

**The Experiment Report of**

***Deep Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **褚涛**

**Student ID 201710106550**

**E-mail 1042832684@qq.com**

**Tutor**   **刘琼**

**Date submitted** **2017.12.7**

1. **Topic:**

Linear Regression, Linear Classification and Gradient Descent

1. **Time:**

2017.12.2—2017.12.7

**3. Reporter:**

Chu Tao

1. **Purposes:**

Further understand of linear regression and gradient descent.

Conduct some experiments under small scale dataset.

Realize the process of optimization and adjusting parameters.

1. **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.

1. **Experimental steps:**

The experimental code and drawing are completed on jupyter.

**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 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.

Finishing experiment report according to result: The template of report can be found in example repository.

**7. Code:**

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

**#the iteration process of** **Linear Regression**

for i in range(it):

y\_pre=X\_train\*W.T+b

L[i]=1/2\*sum((y\_pre-y\_train)\*\*2) #compute loss function

D=-(y\_pre-y\_train).T\*X\_train # compute the inverse number of gradients

W=W+0.001\*D #update weight

b=b-0.001\*np.sum(y\_pre-y\_train)

y\_pre\_test=X\_test\*W.T+b #predict test set

L0[i]=1/2\*sum((y\_pre\_test-y\_test)\*\*2) #compute loss function of test set

**#the iteration process of Linear Classification**

for i in range(it):

y\_pre=X\_train\*W.T+b

L[i]=sum(np.max([[0]\*ntr,-y\_pre\*y\_train],axis=0)) #compute loss function

res=y\_pre\*y\_train

ac[i]=np.array((res[res>0].shape))/ntr # compute classification accuracy of training sets

D=np.max([[0]\*ntr,-y\_pre\*y\_train],axis=0)\*y\_train\*X\_train # compute the inverse number of gradients

W=W+0.0001\*D #update weight

b=b+0.0001\*np.sum(y\_train\*np.max([[0]\*ntr,-y\_pre\*y\_train],axis=0))

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

Select 80% of data set for training and 20% for testing

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

Linear Regression: All weights are set zero

Linear Classification: All weights are set random number in (0,1)

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

Linear Regression:

loss function:

gradient function:

Linear Classification:

loss function:

gradient function:

**11. Experimental results and curve:**

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

Linear Regression: iteration=100, leraning rate=0.001

Linear Classification: iteration=100, leraning rate=0.0001

## Assessment Results (based on selected validation):

Linear Regression: best Loss=4530.52503854

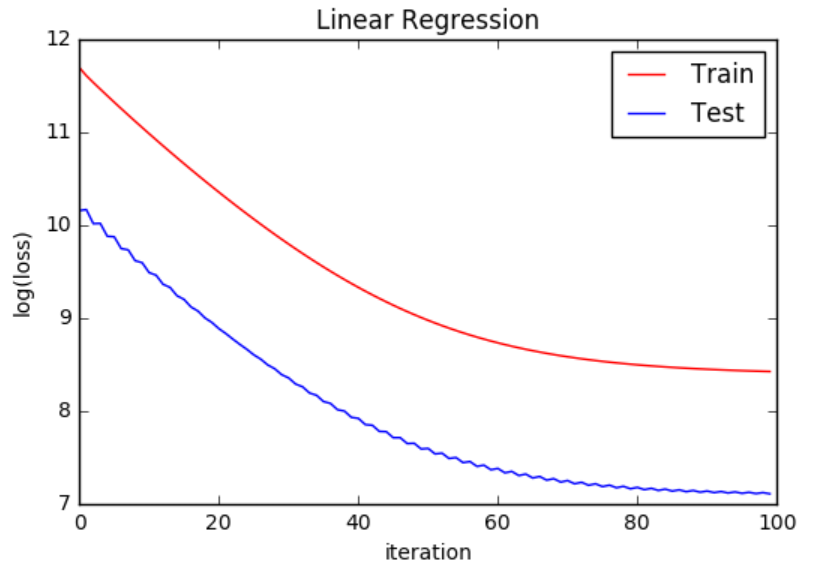
Linear Classification: best Loss= 7.89327186637

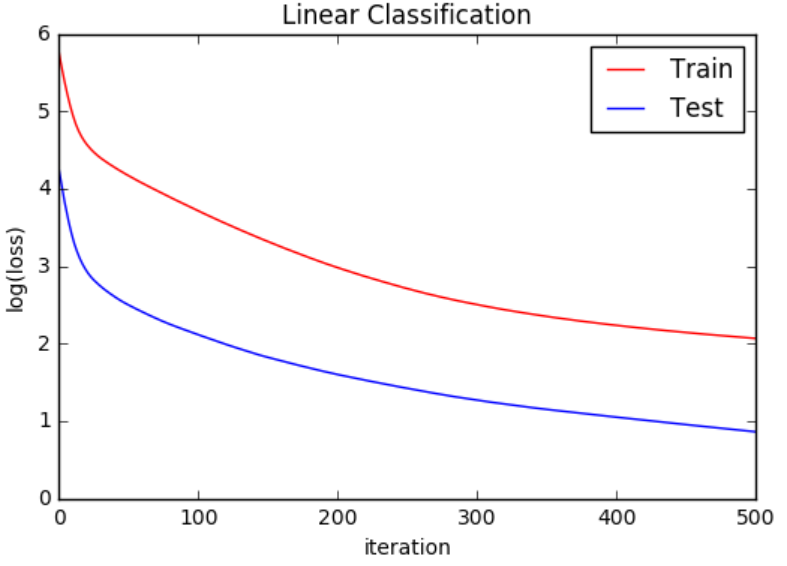
## Predicted Results (Best Results):

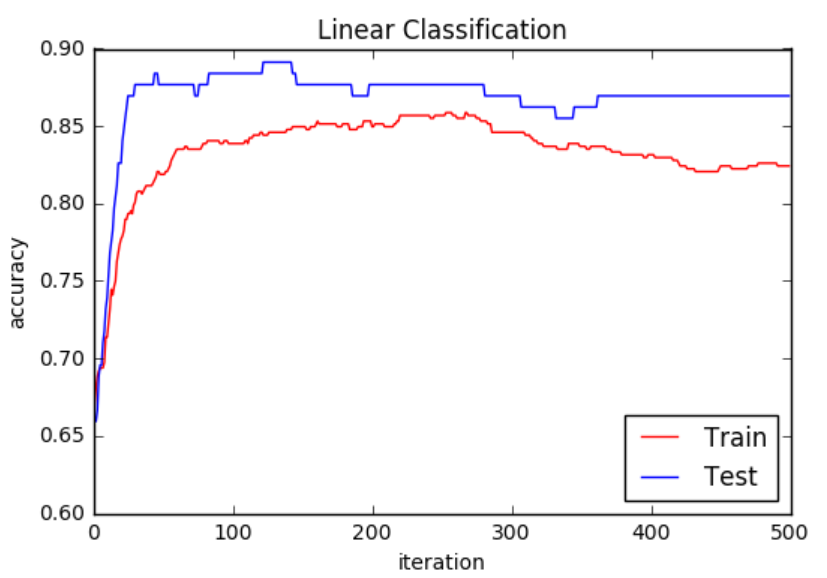
Linear Regression: best Loss= 1214.90769544

Linear Classification: best Loss= 2.36456842678

## Loss curve:







**12. Results analysis:**

Loss value decreases as the number of iterations increases. But when the number of iterations of the classification is increased, due to over fitting, the classification accuracy will rise first and then decrease.

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

The linear regression considers the loss value of all points to make all points as good as possible. But the linear classification only considers the loss values of the wrong points of classification to get more correct classification samples.

**14. Summary:**

By using linear regression and linear classification, we can solve some simple linear problems. But it is incapable of action for complex problems. And there exists the overfitting problem in linear methods. However, problems are always from simple to complex, nonlinear and complex methods change a little based on linear method. Therefore, this experiment let me learn some basic but important knowledge.