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:
Ruolin Liang

Supervisor:
Qingyao Wu

Student ID：2017210460290

Grade:
graduate

December 12, 2017

Linear Regression, Linear Classification and Gradient Descent

Abstract—Linear regression belongs to supervised learning, so the method and supervised learning should be the same, given a training set, a first according to the training set to study out a linear function, and then test the function of training is good (that is, the function fitting enough training set data), picked out the best function (cost function minimum). The goal of linear regression is simple, which is to use a line to fit these points, and to minimize the error between the point set and the fitting functionMotivation of Experiment is further understand of linear regression and gradient descent,.conduct some experiments under small scale dataset and realize the process of optimization and adjusting parameters.

I. INTRODUCTION

The purpose of regression is to predict the target value of another numerical data by several known data. Suppose that the characteristic and the result satisfy the linear relation, that is, to satisfy one formula h (x), the independent variable of this formula is known as the data x, and the function value h(x) is the target value to be predicted. This formula is called regression equation, and the process of getting this equation is called regression. Gradient descent is an algorithm for finding the minimum value of a function. May get local optimal gradient descent, but we have shown that linear regression in optimization problem is only one of the most advantages, because of the loss function J (theta) is a convex quadratic function, won't produce the situation of the local optimum. Instead of scanning the entire training dataset, the stochastic gradient descent is calculated by randomly selecting one data at a time of the fastest decline, which speeds up the iteration. The gradient descent is not converging at the fastest direction of the J (theta), but the oscillations tend to be very small. Support vector machine (SVM) is a kind of second class classification model, the basic model is defined as the characteristic space interval larger linear classifier, the learning strategy is the interval is russsian, finally can be converted into a convex quadratic programming problem. Given some data points, they belong to two different classes, and now you have to find a linear classifier to divide the data into two classes. If expressed in x data points, expressed in y type (y can take 1 or 1, representing two different classes), a linear classifier learning objective is to be found in n dimensional data space a hyperplane (the hyper plane).

II. METHODS AND THEORY

Linear Regression: The target value of another numerical data is predicted by several known data.
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
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.
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
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.
Repeate step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.
Linear Regression:

$$h(x) = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Loss Function:

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

gradient descent:

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}.$$

III. EXPERIMENT

Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features. You are expected to download scaled edition. After downloading, you 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. You are expected to download scaled edition. After downloading, you are supposed to divide it into training set, validation set

Environment for experiment is anaconda3It is includeg following python package: sklearn,numpy,jupyter,matplotlib.

By changing the learning rate, the epoch will observe the experimental results.

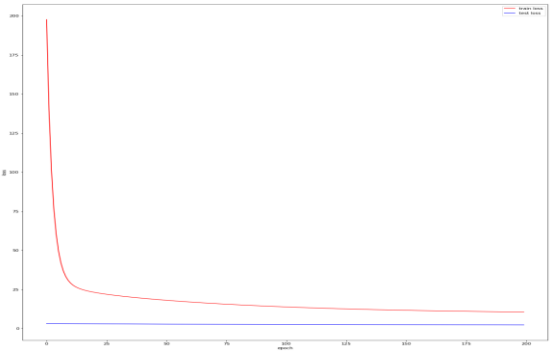
The code is following:

```
from sklearn.datasets import load_svmlight_file
import numpy as np
from sklearn.model_selection import train_test_split


import matplotlib.pyplot as plt

x,y=load_svmlight_file("/home/lrl/ML-logits/housing_scale
")
x=x.todense()

m,n=np.shape(x)
y=y.reshape((m,1))
x_data=np.ones((m,n+1))
x_data[:,:-1]=x[:,:]

x_train,x_test,y_train,y_test=train_test_split(x_data,y,test_s
ize=0.2)


theta=np.ones((n+1,1))


def batchGradienDenscent(epoch,x,y,theta,learning_rate):
    Cost=[]
    for i in range(0,epoch):

        hypothesis=np.dot(x,theta)

        loss=hypothesis-y
        gradient=np.dot(x.T,loss)/m
        theta=theta-learning_rate*gradient
        #cost=np.average(np.abs(np.dot(x,theta)-y))
        cost=1.0/2/m*np.sum(np.square(np.dot(x,theta)-y))
        Cost.append(cost)

    return Cost,theta


learning_rate=0.05
epoch=200
cost_train,theta =
batchGradienDenscent(epoch,x_train,y_train,theta,learning_ra
te)
print theta
cost_test,theta =
batchGradienDenscent(epoch,x_test,y_test,theta,learning_rate)
print theta
epoches=range(200)
plt.figure(figsize=(18,20))
plt.plot(epoches,cost_train,"-",color="r",label="train loss")
plt.plot(epoches,cost_test,"-",color="b",label="test loss")
plt.xlabel("epoch")
plt.ylabel("loss")
plt.legend()
```

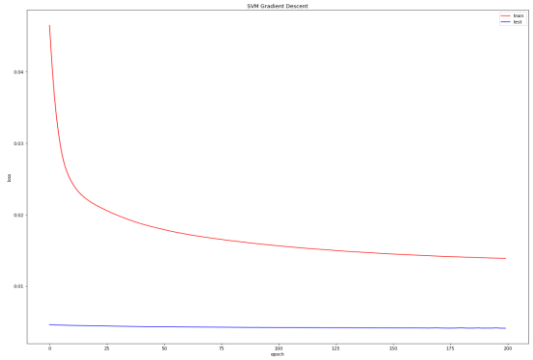
As can be seen from the above picture, the training set has just started to lose a lot, and as the epoch increases, the loss gradually decreases. The validation set verifies the training results and shows the ideal training results. In the experiment, the gradient descent rate can be changed by changing the learning rate

```
from sklearn.datasets import load_svmlight_file
import numpy as np
from sklearn.model_selection import train_test_split


import matplotlib.pyplot as plt

x,y=load_svmlight_file("/home/lrl/ML-logits/classification/
australian_scale")
x=x.todense()
m,n=np.shape(x)
y=y.reshape((m,1))
x_data=np.ones((m,n+1))
x_data[:,:-1]=x[:,:]

x_train,x_test,y_train,y_test=train_test_split(x_data,y,test_s
ize=0.2)
w=np.random.random(size=(n+1,1))
def SVMGradDes(x,y,epoch,learning_rate,w,c):
    losses=[]
    for epoch in range(epoch):
        h=1-y*np.dot(x,w)
        print h.shape
        tmp=np.where(h>0,y,0)

        w=w-learning_rate*(w-c*np.dot(x.T,tmp))
        y_predict=np.where(np.dot(x,w)>0,1,-1)

loss=np.sum(w*w)+c*np.sum(np.maximum(1-y*np.dot(x,w),
0))
        losses.append(loss/m)
```

```
        return losses
    epoch=200
    learning_rate=0.005
    c=0.05
    losses_train=SVMGradDes(x_train,y_train,epoch,learning_
rate,w,c)

    losses_test=SVMGradDes(x_test,y_test,epoch,learning_rate
,w,c)

    plt.figure(figsize=(20,15))
    plt.plot(losses_train,color="r",label="train")
    plt.plot(losses_test,color="b",label="test")
    plt.legend()
    plt.xlabel("epoch")
    plt.ylabel("loss")
    plt.title("SVM Gradient Descent")
    plt.show()
```



This is Linear Regression



This is Linear Classification

As can be seen from the above image, the data is classified by training SVM model, and the larger the epoch, the higher the accuracy of data classification, the smaller the loss. The results of the experiment were also related to the learning rate. The result of the validation set shows that the training result is better.

IV. CONCLUSION

Gradient descent method is an optimization problem solving algorithm. There are two kinds of iterative thinking: batch gradient and stochastic gradient. They have the following differences:The batch gradient converges slowly and the stochastic gradient converges rapidly.The batch gradient is the summary error of all samples before the theta update, and the weight of the gradient descent is updated by examining a sampleThe overhead of batch gradient is large, and the cost of stochastic gradient is small.Use gradient descent method to find the best learning efficiency. This allows for the minimum number of iterations to achieve the precision we need.The experiment results can be improved by changing the learning rate and the parameters of the epoch.