

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  wende zhu

**Student ID 201530613948**

**E-mail 201530613948@mail.scut.edu.cn**

**Tutor**

**Date submitted** **2017. 12.**  8

**1. Topic:** Linear Regression, Linear Classification and Gradient Descent

**2. Time:** 2017/12/8

**3. Reporter:** wende zhu

**4. 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.

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

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.

**6. 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 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 G as D.

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:**

**Regression:**

def compute\_gradient(b\_current,m\_current,x,y,learning\_rate):

b\_gradient = 0

m\_gradient = 0

n=len(y)

N = float(n)

#x = data[:,0]

#y = data[:,1]

#temp = np.ones((n, 1))\*b\_current

y.shape=(n,1)

m\_current.shape=(13,1)

b\_gradient = -(2/N)\*(y-x.todense().dot(m\_current)-b\_current)

b\_gradient = b\_gradient.mean()

m\_gradient = -(2/N)\*x.todense().T\*(((y-x.todense().dot(m\_current)-b\_current)))

m\_gradient = m\_gradient.mean(axis=0)

new\_b = b\_current-(learning\_rate\*b\_gradient)

new\_m = m\_current-(learning\_rate\*m\_gradient)

return[new\_b,new\_m]

**Classification:**

def compute\_gradient(b\_current,m\_current,x,y,learning\_rate,c=0.9):

b\_gradient = 0

m\_gradient = 0

gw=0

gb=0

n=len(y)

N = float(n)

i = random.randint(0,n-1)

y.shape=(n,1)

m\_current.shape=(14,1)

if((1-y[i]\*(x[i,:].dot(m\_current))>=0)):

gw=-y[i]\*x[i,:]

gb=-y[i]

m\_gradient = c \* (gw.T)

else:

gw = 0

gb = 0

m\_gradient = m\_current

b\_gradient = c\*gb

new\_b = b\_current-(learning\_rate\*b\_gradient)

new\_m = m\_current-(learning\_rate\*m\_gradient)

return[new\_b,new\_m]

(Fill in the contents of 8-12 respectively for linear regression and linear classification)

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

**9. The initialization method of model parameters:** zeros, all are zero

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

**Regression:**

**Classification:**

**11. Experimental results and curve:**

## Hyper-parameter selection (η, epoch, etc.): LR: learning rate 0.01 LC: learning rate: 0.008, C: 0.9

## Assessment Results (based on selected validation):

## Predicted Results (Best Results):

LR:

b = 14.948480865819459

m=[[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]

[-2.32252712]]

LC:

b = [-0.2376]

m=[[-0.0015084 ]

[ 0.01125958]

[ 0.00431275]

[ 0.02905825]

[ 0.11434719]

[ 0.01627837]

[ 0.01818385]

[ 0.84958798]

[ 0.11472354]

[-0.00970689]

[ 0.00264126]

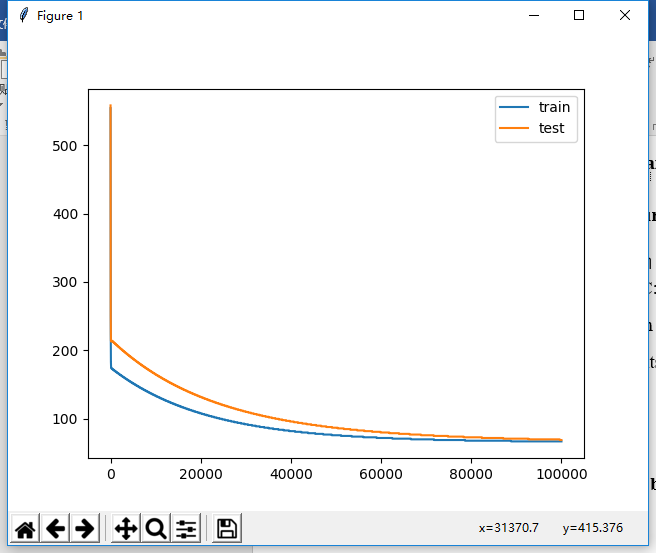
[ 0.02402752]

[-0.11336276]

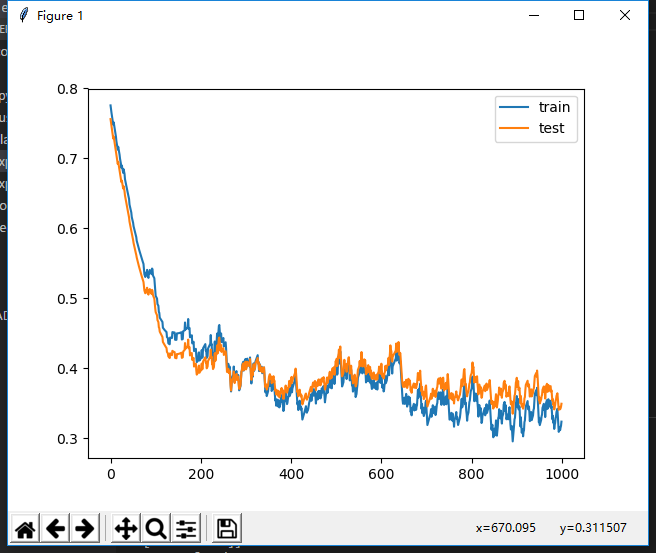
[-0.04270863]]

## Loss curve:

LR:



LC:



**12. Results analysis:**

In linear regression, I have adjust learning rate from 0.0001 to 1, finally 0.01 is the best rate. The loss function decrease less and less in after 100 iterators. So the learning rate is still so large.

In linear classification, I have adjust the gradient, in PPT, gradient about w has a item which is w, I find it maybe be wrong, when I delete it the loss function has a good curve.

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

Differences: LR and LC’s gradient is different, LC’s gradient is section function, but LR’s is continuous function.

**14. Summary:** LR and LC in gradient decent is almost similar. But SVM has other algorithm to optimize such as SMO. But I don’t understand it.