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:
Da Ren

Supervisor:
Qingyao Wu

Student ID:
201720144900

Grade:
Graduate

December 14, 2017

Linear Regression, Linear Classification and Gradient Descent

***Abstract*—Gradient descent plays an important role in machine learning. It can help us get the global optimal value of a function. However, one of the most important task in machine learning is to get the optimal value for loss function. Gradient descent is widely used in this task. I use gradient descent to optimize two tasks: liner regression and linear classification. The experiments show that gradient descent can help us minimize the cost function.**

I. Introduction

GRADIENT descent is a important method in machine learning. The most important tasks in many machine learning models is to update or calculate their parameters which can lead the corresponding loss functions value to be minimal value. There are many methods to solve this problem. Gradient descent is one of the most widely used method in this task. It can help us minimize our cost function in an efficient way. Therefore, it is a significant approach in machine learning. In the paper, I implement the standard gradient descent to optimize the loss function in linear regression and linear classification. The experiments' results show that gradient descent can minimize the loss function.

II. Methods and Theory

Gradient descent is flexible since it can be easily used in different kinds of loss function. It is based on the observation that if the multi-variable function $F(x)$ is defined and differentiable in a neighborhood of a point a, then $F(x)$ decreases fastest if one goes from a in the direction of the negative gradient of F at a. Therefore, if we want to minimize the value of $F(x)$, we can move a against the gradient of F. Therefore, a can be updated by Eq. (1).

$$a_{t+1} = a_t + \lambda \Delta F(a_t) \tag{1}$$

where a_t is the value of a at time t, λ is the learning rate which is always set to be a small value like 0.1. $\Delta F(a_t)$ is the gradient value of $F(a_t)$.

There are two experiments in my work. One is linear regression, the other is linear classification. Therefore I have to choose different loss function to different tasks.

Linear regression is a linear approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables denoted X. It can be calculated as following

$$y' = W^T X + b \tag{2}$$

where W is the weight matrix, and b is the bias variable. However, we can add a new column to X to be X' where the column is all 1. Therefore, we can not use b anymore, since the weight of the all 1 column can act the role of b. y can be calculated as following

$$y' = W^T X' \tag{3}$$

Then, I choose the loss function which is described in Eq. (4)

$$L_1 = 1/2m \sum_{i=1}^{m} (y_i' - y_i)^2 \tag{4}$$

where m is the total number of data and y_i is the value of training data. y_i' is the value which is calculated by linear model. To use the gradient descent to optimize the model, we have to calculate the derivation of Eq. (4).

$$\Delta L_1 / \Delta w = 1/(2m) \sum_{i=1}^{m} 2(y_i' - y_i) x_i = 1/m \sum_{i=1}^{m} (y_i' - y_i) x_i \tag{5}$$

Now, we can use Eq. (5) to calculate derivative at each step and update weight in Eq. (1). After several iterations with a suitable learning rate, we can get a weight matrix which can lead the loss function's value into global minimal.

In the linear classification task, I also use the Eq. (3). Since it is a different task, I choose another loss function. The loss function I choose is called hinge loss, which can be described in Eq. (6)

$$L_2 = \left\| W \right\|^2 /2 + C/m \sum_{i=1}^{m} max(0, 1 - y_i (W^T x_i + b)) \tag{6}$$

where C is a hyperparameter which can be adjusted in different dataset.

To use the gradient descent to optimize the weight matrix, we have to calculate the the derivation of Eq. (6).

$$\Delta L_2 / \Delta w = W + C/m \sum_{i=1}^{m} g_w(x_i) \tag{7}$$

where

$$g_w(x_i) = \begin{cases} -y_i x_i & 1 - y_i (W^T x_i') \geq 0 \\ 0 & 1 - y_i (W^T x_i') < 0 \end{cases} \tag{8}$$

We can update the linear classification according to Eq. (1) and use Eq. (8) to calculate the derivative.

After choosing the loss function and calculate their corresponding derivation, I can use gradient descent to train these two models. My experiment will be described in following.

III. Experiments

There will be two experiments which will be introduced. One is training the linear regression and the other is training the linear classification. Both of them is trained in gradient descent.

A. Dataset

This is two different tasks, so we have to use two different datasets. To train the linear regression model, I use the housing data in LIBSVM Data. There are 506 data and each of them has 13 attributes. For the linear classification tasks, I use the australian data in LIBSVM Data. There are 690 data and each of them has 14 attributes. All the datasets I use is the scaled version.

B. Implementation

In this section, I will introduce the linear regression experiment and linear classification experiment in detail.

In the linear regression experiment, the learning rate λ is set to be 0.1. The weight matrix is initialized randomly. After I read the data from file, I split the dataset into training set and validation set. There are 354 data in training set and 152 data in validation set. The iteration's number is set to 300. The loss value of training set and validation set is described in Fig. 1.

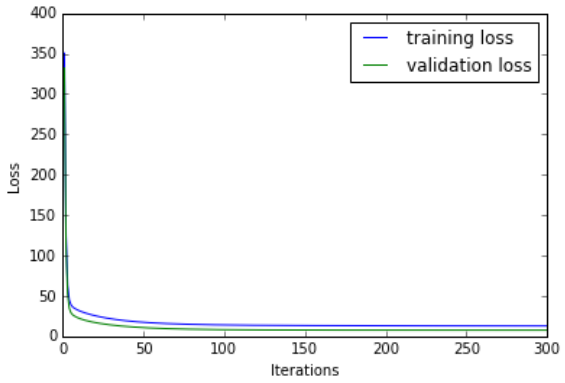

Fig. 1. The loss value of training set and test set in linear regression.

We can find that both of the loss of training set and test set decrease quickly at the beginning and close to unchange at the end of iterations. It means that the value of loss function is closed to global minimal value.

In the linear classfication experiment, the learning rate λ is set to be 0.1. The weight matrix is initialized randomly. I split the dataset into training set and validation set. There are 483 data in training set and 207 data in validation set. The iteration's number is set to 300. The loss value of training set and validation set is described in Fig. 2.

We can find that both of the loss of training set and test set decrease quickly at the beginning and close to unchange at the end of iterations. It means that the value of loss function is closed to global minimal value.

After finish training, I set a threshold to classify the data in validation set. I set the threshold to be 0, if the value calculated by model is smaller than 0, the corresponding data will be classified into one class (the label of this class is -1). If the value is larger or equal than 0, the corresponding data will be classified into another class (the label of this class is 1). And then I calculate the precision, recall and f1-score which is shown in Table I.

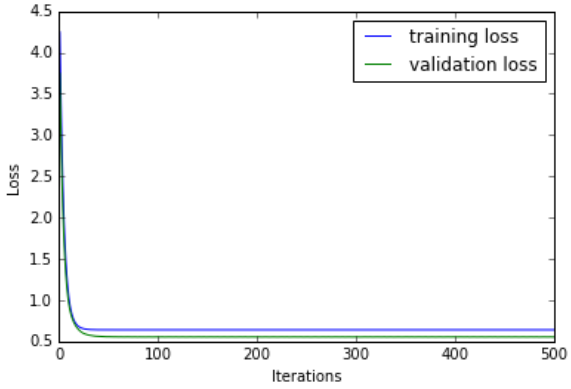

Fig. 2. The loss value of training set and test set in linear classification.

TABLE I
CLASSIFICATION RESULTS

	precision	recall	f1-score	support
class1	0.93	0.81	0.87	117
class2	0.79	0.92	0.85	90
avg / total	0.87	0.86	0.86	207

From Table I, we can find that the precision, recall and f1-score of classification is larger than 0.85. It is a good result of linear classification.

IV. CONCLUSION

In this experiment, I implement gradient descent in linear regression and linear classification. Both of their weight matrix is updated so that corresponding loss functions can be minimized. However, the loss function I use is convex, so that I don't need to care about the problem of local minimal. If the loss function is non-convex, gradient descent maybe fail to find the global minimal. Learning how to use gradient descent to get global minimal in non-convex function is my future work.