

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **王跃**

**Student ID 201530612941**

**E-mail 602056964@qq.com**

**Tutor**   **谭明奎**

**Date submitted** **2017. 12 .04**

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

**2. Time:2017.12.02**

**3. Reporter:王跃**

**4. Purposes:**

1. Further understand of linear regression and gradient descent.
2. Conduct some experiments under small scale data set.
3. Realize the process of optimization and adjusting parameters.
4. **Data sets and data analysis:**

Linear Regression uses [Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html" \l "housing" \t "https://www.zybuluo.com/chenyaofo/note/_blank) in [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "https://www.zybuluo.com/chenyaofo/note/_blank), including 506 samples and each sample has 13 features. You are expected to download scaled edition.

Linear classification uses [australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html" \l "australian" \t "https://www.zybuluo.com/chenyaofo/note/_blank) in [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "https://www.zybuluo.com/chenyaofo/note/_blank), including 690 samples and each sample has 14 features. You are expected to download scaled edition.

1. **Experimental steps:**

The step of 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\_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: 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:  W= W+η\*D,η is learning rate, a hyper-parameter that we can adjust.
8. Get the loss L\_train under the training set and  L\_validation by validating under validation set.
9. Repeated step 5 to 8 for several times, and drawing graph of  L\_train as well as  L\_validation with the number of iterations.

The step of 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:  W= W+η\*D,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  L\_train under the train set and  L\_validation by validating under validation set.
9. Repeated step 5 to 8 for several times, and drawing graph of  L\_train as well as  L\_validatioin with the number of iterations.
10. **Code:**

The code of Linear regression and gradient Descent：

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_svmlight\_file

from sklearn.model\_selection import train\_test\_split

dir = "./jiqixuexi/"

def get\_data():

data = load\_svmlight\_file(dir + "housing\_scale.txt")

return data[0], data[1]

# get\_loss

def get\_loss(lda,w,x,y):

return (1/x.shape[0])\*((lda/2)\*np.dot(w.T,w)+0.5\*np.dot((y-(np.dot(x,w)).T),(y-np.dot(x,w))))

X, y = get\_data()

# huafengdhujuji

w = np.zeros((14,))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

a = np.ones((X\_train.shape[0],1))

b = np.ones((X\_test.shape[0],1))

X\_train = np.column\_stack((X\_train.toarray(),a))

X\_test = np.column\_stack((X\_test.toarray(),b))

loss =[]

loss\_test= []

i=1000

while i>0:

grad =( 1/(X\_train.shape[0]))\*(0.05\*w + np.dot(((X\_train).T),(np.dot(X\_train,w)-y\_train)))

w = w-0.01\*grad

loss.append(get\_loss(0.05,w,X\_train,y\_train))

loss\_test.append(get\_loss(0.05,w,X\_test,y\_test))

i=i-1

x = []

for i in range(1000):

x.append(i+1)

plt.figure()

l1,= plt.plot(x, loss)

l2,=plt.plot(x, loss\_test, color='red', linewidth=1.0, linestyle='--')

plt.xlabel('times')

plt.ylabel('loss')

plt.legend(handles=[l1, l2,], labels=['up', 'down'], loc='best')

plt.show()

The code of Linear Classification and Gradient Descent：

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_svmlight\_file

from sklearn.model\_selection import train\_test\_split

dir = "./jiqixuexi/"

# read file

def get\_data():

data = load\_svmlight\_file(dir + "australian\_scale.txt")

return data[0], data[1]

# gradient

def gradient(w,x,y):

grad = np.zeros((15,))

for i in range(x.shape[0]):

if (1-y[i]\*np.dot(w.T,x[i].T))>0 :

grad += w - y[i]\*(x[i].T)

if (1-y[i]\*np.dot(w.T,x[i].T))<=0 :

grad += w

shample = x.shape[0]

grad = (1/shample)\*grad

return grad

# get\_loss

def get\_loss(w,x,y):

los = 0

for i in range(x.shape[0]):

if (1-y[i]\*np.dot(w.T,x[i].T))>0 :

los += 1-y[i]\*np.dot((w.T),(x[i].T))

if (1-y[i]\*np.dot(w.T,x[i].T))<=0 :

los += 0

return (0.5\*np.dot((w.T),w)+los)/x.shape[0]

X,y = get\_data()

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

a = np.ones((X\_train.shape[0],1))

b = np.ones((X\_test.shape[0],1))

X\_train = np.column\_stack((X\_train.toarray(),a))

X\_test = np.column\_stack((X\_test.toarray(),b))

loss =[]

loss\_test= []

grad= np.zeros((15,))

w = np.zeros((15,))

time = 1000

while time>0:

grad = gradient(w,X\_train,y\_train)

w = w - 0.05\*grad

loss.append(get\_loss(w,X\_train,y\_train))

loss\_test.append(get\_loss(w,X\_test,y\_test))

time = time-1;

x = []

for i in range(1000):

x.append(i+1)

plt.figure()

l1,= plt.plot(x, loss)

l2,=plt.plot(x, loss\_test, color='red', linewidth=1.0, linestyle='--')

plt.xlabel('times')

plt.ylabel('loss')

plt.legend(handles=[l1, l2], labels=['up', 'down'], loc='best')

plt.show()

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

Linear regression: Outflow method

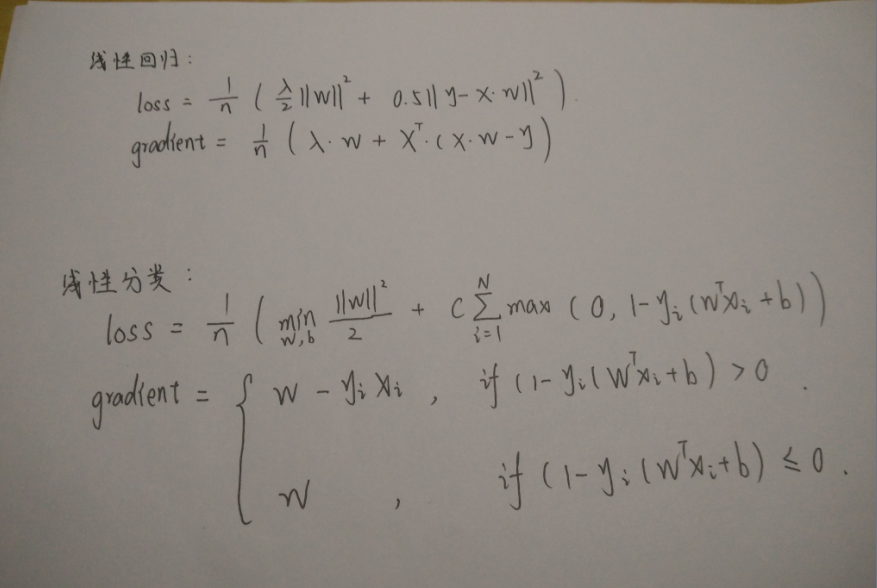
Linear classification: Outflow method

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

Linear regression: all zero initialization

Linear Category: All zero initialization

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



1. **Experimental results and curve:**

Linear regression:

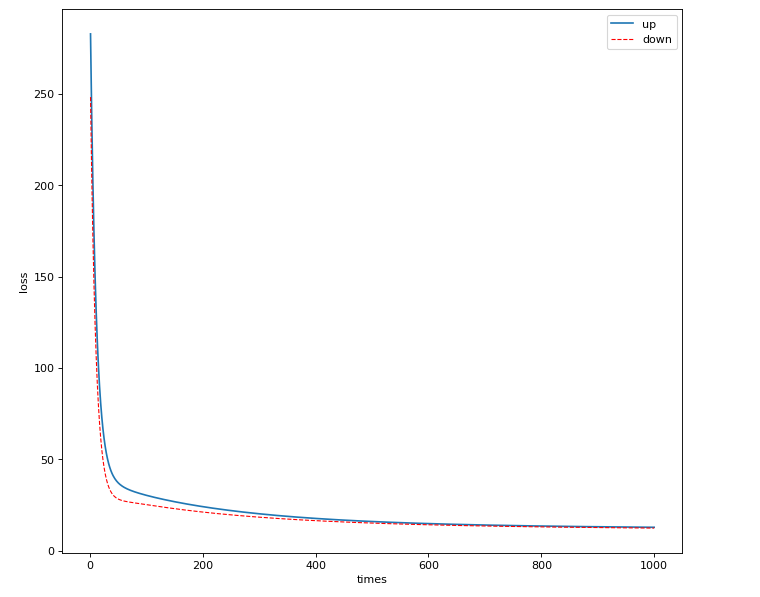
Hyper-parameter selection (η, epoch, etc.): η=0.01

Lambda=0.05

## Assessment Results (based on selected validation):Choose the appropriate loss function to evaluate the result.

## Predicted Results (Best Results): The forecast results are shown graphically.

## Loss curve:



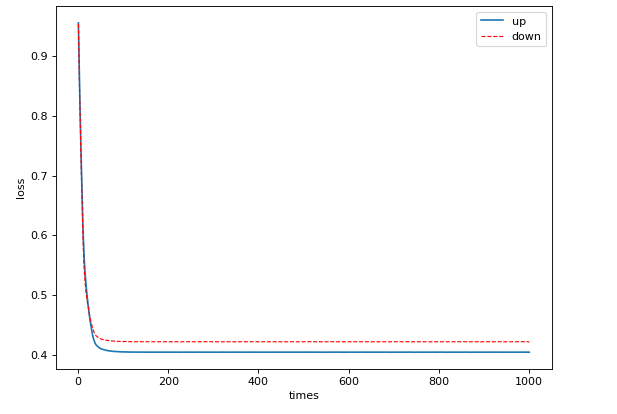
Linear classification

## Hyper-parameter selection (η, epoch, etc.):η=0.05

## Assessment Results (based on selected validation):Choose the appropriate loss function to evaluate the result.

## Predicted Results (Best Results):The forecast results are shown graphically.

## Loss curve:



1. **Results analysis:**

Linear regression:

Through the adjustment of parameters can get a better regression results.

Linear classification:

Through the adjustment of multiple hyper-parameters, a better SVM model can be trained on the training set.

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

In the classification problem, the output not only allows two values, but also multiple values, which are discrete. In the regression problem, the output can take any real number and be continuous. The difference between classification and regression is the type of output variable. Quantitative output is called regression, or continuous variable prediction; qualitative output is called classification, or discrete variable prediction.

1. **Summary:**

Through the study of this experiment, further understand the nature and realization of linear regression and linear classification. Through the learning of gradient descent method, we further understand the important content of gradient learning.