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:
HuiHan

Supervisor:
Qingyao Wu

Student ID:
201721045572

Grade:
Graduate

December 14, 2017

Linear Regression, Linear Classification and Gradient Descent

***Abstract*—Linear Regression, Linear Classification and Gradient Descent are the most basic knowledge points of machine learning.This article describes the use of gradient descent method to solve linear regression and linear classification.Through this experiment to understand and master the basic method of machine learning experiment.**

I. Introduction

THERE are two general problems with this experiment: linear regression and gradient descent, linear classification and gradient descent. Simple linear regression describes the linear relationship between a predictor variable, plotted on the x-axis, and a response variable, plotted on the y-axis. Linear classification divides the data into two categories by a linear function. Through the gradient descent method to optimize linear programming and linear classification, get a more accurate function. The purpose of the experiment includes the following points: (1)Further understand of linear regression and gradient descent. (2)Conduct some experiments under small scale dataset. (3)Realize the process of optimization and adjusting parameters.

II. Methods and Theory

A. Linear Regression and Gradient Descent

First we should determine the model function:

$$f(x) = W^T X + b \tag{1}$$

Then we can deduce the loss function:

$$L_D(W) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - W^T x_i)^2 \tag{2}$$

Training find minimizer of Loss function:

$$W^* = \arg\min_w L_D(W) \tag{3}$$

The method of gradient descent is used to optimize parameters W and b.Derive the gradient formula of W and b respectively:

$$G_W = \frac{1}{n}\sum_{i=1}^{n}(x_i W + b - y_i)x_i \tag{4}$$

$$G_b = \frac{1}{n}\sum_{i=1}^{n}(x_i W + b - y_i) \tag{5}$$

The parameters W and b can be updated with gradient descent:

$$W_t = W_{t-1} + \eta D_W \tag{6}$$

$$b_t = b_{t-1} + \eta D_b \tag{7}$$

B. Linear Classification and Gradient Descent

First we should determine the model function:

$$f(x) = W^T X + b \tag{8}$$

Then we can deduce the loss function:

$$L = \frac{||W||^2}{2} + \frac{C}{n}\sum_{i=1}^{n} max(0, 1 - y_i(W^T x_i + b)) \tag{9}$$

The method of gradient descent is used to optimize parameters W and b.Derive the gradient formula of W and b respectively:

$$G_W = W + \frac{C}{n}\sum_{i=1}^{n}(-y_i x_i) \tag{10}$$

$$G_b = \frac{C}{n}\sum_{i=1}^{n}(-y_i) \tag{11}$$

But the above two formulas have one common condition:$1 - y_i(W^T x_i + b) > 0$

The parameters W and b can be updated with gradient descent:

$$W_t = W_{t-1} + \eta D \tag{12}$$

$$b_t = b_{t-1} + \eta D \tag{13}$$

III. Experiments

A. Dataset

Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features. Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features. Data sets should be divided into training set, validation set.The experimental environment is a direct installation of anaconda3, which avoids the installation of many dependent packages.

B. Implementation

1) Linear Regression and Gradient Descent: The experimental steps are as follows:

(1). Use load_svmlight_file function in sklearn library to load the experiment data.

(2). Divide dataset into training set and validation set using train_test_split function.

(3). Initialize linear model parameters. Set all parameter into one.

(4). Choose loss function and derivation.

(5). Calculate gradient G toward loss function from all samples.

(6). Denote the opposite direction of gradient G as D.

(7). Update model:$W_t = W_t - 1 + \eta$, η is learning rate, set as 0.02 .

(8). Get the loss L_{train} under the training set and $L_{validation}$ by validating under validation set.

(9). Repeat step 4 to 8 for several times, and drawing graph of L_{train} as well as $L_{validation}$ with the number of iterations.

Experimental implementation of the key code is as follows:

```
def get_data():
    data =
        load_svmlight_file("data\\housing_scale")
    return data[0], data[1]
X, y = get_data()
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.2)
def hy(W,b,X1,y1):
    loss=np.sum(np.square(X1*W + b - y1))/
        (2*X1.shape[0])
    return loss
def main(n,X_train,y_train,X_test,y_test):
    W=np.ones(X.shape[1])
    b=1
    lenrat=0.02
    train = []
    validation = []
    for i in range(n):
        train.append(hy(W,b,X_train,y_train))
        validation.append(hy(W,b,X_test,y_test))
        DW= -(X_train*W + b - y_train)*X_train
            / X_train.shape[0]
        Db = -np.sum((X_train*W + b - y_train))
            / X_train.shape[0]
        W = W + lenrat*DW
        b = b + lenrat*Db
    return train,validation
```

The experimental results,which draws graph of L_{train} as well as $L_{validation}$ with the number of iterations,are as follows Fig. 1.

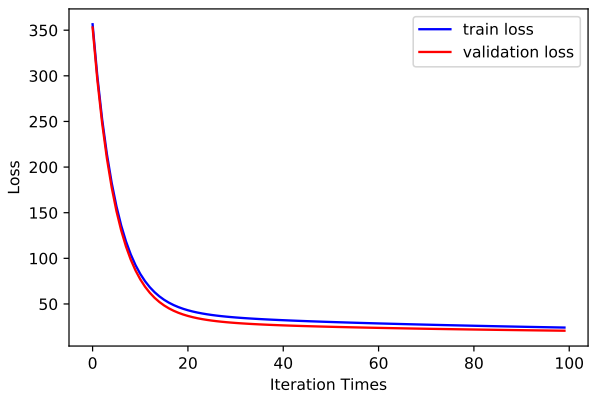

Fig. 1. Loss function curve in linear regression and gradient descent experiments

Linear Regression and Gradient Descent experimental results show that as the number of iterations increases, the cost of the training set and the verification set decreases rapidly and finally tends to be stable. Experimental results and guess the same result, so the experiment was successful.

2) Linear Classification and Gradient Descent: The experimental steps are as follows:

(1). Use load_svmligh_file function in sklearn library to load the experiment data.

(2). Divide dataset into training set and validation set using train_test_split function.

(3). Initialize SVM model parameters. Set all parameter into zore.

(4). Choose loss function and derivation.

(5). Calculate gradient G toward loss function from all samples.

(6). Denote the opposite direction of gradient G as D.

(7). Update model: $W_t = W_(t-1) + \eta D$, η is learning rate, set as 0.01 .

(8). Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Get the loss L_{train} under the training set and $L_{validation}$ by validating under validation set.

(9). Repeat step e to h for several times, and drawing graph of L_{train} as well as $L_{validation}$ with the number of iterations.

Experimental implementation of the key code is as follows:

```
def get_data():
    data =
        load_svmlight_file("data\\australian_scale")
    return data[0], data[1]
X, y = get_data()
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.2)
def cost(W,b,X,Y):
    C = 5
    cost = np.sum(np.square(W))/2 +
        C*np.sum(np.maximum(0,1 -
        Y*(np.dot(W.T,X)+b))) / X.shape[1]
    return cost
def loss(W,b,X,Y):
    C = 5
    GW = np.zeros(X.shape[1])
    Gb = 0
    filt = (1-Y*(np.dot(W.T,X)+b))>0
    GW = W - C*np.dot(Y*filt,X.T).T/X.shape[1]
    Gb = -C*np.sum(Y*filt*b)/X.shape[1]
    return GW,Gb
def main(n,X_train,y_train,X_test,y_test):
    lenrate = 0.01
    trainCost = []
    validCost = []
    W = np.zeros((X_train.shape[0],1))
    b = 0
    for i in range(n):
        GW,Gb = loss(W,b,X_train,y_train)
        train = cost(W,b,X_train,y_train)
        trainCost.append(train)
        valid = cost(W,b,X_test,y_test)
        validCost.append(valid)
        DW = -GW
        Db = -Gb
        W = W + lenrate*DW
        b = b + lenrate*Db
    return trainCost,validCost
```

The experimental results,which draws graph of L_{train} as well as $L_{validation}$ with the number of iterations,are as follows Fig. 2.

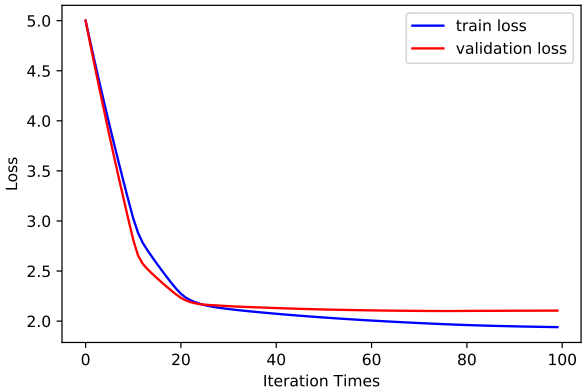

Fig. 2. Loss function curve in linear classification and gradient descent experiments

LinearClassification experimental results show that as the number of iterations increases, the cost of training set and verification set decreases rapidly. However, the final result is not completely stable. The experimental results can be improved by optimizing the code and adjusting the parameters.

IV. Conclusion

I learned a lot from this experiment. First, I got a deeper understanding of linear regression, linear classification and gradient descent. Second, I learned how to use a fixed dataset to train a function model and how to divide the dataset and adjust the parameters during the experiment.