

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **李寿斌**

**Student ID 201530612040**

**E-mail 1366381738@qq.com**

**Tutor**   **谭明奎**

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

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

**2. Time: 2017.12.2**

**3. Reporter:李寿斌**

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

Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features. Divided it into training set, validation set.

Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features. Divided it into training set, validation set.

1. **Experimental steps:**

*Linear Regression and Gradient Descent*

1. Load the experiment data.
2. Decide dataset into training set and validation set.
3. Initialize linear model parameters randomly.
4. Choose loss function and derivation.
5. Calculate gradient G toward loss function from all samples.
6. Denote the opposite direction of gradient *G* as *D*.
7. Update model: *Wt = Wt-1 +η\*D*.
8. Get the loss *Ltrain* under the training set and *Lvalidation* by validating under validation set.
9. Repeat step 5 to 8 for several times.
10. Drawing graph of *Ltrain* as well as *Lvalidation*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 randomly.
4. Choose loss function and derivation.
5. Calculate gradient *G* toward loss function from all samples.
6. Denote the opposite direction of gradient *G* as *D*.
7. Update model: *Wt = Wt-1 +η\*D*.
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 *Ltrain* under the training set and *Lvalidation* by validating under validation set.
9. Repeat step 5 to 8 for several times.
10. Drawing graph of *Ltrain* as well as *Lvalidation*with the number of iterations.

**7. Code:**

Linear Regression***:***

def gradientDescent(X\_train, Y\_train,x,y, theta, alpha, num\_iters):

m = Y.shape[0]

n = y.shape[0]

#store the history loss

L\_history = np.zeros((num\_iters, 1))

L\_historyV = np.zeros((num\_iters, 1))

for iter in range(num\_iters):

theta = theta - (alpha/m) \* (X\_train.T.dot(X\_train.dot(theta) - Y\_train)) #update

L\_history[iter] = computeLoss(X\_train, Y\_train, theta)

L\_historyV[iter] = computeLoss(x, y, theta)

return L\_history,L\_historyV,theta

Linear Classification：

G\_w = np.zeros([x\_train.shape[1],1])

G\_b = 0

x.append(n+1) #Update x-axis

#Step5:Calculate gradient G toward loss function from all samples.

for n in range(0,m\_t):

result = 1 - y\_train[n]\*(w.T\*x\_train[n].T+b)

if(result>=0):

G\_w = G\_w + (w-C\*x\_train[n].T\*y\_train[n])

G\_b = G\_b + (-C\*y\_train[n])

elif(result<0):

G\_w = G\_w + w

G\_b = G\_b

#Step6,7:Update model parameters.

w = w - learn\_rate\*G\_w

b = b - learn\_rate\*G\_b

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

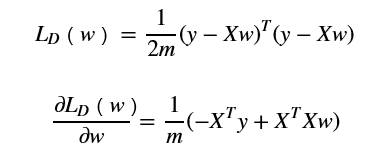
Linear Regression and Linear Classification: hold-out

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

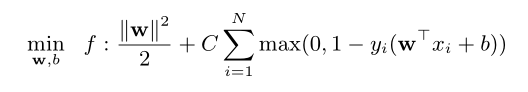
Linear Regression and Linear Classification: randomly

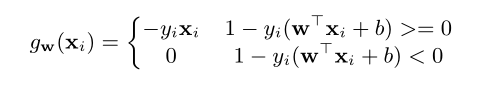
**10. The selected loss function and its derivatives:**

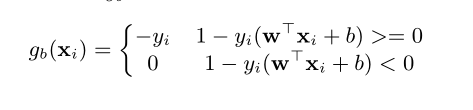
Linear Regression:

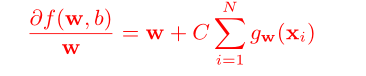


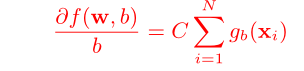
Linear Classification:











**11. Experimental results and curve:**

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

## Assessment Results (based on selected validation):

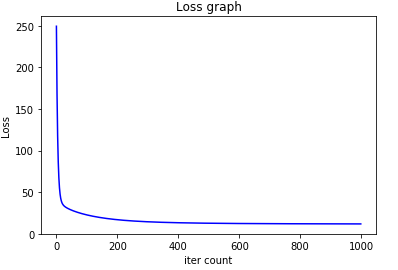
## Predicted Results (Best Results):

## Loss curve:

Linear Regression：

η = 0.03

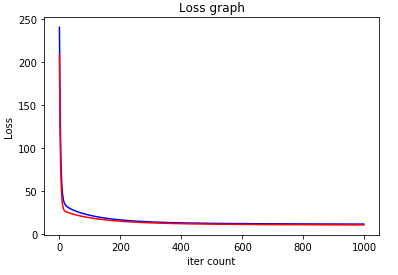
Assessment Results：



Predicted Results



Loss curve

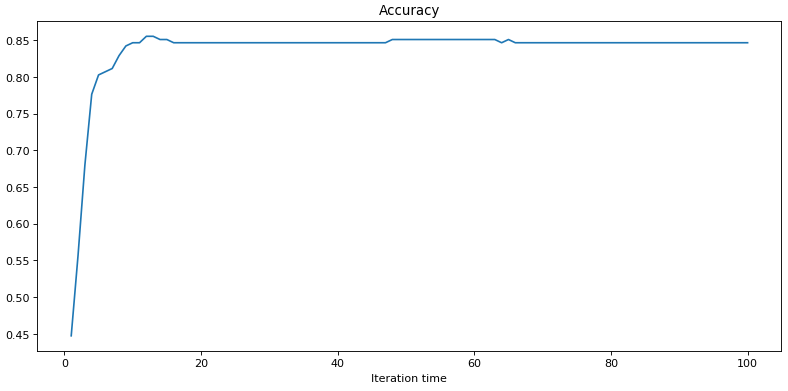


Linear Classification：

C = 10000

η = 0.01/C= 0.000000001

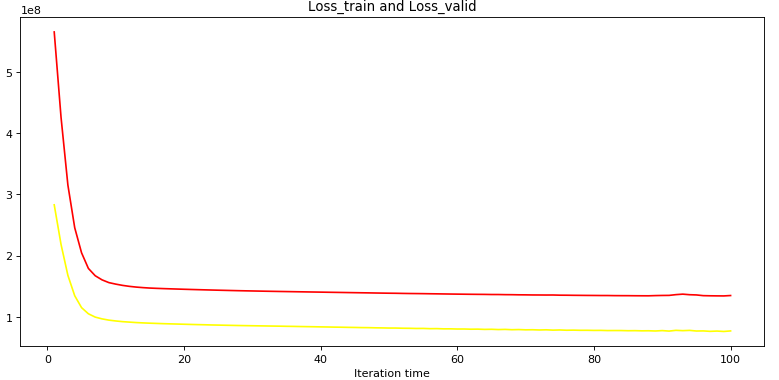
Assessment Results



## Predicted Results:



Loss curve



**12. Results analysis:**

*Linear Regression:*

Whenη = 0.03, after about 700 times iteration, the Linear Regression and Gradient Descent method can get a model whose loss of validation set is lowest. And the loss of validation set decreases steadily.

*Linear Classification:*

When C =1000000 and η = 0.001/C, during the first 20 times iteration ,the Linear Classification and Gradient Descent method can get a best accuracy of validation set which is up to 0.85 , later iteration the accuracy of validation shaking around 0.84 .

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

*Similarities:*

1).Both of them can predict the y according to features.use features to predict Y

2).The structure of parameters is similar.

3). use gradient descent method to optimize the parameters.

*Difference:*

1).The label of linear regression is continuous, but the label of linear classification is discrete.

2).Linear classification uses surrogate loss function to replace *l0/1(z)*function when optimize loss function. But linear regression can optimize loss function directly.

3).Linear regression can be assessed by mean square error. Linear classification can be assessed by accuracy.

**14. Summary:**

Linear regression and linear classification are the basic models of machine learning. In this experiment, we use gradient descent method to optimize their model parameters.