

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **阮子琦**

**Student ID 201530612668**

**E-mail** [609387481@qq.com](mailto:609387481@qq.com)

**Tutor**   **Prof.Mingkui Tan**

**Date submitted** **2017. 12 .07**

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

Linear Classification and Gradient Descent

**2. Time:** 2017-12-04

**3. Reporter:**阮子琦

**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:**

In the regression experiment,we use Housing in LIBSVM Data, including 506 samples and each sample has 13 features.It is a regression data set.

In the classification experiment,we use australian in LIBSVM Data, including 690 samples and each sample has 14 features.It is a classification data set.

**6. Experimental steps:**

The experimental steps of regression experiment is as followed:

1. Load the experiment data. You can use [load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "_blank) 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" \t "_blank) 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.

The experimental steps of classification experiment is as followed:

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:**

**1.**The code used to train data in regression experiment is below:

loss\_train\_tuple = []

loss\_test\_tupe = []

for i in range(1000):

for j in range(col\_theta):

descent = compute\_descent(j)

theta\_array[j] = theta\_array[j] - descent

loss\_train\_tuple.append(train\_loss())

loss\_test\_tupe.append(test\_loss())

def compute\_descent(num):

sum = 0

for x,y in zip(x\_train\_array,y\_train):

sum = sum + ( function(x) - y )\*x[j]

return rate \* sum / row\_train

def function(x):

sum\_median = 0

for i in range(col\_theta):

sum\_median = sum\_median + theta\_array[i]\*x[i]

return sum\_median

**2.**The code used to train data in regression experiment is below:

loss\_train\_tuple = []

loss\_test\_tupe = []

for i in range(5000):

descent\_array = compute\_descent()

theta\_array = theta\_array - rate \* descent\_array

loss\_train\_tuple.append(train\_loss())

loss\_test\_tupe.append(test\_loss())

def compute\_descent():

return theta\_array + C \* compute\_sum()

def compute\_sum():

sum = np.zeros(15)

for x,y in zip(x\_train\_array,y\_train):

if 1 - y \* compute\_function(x) >= 0 :

sum = sum - y \* x

return sum

def compute\_function(x):

sum = 0

for i in range(col\_theta):

sum = sum + theta\_array[i] \* x[i]

return sum

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

In the both experiments, we hold out 20% data as testing data, and the other 80% are used as training data.

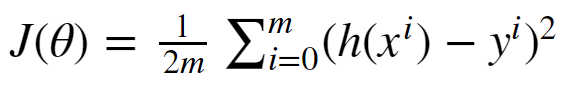
**9. The initialization method of model parameters:**

In the regression experiment, we initiate all theta vecor by setting all it to zero, and we set learning rate to 0.001.

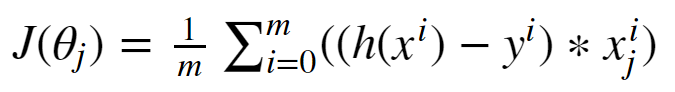
In the classification experiment, we initiate all theta vecor by setting all it to zero, and we set learning rate to 0.001 and set C to 0.1.

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

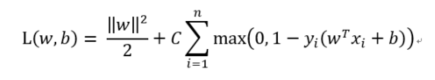
**1.**In the regression experiment, the loss function we used is below:

**.**

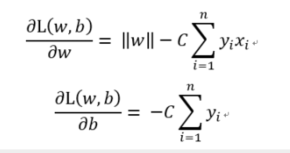
And its derivatives is as below:



1. In the classification experiment, the loss function we used is below:



And its derivatives is as below:



**11. Experimental results and curve:**

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

In the regression experiment we set the learing rate to 0.001.

In the classification experiment, we set learning rate to 0.001 and set C to 0.1.

## Assessment Results (based on selected validation):

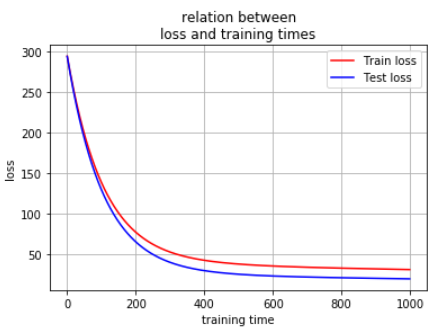
In the both experiment, we evaluate the loss of the training data and testing data as our assessment results.

## Predicted Results (Best Results):

In both experiments, the loss are predicted to descend as the training times is ascending at the beginning , and go smoothly later.

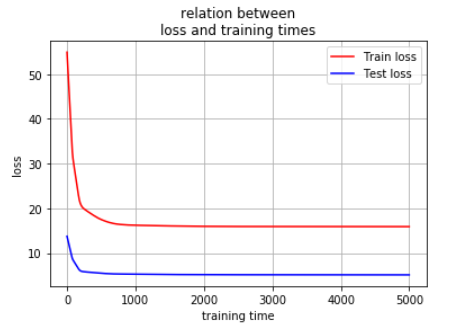
## Loss curve:

1. In the regression experiment, the curve is below:



Figure

2. In the classification experiment, the curve is below:



Figure

**12. Results analysis:**

1. In the regression experiment , as the training time ascend, the cruve of training loss and testing loss both descend, and will go to different convergence at the end.And the two cruve is very close.

2. In the classification experiment , as the training time ascend, the cruve of training loss and testing loss both descend, and will go to different convergence at the end.And the two cruve is not close because of the difference of the number of the data set..

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

The Similarities:

1.both of them can use gradient descent to finish the pridicton. 2.the process of both are almost the same.

The differences:

1.linear regression is to be used to solved the regression problem, and linear classification is to be used to solved the classification problem.

2.linear classification is need to select a threshold, but the other needn’t.

**14. Summary:**

As the training times ascending, the predicted results is more and more closed to the true result.