

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **林英杰**

**Student ID 201530612286**

**E-mail 1142983747@qq.com**

**Tutor**   **谭明奎**

**Date submitted** **2017.12.08**

**1. Topic:**[Linear Regression, Linear Classification and Gradient Descent](https://www.zybuluo.com/chenyaofo/note/949886)

**2. Time:** 2017.12.02

**3. Reporter:** 林英杰

**4. Purposes:**

Further understand of linear regression and gradient descent.

Conduct some experiments under small scale dataset.

Realize the process of optimization and adjusting parameters.

**5. Data sets and data analysis:**

Linear Regression uses [Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#housing) in [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/), 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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#australian) in [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/), 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.

**6. Experimental steps:**

Linear Regression and Gradient Descent

1. Load the experiment data. You can use [load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html) function in sklearn library.
2. Devide dataset. You should divide dataset into training set and validation set using [train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) 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.

**7. Code:**

Linear Regression

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

from sklearn.model\_selection import train\_test\_split

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("G:\\学习资料\\大三上\\机器学习\\experiment\\experiment1\\housing\_scale.txt")

return data[0], data[1]

import numpy as np

import matplotlib.pyplot as plt

X, y = get\_data()

X = X.toarray()

n,m=np.shape(X)

X=np.hstack((X,np.ones((n,1))))

y = y.reshape((n,1))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y)

rate=0.1 #learning rate

w=np.ones((m+1,1)) #initialize the w

num=500 #times

loss\_train=np.zeros((num)) #the loss of train set

loss\_test=np.zeros((num)) #the loss of test set

def h(x):

return x.dot(w)

for i in range (num):

w+=rate\*X\_train.T.dot((y\_train-h(X\_train)))/(X\_train).shape[0]

loss\_train[i]=(y\_train-h(X\_train)).T.dot((y\_train-h(X\_train)))/(X\_train).shape[0]

loss\_test[i]=(y\_test-h(X\_test)).T.dot((y\_test-h(X\_test)))/(X\_test).shape[0]

print(loss\_test[i])

i =range(num)

#Draw graphs of Ltrain and Ltest with the number of iterations

plt.plot(i, loss\_train,'g',label = 'train')

plt.plot(i,loss\_test,'y',label = 'test')

plt.legend(loc='upper right')

plt.xlabel('times')

plt.ylabel('loss')

plt.show()

Linear Classification

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

from sklearn.model\_selection import train\_test\_split

mem = Memory("./mycache")

@mem.cache

def get\_data():

data = load\_svmlight\_file("G:\\学习资料\\大三上\\机器学习\\experiment\\experiment1\\australian\_scale.txt")

return data[0], data[1]

import numpy as np

import matplotlib.pyplot as plt

X, y = get\_data()

X = X.toarray()

n,m=np.shape(X)

X=np.hstack((X,np.ones((n,1))))#add a coloum of 1 after X

y = y.reshape((n,1))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y)

rate=0.1 #learning rate

w=np.ones((m+1,1)) #initialize the w

one=np.ones((X\_train.shape[0],1)) #create a 1 array of (X\_train.shape[0],1)

one2=np.ones((X\_test.shape[0],1)) #create a 1 array of (X\_test.shape[0],1)

num=500 #times

loss\_train=np.zeros((num)) #the loss of train set

loss\_test=np.zeros((num)) #the loss of test set

def h(x):

return x.dot(w)

#train set

for i in range (num):

temp\_loss=0.0

temp\_loss2=0.0

temp\_w=np.zeros((m+1,1)) #initialize the temp\_w

temp=one-y\_train\*h(X\_train)

for j in range(temp.shape[0]):

if(temp[j]>0):

temp\_w+=(y\_train[j]\*X\_train[j].T).reshape((15,1))

temp\_loss+=temp[j]

temp\_w=w-temp\_w/(X\_train.shape[0])

w-=rate\*temp\_w #update the w

loss\_train[i]=temp\_loss+0.5\*np.sum(w\*w)

#test set

temp2=one2-y\_test\*h(X\_test)

for k in range(temp2.shape[0]):

if(temp2[k]>0):

temp\_loss2+=temp2[k]

loss\_test[i]=temp\_loss2+0.5\*np.sum(w\*w)

print(loss\_test[i])

i =range(num)

#Draw graphs of Ltrain and Ltest with the number of iterations

plt.plot(i, loss\_train,'g',label = 'train')

plt.plot(i,loss\_test,'y',label = 'test')

plt.legend(loc='upper right')

plt.xlabel('times')

plt.ylabel('loss')

plt.show()

**8. Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.):**

Using the cross-validation method, the data set is divided into training set and verification set at random, which does not divide the test set.

**9.The initialization method of model parameters:**

All zero initialization

1. **The selected loss function and its derivatives:**

Linear Regression

Loss function: 

Multi-attribute equivalent to 

(Where m is the number of samples, X is a matrix of 1's added to the right of the original X)

Gradient: the gradient of  

For w14(b) x=1

The gradient vector of the entire vector 

Linear Classification

Loss function: 

Gradient: the gradient of wi 

**11. Experimental results and curve:**

## Hyper-parameter selection (η, epoch, etc.):

For Linear Regression

## η is 0.1

For Linear Classification

## η is 0.01

## Assessment Results (based on selected validation):

Linear Regression

(1)After iteration 100 times Ltrain = 26.0673790326, Ltest = 22.2608265437

W=

[[-5.9461104 ]

[-0.55493657]

[-2.05196066]

[ 0.67528403]

[-2.05953383]

[ 8.32592024]

[ 0.70805464]

[-3.70295323]

[ 1.61532602]

[-0.65370171]

[-4.32339231]

[ 3.57839212]

[-8.41881057]

[ 8.58479336]]

(2)After iteration 1000 times Ltrain = 22.3751152863, Ltest = 21.1768932027

W=

[[-5.68939764]

[ 2.73617187]

[ 0.08414575]

[ 0.83096533]

[-4.49936836]

[ 9.32535197]

[ 0.23864832]

[-8.94874777]

[ 3.27689477]

[-2.55759696]

[-4.80965596]

[ 1.96802414]

[-9.73711655]

[ 9.83780719]]

(3)After iteration 10000 times Ltrain = 21.9852310425, Ltest = 22.3832134546

W=

[[ -4.80454296]

[ 2.29937883]

[ 0.61177442]

[ 1.52318644]

[ -4.24235742]

[ 10.42565509]

[ -0.17487613]

[ -8.3439086 ]

[ 3.4031266 ]

[ -3.48922815]

[ -4.62762609]

[ 1.53223491]

[ -8.53327522]

[ 12.3921828 ]]

Linear Classification

(1)After iteration 100 times Ltrain = 234.689951735, Ltest = 95.6897212592

W=

[[ 0.36695443]

[ 0.28390532]

[ 0.25890898]

[ 0.33762654]

[ 0.40483045]

[ 0.36843318]

[ 0.2232014 ]

[ 0.53779172]

[ 0.42236547]

[ 0.21555724]

[ 0.28945046]

[ 0.36117603]

[ 0.21373391]

[ 0.1988344 ]

[ 0.54123203]]

(2)After iteration 1000 times Ltrain = 227.673224456, Ltest = 70.2946270586

W=

[[-0.00807003]

[ 0.0340721 ]

[ 0.04284098]

[ 0.04451316]

[ 0.12745055]

[ 0.08491518]

[ 0.045253 ]

[ 0.53646754]

[ 0.28503643]

[ 0.02966325]

[ 0.01513569]

[ 0.02505062]

[-0.02346342]

[-0.00060444]

[ 0.0151982 ]]

(3)After iteration 10000 times Ltrain =209.209994141, Ltest = 77.3699752742

W=

[[ 0.00404296]

[ 0.02238391]

[ 0.02879584]

[ 0.04545502]

[ 0.12096061]

[ 0.05920587]

[ 0.03618177]

[ 0.55101633]

[ 0.32277649]

[ 0.03357869]

[ 0.00704839]

[ 0.01033566]

[-0.02917938]

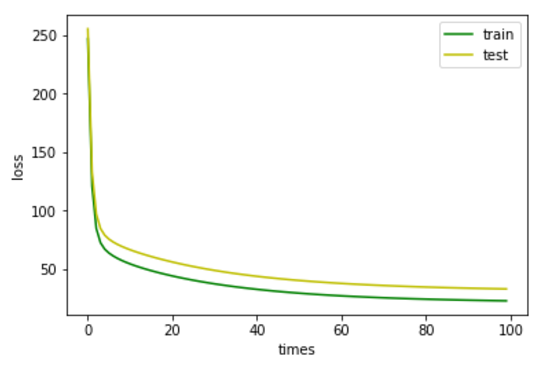
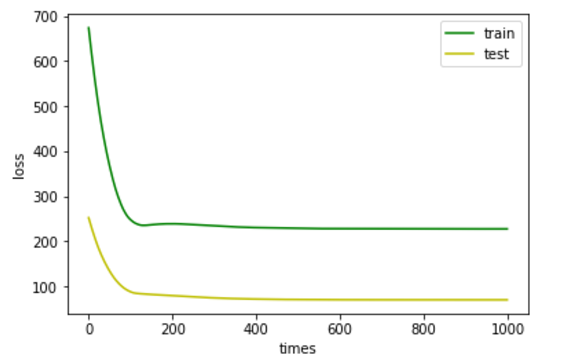
[-0.00421108]

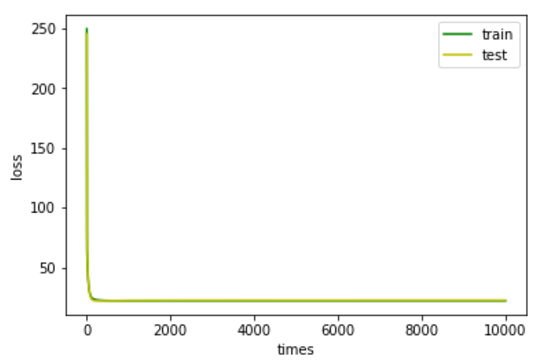
[ 0.01509854]]

## Predicted Results (Best Results):

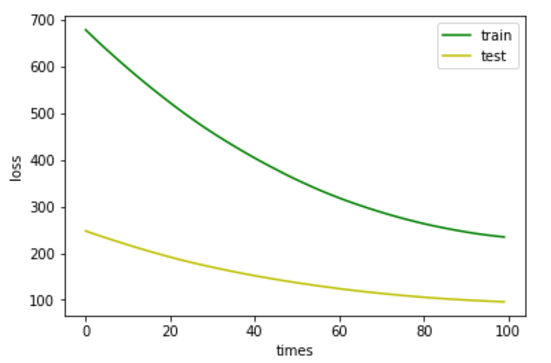
## Loss curve:

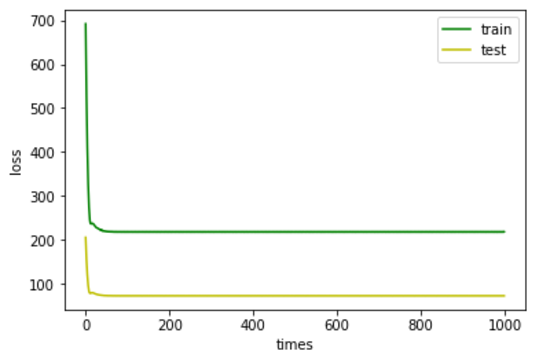
Linear Regression

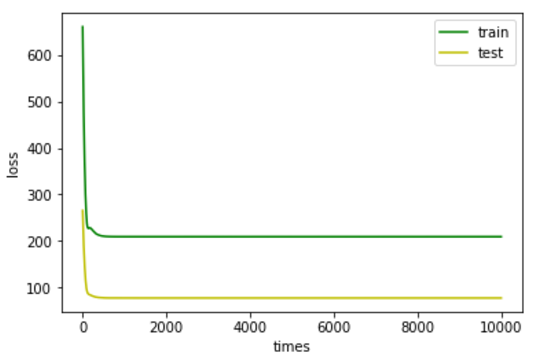
 



Linear Classification







1. **Results analysis:**

Linear Regression

The Loss function value drops rapidly after the previous update iteration cycles. After about 100 iterations, the Loss value has dropped to about 1/10 of the initial value. After 1000 iterations, the Loss value has been reduced to a very small extent. However, As the iteration has been shown to be a very small decline, we can expect it to continue to decline.

Ltrain and Lvalidation keep small differences, and gradually stabilize within a very small range as the iteration proceeds (especially after 50 iterations).

Linear Classification

In the first 20 iterations, the Loss function can rapidly decrease, and the decrease decreases significantly after 20 iterations. After about 100 iterations, the Loss value has dropped to about 1/3 of the initial value, and the subsequent decrease is also very significant Small, but also showed a small increase, you can think Loss value fluctuations in the vicinity of a real value fluctuations, ignoring the error can be regarded as stable. The proportions of Ltrain and Lvalidation are basically the same as the proportions of training set and test set

1. **Similarities and differences between linear regression and linear classification:**

The most basic commonality between linear regression and linear classification is the construction of a model by the linear formula of .

The value of the label y in the linear classification is limited to a few (in this experiment, two) discrete values, but the same value can simultaneously correspond to the value of x in a certain region in the super space. In the linear regression, only a small region , Even a specific x value corresponds to a y value.

From the experimental results, the magnitude of the Loss value of the linear classification is always proportional to the sample size of the dataset. For a dataset of different sizes, there is a certain probability of a mismatch between models with the same "training level", so the larger the dataset , The greater the Loss value.

1. **Summary:**

Through this experiment, I realized that through the training of data to learn the model process, a deeper feeling of the significance of machine learning. Let me better grasp the syntax of python and various packages, but also let me consolidate the knowledge of linear algebra. In this experiment, I saw the similarities and differences between linear regression and linear classification. My understanding of these two concepts is even more profound. I am also more familiar with and understand the gradient descent method