The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Li Jianxiong

Supervisor:
Qingyao Wu

Student ID:
201721045640

Grade:
Graduate

December 13, 2017

Linear Regression, Linear Classification and Gradient Descent

[1]Abstract—

Regression analysis is a method of statistically analyzing data in order to predict the target value of another numerical data from several known data. Assuming that the feature and the result satisfy a linear relationship, that is, satisfying a calculation formula h (x), the argument of this formula is the known data x, and the function value h (x) is the target value to be predicted. This formula is called regression equation, the process of obtaining this equation is called regression.Linear regression is to assume that this method is a linear equation, that is, assume that this equation is a multivariate equation.The purpose of classification is to learn a classification function or classification model (or classifier) to predict unknown categories.

I. INTRODUCTION

The motivation for this experiment is that the motivation for this experiment is to compare and understand the difference between a gradient descent and a stochastic gradient descent, and practice on a small-scale dataset to understand the process of optimization and tuning.

II. METHODS AND THEORY

Linear regression belongs to supervised learning. Therefore, the method should be the same as supervised learning. Given a training set, a linear function is learned from this training set, and then the training of the function is good (that is, whether the function is fit enough for training Set data), pick out the best function (Loss function minimum).

The linear classification method we use is SVM.SVM is a discriminant classifier defined by classification hyperplane. That is to say, given a group of labeled training samples, the algorithm will output an optimal hyperplane to classify the new samples (test samples).

In the machine learning algorithm, when the loss function is minimized, the step-by-step iterative solution can be obtained by the gradient descent method to obtain the minimized loss function and the model parameter values.Gradient descent may not be able to find the global optimal solution, it may be a local optimal solution. Of course, if the loss function is a convex function, the solution obtained by the gradient descent method must be the global optimal solution.

III. EXPERIMENT

1.Purposes:

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.

2.Data sets and data analysis:

Linear Regression uses **Housing** in LIBSVM Data, including 506 samples and each sample has 13 features. We are supposed to divide it into training set, validation set.

Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features. We are supposed to divide it into training set, validation set.

3. Experimental steps

Linear Regression and Gradient Descent

1. Load the experiment data. You can use load_svmlight_file function in sklearn library.
2. Devide dataset. You should divide dataset into training set and validation set using train_test_split function. Test set is not required in this experiment.
3. Initialize linear model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.
4. Choose loss function and derivation: Find more detail in PPT.
5. Calculate gradient toward loss function from all samples.
6. Denote the opposite direction of gradient as .
7. Update model: . is learning rate, a hyper-parameter that we can adjust.
8. Get the loss under the training set and by validating under validation set.
9. Repeate step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

Linear Classification and Gradient Descent

1. Load the experiment data.
2. Divide dataset into training set and validation set.
3. Initialize SVM model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.

4. Choose loss function and derivation: Find more detail in PPT.
5. Calculate gradient toward loss function from all samples.
6. Denote the opposite direction of gradient as .
7. Update model: . is learning rate, a hyper-parameter that we can adjust.
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 under the trainin set and by validating under validation set.
9. Repeate step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

4.The selected loss function and its derivatives:

Linear regression

loss function:

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$$

derivatives:

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

Linear classification

loss function:

$$J(\theta_j) = \frac{1}{n}\sum_{i=1}^{n}\max\{0, 1 - y^{(i)}(w_j x^{(i)} + b)\}$$

derivatives:

$$\frac{\partial}{\partial\theta}J(\theta) = -\frac{1}{n}\sum_{i=1}^{m} y_i \mathbf{x}_i$$

$$st. \qquad y_i \mathbf{w}^T \mathbf{x}_i < 1 \qquad i = 1,2...n$$

5.Core code

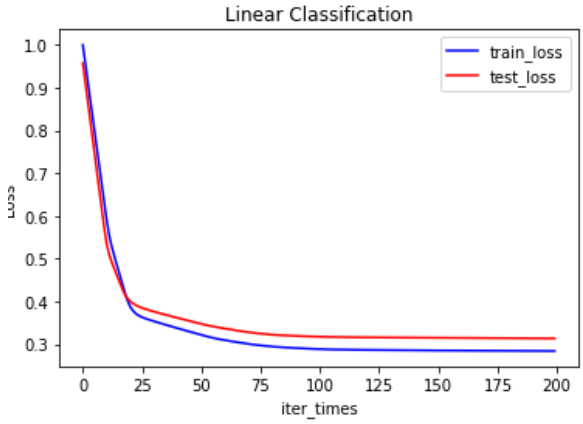
Linear classification

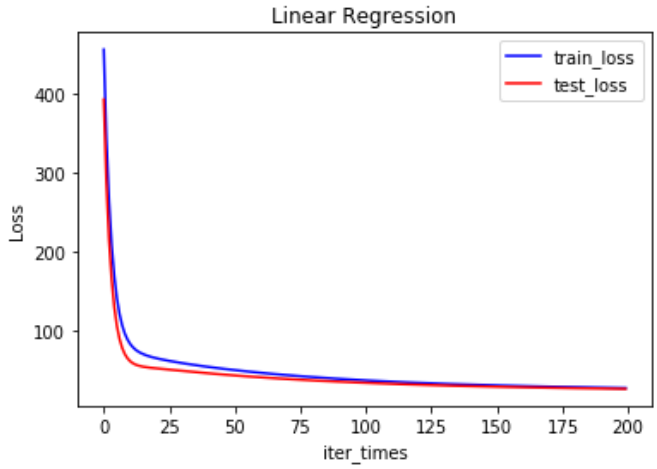
```
def GradientDescent(X_train, y_train, learn_rate, iter_times, X_test, y_test):
    w = np.zeros((X_train.shape[1] + 1,1))
    X_train = np.c_[X_train, ones(X_train.shape[0])]
    X_train = matrix(X_train)
    y_train = matrix(y_train)
    y_train= y_train.T
    X_test = np.c_[X_test, ones(X_test.shape[0])]
    X_test = matrix(X_test)
    y_test = matrix(y_test)
    y_test = y_test.T
     w = matrix(w)
     train_loss = []
     test_loss = []
     for i in range(iter_times):
         f = multiply(y_train, X_train * w)
          w = w + learn_rate*X_train.T*(multiply(f < 1, y_train))
         z = matrix(np.c_[array(1-f), zeros(f.shape)])
         L = (z.max(1).sum())/X_train.shape[0]
         train_loss.append(L)
         f2 = multiply(y_test, X_test * w)
         z2 = matrix(np.c_[array(1-f2), zeros(f2.shape)])
         L2 = (z2.max(1).sum())/X_test.shape[0]
         test_loss.append(L2)
    return w, train_loss, test_loss
```

Linear regression

```
def GradientDescent(X_train, y_train, learn_rate, iter_times, X_test, y_test):
    w = np.zeros((X_train.shape[1] + 1,1))
    X_train = np.c_[X_train, ones(X_train.shape[0])]
    X_train = matrix(X_train)
    X_test = np.c_[X_test, ones(X_test.shape[0])]
    X_test = matrix(X_test)
    y_train = matrix(y_train)
    y_train = y_train.T
    y_test = matrix(y_test)
    y_test = y_test.T
    w = matrix(w)
    train_loss = []
    test_loss = []
    for i in range(iter_times):
            w = w + learn_rate*X_train.T*(y_train - X_train * w)
            L = ((y_train - X_train * w).T * (y_train - X_train* w))/X_train.shape[0]
         train_loss.append(array(L)[0][0])
          L2 = ((y_test - X_test * w).T * (y_test - X_test * w))/X_test.shape[0]
         test_loss.append(array(L2)[0][0])
    return w, train_loss, test_loss
```

6. Experimental results and curve





IV. CONCLUSION

The minimum value of the Loss function is found by gradient descent algorithm to get the optimal solution of linear regression and linear classification function.

Through the experimental results we can see the experimental data fitting degree is very high, linear classification and linear regression combined with gradient decline is a very good method.