South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Shaojia hong

Supervisor:
Qingyao Wu

Student ID：
201720144993

Grade:
Graduate

December 14, 2017

Linear Regression, Linear Classification and Gradient Descent

Abstract—

I. INTRODUCTION

Linear regression describes the linear relationship between a predictor variable, plotted on the x-axis, and a response variable,plotted on the y-axis,the target is to learn a hypothesis or model $f : X \rightarrow Y$

Linear Classification is a Classification that given training data (xi, yi) for i = 1 . . . n, with $x_i \in R^m$ and $y_i \in \{-1, 1\}$, learn a classfier f(x) such that $f(x_i)\begin{cases}\geq 0 & y_i=+1\\ <0 & y_i=-1\end{cases}$ and $y_i f(x_i) > 0$ for a correct classification.

In order to further understand of linear regression and gradient descent.we do some experiments under small scale data set. And to realize the process of optimization and adjusting parameters.

II. METHODS AND THEORY

A. Dataset

We use two data sets,

The Linear Regression experiments uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features.And divided it into training set and verification set.

The Linear classification experiments uses australian in LIBSVM Data, including 690 samples and each sample has 14 features.

And divided it into training set and verification set.

B. Experimental environment

Python3 and at least the following Python packages are included such as sklearn，numpy，jupyter，matplotlib.

It is recommended to install anaconda3 directly, which has built in the above Python packages.The experimental code and drawing are all done on jupyter.

C. Steps

The step of 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 G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D.
7. Update model: W= W+ η *D, η is learning rate, a hyper-parameter that we can adjust.
8. Get the loss L_train under the training set and L_validation by validating under validation set.
9. Repeated step 5 to 8 for several times, and drawing graph of L_train as well as L_validation with the number of iterations.

The step of 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 G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D.
7. Update model: W= W+ η *D, η 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 L_train under the train set and L_validation by validating under validation set.
9. Repeated step 5 to 8 for several times, and drawing graph of L_train as well as L_validatioin with the number of iterations.

III. EXPERIMENT

A. Formula

1) Linear regression formula

Target function is

$$w = w - \frac{\alpha}{m} * (x * (wx + b - y))$$

Loss function is

$$J = \frac{1}{2m}(wx + b - y)^2$$

2) Linear Classification formula

Target function is

$$g_w(x_i) = \begin{cases} -y_i x_i & 1 - y_i(w^T x_i + b) >= 0 \\ 0 & 1 - y_i(w^T x_i + b) < 0 \end{cases}$$

$$g_b(x_i) = \begin{cases} -y_i & 1 - y_i(w^T x_i + b) >= 0 \\ 0 & 1 - y_i(w^T x_i + b) < 0 \end{cases}$$

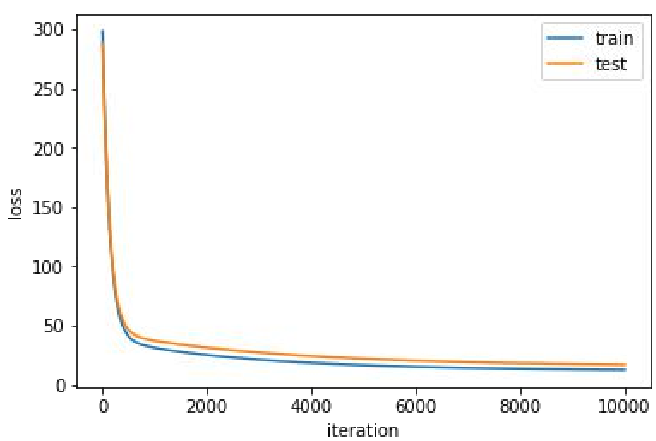
$$\Delta_w L(w, b) = w + \frac{C}{n}\sum_{i=1}^{n} g_w(x_i)$$

Loss function is

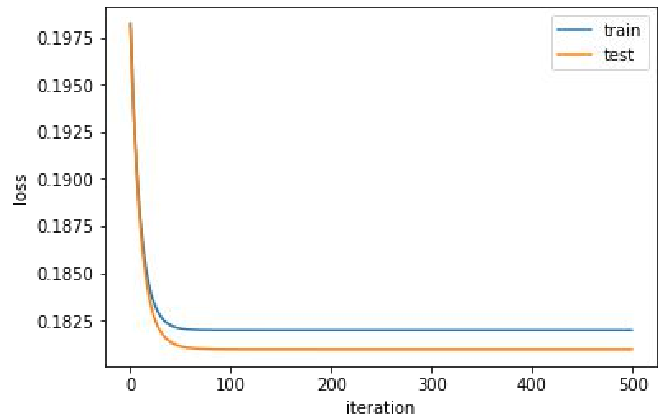
$$\Delta_b L(w, b) = w + \frac{C}{n}\sum_{i=1}^{n} g_b(x_i)$$

B. Experimental results

1. Linear regression result is the following diagram and the linear regression parameter is such
Learn_rate=0.001
maxIteration = 10000



2. Linear Classification result is the following diagram and the linear regression parameter is such
iteration = 500
learning_rate = 0.05



C. code

1) The code of Linear regression and gradient Descent:

```
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
import numpy as np
from matplotlib import pyplot
train,target=load_svmlight_file('F:\housing_scale')
#read data
x_train,x_test,y_train,y_test=
train_test_split(train,target,test_size=  0.2,random_state  =
0)#切分数据集

#compuet loss
def computeCost(X, y, theta):
    m = y.shape[0]
    J = (np.sum((X.dot(theta) - y)**2)) / (2*m)
    return J

def gradientDescent(X, y, theta, alpha, num_iters):
    m = y.shape[0]
    # loss
    J_history = np.zeros((num_iters, 1))
    for iter in range(num_iters):
        J_history[iter] = computeCost(X, y, theta)
        theta = theta - (alpha/m) * (X.T.dot(X.dot(theta) - y))
    return theta,J_history

m, n = np.shape(x_train)
theta= np.zeros((n,1))#init parameter
alpha = 0.001#learnrate
maxIteration = 10000#Iteration number
y_train=y_train.reshape(m,1)
m, n = np.shape(x_test)
y_test=y_test.reshape(m,1)
theta_train,loss_iteration_train=
gradientDescent(x_train,y_train, theta, alpha, maxIteration)
```

```
theta_test,loss_iteration_test = gradientDescent(x_test,y_test, theta, alpha, maxIteration)
pyplot.plot(loss_iteration_train, mfc='w',label='train')
pyplot.plot(loss_iteration_test, mfc='w',label='test')
pyplot.legend()
pyplot.xlabel("iteration")
pyplot.ylabel("loss")
pyplot.show()
```

2) The code of Linear Classification and gradient Descent:

```
from sklearn.datasets import load_svmlight_file
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
from matplotlib import pyplot
import pandas as pd
X,y=load_svmlight_file(r'F:\\australian_scale')
X_train,X_validation,y_train,y_validation=train_test_split(
X,y,test_size=0.25)#分割数据集
y_train=np.reshape(y_train,(len(y_train),1))
y=np.mat(y)
#添加一列 1 在最后一列
b=np.ones((np.shape(X_train)[0],1))
#print(X_train.shape)
X_train=np.column_stack((X_train.toarray(),b))
#print(X_train.shape)
b1=np.ones((np.shape(X_validation)[0],1))
X_validation=np.column_stack((X_validation.toarray(),b1))
m=X_train.shape[1]
loss_train=[]
loss_validation=[]
w = np.zeros((15,))
C = 0.2
iteration = 500
learning_rate = 0.05
def get_loss(X,w,y):
    loss=0
    for i in range(X.shape[0]):
        #loss+=max(0,1-np.sum((y.T)*np.dot(X,w)))
        loss+=max(0,1-y[i]*np.dot(w.T,X[i]))
    loss=(C/X.shape[0])*loss+1/2*(w.T.dot(w))
    return loss

def gradient(X,w,y):
    sum = 0
    for j in range(X.shape[0]):
        if (1-y[j]*np.dot(w.T,X[j]))>=0:
            sum+=-y[j]*X[j]
    w = w + (C/X.shape[0])*sum
    return w


    # w 初始化为 1 维度 m*1
loss_train=[]
loss_validation=[]
for i in range(0,iteration):#it 次迭代
    gra=gradient(X_train,w,y_train)
    w=w-learning_rate*gra
    loss_train.append(get_loss(X_train,w,y_train))

loss_validation.append(get_loss(X_validation,w,y_validation)
)

pyplot.plot(loss_train, mfc='w',label='train')
pyplot.plot(loss_validation, mfc='w',label='test')
pyplot.legend()
pyplot.xlabel("iteration")
pyplot.ylabel("loss")
pyplot.show()
```

IV. CONCLUSION

A. Results analysis

Linear regression:
Through the adjustment of parameters, we can get a better regression results.
Linear classification:
Through the adjustment of multiple hyper-parameters,we can get a better SVM model

B. Summary

Through this experiment,I further understood the principle of linear regression ,linear classification and gradient descent. By learning the gradient descent method, we can further understand the important content of the gradient learning, and realize the process of optimizing and adjusting the parameters.