1 + ... + w_m x_m \\ &= \sum_{j=1}^{n} w_j x_j + w_0 \\ &= \mathbf{w}^T \mathbf{x} + w_0 \end{aligned} \tag{2}$$

To evaluate and update our model, we need to define the difference between our predict and the groundtruth. We use loss function **Least squared loss** to measure the difference.

$$\begin{aligned} \mathcal{L}_D(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 \\ &= \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \end{aligned} \tag{3}$$

And our goal is to find **w*** that minimizes loss $\mathcal{L}$.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}_D(\mathbf{w}) \tag{4}$$

B. Closed-form Solution

To calculate the gradient of loss function, we first need to represent the loss function as a function in matrix form:

$$\begin{aligned} \mathcal{L}_D(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i; \mathbf{w}))^2 \\ &= \frac{1}{2} (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})) \\ &= \frac{1}{2} (\mathbf{y}^T \mathbf{y} - 2\mathbf{w}^T \mathbf{X}^T \mathbf{y} + \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}) \end{aligned} \tag{5}$$

And then we calculate the partial derivative of the loss function for the parameter $\mathbf{w}$:

$$\begin{aligned} \frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} &= \frac{1}{2} (\frac{\partial \mathbf{y}^T \mathbf{y}}{\partial \mathbf{w}} - \frac{\partial 2\mathbf{w}^T \mathbf{X}^T \mathbf{y}}{\partial \mathbf{w}} + \frac{\partial \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w}}{\partial \mathbf{w}}) \\ &= \frac{1}{2} (-2\mathbf{X}^T \mathbf{y} + (\mathbf{X}^T \mathbf{X} + (\mathbf{X}^T \mathbf{X})^T)\mathbf{w}) \\ &= -\mathbf{X}^T \mathbf{y} + \mathbf{X}^T \mathbf{X}\mathbf{w} \end{aligned} \tag{6}$$

To get analytical solution, we make the partial derivative equals to zero and calculate the value of parameter w:

$$\begin{aligned} \frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} &= -\mathbf{X}^T \mathbf{y} + \mathbf{X}^T \mathbf{X}\mathbf{w} = 0 \\ &\Rightarrow \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \end{aligned} \tag{7}$$

As we have seen in the equation, we have to calculate the inverse matrix of $\mathbf{X}^T\mathbf{X}$, which is not only time consuming but impossible for most matrices. Thus most of the time, we will use gradient descent to update our parameters, while stochastic gradient descent is a typical variant of gradient descent.

C. Gradient Descent

In the gradient descent, I minimize loss by repeated gradient steps(when no closed form). First, I compute gradient of loss with respect to parameters : $\frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}}$. Then I update parameters with rate η which is called learning rate.

$$\mathbf{w}' \to \mathbf{w} - \eta \frac{\partial \mathcal{L}_D(\mathbf{w})}{\partial \mathbf{w}} \tag{8}$$

D. Stochastic Gradient Descent

The main idea of stochastic gradient descent (SGD) are to estimate function and gradient from a small, current subsample of training data and with enough iterations and data, loss function will converge to the true minimum. In SDG the true gradient of $\mathcal{L}(w)$ is approximated by agradient of a single example:

$$\mathbf{w} := \mathbf{w} - \eta \nabla \mathcal{L}_i(\mathbf{w}) \tag{9}$$

Although random noise is introduced, it behaves like gradient descent in its expectation.

III. Experiments

A. Dataset

The dataset I used in this experiment is scaled Housing dataset from LIBSVM Data. There are 506 instances and 13 features in this dataset. The data had been scaled into [-1,1] for better prediction. What we need to do is to predict the house price according to its features like averge number of rooms per dwelling. Because they don't divide the training set and test set, we need to divide the training set and test set by ourselves.

B. Implementation

1) Closed-form Solution: .

First of all, I load the Housing dataset using $load\_svmlight\_file$ function in sklearn library. Then, I divide dataset into training set and validation set using $train\_test\_split$ function. I set the test size to 0.25.

When it comes to initialize parameters. I choose to set all parameter into zero, initialize it randomly and with normal distribution separately to determine which one is the best.After that, I calculate the closed-form solution by equation 7 and update parameters. And then I calculate loss of both training set and validation set. I repeat this process 300 times and output the loss during each process.

Finally, I plot both training loss and validation loss during parameter update. As we can see from Fig.1, the closed-form solution finds the best parameters at the first iteration, the loss of training set and validation set are stable. And then I plot the validation rmse variation with the number of iterations, the rmse of valiadation is still stable. Finally, I output and plot the mean loss with different parameter initializers, as shown in TABLE I and Fig. 3, the initializer in closed-form solution doesn't affect the loss function at all.

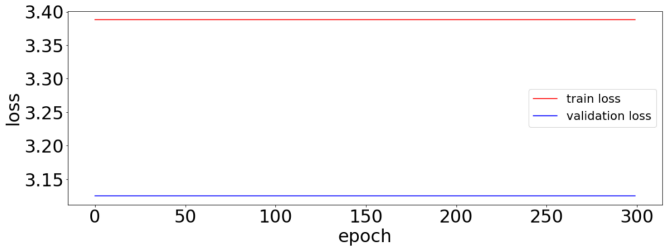

Fig. 1. The loss varing with the number of iterations.

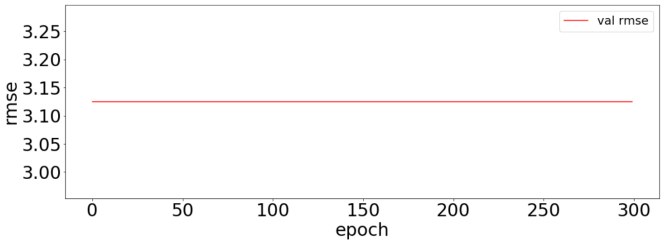

Fig. 2. The rmse varing with the number of iterations.

TABLE I
MEAN LOSS WITH DIFFERENT INITIALIZER

Initializer	Mean Loss
Zero Initializer	3.394
Random Initializer	3.394
Normal Initializer	3.394

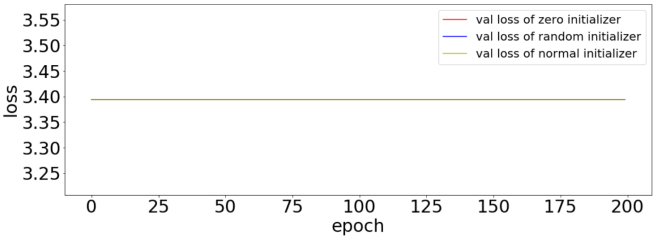

Fig. 3. The loss varing with the number of iterations and different intializers.

2) Stochatic Gradient Descent: .

As the same in closed-form solution experiment, I read training data and split it into two parts. During training, I set learning rate to 0.001 and max epoch to 300. I still try three types of parameter initializers. When it comes to parameters update, I first randomly choose a sample from training set and update parameters according to its partial derivative, and I calculate and store the training loss and validation loss with parameters just update.

As shown in Fig. 4, the training loss is unstable, but the validation loss and rmse are stable decline. Different initializers bring different loss mean values, which means that choosing different initializers in sgd will have a certain impact on the results, and zero intializer seems better than other two initializers in this dataset.

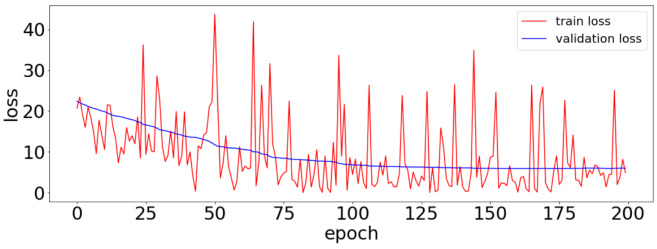

Fig. 4. The loss varing with the number of iterations.

Finally, I compared the result of two methods mentioned above. We can know from Table III that closed-form solution

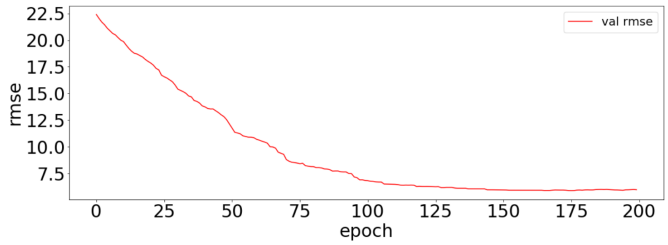

Fig. 5. The rmse varing with the number of iterations.

TABLE II
MEAN LOSS WITH DIFFERENT INITIALIZER

Initializer	Mean Loss
Zero Initializer	10.123
Random Initializer	10.553
Normal Initializer	11.342

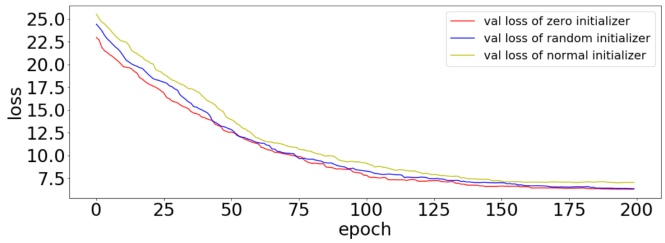

Fig. 6. The loss varing with the number of iterations and different intializers.

has lower validation loss than sgd. The final training loss of sgd is lower than that of closed form. This result means that sgd may be better than closed form. After all, it contains many random factors in training process, but it has two more hyper-parameters that need to be artificially adjusted.

TABLE III
MEAN LOSS WITH DIFFERENT INITIALIZER

Metrics	Closed-form	Sgd
Final training loss	3.3876407032391738	3.068910779095976
Final validation loss	3.1251863328930667	5.648078628287903

IV. CONCLUSION

In this experiment, I implement linear regression and update parameters with closed-form solution and stochastic gradient descent, and I make comparison between the two update methods mentioned above. From what has been shown above, we may safely draw the conclusion that the closed-form solution is accurate and doesn't need iterations to update, while the sgd needs several iterations to reach a good solution. But we need to try other kinds of dataset to evalute these two methods better.

During this experiment, I realized what I learned in class, which enhanced my code practice ability. By comparing these two methods and writing experimental reports, I came into contact with a little scientific research. And it has increased my knowledge.