South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Shaojia hong

Supervisor:
Qingyao Wu

Student ID：
201720144993

Grade:
Graduate

December 14, 2017

Logistic Regression, Linear Classification and Gradient Descent

Abstract—

I. Introduction

Logistic regression is a linear classification model. The difference between linear regression and linear regression is that in order to output large numbers of linear regression, for example, from negative infinity to positive infinity, it is compressed to 0 and 1.only need a logistic function is that $g(z)=\frac{1}{1+e^{-z}}$.

Linear Classification is a Classification that given training data (xi, yi) for i = 1 . . . n, with $x_i \in R^m$ and $y_i \in \{-1, 1\}$, learn a classfier f(x) such that $f(x_i)\begin{cases}\ge 0 & y_i=+1\\ <0 & y_i=-1\end{cases}$ and $y_i f(x_i)>0$ for a correct classification.

In order to further understand of the difference and connection between the gradient descent and the random gradient descent,and the difference and connection between logistic regression and linear classification is compared.Finally further understand the principle of SVM and practice it on larger data.Finally we use the SGD,NAG,RMSProp, AdaDelta, and Adam of gradient methods to gradient descent,and compare the loss of five methods.

II. Methods and Theory

A. Dataset

We use a data set,the experiments uses the data set of a9a in LIBSVM Data, including 32561 / 16281(testing) samples and each sample has 123/123 (testing) features.

B. Experimental environment

Python3 and at least the following Python packages are included such as sklearn，numpy，jupyter，matplotlib.
It is recommended to install anaconda3 directly, which has built in the above Python packages.The experimental code and drawing are all done on jupyter.

C. Steps

The step of Logistic regression and gradient Descent：

1.Read the experimental training set and the validation set.
2.init the parameter of logistic regression model ,the initialization can consider all zero initialization, random initialization or normal distribution initialization.
3.Select the Loss function and seek guidance for it. The process is detailed in the courseware ppt.
4.The gradient of a partial sample to the Loss function G is obtained.
5.Update the model parameters using different optimization methods (NAG, RMSProp, AdaDelta, and Adam).
6.Choosing the appropriate threshold, we will verify that the mark of the concentrated calculation is more than the threshold as a positive class, and otherwise as a negative class. Test and get the Loss function values $L_{NAG}, L_{RMSProp}, L_{AdaDelta}$和$L_{Adam}$ of different optimization methods on the validation set.
7.Repeat step 4-6 several times, draw the graph of $L_{NAG}, L_{RMSProp}, L_{AdaDelta}$和$L_{Adam}$, and change graphs with the number of iterations.

The step of Linear Classification and gradient Descent：

1.Read the experimental training set and the validation set.
2.The support vector machine model parameter initialization can consider all zero initialization, random initialization or normal distribution initialization.
3.Select the Loss function and seek guidance for it. The process is detailed in the courseware ppt.
4.The gradient of a partial sample to the Loss function G is obtained.
5.Update the model parameters using different optimization methods (NAG, RMSProp, AdaDelta, and Adam).
6.Choosing the appropriate threshold, we will verify that the mark of the concentrated calculation is more than the threshold as a positive class, and otherwise as a negative class. Test and get the Loss function values $L_{NAG}, L_{RMSProp}, L_{AdaDelta}$和$L_{Adam}$ of different optimization methods on the validation set.
7.Repeat step 4-6 several times, draw the graph of $L_{NAG}, L_{RMSProp}, L_{AdaDelta}$和$L_{Adam}$, and change graphs with the number of iterations.

III. Experiment

A. Formula

1) Logistic regression formula

Logistic function is

$$h_w(x_i) = \frac{1}{1+e^{-wx}}$$

Target function is

$$\mathrm{w} = w - \frac{1}{n}\sum_{i=1}^{n} \alpha(h_w(x_i) - y_i)x_i$$

Loss function is

$$\mathrm{J}(w) = \frac{1}{n}[\sum_{i=1}^{n} y_i \log h_w(x_i) + (1-y_i)\log(1-h_w(x_i))]$$

SGD function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1})$$

$$\theta_t \leftarrow \theta_{t-1} - \eta g_t$$

NAG function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1} - \gamma v_{\mathrm{t-1}})$$

$$v_t \leftarrow \gamma v_{\mathrm{t-1}} + \eta g_t$$

$$\theta_t \leftarrow \theta_{t-1} - v_t$$

RMSProp function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1})$$

$$\mathrm{G}_t \leftarrow \gamma \mathrm{G}_{\mathrm{t-1}} + (1-\gamma) g_t \Theta g_t$$

$$\theta_t \leftarrow \theta_{t-1} - \frac{\eta}{\sqrt{G_t + \varepsilon}} \Theta g_t$$

AdaDelta function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1})$$

$$\mathrm{G}_t \leftarrow \gamma \mathrm{G}_{\mathrm{t}} + (1-\gamma) g_t \Theta g_t$$

$$\Delta\theta_t \leftarrow -\frac{\sqrt{\Delta_{t-1}+\varepsilon}}{\sqrt{G_t+\varepsilon}} \Theta g_t$$

$$\theta_t \leftarrow \theta_{t-1} - \Delta\theta_t$$

$$\Delta_t \leftarrow \gamma\Delta_{t-1} + (1-\gamma)\Delta\theta_t \Theta \Delta\theta_t$$

Adam function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1})$$

$$\mathrm{m}_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$\mathrm{G}_t \leftarrow \gamma \mathrm{G}_{\mathrm{t}} + (1-\gamma) g_t \Theta g_t$$

$$\alpha \leftarrow \eta \frac{\sqrt{1-\gamma^t}}{\sqrt{1-\beta^t}}$$

$$\theta_t \leftarrow \theta_{t-1} - \alpha \frac{m_t}{\sqrt{G_t + \varepsilon}}$$

2) Linear Classification formula

Target function is

$$\mathrm{g}_w(x_i) = \begin{cases} -y_i x_i & 1-y_i(w^T x_i + b) >= 0 \\ 0 & 1-y_i(w^T x_i + b) < 0 \end{cases}$$

$$\mathrm{g}_{\mathrm{b}}(x_i) = \begin{cases} -y_i & 1-y_i(w^T x_i + b) >= 0 \\ 0 & 1-y_i(w^T x_i + b) < 0 \end{cases}$$

$$\Delta_{\mathrm{w}} L(w, \mathrm{b}) = w + \frac{C}{n}\sum_{i=1}^{n} g_w(x_i)$$

Loss function is

$$\Delta_{\mathrm{b}} L(w, \mathrm{b}) = w + \frac{C}{n}\sum_{i=1}^{n} g_b(x_i)$$

SGD function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1})$$

$$\theta_t \leftarrow \theta_{t-1} - \eta g_t$$

NAG function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1} - \gamma v_{\mathrm{t-1}})$$

$$v_t \leftarrow \gamma v_{\mathrm{t-1}} + \eta g_t$$

$$\theta_t \leftarrow \theta_{t-1} - v_t$$

RMSProp function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1})$$

$$\mathrm{G}_t \leftarrow \gamma \mathrm{G}_{\mathrm{t-1}} + (1-\gamma) g_t \Theta g_t$$

$$\theta_t \leftarrow \theta_{t-1} - \frac{\eta}{\sqrt{G_t + \varepsilon}} \Theta g_t$$

AdaDelta function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1})$$

$$\mathrm{G}_t \leftarrow \gamma \mathrm{G}_{\mathrm{t}} + (1-\gamma) g_t \Theta g_t$$

$$\Delta\theta_t \leftarrow -\frac{\sqrt{\Delta_{t-1}+\varepsilon}}{\sqrt{G_t+\varepsilon}} \Theta g_t$$

$$\theta_t \leftarrow \theta_{t-1} - \Delta\theta_t$$

$$\Delta_t \leftarrow \gamma\Delta_{t-1} + (1-\gamma)\Delta\theta_t \Theta \Delta\theta_t$$

Adam function is

$$\mathrm{g}_t \leftarrow \nabla J_i(\theta_{t-1})$$

$$\mathrm{m}_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$\mathrm{G}_t \leftarrow \gamma \mathrm{G}_{\mathrm{t}} + (1-\gamma) g_t \Theta g_t$$

$$\alpha \leftarrow \eta \frac{\sqrt{1-\gamma^t}}{\sqrt{1-\beta^t}}$$

$$\theta_t \leftarrow \theta_{t-1} - \alpha \frac{m_t}{\sqrt{G_t + \varepsilon}}$$

B. Experimental results

1. Logistic regression result is the following diagram and the linear regression parameter is such

SGD method parameter:

η =0.005

NAG method parameter:

$\gamma = 0.9$

η =0.005

RMSProp method parameter:

$\gamma = 0.9$

$\varepsilon = 10^{-6}$

η =0.001

AdaDelta method parameter:

$\gamma = 0.95$

$\varepsilon = 10^{-6}$

Adam method parameter:

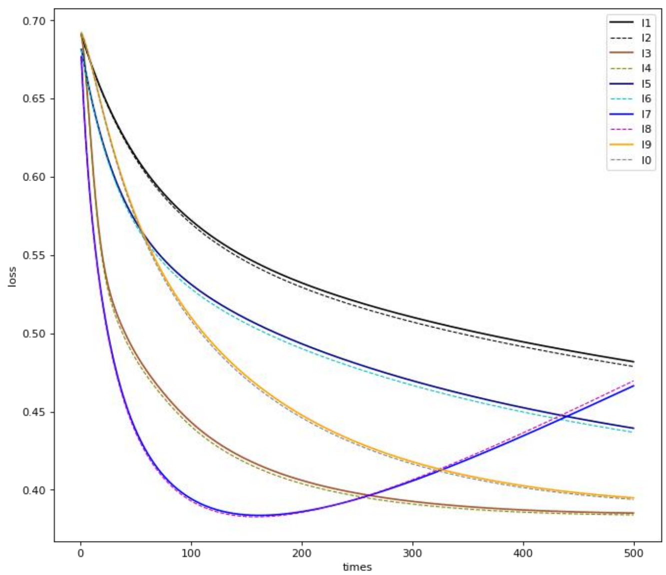
β =0.9

$\gamma = 0.999$

$\varepsilon = 10^{-6}$

η =0.001

l1,l2 is SGD method loss and test loss line
l3,l4 is NAG method loss and test loss line
l5,l6 is RMSProp method loss and test loss line
l7,l8 is AdaDelta method loss and test loss line
l9,l0 is Adam method loss and test loss line



2. Linear Classification result is the following diagram and the linear regression parameter is such

SGD method parameter:

η =0.001

NAG method parameter:

$\gamma = 0.9$

η =0.001

RMSProp method parameter:

$\gamma = 0.9$

$\varepsilon = 10^{-8}$

η =0.001

AdaDelta method parameter:

$\gamma = 0.9$

$\varepsilon = 10^{-6}$

Adam method parameter:

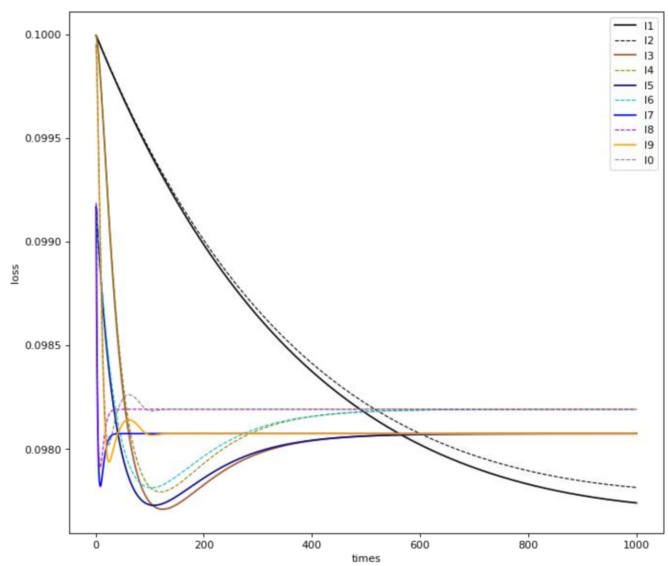
β =0.9

$\gamma = 0.999$

$\varepsilon = 10^{-6}$

η =0.001

l1,l2 is SGD method loss and test loss line
l3,l4 is NAG method loss and test loss line
l5,l6 is RMSProp method loss and test loss line
l7,l8 is AdaDelta method loss and test loss line
l9,l0 is Adam method loss and test loss line



C. code

1) The code of Logistic regression and gradient Descent:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import     load_svmlight_file
import random
import math
```

```
# get_testdata
def get_testdata():
   data = load_svmlight_file("F:\\a9a.t")
   for i in range( data[1].shape[0]):
      if   data[1][i] ==(-1):
          data[1][i]=0
   return data[0], data[1]
# get_traindata
def get_traindata():
   data = load_svmlight_file("F:\\a9a")
   for i in range( data[1].shape[0]):
      if   data[1][i] ==(-1):
          data[1][i]=0
   return data[0], data[1]


def get_loss(w,x,y):
   loss = 0
   for i in range(x.shape[0]):
      h = 1/(1+math.exp((-1)*np.dot((w.T),(x[i].T))))
      l =-1*(y[i]*math.log(h)+(1-y[i])*math.log(1-h))
      loss += l
   return loss/(x.shape[0])

def gradient(w,x,y,index):
   gradient=np.zeros((124,))
   for i in index:
      h = 1/(1+math.exp((-1)*np.dot((w.T),(x[i].T))))
      gradient +=(h-y[i])*(x[i].T)
   return gradient*(1/100)



# read_files
X_train,y_train = get_traindata()
X_test,y_test = get_testdata()
a = np.ones((X_train.shape[0],1))
b = np.zeros((X_test.shape[0],1))
c = 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))
X_test = np.column_stack((X_test,c))
iteration=500


w1= np.zeros((124,))
learn_rate1=0.005
loss_SGD = []
loss_SGD_test = []
def SGD(w,x,y,x_t,y_t,index):
   g=gradient(w,x,y,index)
   w=w-learn_rate1*g
   loss_SGD.append(get_loss(w,x,y))
   loss_SGD_test.append(get_loss(w,x_t,y_t))
   return w
w2= np.zeros((124,))
v2=np.zeros((124,))
gama2=0.9
learn_rate2=0.005
loss_NAG = []
loss_NAG_test = []
def NAG(w,v,x,y,x_t,y_t,index):
    g=gradient(w-gama2*v,x,y,index)
    v=gama2*v+learn_rate2*g
    w=w-v
    loss_NAG.append(get_loss(w,x,y))
    loss_NAG_test.append(get_loss(w,x_t,y_t))
    return w,v


w3=np.zeros((124,))
G3=np.zeros((124,))
gama3=0.9
e=10**(-6)
learn_rate3=0.001
loss_RMSProp = []
loss_RMSProp_test = []
def RMSProp(w,G,x,y,x_t,y_t,index):
   g=gradient(w,x,y,index)
   G=G*gama3+(1-gama3)*g*g
   G_temp=G
   for j in range(G.shape[0]):
      G_temp[j]=learn_rate3/math.sqrt(G_temp[j]+e)
   w=w-G_temp*g
   loss_RMSProp.append(get_loss(w,x,y))
   loss_RMSProp_test.append(get_loss(w,x_t,y_t))
   return w,G

w4=np.zeros((124,))
t4=np.zeros((124,))
G4=np.zeros((124,))
gama4=0.95
e=10**(-6)
loss_AdaDelta = []
loss_AdaDelta_test = []
def AdaDelta(w,G,t,x,y,x_t,y_t,index):
   g=gradient(w,x,y,index)
   G=gama4*G+(1-gama4)*np.square(g)
   G_temp=G+e
   t_temp=t+e
   for j in range(G.shape[0]):

G_temp[j]=-((math.sqrt(t_temp[j]))/(math.sqrt(G_temp[j])))
   w_temp=G_temp*g
   w=w+w_temp
   t=gama4*t+(1-gama4)*np.square(w_temp)
   loss_AdaDelta.append(get_loss(w,x,y))
   loss_AdaDelta_test.append(get_loss(w,x_t,y_t))
   return w,G,t

w5=np.zeros((124,))
```

```
m5=np.zeros((124,))
a5=np.zeros((124,))
G5=np.zeros((124,))
peta5=0.9
gama5=0.999
e=10**(-6)
learn_rate5=0.001
loss_Adam = []
loss_Adam_test = []
def Adam(w,G,m,t,x,y,x_t,y_t,index):
    g=gradient(w,x,y,index)
    m=peta5*m+(1-peta5)*g
    G=gama5*G+(1-gama5)*g*g

alpa=learn_rate5*((math.sqrt(1-math.pow(gama5,t)))/(math.sq
rt(1-math.pow(peta5,t))))
    G_temp=G+e
    for j in range(G.shape[0]):
        G_temp[j]=m[j]/(math.sqrt(G_temp[j]))
    w=w-alpa*G_temp
    loss_Adam.append(get_loss(w,x,y))
    loss_Adam_test.append(get_loss(w,x_t,y_t))
    return w,G,m


index= random.sample(range(X_train.shape[0]),100)
for i in range(iteration):
    w1=SGD(w1,X_train,y_train,X_test,y_test,index)
    w2,v2=NAG(w2,v2,X_train,y_train,X_test,y_test,index)

w3,G3=RMSProp(w3,G3,X_train,y_train,X_test,y_test,index)

w4,G4,t4=AdaDelta(w4,G4,t4,X_train,y_train,X_test,y_test,in
dex)

w5,G5,m5=Adam(w5,G5,m5,i+1,X_train,y_train,X_test,y_tes
t,index)
    print(i)


x = []
for i in range(iteration):
    x.append(i+1)
plt.figure(figsize=(10,9), dpi=80)
l1,= plt.plot(x, loss_SGD,color='black')
l2,=plt.plot(x, loss_SGD_test, color='black', linewidth=1.0,
linestyle='--')
l3,=plt.plot(x, loss_NAG, color='sienna')
l4,=plt.plot(x, loss_NAG_test, color='olive', linewidth=1.0,
linestyle='--')
l5,=plt.plot(x, loss_RMSProp, color='navy')
l6,=plt.plot(x, loss_RMSProp_test, color='c', linewidth=1.0,
linestyle='--')
l7,=plt.plot(x, loss_AdaDelta, color='blue')
l8,=plt.plot(x, loss_AdaDelta_test, color='m', linewidth=1.0,
linestyle='--')
l9,= plt.plot(x, loss_Adam,color='orange')
l0,=plt.plot(x, loss_Adam_test, color='gray', linewidth=1.0,
linestyle='--')
plt.xlabel('times')
plt.ylabel('loss')
plt.legend(handles=[l1, l2,l3,l4,l5,l6,l7,l8,l9,l0,], labels=['l1',
'l2','l3', 'l4','l5', 'l6','l7', 'l8','l9', 'l0'], loc='best')
plt.show()
```

2) The code of Linear Classification and gradient Descent:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_svmlight_file
import random
import math



# get_testdata
def get_testdata():
    data = load_svmlight_file("F:\\a9a.t")
    for i in range( data[1].shape[0]):
        if  data[1][i] ==(-1):
            data[1][i]=0
    return data[0], data[1]
# get_traindata
def get_traindata():
    data = load_svmlight_file("F:\\a9a")
    for i in range( data[1].shape[0]):
        if  data[1][i] ==(-1):
            data[1][i]=0
    return data[0], data[1]

C = 0.1
def get_loss(w,x,y):
    loss=0
    for i in range(x.shape[0]):
        loss+=max(0,1-y[i]*np.dot(w.T,x[i]))
    loss=(C/x.shape[0])*loss+1/2*(w.T.dot(w))
    return loss

def gradient(w,x,y,index):
    sum = 0
    for j in index:
        if (1-y[j]*np.dot(w.T,x[j])>=0):
            sum+=-y[j]*x[j]
    return w+(C/100)*sum

# read_files
X_train,y_train = get_traindata()
X_test,y_test = get_testdata()
a = np.ones((X_train.shape[0],1))
b = np.zeros((X_test.shape[0],1))
c = 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))
X_test = np.column_stack((X_test,c))
iteration=1000


w1= np.zeros((124,))
learn_rate1=0.001
loss_SGD = []
loss_SGD_test = []
def SGD(w,x,y,x_t,y_t,index):
    g=gradient(w,x,y,index)
    w=w-learn_rate1*g
    loss_SGD.append(get_loss(w,x,y))
    loss_SGD_test.append(get_loss(w,x_t,y_t))
    return w

w2= np.zeros((124,))
v2=np.zeros((124,))
gama2=0.9
loss_NAG = []
loss_NAG_test = []
learn_rate2=0.001
def NAG(w,v,x,y,x_t,y_t,index):
     g=gradient(w-gama2*v,x,y,index)
     v=gama2*v+learn_rate2*g
     w=w-v
     loss_NAG.append(get_loss(w,x,y))
     loss_NAG_test.append(get_loss(w,x_t,y_t))
     return w,v


w3=np.zeros((124,))
G3=np.zeros((124,))
gama3=0.9
e=10**(-8)
loss_RMSProp = []
loss_RMSProp_test = []
learn_rate3=0.001
def RMSProp(w,G,x,y,x_t,y_t,index):
    g=gradient(w,x,y,index)
    G=G*gama3+(1-gama3)*np.square(g)
    G_temp=G
    for j in range(G.shape[0]):
        G_temp[j]=learn_rate3/math.sqrt(G_temp[j]+e)
    w=w-G_temp*g
    loss_RMSProp.append(get_loss(w,x,y))
    loss_RMSProp_test.append(get_loss(w,x_t,y_t))
    return w,G

w4=np.zeros((124,))
t4=np.zeros((124,))
G4=np.zeros((124,))
gama4=0.9
e=10**(-6)
loss_AdaDelta = []
loss_AdaDelta_test = []
def AdaDelta(w,G,t,x,y,x_t,y_t,index):
    g=gradient(w,x,y,index)
    G=gama4*G+(1-gama4)*g*g
    G_temp=G+e
    t_temp=t+e
    for j in range(G.shape[0]):

G_temp[j]=-((math.sqrt(t_temp[j]))/(math.sqrt(G_temp[j])))
    w_temp=G_temp*g
    w=w+w_temp
    t=gama4*t+(1-gama4)*np.square(w_temp)
    loss_AdaDelta.append(get_loss(w,x,y))
    loss_AdaDelta_test.append(get_loss(w,x_t,y_t))
    return w,G,t

w5=np.zeros((124,))
m5=np.zeros((124,))
a5=np.zeros((124,))
G5=np.zeros((124,))
peta5=0.9
gama5=0.999
e=10**(-6)
learn_rate5=0.001
loss_Adam = []
loss_Adam_test = []
def Adam(w,G,m,t,x,y,x_t,y_t,index):
    g=gradient(w,x,y,index)
    m=peta5*m+(1-peta5)*g
    G=gama5*G+(1-gama5)*g*g

alpa=learn_rate5*((math.sqrt(1-math.pow(gama5,t)))/(math.sq
rt(1-math.pow(peta5,t))))
    G_temp=G+e
    for j in range(G.shape[0]):
        G_temp[j]=m[j]/(math.sqrt(G_temp[j]))
    w=w-alpa*G_temp
    loss_Adam.append(get_loss(w,x,y))
    loss_Adam_test.append(get_loss(w,x_t,y_t))
    return w,G,m


index= random.sample(range(X_train.shape[0]),100)
for i in range(iteration):
    w1=SGD(w1,X_train,y_train,X_test,y_test,index)
    w2,v2=NAG(w2,v2,X_train,y_train,X_test,y_test,index)

w3,G3=RMSProp(w3,G3,X_train,y_train,X_test,y_test,index)

w4,G4,t4=AdaDelta(w4,G4,t4,X_train,y_train,X_test,y_test,in
dex)

w5,G5,m5=Adam(w5,G5,m5,i+1,X_train,y_train,X_test,y_tes
t,index)
    print(i)



x = []
```

```
for i in range(iteration):
    x.append(i+1)
plt.figure(figsize=(10,9), dpi=80)
l1,= plt.plot(x, loss_SGD,color='black')
l2,=plt.plot(x, loss_SGD_test, color='black', linewidth=1.0,
linestyle='--')
l3,=plt.plot(x, loss_NAG, color='sienna')
l4,=plt.plot(x, loss_NAG_test, color='olive', linewidth=1.0,
linestyle='--')
l5,=plt.plot(x, loss_RMSProp, color='navy')
l6,=plt.plot(x, loss_RMSProp_test, color='c', linewidth=1.0,
linestyle='--')
l7,=plt.plot(x, loss_AdaDelta, color='blue')
l8,=plt.plot(x, loss_AdaDelta_test, color='m', linewidth=1.0,
linestyle='--')
l9,= plt.plot(x, loss_Adam,color='orange')
l0,=plt.plot(x, loss_Adam_test, color='gray', linewidth=1.0,
linestyle='--')
plt.xlabel('times')
plt.ylabel('loss')
plt.legend(handles=[l1, l2,l3,l4,l5,l6,l7,l8,l9,l0,], labels=['l1',
'l2','l3', 'l4','l5', 'l6','l7', 'l8','l9', 'l0'], loc='best')
plt.show()
```

IV. CONCLUSION

A. Results analysis

Through doing the Logistic Regressionand Linear Classification experiment,we can see that select different descent method, the loss descent rate are different.
The SGD method drop the slowest ,then is RMSProp,Adam,NAG,AdaDelta.

B. Summary

Through this experiment,I further understood the principle of logistic regression and linear classification and the SGD,NAG,RMSProp, AdaDelta, and Adam gradient descent. By learning and compare the five gradient descent method, we can further understand the important content of the gradient learning, and realize the process of optimizing and adjusting the parameters.