

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **Wu Xianzhe**

**Student ID 201530613078**

**E-mail 494669733@qq.com**

**Tutor**   **Tan Mingkui**

**Date submitted** **2017. 12 . 6**

**1. Topic:**Linear regression , classification and gradient descent

**2. Time:** 2017.12.2-2017.12.8

**3. Reporter:**Wu Xianzhe

**4. Purposes:**

1) Understand more about the principles of linear regression and gradient descent

2) Practice on data sets of small scale

3) Experience the process of optimization and adjusting parameters

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

In linear regression , we use scale version of data set “housing” , which contains 506 samples . There are 13 attributes in each sample .

In linear classification , we use scale version of data set “australian” , which contains 690 samples . There are 14 attributes in each sample .

1. **Experimental steps:**

Linear regression and gradient descent :

1. Use load\_svmlight\_file function to read the data
2. Use train\_test\_split function to divide the data set into training set and validation set
3. Initialize the linear model parameters . (I chose to initialize all parameters as 0 )
4. Select loss function and compute its derivation .
5. Compute the gradient G of all samples to the loss function
6. Set the learning rate n and update the model parameter Wt=Wt-1-nG
7. Test the model on training set and record the value of loss function as Ltrain; Test the model on validation set and record the value of loss function as Lvalidation;
8. Repeat step 4 to 7 for a couple of times and draw the graph to show how Ltrain and Lvalidation change with iteration time .

Linear classification and gradient descent :

1. Use load\_svmlight\_file function to read the data
2. Use train\_test\_split function to divide the data set into training set and validation set
3. Initialize the SVM parameters . (I chose to initialize all parameters as 0 )
4. Select loss function and compute its derivation .
5. Compute the gradient G of all samples to the loss function
6. Set the learning rate n and update the model parameter Wt=Wt-1-nG
7. Select proper value as threshold . Mark the results bigger than threshold as positive class and mark those smaller than threshold as negative class . Test the model on training set and record the value of loss function as Ltrain; Test the model on validation set and record the value of loss function as Lvalidation;
8. Repeat step 4 to 7 for a couple of times and draw the graph to show how Ltrain and Lvalidation change with iteration time .
9. **Code:**

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

Key code of linear regression :

#初始化参数 选择全零初始化

W=np.zeros((col+1,1))

#确定学习率和训练次数

n=0.00001

count=0

max\_count=1000

#用数组记录loss

loss\_train=np.zeros(max\_count)

loss\_validation=np.zeros(max\_count)

while(count<max\_count):

G=-X\_train.T.dot(y\_train)+X\_train.T.dot(X\_train).dot(W)

loss\_train[count]=(y\_train-X\_train.dot(W)).T.dot(y\_train-X\_train.dot(W))

loss\_validation[count]=(y\_test-X\_test.dot(W)).T.dot(y\_test-X\_test.dot(W))

W=W-n\*G

count+=1

Key code of linear classification:

#初始化参数 选择全零初始化

W=np.zeros((col+1,1))

#确定学习率和训练次数

n=0.0001

count=0

max\_count=100

#用数组记录loss

loss\_train=np.zeros(max\_count)

loss\_validation=np.zeros(max\_count)

while(count<max\_count):

i=0

j=0

G\_hinge=np.zeros((col+1,1))

loss\_hinge\_train=0

loss\_hinge\_validation=0

while(i<row\_train):

if(y\_train[i].dot(W.T.dot(X\_train[i]))<1):

y\_train\_cell=y\_train[i].reshape(1,1)

X\_train\_cell=X\_train[i].reshape(15,1)

G\_hinge+=-X\_train\_cell.dot(y\_train\_cell)

loss\_hinge\_train+=1-y\_train[i].dot(W.T.dot(X\_train[i]))

i+=1

else:

i+=1

while(j<row\_test):

if(y\_test[j].dot(W.T.dot(X\_test[j]))<1):

loss\_hinge\_validation+=1-y\_test[j].dot(W.T.dot(X\_test[j]))

j+=1

G=W+G\_hinge

loss\_train[count]=loss\_hinge\_train

loss\_validation[count]=loss\_hinge\_validation

W=W-n\*G

count+=1

count=0

while(count<max\_count):

loss\_train[count]=loss\_train[count]/row\_train

loss\_validation[count]=loss\_validation[count]/row\_test

count+=1

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

Regression:hold-out

Classification:hold-out

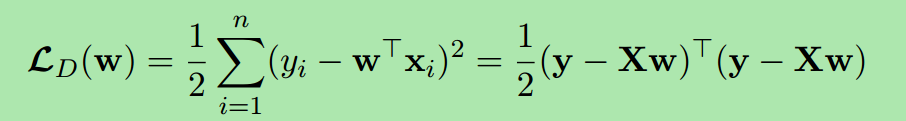
1. **The initialization method of model parameters:**

I initialize all parameters as zero .

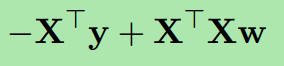
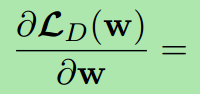
1. **The selected loss function and its derivatives:**

Linear regression:

Loss function:

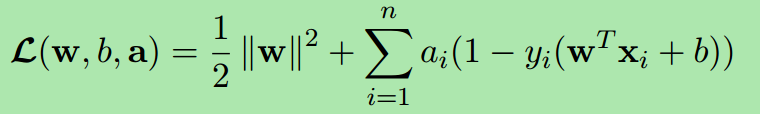


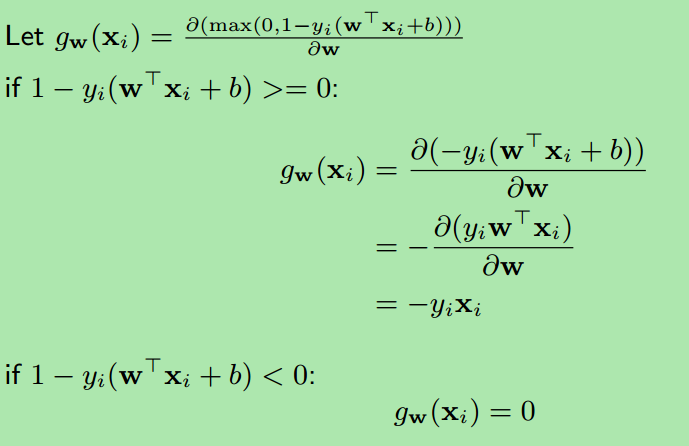
Derivative:

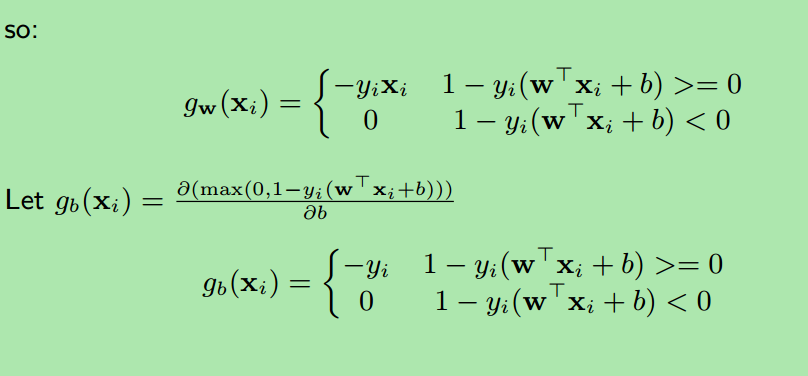


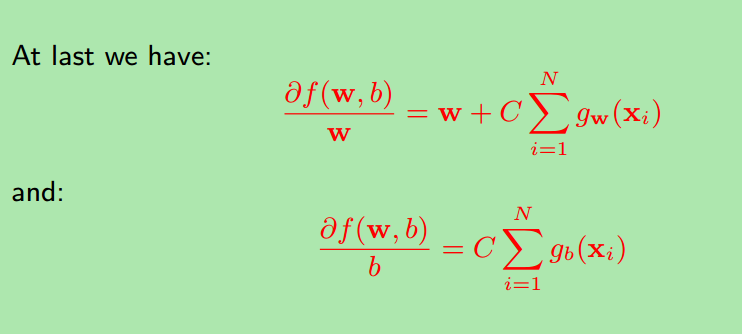
Linear classification:

Loss function:



Derivative



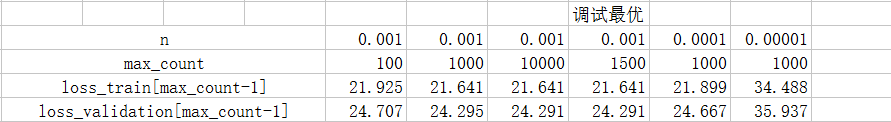


1. **Experimental results and curve:**

Linear regression:

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

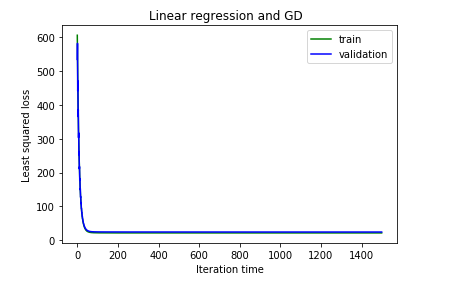
## Assessment Results (based on selected validation):



## Predicted Results (Best Results):

When n equals to 0.001 and max\_count equals to 1500 .

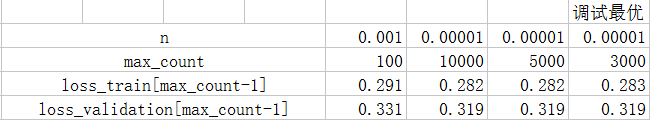
## Loss curve:



Linear classification:

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

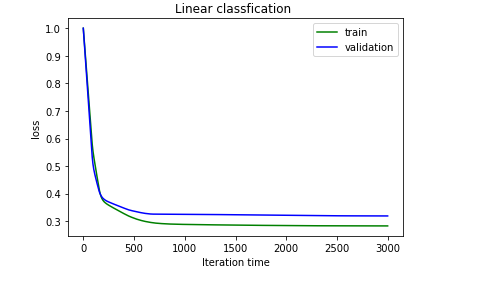
## Assessment Results (based on selected validation):



## Predicted Results (Best Results):

When n equals to 0.00001 and max\_count equals to 3000

## Loss curve:



1. **Results analysis:**

In linear regression , we set n as 0.001 and train the model on training set for 1500 times so that we can make the loss converge . and if n is bigger than 0.005 , we will be failed to converge the results .

And in linear classification , we set n as 0.00001 because we won’t get convergent curve when we use any value which is bigger than 0.00001 . Than 3000 times of training is enough .

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

For data sets , the inputs are similar in two cases while the outputs differ a lot . In linear regression , one output refers to one input generally . But in linear classification , one output(+1 or -1) may refers to a group of inputs .

For operation , linear regression finds out a linear model to try to fit and predict each input&output pair . And generally linear classification needs us to get a proper hyperplane so that we can divide all the inputs into two classes .

**14. Summary:**

Well , um , to be honest , before conducting this experiment , I knew nothing about python . So this is also the first time that I use this language to achieve something .

The experiment taught me more details about linear regression , linear classification and gradient descent . Actually , I just understand the general concept about regression and classification . And some skills like SVM , SGD are ambiguous to me . This time I set the parameters and adjusted them and established the model on my own .

All of these made me understand more about the knowledge .