

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **文威**

**Student ID 201530613009**

**E-mail wenwei567@gmail.com**

**Tutor**   **谭明奎**

**Date submitted** **2017.12.3**

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

**2. Time: 2017.12.2**

**3. Reporter:文威**

**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:**

For the Linear Regression experiment, I use the Housing data in LIBSVM Data, which includes 506 samples and each sample has 13 features. For the Linear Classification, I use the Australian data in LIBSVM Data, which includes 690 samples and each sample has 14 features.

1. **Experimental steps:**

**Linear Regression:**

1)Load the experiment data. You can use load\_svmlight\_file function in sklearn library.

2)Divide 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:

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)Repeat step 5 to 8 for several times, and drawing graph of  as well as  with the number of iterations.

**Linear Classification：**

1.Load the experiment data.

2.Divide data set into training set and validation set.

3.Initialize SVM model parameters.

4.Choose loss function：

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 training set and  by validating under validation set.

9.Repeat step 5 to 8 for several times, and drawing graph of  as well as  with the number of iterations.

1. **Code:**

**Linear Regression:**

def GradientDescent(X,y,w,iteration,lr,train\_loss,X\_test,y\_test,test\_loss ):

for i in range(iteration):

gradient = Gradient(X,y,w)

w = w - lr\*gradient

train\_loss.append(Loss(X,y,w))

test\_loss.append(Loss(X\_test,y\_test,w))

return w

**Linear Classification：**

def Gradient(X,y,w):

c = 0.9

n\_sample,n\_feature = X.shape

f1 = np.ones((n\_sample,1)) - y \* np.dot(X,w)

gradient = np.zeros((1,n\_feature))

for i in range (n\_sample):

if(f1[i,0] < 0):

gradient = gradient + w.T;

else:

gradient = gradient + w.T - (c \* np.dot(y.T,X))

gradient = gradient/n\_sample

return gradient

def GradientDescent(X,y,w,iteration,lr,train\_loss,X\_test,y\_test,test\_loss):

for i in range(iteration):

gradient = Gradient(X,y,w)

w = w - lr \* gradient.T

train\_loss.append(Loss(X,y,w))

test\_loss.append(Loss(X\_test,y\_test,w))

return w

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

I use hold-out method for both experiments.

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

I initialize all parameter to zero in linear regression experiment; I initialize w parameter to zero and initialize C parameter to 1 in linear classification experiment.

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

**Linear Regression:**

**Loss function:**

****

**Gradient function:**

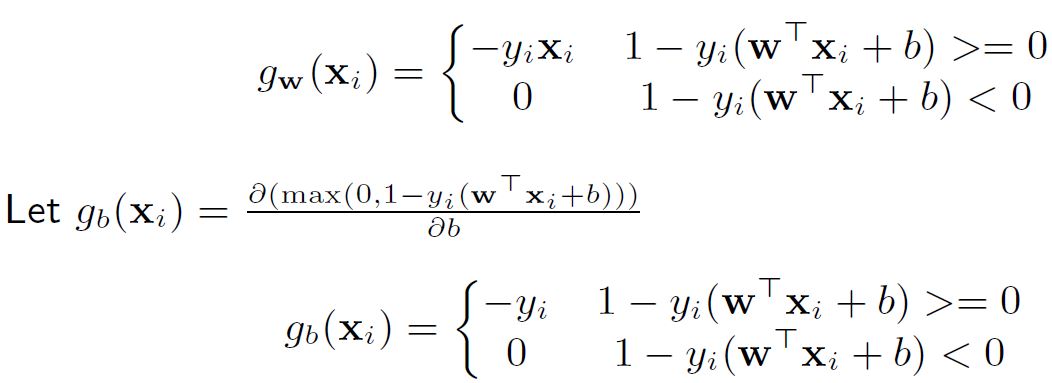
****

**Linear Classification：**

**Loss function:**

****

**Gradient function:**







**11. Experimental results and curve:**

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

Linear Regression: η = 0.01 ,epoch = 600

Linear Classification： η =0.00001, epoch = 700, C= 0.9 (hinge loss parameter)

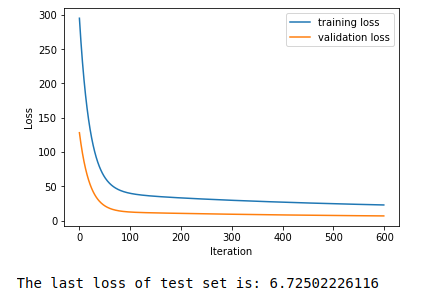
## **Assessment Results (based on selected validation):**

For the Linear Regression, I get the loss = 6.73 based on selected validation.

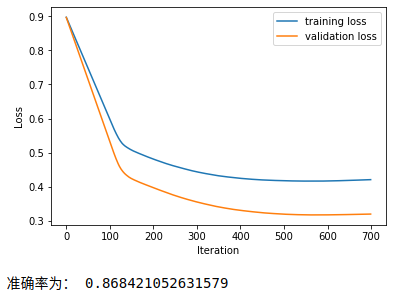
For the Linear Classification, I get the accuracy = 96.8% based on selected validation.

## **Loss curve:**

Linear Regression:



Linear Classification:



1. **Results analysis:**

In both experiment, the loss function decrease significantly at the first several iterations, then get down to a steady value.

If the learning rate is too large, then the loss function can't converge, even going to infinity. If the learning rate is too small,then the loss function will converge so slow.

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

Similarity:

1)Both of the linear regression and linear classification use a linear model to solve problem. They both use hyper surface.

2)They have the same optimization algorithm. Both of they use gradient descent algorithm to optimize the parameter of the models.

Difference:

1)The objective function is different. For linear regression, we need to get a model which predicted value get closer to our answer. For linear classification, we need to get a larger margin with the giving data.

2)The use of linear regression and linear classification model is different. Linear regression is to solve continuous value’s prediction. But linear classification is to divide two class of data.

1. **Summary:**

In this experiment, I build a linear regression classification and a linear classification model. I use the gradient descent algorithm to optimize the parameter of these two model. And I realize that it is important to set proper initial value and learning rate to get a better result.

This experiment make me more familiar with the calculation of matrix，the use of anaconda and the realization of machine learning. I get a big harvest by this experiment.