The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Li Zhang

Supervisor:
Qingyao Wu

Student ID：
201721045909

Grade:
Postgraduate

December 14, 2017

Linear Regression, Linear Classification and Gradient Descent

Abstract—Linear regression and linear classification are the basic algorithm in the field of machine learning. Gradient Descent is an effective way to find the good model parameters for above algorithm. The experiments indicate that the algorithms are effective.

I. INTRODUCTION

Linear regression and linear classification are the basic algorithm in the field of machine learning. Gradient Descent is an effective way to find the good model parameters for above algorithm. This report talks about the experiment of linear regression and linear classification. Besides, the gradient descent is used to find the good parameters.

The rest paper is organized as follow. Section II contains the experiment steps. Section III describes the code and result for the two experiments. Section IV concludes the report.

II. METHODS AND THEORY

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.

Then the experiment will be performed by the following steps:

1) Download the dataset to local host machine.
2) Load dataset into memory.
3) Split dataset into training set and validation set.
4) Create and fill necessary data structures.
5) Write functions for calculating loss and gradient (different in regression and classification).
6) Set parameters (learning rate and the number of iterations).
7) Initiate weights (using normal distribution).
8) Calculate the gradient and update weights according to the gradient calculated in each iteration.
9) Change parameters and run again.
10) Draw plot for experiment result.

III. EXPERIMENT

A. Codes of experiments

The codes of two experiments are shown as follow,

1) Linear Regression

```
#linear regression
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split

#read the data
housing_path = './housing_scale.txt'
X, y = load_svmlight_file(housing_path)

#add 1 for each row
X = X.toarray()
row,col=X.shape
X=np.column_stack((X,np.ones(row)))

#divide data into training and validation
X_train, X_validation, y_train, y_validation =
train_test_split(X, y, test_size=0.2, random_state=30)

#initialize by zeros
W = np.zeros(col+1)

#define learn rate and itreation, loss array
learn_rate = 0.02
iteration = 500

loss_train = np.zeros(interation)
loss_validation = np.zeros(interation)

#loss function
def get_loss(X, W, y):
    diff = y - np.dot(X, W)
    loss = np.dot(diff,diff.T) / (2 * X.shape[0])
    return loss

#gradient function
def get_gradient(X, W, y):
    diff = y - np.dot(X, W)
    G = - np.dot(diff,X)/ X.shape[0]
    return G

#start interation
for i in range(iteration):
    # get loss
    loss_train[i] = get_loss(X_train, W, y_train)
    loss_validation[i] = get_loss(X_validation, W,
y_validation)

    #get gradient and update W
    G = get_gradient(X_train, W, y_train)
```

```
    W = W - learn_rate * G

#draw the result
plt.plot(loss_train,label="loss_train")
plt.plot(loss_validation,label="loss_validation")
plt.legend()
plt.xlabel("iteration")
plt.ylabel("loss")
plt.title("Linear regression")
plt.show()
print(loss_train[iteration-1])
print(loss_validation[iteration-1])
```

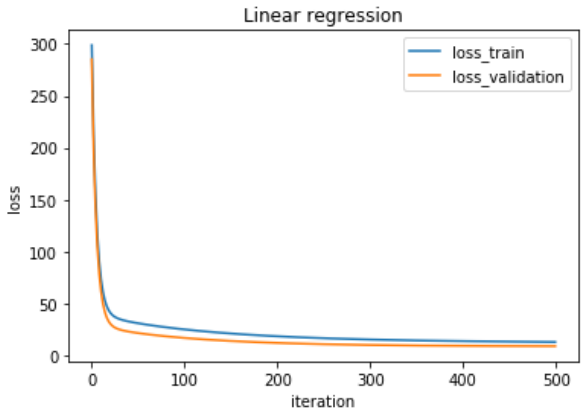
2) Linear Classification

```
#linear classification
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split

#read the data
housing_path = './australian_scale.txt'
X, y = load_svmlight_file(housing_path)

#add 1 for each row
X = X.toarray()
row,col=X.shape
X=np.column_stack((X,np.ones(row)))

#divide data into training and validation
X_train, X_validation, y_train, y_validation =
train_test_split(X, y, test_size=0.2, random_state=30)

#initialize by zeros
W = np.zeros(X_train.shape[1])

#define learn rate, itreation, lambdal, loss array, accuracy
array
learn_rate = 0.05
iteration = 500
lambdal = 0.01

loss_train = np.zeros(iteration)
loss_validation = np.zeros(iteration)
accuracy = np.zeros(iteration)

#loss function
def get_loss(X, W, y, lambdal, W_0):
    diff = np.ones(y.shape[0]) - y * np.dot(X, W)
    diff[diff < 0] = 0
    loss = np.sum(diff) / X.shape[0] +
np.dot(W_0,W_0.T)/2*lambdal
    return loss

#gradient function
def get_gradient(X, W, y, lambdal, W_0):
    diff = np.ones(y.shape[0]) - y * np.dot(X, W)
    y_get = y.copy()
    y_get[diff <= 0] = 0
    G = -np.dot(y_get,X) / X.shape[0] + W_0 * lambdal
    return G

#accuracy function
def get_accuracy(x, W, y):
    preY = np.dot(X,W)
    count = np.sum(preY * y >0)
    Accuracy = count / X.shape[0]
    return Accuracy


#start iteration
for i in range(iteration):
    W_0 = W.copy()
    W_0[col-1]= 0
    # get loss
    loss_train[i] = get_loss(X_train, W, y_train, lambdal,
W_0)
    loss_validation[i] = get_loss(X_validation, W,
y_validation, lambdal, W_0)

    #get accuracy
    #accuracy[i] = get_accuracy(X_validation, W,
y_validation)

    #get gradient and update W
    G = get_gradient(X_train, W, y_train, lambdal, W_0)
    W = W - learn_rate * G

#draw the result
plt.plot(loss_train,label="loss_train")
plt.plot(loss_validation,label="loss_validation")
plt.legend()
plt.xlabel("iteration")
plt.ylabel("loss")
plt.title("Linear classification")
plt.show()
print(loss_train[iteration-1])
print(loss_validation[iteration-1])
```

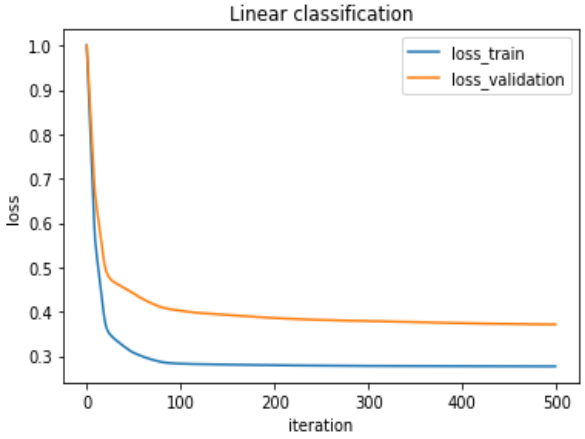
B. Experiment results

1) Linear Regression

In this set of experiments, we set the learning rate to 0.02 and iteration number to 500. The result is shown in the following figure. The validation loss can get to 9.08 after the iteration.



2)Linear Classification
In this set of experiments, we set the learning rate to 0.05 and iteration number to 500. The result is shown in the following figure. The validation loss can get to 0.37 after the iteration.



IV. CONCLUSION

After the experiments, we know the parameters such as learning rate and iteration are very important. If the iteration number is less, the loss of the model would be high. If the learning rate is too high, the loss function would become not convergent.