

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **林英杰**

**Student ID 201530612286**

**E-mail 1142983747@qq.com**

**Tutor**   **谭明奎**

**Date submitted** **2017.12.15**

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

**2. Time:** 2017.12.9

**3. Reporter:** 林英杰

**4. Purposes:**

（1）Compare and understdan the difference between gradient descent and stochastic gradient descent.

（2）Compare and understand the differences and relationships between Logistic regression and linear classification.

（3）Further understand the principles of SVM and practice on larger data.

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

Experiment uses [a9a](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#a9a) of [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/), including 32561/16281(testing) samples and each sample has 123/123 (testing) features. Please download the training set and validation set.

**6. Experimental steps:**

Logistic Regression and Stochastic Gradient Descent

1. Load the training set and validation set.
2. Initalize logistic regression model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
4. Calculate gradient G toward loss function from partial samples.
5. Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam).
6. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Predict under validation set and get the different optimized method loss LNAG，LRMSProp，LAdaDelta  and  LAdam.
7. Repeate step 4 to 6 for several times, and drawing graph of LNAG，LRMSProp， LAdaDelta  and LAdam with the number of iterations.

Linear Classification and Stochastic Gradient Descent

1. Load the training set and validation set.
2. Initalize SVM model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
4. Calculate gradient G toward loss function from partial samples.
5. Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam).
6. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Predict under validation set and get the different optimized method loss LNAG，LRMSProp，LAdaDelta and LAdam.
7. Repeate step 4 to 6 for several times, and drawing graph of LNAG，LRMSProp，LAdaDelta  and LAdam with the number of iterations.

**7. Code:**

Logistic Regression and Stochastic Gradient Descent

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

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

mem = Memory("./mycache")

@mem.cache

def get\_data(path):

data = load\_svmlight\_file(path)

return data[0], data[1]

import numpy as np

import matplotlib.pyplot as plt

X\_train, y\_train= get\_data("G:\\学习资料\\大三上\\机器学习\\experiment\\experiment2\\a9a.txt")

X\_test,y\_test= get\_data("G:\\学习资料\\大三上\\机器学习\\experiment\\experiment2\\a9a.t")

X\_train = X\_train.toarray()

X\_test = X\_test.toarray()

y\_train[y\_train==-1]=0

y\_test[y\_test==-1]=0

X\_train=np.hstack((X\_train,np.ones((X\_train.shape[0],1))))#add a coloum of 1 after X\_train

y\_train = y\_train.reshape((y\_train.shape[0],1))

X\_test=np.hstack((X\_test,np.zeros((X\_test.shape[0],1))))#add a coloum of 0 after X\_test

X\_test=np.hstack((X\_test,np.ones((X\_test.shape[0],1))))#add a coloum of 1 after X\_test

y\_test = y\_test.reshape((y\_test.shape[0],1))

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

w=np.ones((X\_train.shape[1],1)) #initialize the w

w1=np.ones((X\_train.shape[1],1)) #initialize the w1

w2=np.ones((X\_train.shape[1],1)) #initialize the w2

w3=np.ones((X\_train.shape[1],1)) #initialize the w3

w4=np.ones((X\_train.shape[1],1)) #initialize the w4

P=0

P1=0

P2=0

P3=0

P4=0

num=500 #times

loss\_test\_SGD=np.zeros((num)) #the loss of test set

loss\_test\_NAG=np.zeros((num))

loss\_test\_RMSProp=np.zeros((num))

loss\_test\_AdaDelta=np.zeros((num))

loss\_test\_Adam=np.zeros((num))

def h(x,W):

return np.exp(x.dot(W))

def sample(x,y,n):

index\_sample=np.random.randint(0,x.shape[0]-1,n)

x\_sample=np.zeros((n,x.shape[1]))

y\_sample=np.zeros((n,y.shape[1]))

index=0

for i in index\_sample:

x\_sample[index]=x[i]

y\_sample[index]=y[i]

index=index+1

return x\_sample,y\_sample

def right\_rate(W):

p=X\_test.dot(W)

right=0

for i in range(X\_test.shape[0]):

if p[i]>=0:

p[i]=1

else:

p[i]=0

if p[i]==y\_test[i]:

right=right+1

P=right/X\_test.shape[0]

return P

#SGD

rate=0.5 #learning rate

#NAG element

v=0.1

gama\_NAG=0.9

rate\_NAG=0.05

#RMSProp

gama\_RMSProp=0.9

rate\_RMSProp=0.01

epsilon\_RMSProp=0.00000001

G\_RMSProp=0

#AdaDelta

G\_AdaDelta=0

gama\_AdaDelta=0.95

delta=0.01

epsilon\_AdaDelta=0.00000001

#Adam

G\_Adam=0

beta=0.9

gama\_Adam=0.999

rate\_Adam=0.02

epsilon\_Adam=0.00000001

m=0

for i in range(num):

X\_sample,y\_sample=sample(X\_train,y\_train,64)

one\_sample=np.ones((X\_sample.shape[0],1))

#SGD

J=X\_sample.T.dot((h(X\_sample,w)/(one\_sample+h(X\_sample,w))-y\_sample))/X\_sample.shape[0]

w-=rate\*J

loss\_test\_SGD[i]=(one\_test.T.dot((np.log(one\_test+h(X\_test,w))-X\_test.dot(w)\*y\_test)))/X\_test.shape[0]

if right\_rate(w)>P:

P=right\_rate(w)

#NAG

J1=X\_sample.T.dot((h(X\_sample,w1)/(one\_sample+h(X\_sample,w1))-y\_sample))/X\_sample.shape[0]

g\_NAG=J1-gama\_NAG\*v

v=gama\_NAG\*v+rate\_NAG\*g\_NAG

w1=w1-v

loss\_test\_NAG[i]=(one\_test.T.dot((np.log(one\_test+h(X\_test,w1))-X\_test.dot(w1)\*y\_test)))/X\_test.shape[0]

if right\_rate(w1)>P1:

P1=right\_rate(w1)

#RMSProp

J2=X\_sample.T.dot((h(X\_sample,w2)/(one\_sample+h(X\_sample,w2))-y\_sample))/X\_sample.shape[0]

g\_RMSProp=J2

G\_RMSProp=gama\_RMSProp\*G\_RMSProp+(1-gama\_RMSProp)\*g\_RMSProp\*g\_RMSProp

w2-=rate\_RMSProp/(np.sqrt(G\_RMSProp+epsilon\_RMSProp))\*g\_RMSProp

loss\_test\_RMSProp[i]=(one\_test.T.dot((np.log(one\_test+h(X\_test,w2))-X\_test.dot(w2)\*y\_test)))/X\_test.shape[0]

if right\_rate(w2)>P2:

P2=right\_rate(w2)

#AdaDelta

J3=X\_sample.T.dot((h(X\_sample,w3)/(one\_sample+h(X\_sample,w3))-y\_sample))/X\_sample.shape[0]

g\_AdaDelta=J3

G\_AdaDelta=gama\_AdaDelta\*G\_AdaDelta+(1-gama\_AdaDelta)\*g\_AdaDelta\*g\_AdaDelta

delta\_Q=-(np.sqrt(delta+epsilon\_AdaDelta)/np.sqrt(G\_AdaDelta+epsilon\_AdaDelta))\*g\_AdaDelta

w3+=delta\_Q

delta=gama\_AdaDelta\*delta+(1-gama\_AdaDelta)\*delta\_Q\*delta\_Q

loss\_test\_AdaDelta[i]=(one\_test.T.dot((np.log(one\_test+h(X\_test,w3))-X\_test.dot(w3)\*y\_test)))/X\_test.shape[0]

if right\_rate(w3)>P3:

P3=right\_rate(w3)

#Adam

J4=X\_sample.T.dot((h(X\_sample,w4)/(one\_sample+h(X\_sample,w4))-y\_sample))/X\_sample.shape[0]

g\_Adam=J4

m=beta\*m+(1-beta)\*g\_Adam

G\_Adam=gama\_Adam\*G\_Adam+(1-gama\_Adam)\*g\_Adam\*g\_Adam

alpha=rate\_Adam\*(np.sqrt(1-gama\_Adam)/(1-beta))

w4-=alpha\*(m/np.sqrt(G\_Adam+epsilon\_Adam))

loss\_test\_Adam[i]=(one\_test.T.dot((np.log(one\_test+h(X\_test,w4))-X\_test.dot(w4)\*y\_test)))/X\_test.shape[0]

if right\_rate(w4)>P4:

P4=right\_rate(w4)

i =range(num)

#Draw graphs of Ltrain and Ltest with the number of iterations

plt.rcParams['figure.figsize']= (20,10)

plt.plot(i,loss\_test\_SGD,label = 'loss\_SGD')

plt.plot(i,loss\_test\_NAG,label = 'loss\_NAG')

plt.plot(i,loss\_test\_RMSProp,label = 'loss\_RMSProp')

plt.plot(i,loss\_test\_AdaDelta,label = 'loss\_AdaDelta')

plt.plot(i,loss\_test\_Adam,label = 'loss\_Adam')

plt.legend(loc='upper right')

plt.xlabel('times')

plt.ylabel('loss')

plt.show()

print(P)

print(P1)

print(P2)

print(P3)

print(P4)

Linear Classification and Stochastic Gradient Descent

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

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

mem = Memory("./mycache")

@mem.cache

def get\_data(path):

data = load\_svmlight\_file(path)

return data[0], data[1]

import numpy as np

import matplotlib.pyplot as plt

X\_train, y\_train= get\_data("G:\\学习资料\\大三上\\机器学习\\experiment\\experiment2\\a9a.txt")

X\_test,y\_test= get\_data("G:\\学习资料\\大三上\\机器学习\\experiment\\experiment2\\a9a.t")

X\_train = X\_train.toarray()

X\_test = X\_test.toarray()

X\_train=np.hstack((X\_train,np.ones((X\_train.shape[0],1))))#add a coloum of 1 after X\_train

y\_train = y\_train.reshape((y\_train.shape[0],1))

X\_test=np.hstack((X\_test,np.zeros((X\_test.shape[0],1))))#add a coloum of 0 after X\_test

X\_test=np.hstack((X\_test,np.ones((X\_test.shape[0],1))))#add a coloum of 1 after X\_test

y\_test = y\_test.reshape((y\_test.shape[0],1))

w=np.ones((X\_train.shape[1],1)) #initialize the w

w1=np.ones((X\_train.shape[1],1)) #initialize the w1

w2=np.ones((X\_train.shape[1],1)) #initialize the w2

w3=np.ones((X\_train.shape[1],1)) #initialize the w3

w4=np.ones((X\_train.shape[1],1)) #initialize the w4

P=0

P1=0

P2=0

P3=0

P4=0

num=500 #times

loss\_test\_SGD=np.zeros((num)) #the loss of test set

loss\_test\_NAG=np.zeros((num))

loss\_test\_RMSProp=np.zeros((num))

loss\_test\_AdaDelta=np.zeros((num))

loss\_test\_Adam=np.zeros((num))

def h(x,W):

return x.dot(W)

def sample(x,y,n):

index\_sample=np.random.randint(0,x.shape[0]-1,n)

x\_sample=np.zeros((n,x.shape[1]))

y\_sample=np.zeros((n,y.shape[1]))

index=0

for i in index\_sample:

x\_sample[index]=x[i]

y\_sample[index]=y[i]

index=index+1

return x\_sample,y\_sample

def right\_rate(W):

p=X\_test.dot(W)

right=0

for i in range(X\_test.shape[0]):

if p[i]>=0:

p[i]=1

else:

p[i]=-1

if p[i]==y\_test[i]:

right=right+1

P=right/X\_test.shape[0]

return P

#SGD

rate=0.01 #learning rate

C=100

#NAG element

v=0.1

gama\_NAG=0.9

rate\_NAG=0.0005

C1=100

#RMSProp

gama\_RMSProp=0.9

rate\_RMSProp=0.01

epsilon\_RMSProp=0.00000001

G\_RMSProp=0

C2=100

#AdaDelta

G\_AdaDelta=0

gama\_AdaDelta=0.95

delta=0.01

epsilon\_AdaDelta=0.000001

C3=100

#Adam

G\_Adam=0

beta=0.9

gama\_Adam=0.999

rate\_Adam=0.01

epsilon\_Adam=0.00000001

m=0

C4=100

for i in range (num):

X\_sample,y\_sample=sample(X\_train,y\_train,64)

#SGD

temp\_loss=0.0

temp\_w=0 #initialize the temp\_w

temp=1-y\_sample\*h(X\_sample,w)

for j in range(temp.shape[0]):

if(temp[j]>0):

temp\_w+=(y\_sample[j]\*X\_sample[j].T).reshape((X\_sample.shape[1],1))

J=w-C\*(temp\_w/(X\_sample.shape[0]))

w-=rate\*J #update the w

temp=1-y\_test\*h(X\_test,w)

for k in range(temp.shape[0]):

if(temp[k]>0):

temp\_loss+=temp[k]

loss\_test\_SGD[i]=C\*temp\_loss+0.5\*np.sum(w\*w)

if right\_rate(w)>P:

P=right\_rate(w)

#NAG

temp\_loss=0.0

temp\_w=0 #initialize the temp\_w

temp=1-y\_sample\*h(X\_sample,w1)

for j in range(temp.shape[0]):

if(temp[j]>0):

temp\_w+=(y\_sample[j]\*X\_sample[j].T).reshape((X\_sample.shape[1],1))

J1=w1-C1\*(temp\_w/(X\_sample.shape[0]))

g\_NAG=J1-gama\_NAG\*v

v=gama\_NAG\*v+rate\_NAG\*g\_NAG

w1=w1-v #update the w

temp=1-y\_test\*h(X\_test,w1)

for k in range(temp.shape[0]):

if(temp[k]>0):

temp\_loss+=temp[k]

loss\_test\_NAG[i]=C1\*temp\_loss+0.5\*np.sum(w1\*w1)

if right\_rate(w1)>P1:

P1=right\_rate(w1)

#RMSProp

temp\_loss=0.0

temp\_w=0 #initialize the temp\_w

temp=1-y\_sample\*h(X\_sample,w2)

for j in range(temp.shape[0]):

if(temp[j]>0):

temp\_w+=(y\_sample[j]\*X\_sample[j].T).reshape((X\_sample.shape[1],1))

J2=w2-C2\*(temp\_w/(X\_sample.shape[0]))

g\_RMSProp=J2

G\_RMSProp=gama\_RMSProp\*G\_RMSProp+(1-gama\_RMSProp)\*g\_RMSProp\*g\_RMSProp

w2-=rate\_RMSProp/(np.sqrt(G\_RMSProp+epsilon\_RMSProp))\*g\_RMSProp

temp=1-y\_test\*h(X\_test,w2)

for k in range(temp.shape[0]):

if(temp[k]>0):

temp\_loss+=temp[k]

loss\_test\_RMSProp[i]=C2\*temp\_loss+0.5\*np.sum(w2\*w2)

if right\_rate(w2)>P2:

P2=right\_rate(w2)

#AdaDelta

temp\_loss=0.0

temp\_w=0 #initialize the temp\_w

temp=1-y\_sample\*h(X\_sample,w3)

for j in range(temp.shape[0]):

if(temp[j]>0):

temp\_w+=(y\_sample[j]\*X\_sample[j].T).reshape((X\_sample.shape[1],1))

J3=w3-C3\*(temp\_w/(X\_sample.shape[0]))

g\_AdaDelta=J3

G\_AdaDelta=gama\_AdaDelta\*G\_AdaDelta+(1-gama\_AdaDelta)\*g\_AdaDelta\*g\_AdaDelta

delta\_Q=-(np.sqrt(delta+epsilon\_AdaDelta)/np.sqrt(G\_AdaDelta+epsilon\_AdaDelta))\*g\_AdaDelta

w3+=delta\_Q

delta=gama\_AdaDelta\*delta+(1-gama\_AdaDelta)\*delta\_Q\*delta\_Q

temp=1-y\_test\*h(X\_test,w3)

for k in range(temp.shape[0]):

if(temp[k]>0):

temp\_loss+=temp[k]

loss\_test\_AdaDelta[i]=C3\*temp\_loss+0.5\*np.sum(w3\*w3)

if right\_rate(w3)>P3:

P3=right\_rate(w3)

#Adam

temp\_loss=0.0

temp\_w=0 #initialize the temp\_w

temp=1-y\_sample\*h(X\_sample,w4)

for j in range(temp.shape[0]):

if(temp[j]>0):

temp\_w+=(y\_sample[j]\*X\_sample[j].T).reshape((X\_sample.shape[1],1))

J4=w4-C4\*(temp\_w/(X\_sample.shape[0]))

g\_Adam=J4

m=beta\*m+(1-beta)\*g\_Adam

G\_Adam=gama\_Adam\*G\_Adam+(1-gama\_Adam)\*g\_Adam\*g\_Adam

alpha=rate\_Adam\*(np.sqrt(1-gama\_Adam)/(1-beta))

w4-=alpha\*(m/np.sqrt(G\_Adam+epsilon\_Adam))

temp=1-y\_test\*h(X\_test,w4)

for k in range(temp.shape[0]):

if(temp[k]>0):

temp\_loss+=temp[k]

loss\_test\_Adam[i]=C4\*temp\_loss+0.5\*np.sum(w4\*w4)

if right\_rate(w4)>P4:

P4=right\_rate(w4)

i =range(num)

#Draw graphs of Ltrain and Ltest with the number of iterations

plt.rcParams['figure.figsize']= (20,10)

plt.plot(i,loss\_test\_SGD,label = 'loss\_SGD')

plt.plot(i,loss\_test\_NAG,label = 'loss\_NAG')

plt.plot(i,loss\_test\_RMSProp,label = 'loss\_RMSProp')

plt.plot(i,loss\_test\_AdaDelta,label = 'loss\_AdaDelta')

plt.plot(i,loss\_test\_Adam,label = 'loss\_Adam')

plt.legend(loc='upper right')

plt.xlabel('times')

plt.ylabel('loss')

plt.show()

print(P)

print(P1)

print(P2)

print(P3)

print(P4)

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

Logistic Regression and Stochastic Gradient Descent

zero initialization

Linear Classification and Stochastic Gradient Descent

zero initialization

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

Logistic Regression and Stochastic Gradient Descent

Loss function: 

,

Gradient: the gradient of wi 

For w14(b) x=1

The gradient of w 

Linear Classification and Stochastic Gradient Descent

Loss function: 

Gradient: the gradient of wi 

**10. Experimental results and curve:**(Fill in this content for various methods of gradient descent respectively)

Logistic Regression and Stochastic Gradient Descent

## Hyper-parameter selection:

SGD: η=0.5

NAG: η=0.05, γ=0.9

RMSProp: η=0.01, γ=0.9 ,ε=1e-8

AdaDelta: γ=0.95, ε=1e-8

Adam: η=0.02, γ=0.999, ε=1e-8, β=0.9

Iteration: 500

Number of samples: 64

## Predicted Results (Best Results):

The right rate of SGD: 0.8490264725753947

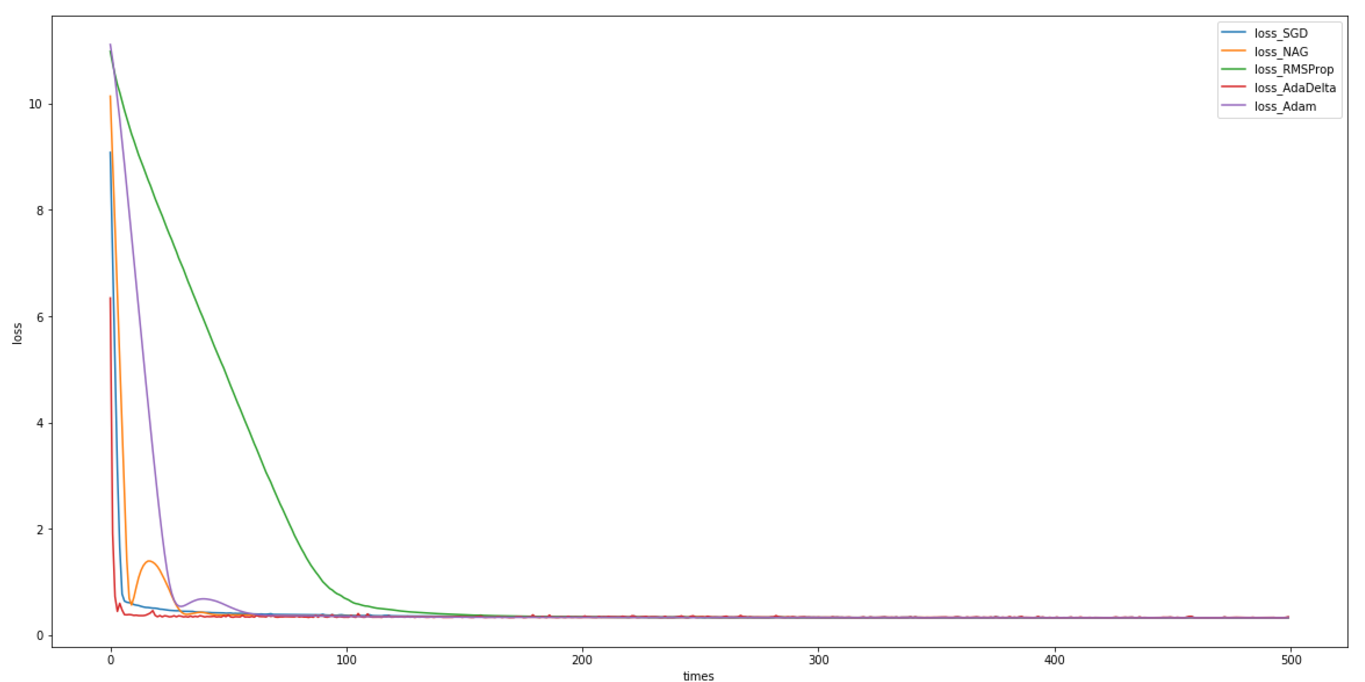
The right rate of NAG: 0.8493335790184878

The right rate of RMSProp: 0.85093.5325225723

The right rate of AdaDelta: 0.8485965235550642

The right rate of Adam: 0.8503777409250046

## Loss curve:



Linear Classification and Stochastic Gradient Descent

## Hyper-parameter selection:

SGD: η=0.01,C=100

NAG: η=0.0005, γ=0.9,C=100

RMSProp: η=0.01, γ=0.9 ,ε=1e-8,C=100

AdaDelta: γ=0.95, ε=1e-8,C=100

Adam: η=0.01, γ=0.999, ε=1e-8, β=0.9,C=100

Iteration: 500

Number of samples: 64

## Predicted Results (Best Results):

The right rate of SGD: 0.8474909403599288

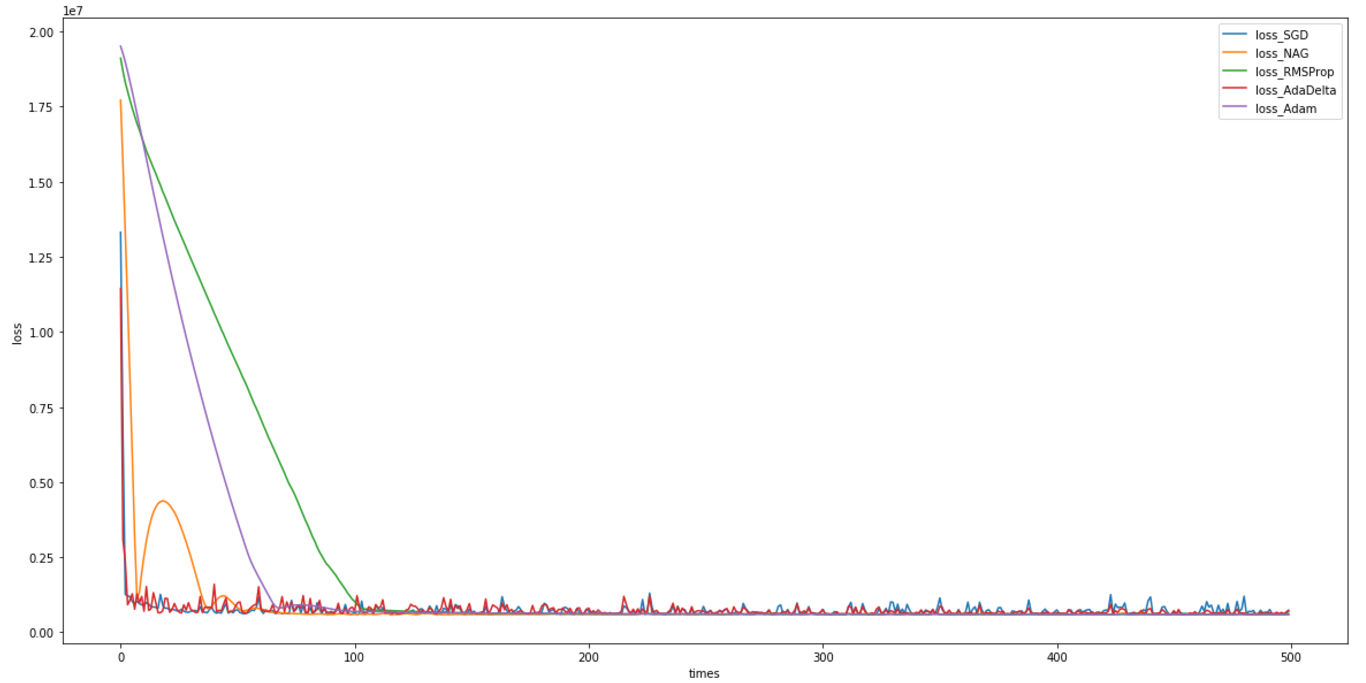
The right rate of NAG: 0.8492721577298692

The right rate of RMSProp: 0.8478594680916406

The right rate of AdaDelta: 0.8455868804127511

The right rate of Adam: 0.8449726675265647

## Loss curve:



**11. Results analysis:**

Logistic Regression and Stochastic Gradient Descent

The five optimization models can significantly reduce the Loss value on the test set in the first 200 iterations, and then slowly decrease and stabilize in 200-500 iterations, and the corresponding correctness in the test set The rate can reach more than 84%.

The downward trend of the five optimization models is consistent, and there is no big data fluctuation, and the curve obtained is relatively smooth.

Linear Classification and Stochastic Gradient Descent

The five optimization models can significantly reduce the Loss value on the test set in the first 200 iterations, and then slowly decrease and stabilize in 200-500 iterations, and the corresponding correctness in the test set The rate can reach more than 84%.

The downward trend of the five optimization models is consistent, but the data of SGD and AdaDElta curves are more volatile than the other three curves, while the other three curves are relatively smooth.

**12. Similarities and differences between logistic regression and linear classification：**

The basic function of logistic regression and linear classification is to construct the model by the linear formula of .Both ways are based on the classification of the problem into linear thinking.

The difference is that logistic regression is achieved by mapping the data to existing functions, whereas linear classification assumes a function to fit the data. Both methods use a gradient descent, and linear regression also uses the SVM method.

**13. Summary:**

This experiment let me learn the method of random gradient descent and five optimization methods. Let me know more about them. At the same time, by constantly adjusting the parameters to achieve better accuracy, I became more aware of the importance of the parameters and realized that it was not easy to learn a good model. This experiment gave me a deeper understanding of the machine learning course.