

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  HuXianlin

**Student ID 201530781104**

**E-mail 865403039@qq.com**

**Tutor**   **TanMingkui**

**Date submitted** **2017.12.8**

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

**2. Time: 2017.12.2**

**3. Reporter:**HuXianlin

**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\_set\_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 G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D .
7. Update model: η. η 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 G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D .
7. Update model: η. η 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. Repeate step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

**7. Code:**

Regression:

for i in range(1,500):

graph\_x.append(i)

G=loss\_g(X\_train,y\_train,w)

w-=n\*G

l\_train=loss(y\_train,X\_train,w)

l\_validation=loss(y\_test,X\_test,w)

Lt.append(l\_train)

Lv.append(l\_validation)

Classification:

for i in range(1,500):

graph\_x.append(i)

G=loss\_g(X\_train,y\_train,w)

w-=n\*G

l\_train=loss(y\_train,X\_train,w)

l\_validation=loss(y\_test,X\_test,w)

Lt.append(l\_train)

Lv.append(l\_validation)

predict=[]

for j in range(1,len(X\_test)):

if f(X\_test[j-1],w)>=0.1:

predict.append(1)

else:

predict.append(-1)

sum=0

for j in range(0,len(predict)-1) :

if predict[j]==y\_test[j]:

sum=sum+1

acc.append(sum/len(predict))

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

hold-out

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

set all parameter into zero

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

Regression：

Classification：

**11. Experimental results and curve:**

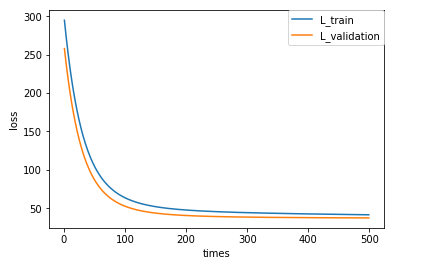
**Regression：**

Hyper-parameter selection：η=0.00001 epoch=500

Assessment Results (based on selected validation): Loss curve converges

Predicted Results (Best Results): Loss curve converges below 25

Loss curve :



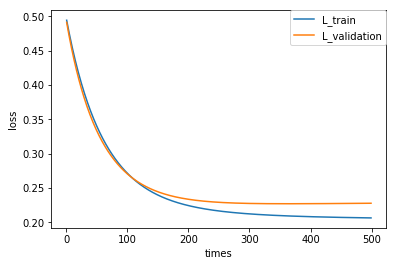
**Classification**：

Hyper-parameter selection：η=0.00001 epoch=500 threshold=0.1

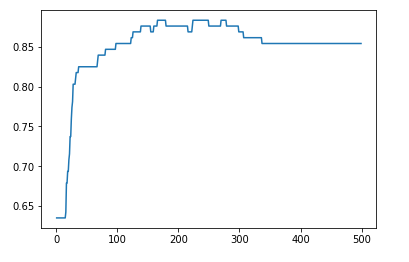
Assessment Results (based on selected validation): Loss curve converges

Predicted Results (Best Results): Loss curve converges below 0.25, accuracy rate=0.86

Loss curve :



accuracy rate：



**12. Results analysis:**

Regression: By 500 iterations, the loss function got stable. The L\_train and L\_validation both dropped below 0.25, proving the experimental result is correct.

Classification: By 500 iterations, the loss function got stable. The L\_train and L\_validation both dropped below 0.25. And the accuracy is stable at 0.86, which is a reasonable value.

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

Overall, the two questions are essentially the same, that is, the fitting (matching) of the model. However, the y value (label) of the classification problem is more discretized, and the same y value may correspond to a large number of x of a certain range.Therefore, the classification problem is usually some x in a certain region corresponds to one y, and the regression problem is usually x in a very small region(or generally one x) to one y.

**14. Summary:**

Through this experiment, I deepened my understanding of linear regression and linear classification problems and learned to use ipython notebook and git. During the experiment I encountered many problems, such as code bugs, not familiar with the python library, do not use tools. It took me a lot of time , but eventually I finished it and learned a lot from it.