

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **吕睿**

**学 号 201530612477**

**邮 箱 729239617@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12 月 15 日**

## 1. 实验题目: 线性回归、线性分类与梯度下降

## 2. 实验时间：2017年 12 月 2 日

## 3. 报告人: 吕睿

## 4. 实验目的:

## (1)对比理解梯度下降和随机梯度下降的区别与联系。

## (2)对比理解逻辑回归和线性分类的区别与联系。

## (3)进一步理解SVM的原理并在较大数据上实践。

## 数据集以及数据分析：

## 实验使用的是LIBSVM Data的中的a9a数据，包含32561 / 16281(testing)个样本，每个样本有123/123 (testing)个属性。请自行下载训练集和验证集。

## 实验步骤:

**逻辑回归与随机梯度下降**

1. 读取实验训练集和验证集。
2. 逻辑回归模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。
3. 选择Loss函数及对其求导，过程详见课件ppt。
4. 求得**部分样本**对Loss函数的梯度G。
5. **使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta和Adam）**。
6. 选择合适的阈值，将验证集中计算结果**大于阈值的标记为正类，反之为负类**。在验证集上测试并得到不同优化方法的Loss函数值LNAG，LRMSProp，LAdaDelta和LAdam。
7. 重复步骤4-6若干次，**画出LNAG，LRMSProp，LAdaDelta和LAdam随迭代次数的变化图**。

**线性分类与随机梯度下降**

1. 读取实验训练集和验证集。
2. 支持向量机模型参数初始化，可以考虑全零初始化，随机初始化或者正态分布初始化。
3. 选择Loss函数及对其求导，过程详见课件ppt。
4. 求得**部分样本**对Loss函数的梯度G。
5. **使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta和Adam）**。
6. 选择合适的阈值，将验证集中计算结果**大于阈值的标记为正类，反之为负类**。在验证集上测试并得到不同优化方法的Loss函数值LNAG，LRMSProp，LAdaDelta和LAdam。
7. 重复步骤4-6若干次，**画出LNAG，LRMSProp，LAdaDelta和LAdam随迭代次数的变化图**。

## 7. 代码内容:

## **逻辑回归和随机梯度下降**

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

X\_train, y\_train = load\_svmlight\_file("a9a.txt")[0], load\_svmlight\_file("a9a.txt")[1]

X\_test, y\_test = load\_svmlight\_file("a9a.t.txt")[0], load\_svmlight\_file("a9a.t.txt")[1]

import numpy as np

X\_train = X\_train.dot(np.eye(X\_train.shape[1]))

X\_test = X\_test.dot(np.eye(X\_test.shape[1]))

X\_train = np.hstack((X\_train,np.ones((X\_train.shape[0],1)))) #把所有x的属性加上一个1，用于和w14（b）相乘

X\_test = np.hstack((X\_test,np.ones((X\_test.shape[0],1)))) #把所有x的属性加上一个0，（补齐数据文件的全零特征）

X\_test = np.hstack((X\_test,np.ones((X\_test.shape[0],1)))) #把所有x的属性加上一个1，用于和w14（b）相乘

def changeLabel(y):#将label中的-1值全部改为0以便设计Loss函数

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

if y[i] == -1:

y[i] = 0

changeLabel(y\_train)

changeLabel(y\_test)

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

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

w\_sgd = np.zeros((X\_train.shape[1],1)) #将SGD模型参数w初始化为全零

w\_nag = np.zeros((X\_train.shape[1],1)) #将NAG模型参数w初始化为全零

w\_rmsp = np.zeros((X\_train.shape[1],1)) #将RMSProp模型参数w初始化为全零

w\_adaD = np.zeros((X\_train.shape[1],1)) #将AdaDelta模型参数w初始化为全零

w\_adam = np.zeros((X\_train.shape[1],1)) #将Adam模型参数w初始化为全零

n = 10000 #设置迭代次数

LSGD = np.zeros((n)) #初始化用于保存LSGD的值随迭代次数变化的数组

LNAG = np.zeros((n)) #初始化用于保存LNAG的值随迭代次数变化的数组

LRMSProp = np.zeros((n)) #初始化用于保存LRMSProp的值随迭代次数变化的数组

LAdaDelta = np.zeros((n)) #初始化用于保存LAdaDelta的值随迭代次数变化的数组

LAdam = np.zeros((n)) #初始化用于保存LAdam的值随迭代次数变化的数组

def Sigmod(w,X):#定义Sigmod函数,输出对每个样本都进行计算后的列向量

return 1/(1+np.exp(-((X.dot(w)))))

def Lfun(w,X,y):#定义Loss函数

m = y.shape[0]

s = Sigmod(w,X)

l = -(y\*np.log(s)+(1-y)\*np.log(1-s))

return l.sum()

def DER\_SGD(w,X,y):#定义随机梯度下降的梯度函数

m = 100 #从数据集中随机取样

index = np.random.randint(0,X.shape[0],m)

X\_sample = np.zeros((m,X.shape[1]))

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

for i in range (m):

X\_sample[m-1] = X[index[m-1]]

y\_sample[m-1] = y[index[m-1]]

return -((X\_sample.T).dot(y\_sample-Sigmod(w,X\_sample)))

def cRate(w,X,y):#定义用于计算预测正确率的函数

m = y.shape[0]

j = (X.dot(w))

c = 0

for i in range (m):

if (j[i]>0 and y[i]==1) or (j[i]<0 and y[i]==0):

c = c + 1

return c/m

def SGD(w,lnrt):

lnrt = lnrt

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

w -= lnrt\*g

LSGD[i] = Lfun(w,X\_test,y\_test)

def NAG(w,lnrt,u):

V = 0

u = u

lnrt = lnrt

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

V = u\*V + lnrt\*g

w -= V

LNAG[i] = Lfun(w,X\_test,y\_test)

def RMSProp(w,lnrt,u,e):

G = 0

u = u

e = e

lnrt = lnrt

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

G = u\*G + (1-u)\*g\*g

w -= (lnrt/((G+e)\*\*0.5))\*g

LRMSProp[i] = Lfun(w,X\_test,y\_test)

def AdaDelta(w,u,e):

G = 0

u = u

delta = 0

e = e

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

G = u\*G + (1-u)\*(g\*g)

wdelta = -(((delta+e)\*\*0.5)/((G+e)\*\*0.5)) \* g

w += wdelta

delta = u\*delta + (1-u)\*(wdelta\*wdelta)

LAdaDelta[i] = Lfun(w,X\_test,y\_test)

def Adam(w,lnrt,u,e,B):

G = 0

u = u

B = B

m = 0

e = e

lnrt = lnrt

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

m = B\*m + (1-B)\*g

G = u\*G + (1-u)\*(g\*g)

w-=(lnrt\*((1-u\*\*(i+1))\*\*0.5)\*m)/((1-B\*\*(i+1))\*((G+e)\*\*0.5))

LAdam[i] = Lfun(w,X\_test,y\_test)

SGD(w\_sgd,0.01)

NAG(w\_nag,0.001,0.9)

RMSProp(w\_rmsp,0.0015,0.99,1e-7)

AdaDelta(w\_adaD,0.95,1e-6)

Adam(w\_adam,0.002,0.999,1e-8,0.9)

import matplotlib.pyplot as plt

#绘制总图

x = np.arange(0,n,1)

plt.rcParams['figure.figsize']=(40,20)

plt.plot(x, LSGD, 'black',label = 'LSGD')

plt.plot(x, LNAG, 'r',label = 'LNAG')

plt.plot(x, LRMSProp, 'b',label = 'LARMSProp')

plt.plot(x, LAdaDelta, 'y',label = 'LAdaDelta')

plt.plot(x, LAdam, 'g',label = 'LAdam')

plt.legend(loc='upper right')

plt.xlabel('Times of iteration')

plt.ylabel('Loss')

plt.show()

## **线性分类和随机梯度下降**

from sklearn.externals.joblib import Memory

from sklearn.datasets import load\_svmlight\_file

X\_train, y\_train = load\_svmlight\_file("a9a.txt")[0], load\_svmlight\_file("a9a.txt")[1]

X\_test, y\_test = load\_svmlight\_file("a9a.t.txt")[0], load\_svmlight\_file("a9a.t.txt")[1]

import numpy as np

X\_train = X\_train.dot(np.eye(X\_train.shape[1]))

X\_test = X\_test.dot(np.eye(X\_test.shape[1]))

X\_train = np.hstack((X\_train,np.ones((X\_train.shape[0],1)))) #把所有x的属性加上一个1，用于和w14（b）相乘

X\_test = np.hstack((X\_test,np.ones((X\_test.shape[0],1)))) #把所有x的属性加上一个0，（补齐数据文件的全零特征）

X\_test = np.hstack((X\_test,np.ones((X\_test.shape[0],1)))) #把所有x的属性加上一个1，用于和w14（b）相乘

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

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

w\_sgd = np.zeros((X\_train.shape[1],1)) #将SGD模型参数w初始化为全零

w\_nag = np.zeros((X\_train.shape[1],1)) #将NAG模型参数w初始化为全零

w\_rmsp = np.zeros((X\_train.shape[1],1)) #将RMSProp模型参数w初始化为全零

w\_adaD = np.zeros((X\_train.shape[1],1)) #将AdaDelta模型参数w初始化为全零

w\_adam = np.zeros((X\_train.shape[1],1)) #将Adam模型参数w初始化为全零

n = 10000 #设置迭代次数

LSGD = np.zeros((n)) #初始化用于保存LSGD的值随迭代次数变化的数组

LNAG = np.zeros((n)) #初始化用于保存LNAG的值随迭代次数变化的数组

LRMSProp = np.zeros((n)) #初始化用于保存LRMSProp的值随迭代次数变化的数组

LAdaDelta = np.zeros((n)) #初始化用于保存LAdaDelta的值随迭代次数变化的数组

LAdam = np.zeros((n)) #初始化用于保存LAdam的值随迭代次数变化的数组

def Lfun(w,X,y):#定义Loss函数

m = y.shape[0]

l = 1-(X.dot(w))\*y

for i in range (m):

if l[i] < 0:

l[i] = 0

return l.sum()+0.5\*np.sum(w\*w)

def DER\_SGD(w,X,y):#定义随机梯度下降的梯度函数

m = 100 #从数据集中随机取样

index = np.random.randint(0,X.shape[0],m)

X\_sample = np.zeros((m,X.shape[1]))

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

for i in range (m):

X\_sample[m-1] = X[index[m-1]]

y\_sample[m-1] = y[index[m-1]]

j = (X\_sample.dot(w))\*y\_sample

o = np.zeros((m,1))

for i in range (m):

if j[i] < 1:

o[i] = y\_sample[i]

return -((X\_sample.T).dot(o))

def cRate(w,X,y):

m = y.shape[0]

j = (X.dot(w))\*y

c = 0

for i in range (m):

if j[i] > 0:

c = c + 1

return c/m

def SGD(w,lnrt):

lnrt = lnrt

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

w -= lnrt\*g

LSGD[i] = Lfun(w,X\_test,y\_test)

def NAG(w,lnrt,u):

V = 0

u = u

lnrt = lnrt

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

V = u\*V + lnrt\*g

w -= V

LNAG[i] = Lfun(w,X\_test,y\_test)

def RMSProp(w,lnrt,u,e):

G = 0

u = u

e = e

lnrt = lnrt

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

G = u\*G + (1-u)\*g\*g

w -= (lnrt/((G+e)\*\*0.5))\*g

LRMSProp[i] = Lfun(w,X\_test,y\_test)

def AdaDelta(w,u,e):

G = 0

u = u

delta = 0

e = e

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

G = u\*G + (1-u)\*(g\*g)

wdelta = -(((delta+e)\*\*0.5)/((G+e)\*\*0.5)) \* g

w += wdelta

delta = u\*delta + (1-u)\*(wdelta\*wdelta)

LAdaDelta[i] = Lfun(w,X\_test,y\_test)

def Adam(w,lnrt,u,e,B):

G = 0

u = u

B = B

m = 0

e = e

lnrt = lnrt

for i in range (n):#迭代若干次，更新模型参数w

g = (DER\_SGD(w,X\_train,y\_train))

m = B\*m + (1-B)\*g

G = u\*G + (1-u)\*(g\*g)

w -= (lnrt\*((1-u\*\*(i+1))\*\*0.5)\*m)/((1-B\*\*(i+1))\*((G+e)\*\*0.5))

LAdam[i] = Lfun(w,X\_test,y\_test)

SGD(w\_sgd,0.001)

NAG(w\_nag,0.0001,0.9)

RMSProp(w\_rmsp,0.001,0.9,1e-8)

AdaDelta(w\_adaD,0.8,1e-6)

Adam(w\_adam,0.001,0.999,1e-8,0.9)

import matplotlib.pyplot as plt

x = np.arange(0,n,1)

plt.rcParams['figure.figsize']=(40,20)

plt.plot(x, LSGD, 'black',label = 'LSGD')

plt.plot(x, LNAG, 'r',label = 'LNAG')

plt.plot(x, LRMSProp, 'b',label = 'LARMSProp')

plt.plot(x, LAdaDelta, 'y',label = 'LAdaDelta')

plt.plot(x, LAdam, 'g',label = 'LAdam')

plt.legend(loc='upper right')

plt.xlabel('Times of iteration')

plt.ylabel('Loss')

plt.show()

## 逻辑回归和随机梯度下降8-12：

## 8.模型参数的初始化方法:

全零初始化。

## 9.选择的loss函数及其导数:

Loss函数：

其中，

梯度： 的梯度

（对于w14(b)而言x=1）

则整个向量的梯度向量

## 10.实验结果和曲线图:

## 超参数选择（η,epoch等）：

SGD的学习率η取0.005

NAG的学习率η取0.001,γ取0.9

RMSProp的学习率η取0.001,γ取0.9,ε取1e-8

AdaDelta的γ取0.9,ε取1e-6

Adam的学习率η取0.002,γ取0.999,ε取1e-8,β取0.9

## 预测结果（最佳结果）：

Wsgd

[ -8.33881251e-01 -3.80854456e-01 1.38862512e-01 4.36959612e-01

3.18640237e-01 -2.59416321e-03 -3.16249133e-01 1.78163547e-01

2.38683655e-01 5.71512368e-03 -5.60261661e-02 -1.53187393e-02

-2.74374610e-03 -1.80426102e-01 2.11619438e-02 -7.83913545e-02

-3.00913622e-02 -5.25264708e-02 3.04510014e-02 -4.87357409e-02

-2.12982950e-01 -2.21452686e-01 2.57797624e-01 2.14565595e-02

1.14191371e-02 -1.17707387e-01 -2.09235986e-01 -5.92052234e-02

2.71828708e-01 -4.37526521e-02 -1.37567269e-01 2.20541442e-01

-6.72581172e-02 -1.58698057e-02 -8.63579390e-01 -2.21452686e-01

-4.87357409e-02 3.28756967e-02 7.80618775e-01 9.56355942e-01

-2.56707353e-01 -7.60723469e-01 -1.76979525e-01 -1.50256986e-02

-7.34182191e-02 6.22497707e-03 1.79109047e-01 -4.62593275e-02

-4.44430160e-01 3.00435133e-01 6.09183312e-01 4.83575768e-01

-3.24058111e-01 -2.65157687e-01 -2.92775811e-02 -3.04784668e-01

-2.42720018e-01 -2.31888415e-02 1.41803325e-01 -1.85606647e-03

6.71009564e-01 -5.97791321e-01 2.93756958e-01 -2.42412384e-01

-1.28782387e-01 -3.16053777e-01 9.68254581e-02 -8.31373250e-02

-6.73145736e-02 -6.12111827e-02 -2.05435723e-01 -2.83615089e-01

-3.66582569e-02 -9.68351825e-01 6.48078479e-01 -5.92644743e-01

2.72371397e-01 -5.95679421e-01 -1.41554897e-01 -1.20580180e-01

8.35239766e-02 4.54017175e-01 1.20676626e-01 6.93923639e-03

3.40509509e-03 -3.42517319e-02 9.94886803e-03 -2.10421557e-02

-5.76639014e-03 1.72882070e-02 9.48279544e-03 -1.74087217e-02

-5.88502339e-02 -1.58798772e-02 1.03443336e-02 -1.37471754e-02

2.56820960e-03 2.60135489e-02 1.67846065e-02 -1.04131887e-02

-2.25372264e-02 -2.39012374e-02 -1.61172572e-01 -2.96195781e-02

1.96186781e-03 1.47667041e-02 -4.70822415e-03 -9.18772966e-03

1.44776437e-03 -9.86239536e-03 -2.10682096e-02 -1.78007638e-02

8.50652093e-03 5.72512724e-03 2.35912962e-03 -4.09122526e-03

-1.32967608e-02 -6.88759217e-03 -3.38042444e-02 -7.72571873e-03

-1.50689852e-02 4.35679099e-04 -9.47113700e-04 -3.20273346e-01]

Wnag

[ -7.97307702e-01 -2.99765776e-01 5.91765986e-02 3.37019546e-01

3.74222541e-01 -5.20451506e-02 -1.97233419e-01 7.16956201e-02

2.52961933e-01 -9.60805074e-03 -2.27481562e-02 -8.31767983e-03

-1.82029514e-04 -1.06589188e-01 -3.02824225e-02 -1.74512434e-01

-2.27309253e-02 7.46017727e-03 1.09875406e-01 -4.75339950e-02

-1.99079865e-01 -2.48488372e-01 1.80150821e-01 6.24375351e-02

8.82808583e-02 -1.11392352e-01 -2.16959874e-01 -5.34526563e-02

2.36132229e-01 -6.31069898e-02 -1.52286748e-01 2.16672725e-01

-1.12521648e-01 -1.53818668e-02 -9.24182001e-01 -2.48488372e-01

-4.75339950e-02 1.50718393e-01 7.42831182e-01 8.71621295e-01

-2.58962467e-01 -7.26049865e-01 -1.79011122e-01 -6.17269269e-03

-6.26283977e-02 3.45484573e-02 1.49229863e-01 -5.13982522e-02

-3.58638248e-01 1.19616328e-01 6.77815922e-01 4.61150547e-01

-2.32742331e-01 -2.61668813e-01 -7.16860339e-02 -2.92868807e-01

-1.86177848e-01 -3.37109325e-02 1.18485634e-01 -2.70193285e-03

5.63822414e-01 -5.15940221e-01 3.60124740e-01 -1.81676909e-01

-1.68671483e-01 -3.84313333e-01 -1.47732434e-01 2.75732528e-03

-4.35551269e-02 -5.43349407e-02 -8.37896156e-02 -3.11634203e-01

-1.50205896e-02 -9.35288749e-01 6.08633956e-01 -6.61594071e-01

3.34939279e-01 -7.82975687e-01 -1.64685511e-01 2.31636608e-02

1.65090712e-01 4.32752033e-01 9.91831735e-02 -1.14258638e-03

2.71321264e-03 1.36260272e-03 5.02399814e-02 8.51491186e-03

-6.93520206e-03 -3.19393032e-02 1.09705152e-02 -9.68659845e-03

-6.20362866e-03 -2.26362354e-02 3.09532862e-03 1.03907625e-04

-4.45511241e-03 4.26331839e-02 9.71769356e-03 -1.31606600e-02

-2.33064013e-02 -1.62800386e-02 -1.91918792e-01 -7.26693057e-03

-1.97136687e-02 1.00754955e-02 -2.49836279e-02 -5.62630222e-03

-2.05043285e-02 1.56141822e-02 3.58109006e-03 -3.77937300e-02

7.11053080e-03 -1.65338267e-02 -1.80959838e-02 -8.13139654e-03

3.35599806e-03 2.16392738e-04 -2.76193958e-02 -1.24249543e-02

-1.58334510e-02 1.54492482e-03 0.00000000e+00 -3.26654792e-01]

Wrmsp

[-1.82290302 -0.39944496 0.08235624 0.29082787 0.12284566 0.03903618

-0.45982501 0.34327776 0.22070314 -0.16079908 -0.00490288 0.

-0.0149968 -0.35588743 0.01863553 -0.02636312 0.01006281 -0.12853012

-0.01691826 -0.11509634 -1.04054239 -0.36147345 0.69177251 -0.16025578

-0.28741606 -0.71770843 -0.88354666 -0.28554298 0.41123283 -0.41703898

-0.54595143 0.48071534 -0.68556458 -0.13618219 -1.11855275 -0.36147345

-0.11509634 -0.1438043 0.50064044 0.46557244 -0.7572294 -0.83374389

-0.80207282 -0.76562394 -0.80431253 -0.04481144 0.13148396 -0.30814729

-0.96978795 -0.00258113 0.5199767 0.39746306 -0.90493394 -0.39760826

-0.25189964 -0.97259283 -0.27938852 -0.36333028 -0.08964824 -0.0262306

0.60522417 -1.52344785 0.1650713 -0.53089281 -0.80208892 -1.00877817

0.0249936 -0.0088516 -0.30003439 -0.51337886 -0.45424556 -0.30866586

0.02391507 -0.4068039 0.94737249 -0.22108305 0.6559244 -0.97229012

-0.43736071 -0.15624541 0.15248095 0.32432245 0.13088012 -0.05821475

-0.2493025 -0.2644763 -0.18685165 -0.12717106 -0.04985173 -0.00438221

0.0308367 -0.04217135 -0.19172402 0.00394091 -0.09513826 -0.06762565

-0.04497839 -0.02686318 -0.05743514 -0.08012644 -0.17765226 -0.18884754

-0.79249134 -0.11143035 -0.10074021 -0.02419776 -0.23831788 -0.10342438

-0.16708397 -0.04304236 -0.18149769 -0.24347408 -0.07003352 -0.17858007

-0.10383786 0.04499836 -0.04165681 0.0046202 -0.28936746 -0.11486311

-0.08647986 0.00820767 0. -0.02094322]

WadaD

[ -1.93301878e+00 -8.22069160e-01 -1.50623377e-01 2.21964943e-01

5.58224335e-02 1.43520181e-01 -4.22363663e-01 1.99380631e-01

-2.57894059e-02 -2.48824398e-01 -2.53983134e-01 -4.42602301e-03

-5.31686084e-03 -4.62733686e-01 -2.64850676e-01 -9.87675225e-02

-1.56456098e-01 -2.50003887e-01 -4.44095985e-02 -3.18571748e-01

-5.21504136e-01 -3.95935610e-01 4.05037027e-01 -3.10012916e-01

-2.83906868e-01 -3.47894252e-01 -5.40189804e-01 -2.81681461e-01

2.04421966e-01 -1.72011489e-01 -4.36850033e-01 2.31007528e-01

-3.21285011e-01 -1.14535831e-01 -1.40203776e+00 -3.95935610e-01

-3.18571748e-01 -3.47290249e-01 6.61504155e-01 6.20417966e-01

-1.07199939e+00 -1.36012398e+00 -7.45506057e-01 -7.27903280e-01

-3.83100305e-01 -8.29051394e-03 -7.06408920e-02 -3.62578149e-01

-1.10556563e+00 7.70965481e-03 4.25930337e-01 3.66465979e-01

-7.08133275e-01 -6.90661214e-01 -4.90801876e-01 -6.69086397e-01

-6.31859511e-01 -1.78591296e-01 -1.76555845e-01 -4.34405001e-03

3.54964393e-01 -2.00721167e+00 -4.85860566e-02 -8.33797041e-01

-8.26960094e-01 -1.30158342e+00 1.46006450e-01 -1.48443437e-01

-2.14071126e-01 -1.95666110e-01 -4.54138991e-01 -4.04106611e-01

5.44130083e-02 -5.56598025e-01 9.18375266e-01 -1.00633234e-01

3.69086759e-01 -1.19147221e+00 -4.08481649e-01 -1.97241388e-01

3.32096966e-02 3.12882675e-01 3.81104155e-01 3.73389777e-04

-2.25347060e-02 -8.81532163e-02 -3.36571496e-02 -6.35512962e-02

-8.90250630e-03 -1.94607422e-02 4.89026149e-03 -2.41266214e-03

-5.70758625e-02 -3.34582680e-03 -3.59101963e-02 -1.75045366e-02

-1.77858969e-02 -3.38318735e-02 -8.66374196e-03 -6.71066905e-02

-5.86298769e-02 -5.07494328e-02 -5.18428236e-01 -2.79340937e-02

-2.65795487e-02 -1.30535484e-02 -3.19224802e-02 -2.22287244e-02

-2.38590328e-02 -2.11147899e-02 -1.89635669e-02 -6.55907212e-02

4.47586529e-03 -5.03438219e-02 -2.47549481e-02 -3.06521134e-02

-1.38838787e-02 -5.68087788e-03 -8.13534106e-02 -2.83939985e-02

-4.55989902e-02 -1.25665784e-02 -4.35707620e-03 3.86378866e-01]

Wadam

[ -1.20026577e+00 -4.34419192e-01 9.31710071e-02 1.60540845e-01

2.62459460e-01 3.22875554e-02 -4.41439646e-01 4.00001489e-01

1.60270304e-01 -1.81587643e-01 -7.12675760e-02 -1.81613357e-01

-5.61797400e-02 -1.96394814e-01 -6.95565191e-02 -3.04551707e-02

-7.26670247e-02 -3.38574231e-02 7.30497811e-02 -5.66624843e-02

-6.30998106e-01 -3.14890421e-01 6.27236642e-01 5.52475453e-02

4.30469746e-02 -6.04143513e-01 -6.19247909e-01 -2.71210834e-01

2.88509779e-01 -4.82257774e-01 -5.35280778e-01 6.21063532e-01

-6.01504347e-01 -3.03745534e-01 -1.15400909e+00 -3.14890421e-01

-5.66624843e-02 9.13432799e-02 5.26945625e-01 5.60999311e-01

-8.05015310e-01 -7.92518272e-01 -4.47851587e-01 -3.41726176e-01

-2.36271968e-01 -2.37051222e-02 5.28264624e-01 -2.09248413e-01

-7.70265393e-01 -8.54555047e-02 6.20435054e-01 5.13326139e-01

-5.61362097e-01 -3.18653598e-01 -2.29076015e-01 -8.19403516e-01

-2.76591045e-01 -4.16824059e-01 5.89768015e-02 -6.34287906e-02

9.65894996e-01 -1.16886101e+00 2.66456072e-01 -4.14731570e-01

-9.09446989e-01 -8.24324700e-01 -8.95547827e-02 9.17395677e-02

-6.83618747e-01 -4.04476046e-01 -1.74506492e-01 -4.03205347e-01

3.01065967e-02 -5.62699471e-01 1.06668136e+00 -3.24746570e-01

6.61355038e-01 -8.92880390e-01 -2.28512025e-01 -3.41779065e-02

7.23902956e-02 2.70438989e-01 -6.23312539e-02 -8.58844935e-02

6.44518846e-02 -2.14056197e-01 1.71978846e-01 -2.42882785e-02

-2.06728940e-01 1.84924738e-01 9.11472183e-02 -1.38456899e-01

-2.28061562e-01 -6.61182369e-02 5.18882273e-02 -1.79641547e-01

-1.51368597e-01 9.28019069e-02 -4.99751890e-02 -5.36254448e-02

-1.34853200e-01 -3.26922197e-01 -2.52209145e-01 -2.41363463e-01

-1.51189898e-01 1.03672090e-01 -4.61736839e-01 -5.51203144e-02

-1.25472601e-01 1.76314993e-01 -1.86640179e-01 -3.54792597e-01

-1.24199249e-01 -2.74168565e-01 -4.35679661e-01 -1.09275927e-03

-1.65536902e-01 3.13813306e-02 -4.97760585e-01 -2.58871332e-01

-2.53485030e-01 2.77617978e-02 0.00000000e+00 -1.46129100e-01]

**SGD模型的最终正确率： 0.8474909403599288**

**NAG模型的最终正确率： 0.8454640378355138**

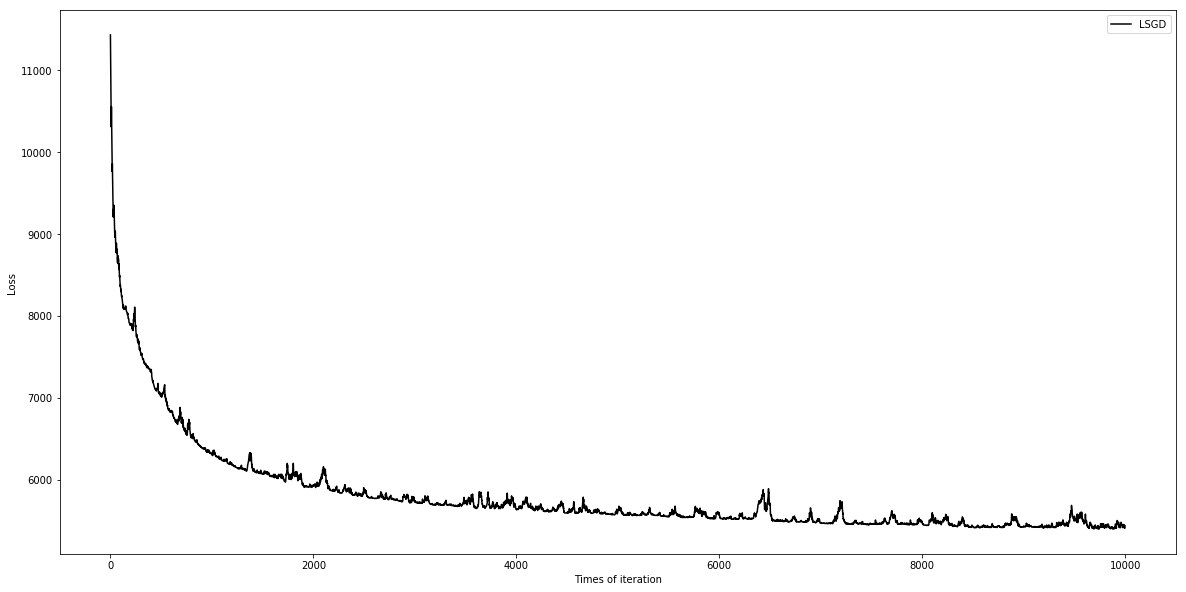
**RMSProp模型的最终正确率： 0.849640685461581**

**AdaDelta模型的最终正确率： 0.8434371353110989**

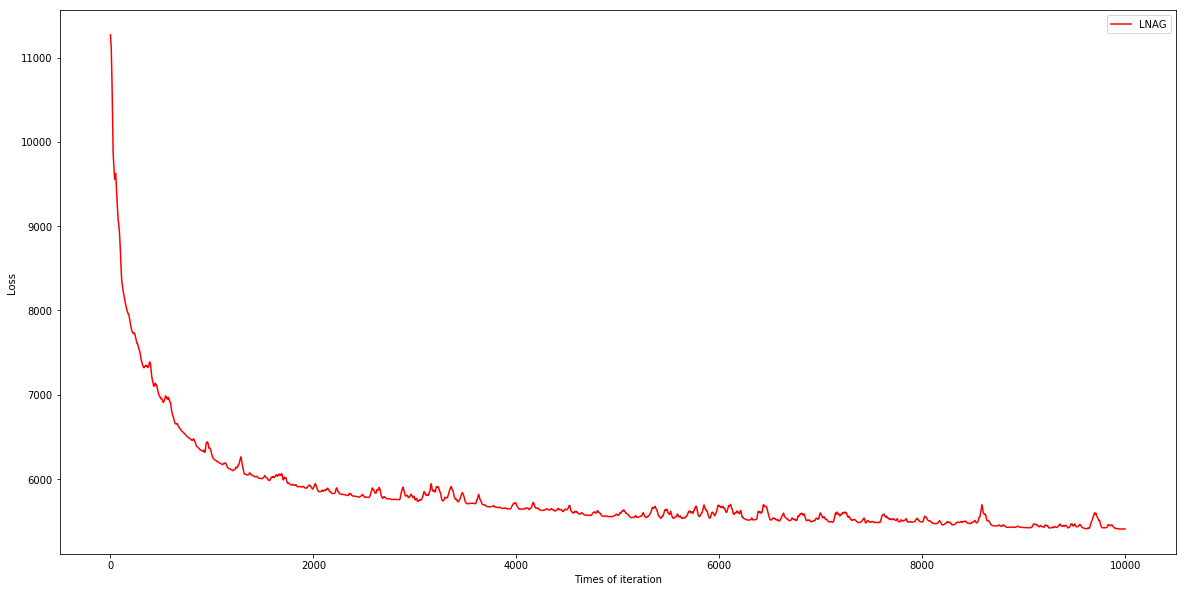
**Adam模型的最终正确率： 0.8486579448436828**

## loss曲线图：

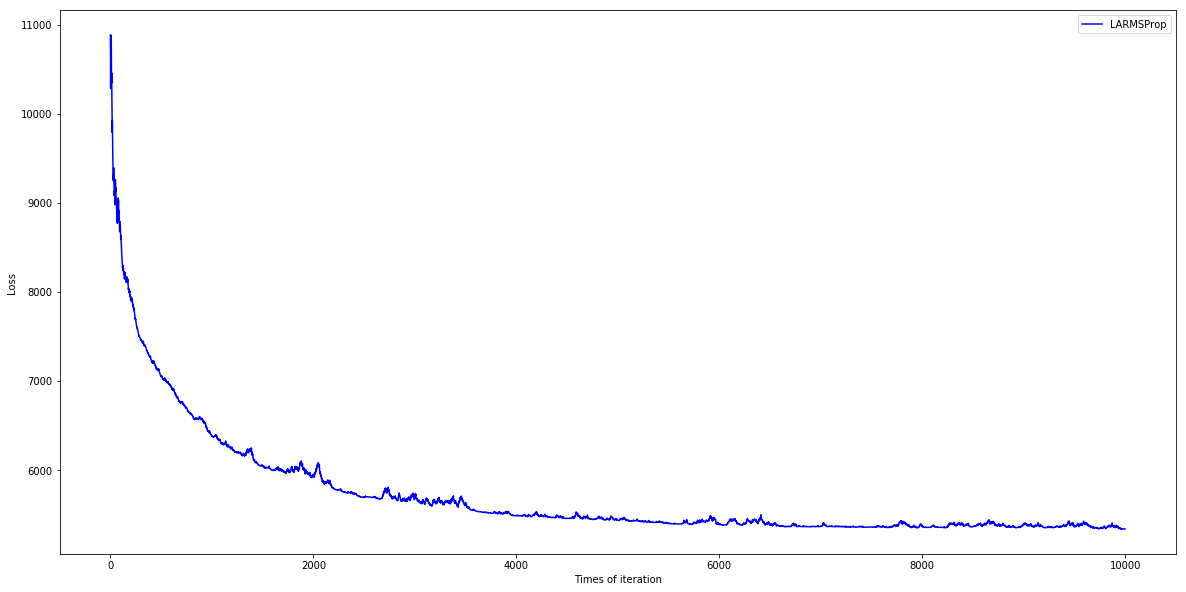
**SGD**

****

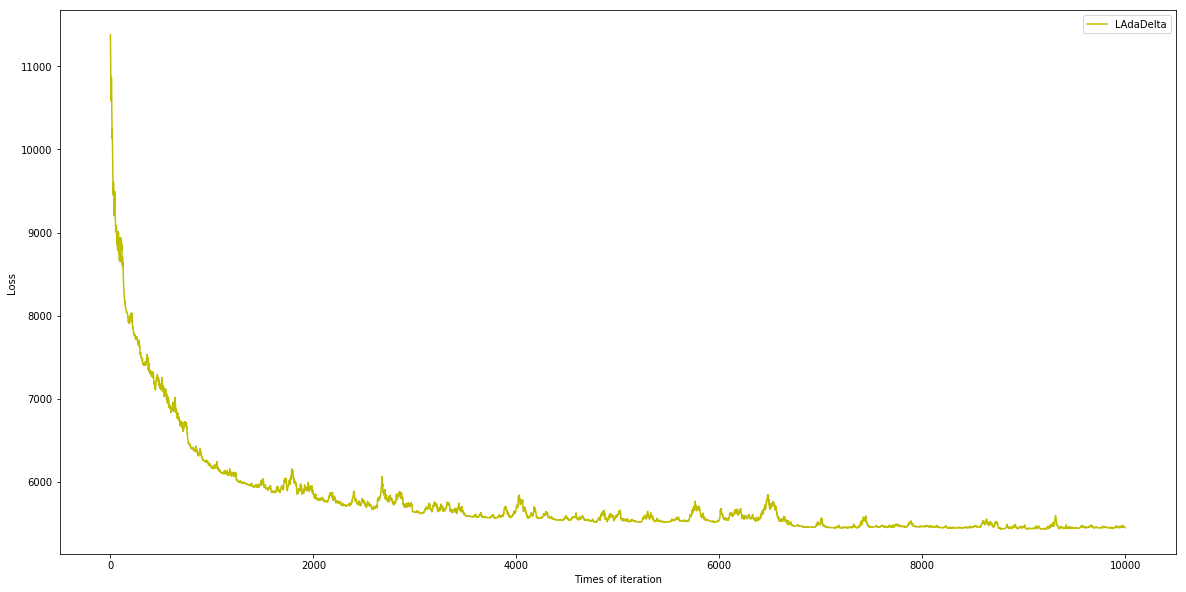
**NAG**

****

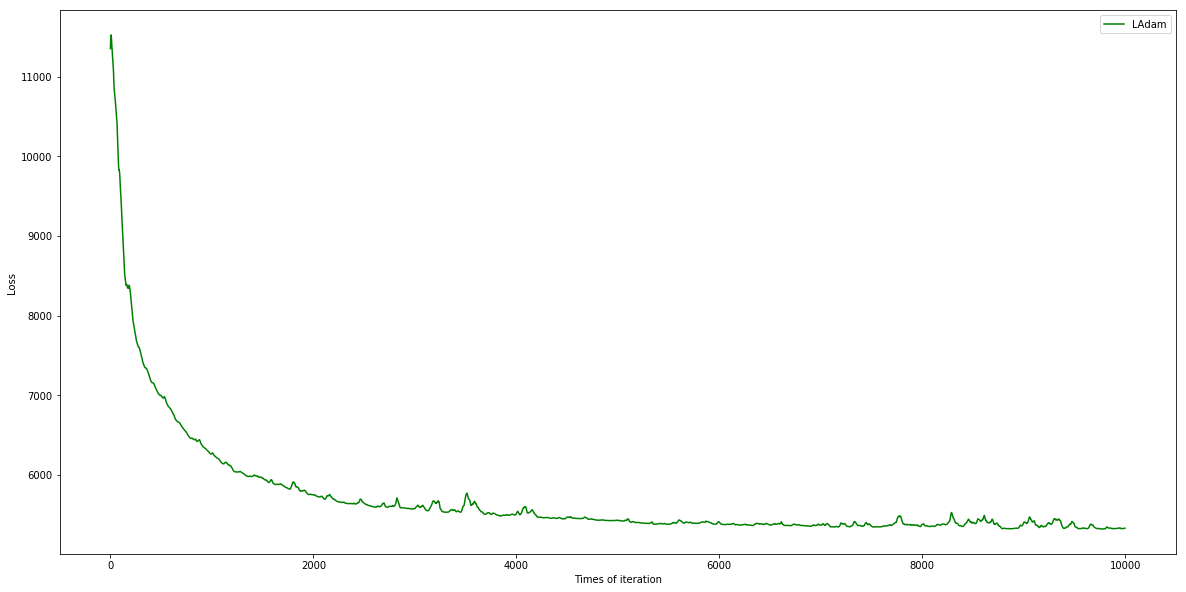
**RMSProp**

****

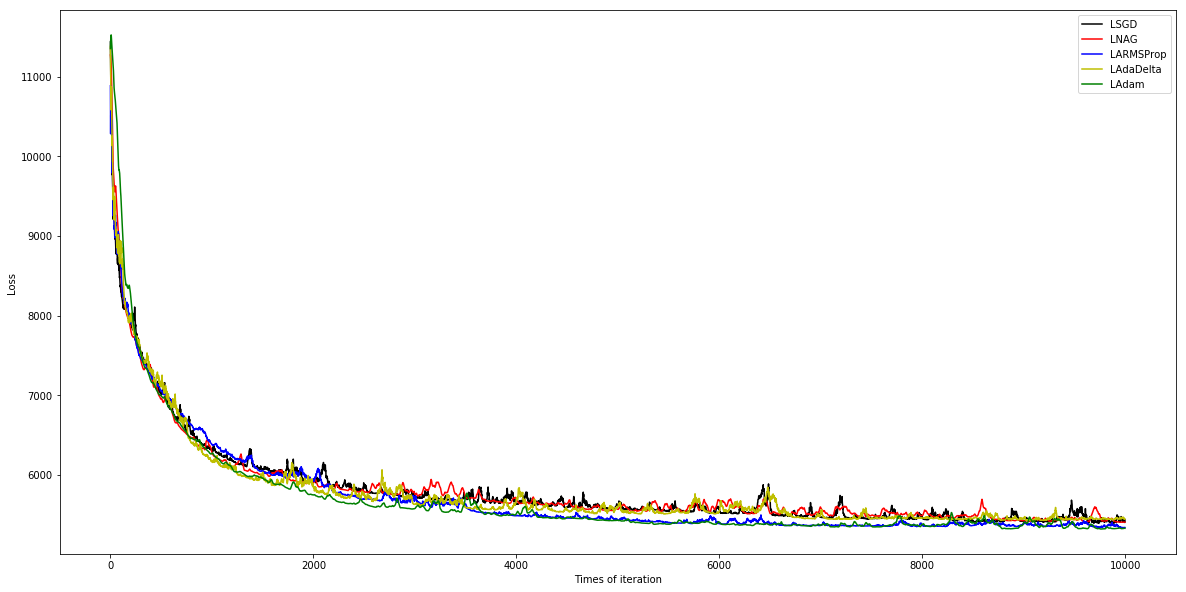
**AdaDelta**

****

**Adam**

****

**总图：**

****

## 11.实验结果分析:

5种优化模型均在前200次迭代内就能使测试集上的Loss值大幅下降，然后在200-2000次迭代内Loss值的降幅逐渐降低，在2000次迭代后Loss值达到6000以下，在2000-10000次迭代内Loss值缓慢降到5500左右，此时在测试集上对应的正确率均可达84%以上。

5种优化模型在经过多次调参以后曲线趋于一致。

以上5种优化模型的曲线在整个迭代中均有不同程度的上下波动,推测是因为Loss函数没有对样本取平均数的原因。

## 线性分类和随机梯度下降8-11：

## 8.模型参数的初始化方法:

全零初始化。

## 9.选择的loss函数及其导数:

Loss函数：

梯度：对wi的梯度

## 10.实验结果和曲线图:

## 超参数选择（η,epoch等）：

SGD的学习率η取0.001

NAG的学习率η取0.0001,γ取0.9

RMSProp的学习率η取0.001,γ取0.9,ε取1e-8

AdaDelta的γ取0.8,ε取1e-6

Adam的学习率η取0.001,γ取0.999,ε取1e-8,β取0.9

预测结果（最佳结果）：

**Wsgd**

[-0.149 -0.127 -0.013 0.085 0.042 -0.01 -0.114 0.056 0.031 -0.034

-0.003 0. 0. -0.052 -0.017 -0.053 -0.011 -0.029 0.096 -0.058

-0.043 -0.142 0.078 -0.029 -0.005 -0.04 -0.059 -0.02 0.096 -0.021

-0.046 0.061 -0.025 -0.005 -0.259 -0.142 -0.058 -0.034 0.331 0.236

-0.096 -0.218 -0.041 -0.016 -0.03 0.003 0.035 -0.099 -0.092 0.051

0.198 0.139 -0.055 -0.047 -0.05 -0.094 -0.059 -0.007 0.006 0.

0.089 -0.092 0.161 -0.157 -0.042 -0.121 -0.023 -0.045 -0.018 -0.016

-0.06 -0.161 -0.001 -0.428 0.266 -0.236 0.074 -0.187 -0.048 -0.085

0.052 0.106 0.007 -0.002 0.003 -0.005 0.002 -0.002 -0.001 -0.006

0.001 0. -0.011 -0.011 0.001 0. -0.001 -0.005 -0.001 -0.005

-0.004 -0.002 -0.052 -0.001 -0.001 0.002 -0.001 -0.002 -0.001 -0.005

-0.007 0.002 -0.001 -0.006 -0.003 -0.002 -0.004 -0.003 -0.005 -0.001

-0.004 0.001 0. -0.162]

**Wnag**

[ -1.38000000e-01 -1.26109933e-01 -8.21258713e-03 5.20340586e-02

4.93587673e-02 -1.33309106e-02 -8.59990674e-02 1.00220123e-01

2.09750370e-02 -3.79123226e-02 -3.80041305e-02 0.00000000e+00

0.00000000e+00 -8.89961833e-02 -3.93767342e-03 7.16594019e-03

-4.29376837e-02 -4.22240940e-02 7.70367965e-02 -7.48064084e-02

-3.80000000e-02 -1.31133072e-01 9.49833891e-02 -1.90000019e-02

-2.10182463e-02 -3.80000000e-02 -6.00000000e-02 -1.20000000e-02

7.31621163e-02 -1.10000000e-02 -4.69999985e-02 6.88457308e-02

-2.19999999e-02 -1.10000000e-02 -2.38999998e-01 -1.31133072e-01

-7.48064084e-02 -4.00182482e-02 3.14028033e-01 2.15899610e-01

-1.07809996e-01 -1.97018801e-01 -3.80005075e-02 -1.99999993e-02

-2.80000000e-02 4.00000000e-03 1.90000000e-02 -7.64980999e-02

-8.69903024e-02 1.49944732e-02 2.15418073e-01 1.17091233e-01

-4.60001024e-02 -7.50550105e-02 -2.90220503e-02 -6.29912720e-02

-3.99981926e-02 -5.00000000e-03 2.99998064e-03 -2.00000000e-03

9.30000000e-02 -9.89997815e-02 1.26899610e-01 -1.48829527e-01

-3.59999993e-02 -1.06999996e-01 -2.29189609e-02 -1.90000005e-02

-2.99999314e-02 -1.70000000e-02 -8.20108013e-02 -1.71719901e-01

7.90206365e-04 -4.26311927e-01 2.55382233e-01 -2.78901873e-01

1.07972179e-01 -1.88150092e-01 -3.70107762e-02 -8.01509259e-02

1.98547491e-02 1.14527351e-01 -2.59297499e-02 3.00000000e-03

-1.00000000e-03 -1.30000000e-02 1.00000000e-03 -3.00000000e-03

-1.00000000e-03 2.00000000e-03 -3.00000000e-03 -4.00000000e-03

-3.00000000e-03 -5.00000000e-03 -6.25267648e-14 -3.00000000e-03

0.00000000e+00 -9.99999261e-04 4.00000000e-03 1.30477769e-18

-3.00000000e-03 -8.00000125e-03 -4.19999999e-02 -5.99995952e-03

-2.00000000e-03 4.00000000e-03 -6.00000000e-03 -2.00000000e-03

-2.00000000e-03 3.00000000e-03 -2.00000000e-03 -4.00000000e-03

4.37415955e-19 -1.00000000e-03 -2.00000000e-03 1.00000000e-03

-3.00000000e-03 -2.00000000e-03 -7.99998432e-03 -4.36967988e-19

-1.00000000e-03 1.00000000e-03 0.00000000e+00 -1.70929694e-01]

**Wrmsp**

[ -2.99383930e-01 -2.84098404e-01 -6.89589379e-02 1.83946367e-01

5.76690129e-02 -2.44741889e-02 -1.59013392e-01 7.11326135e-02

1.10271269e-01 -9.13477551e-03 2.43600303e-02 0.00000000e+00

0.00000000e+00 -6.63904709e-02 -5.72301254e-02 -4.28200040e-02

-6.36947350e-02 1.22561057e-02 8.39385030e-02 -5.58275412e-02

-7.84097978e-02 -2.22527978e-01 2.13421105e-01 -6.69627354e-02

-4.38161938e-02 -8.53866906e-02 -1.23854520e-01 -3.31392054e-02

1.96776137e-01 -2.84578095e-02 -7.52290038e-02 1.53755474e-01

-6.29174909e-02 -1.58113234e-02 -4.59625468e-01 -2.22527978e-01

-5.58275412e-02 -1.05582705e-01 4.63500584e-01 3.15430020e-01

-2.47960653e-01 -3.92014359e-01 -9.27198929e-02 -6.91843737e-02

-4.26492736e-02 1.26491100e-02 8.85211710e-02 -1.00082968e-01

-1.79069029e-01 2.02805102e-02 3.53472866e-01 2.26066656e-01

-1.02867880e-01 -1.12970510e-01 -5.37215076e-02 -1.86006645e-01

-1.47512786e-01 -9.48683251e-03 3.05481549e-02 0.00000000e+00

2.41649828e-01 -2.05811171e-01 2.18013027e-01 -3.23183128e-01

-7.00198067e-02 -2.77257626e-01 -1.23178695e-02 -5.83522399e-02

-5.72060447e-02 -5.63209547e-02 -7.04390611e-02 -2.15024812e-01

-1.08862824e-02 -4.05669144e-01 4.68337415e-01 -2.54653080e-01

2.41498860e-01 -3.22635433e-01 -7.66103678e-02 -1.12078339e-01

1.08345631e-01 1.25511919e-01 -1.15659566e-02 9.48683251e-03

-2.58405262e-04 -3.16227750e-03 -2.18508258e-03 1.26329061e-02

-6.32455500e-03 3.13915458e-03 6.32455498e-03 -6.32455500e-03

-1.29758005e-02 -1.59179446e-02 -1.86643329e-02 6.32455500e-03

6.32455500e-03 -6.00421627e-03 7.03684323e-03 -9.48090868e-03

-6.42201555e-03 -1.58113627e-02 -1.18297457e-01 -1.26491100e-02

-6.32455500e-03 3.16227750e-03 -2.52947180e-02 -3.16227750e-03

-3.16227750e-03 -1.26486363e-02 9.48683251e-03 -1.89731386e-02

-6.32455500e-03 -3.16227750e-03 -6.32455500e-03 6.32455500e-03

3.16227750e-03 0.00000000e+00 1.26491100e-02 -3.16227750e-03

-9.48683250e-03 6.32455500e-03 0.00000000e+00 -1.27159926e-01]

**WadaD**

[ -2.44260589e-01 -1.83161152e-01 -2.25308127e-03 1.92061989e-01

1.39267671e-01 7.02386942e-02 -1.85383150e-01 1.46685222e-01

1.01906139e-01 -7.54929972e-02 -4.28948378e-02 -2.23606239e-03

0.00000000e+00 -7.78591925e-02 -3.48104970e-03 -7.53807326e-03

1.83289811e-02 -3.91947410e-02 4.80703110e-02 -8.86604057e-02

-7.40368468e-02 -1.56974609e-01 1.57949008e-01 1.59002065e-02

-6.55541261e-03 -6.05460337e-02 -8.52912205e-02 -9.00325464e-03

1.62851722e-01 -2.90688110e-02 -7.17368366e-02 1.32242056e-01

-4.02551949e-02 -6.70818716e-03 -3.80613293e-01 -1.56974609e-01

-8.86604057e-02 8.95807062e-03 5.34486041e-01 4.72423025e-01

-8.78589474e-02 -3.11295509e-01 -7.41312269e-02 -2.04363672e-02

-3.14860024e-02 2.23606239e-03 7.85252283e-02 -7.38829016e-02

-1.17848842e-01 5.67582073e-03 3.78516103e-01 3.08302358e-01

-9.43562934e-02 -1.55667951e-01 -1.78399029e-02 -1.34189760e-01

-9.43648968e-02 -2.01245615e-02 -4.64199345e-03 2.23606239e-03

2.07715917e-01 -1.35784843e-01 2.28958546e-01 -2.00114741e-01

-4.92470612e-02 -1.40469167e-01 5.75164128e-02 -5.85644938e-02

-4.71722213e-02 -3.12829631e-02 -5.78087396e-02 -1.70365557e-01

4.41081638e-02 -6.78266801e-01 4.17237216e-01 -4.24147661e-01

1.90400609e-01 -2.52109387e-01 -6.57039514e-02 -6.80602895e-02

9.05458701e-02 1.89524718e-01 1.32969860e-01 2.23606239e-03

-6.70819341e-03 -8.94421240e-03 1.12760793e-02 2.23606238e-03

-6.70818716e-03 4.60273267e-03 1.11802748e-02 -8.94629590e-03

2.35483699e-11 -1.56547281e-02 2.23601584e-03 -2.23606239e-03

-2.23606239e-03 -8.94408784e-03 4.46876508e-03 -8.94424954e-03

-8.94424871e-03 -2.01245665e-02 -8.07686925e-02 -4.47212477e-03

-6.70818716e-03 8.94468136e-03 -6.70818716e-03 -4.47212477e-03

-2.23606239e-03 -8.98405163e-03 -2.23606239e-03 -6.70818716e-03

2.23606239e-03 0.00000000e+00 0.00000000e+00 -4.47212477e-03

-4.47212477e-03 0.00000000e+00 4.55490680e-14 0.00000000e+00

-2.23606239e-03 0.00000000e+00 0.00000000e+00 -2.06759733e-01]

**Wadam**

[-0.43362141 -0.25185534 0.00522976 0.20775879 0.03757524 0.03157098

-0.10621883 0.31036717 0.1594781 -0.15353226 -0.12419719 -0.03167142

0. -0.14301079 0.02294739 0.0101441 -0.03154773 0.00080182

0.09950376 -0.10814786 -0.1963368 -0.28165565 0.48420591 -0.08924095

0.03228623 -0.22869068 -0.27957517 -0.13145875 0.23910102 -0.14645872

-0.17299829 0.44203518 -0.11318398 -0.1317031 -0.55541688 -0.28165565

-0.10814786 -0.03422325 0.43944449 0.31766922 -0.36604805 -0.48230272

-0.16123007 -0.22674538 -0.12144972 0.06250157 0.12234226 -0.22910594

-0.40676797 0.10786053 0.44125944 0.32153005 -0.27476156 -0.24333881

0.01625686 -0.32091362 -0.24008613 -0.05895082 0.00348435 0.

0.38298192 -0.39954886 0.20220438 -0.41291548 -0.18617428 -0.30506948

-0.01605944 -0.0022987 -0.11394047 -0.1314358 -0.10344381 -0.18677898

0.02465051 -0.34914035 0.67962159 -0.18282905 0.36871246 -0.44829862

-0.15940354 -0.05006328 0.06478685 0.20380812 0.01542436 -0.1061737

-0.05126307 -0.07450306 -0.05639253 0.00441159 0. -0.06191248

0.12917423 -0.03877865 -0.05299144 0.00081542 -0.11382374 0.1111535

0. 0.19619232 0.01573409 0.03373661 0.03754171 -0.14388357

-0.27722543 -0.00662018 0.03155485 0.02315502 -0.06008812 -0.00172659

-0.05163124 -0.06754389 0.02058178 -0.1327663 -0.01246329 -0.00437086

-0.02100945 0. -0.03045058 -0.0125085 -0.10652573 -0.03413585

-0.02067143 0.02424852 0. -0.07121325]

**SGD模型的最终正确率： 0.8263005957864996**

**NAG模型的最终正确率： 0.8110066949204594**

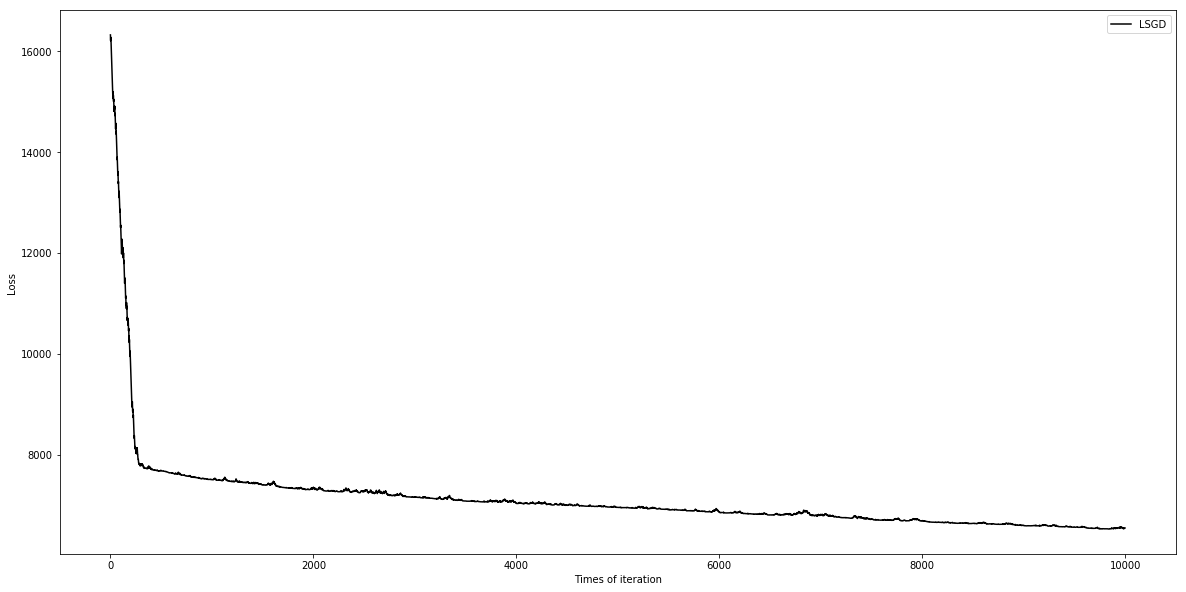
**RMSProp模型的最终正确率： 0.840366070880167**

**AdaDelta模型的最终正确率： 0.8417173392297771**

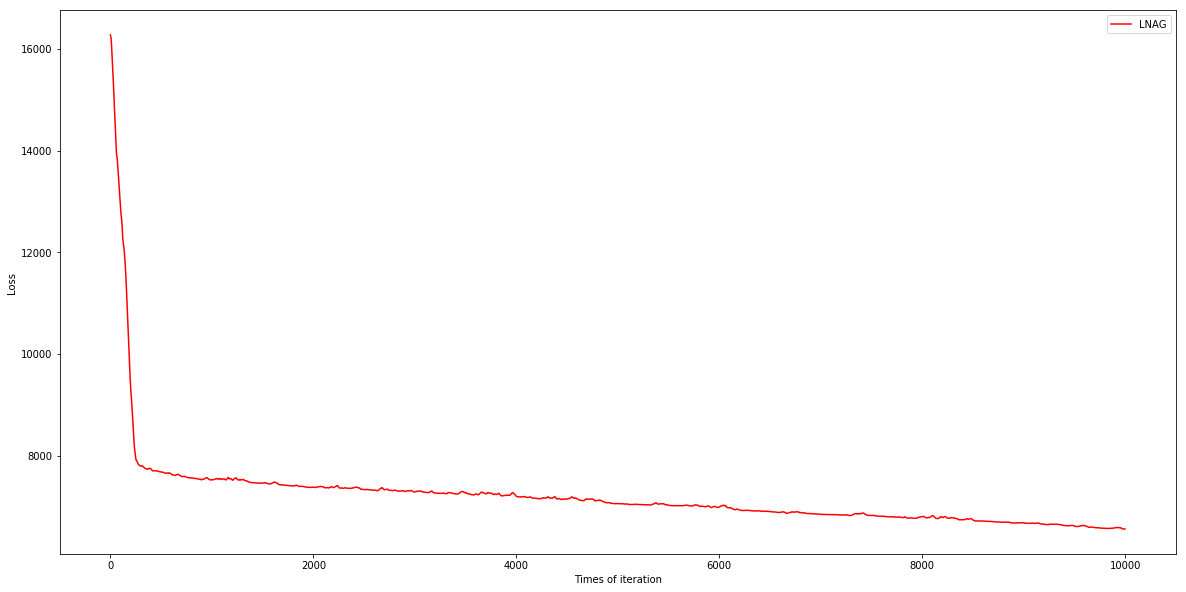
**Adam模型的最终正确率： 0.8458939868558443**

## loss曲线图：

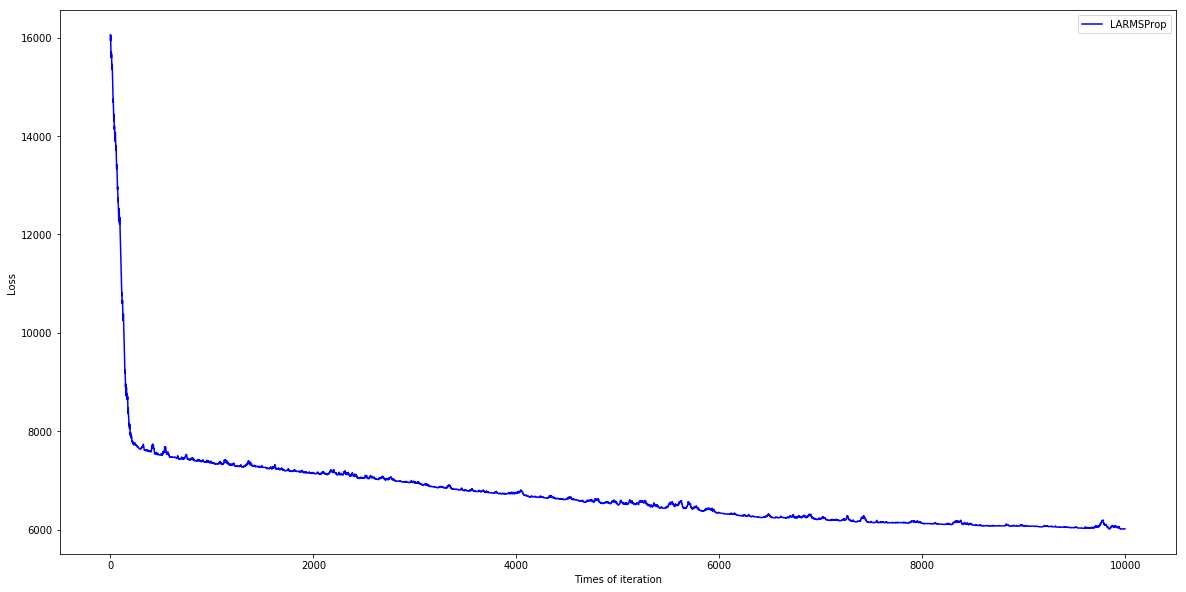
**SGD**

****

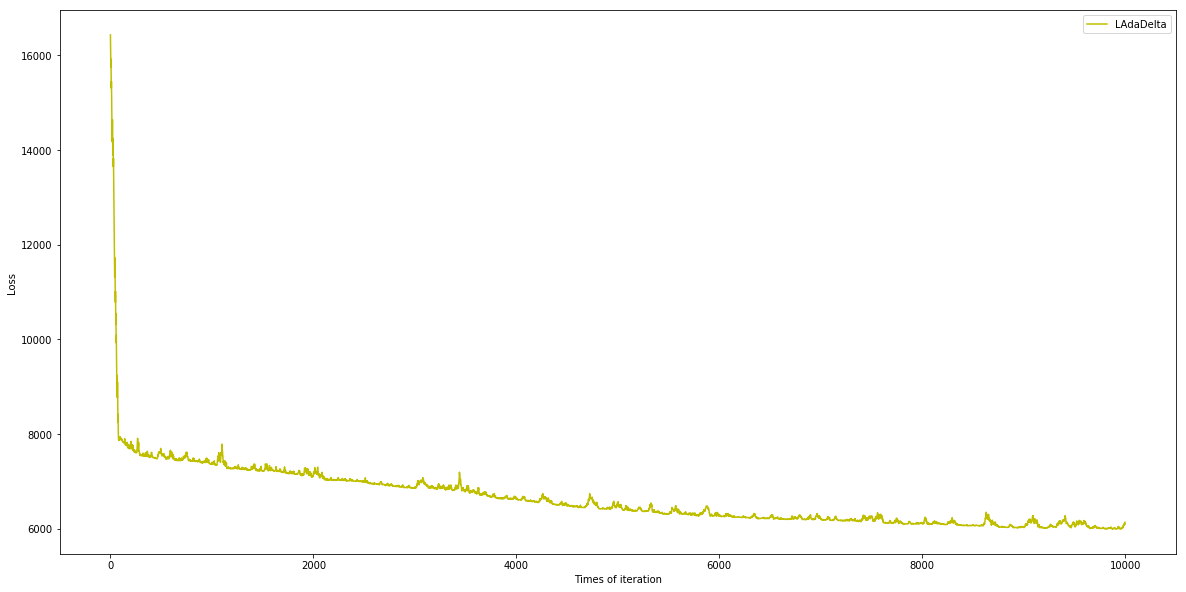
**NAG**

****

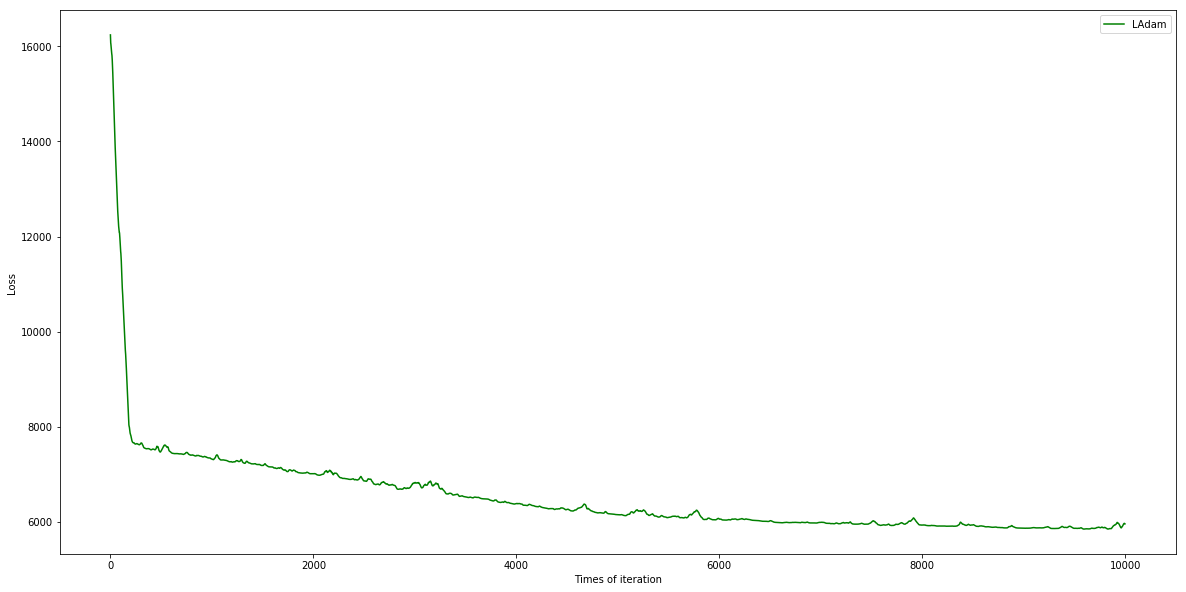
**RMSProp**

****

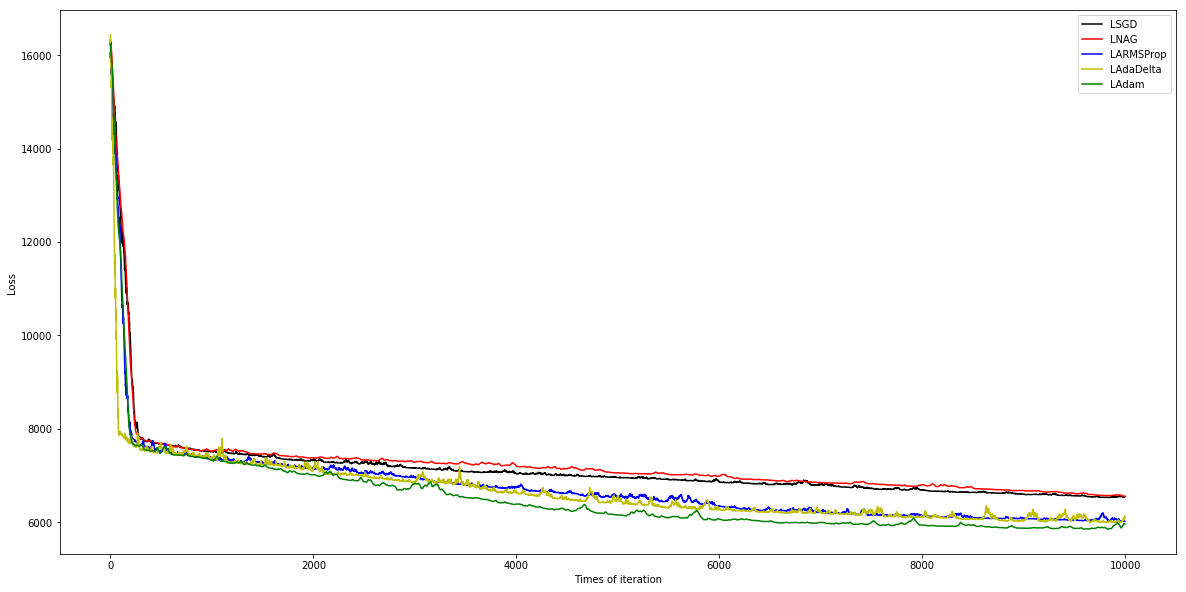
**AdaDelta**

****

**Adam**

****

**总图：**

****

## 11.实验结果分析:

5种优化模型均在前200次迭代内就能使测试集上的Loss值大幅下降，然后在200-10000次迭代内Loss值缓慢地下降，最终达到6000左右，此时在测试集上对应的正确率均可达80%以上。

普通SGD和NAG模型的Loss曲线非常相近，最后的准确率也很相近，而比另外三种模型低一些。

RMSProp，AdaDelta和Adam模型的曲线在迭代后半段能够达到更低的Loss值，其中又以Adam的效果最优。

以上5种优化模型的曲线在整个迭代中均有不同程度的上下波动,推测是因为Loss函数没有对样本取平均数的原因。

## 12.对比逻辑回归和线性分类的异同点：

逻辑回归和线性分类最基本的共同点就是通过

的线性公式构造模型，不同的是逻辑回归再对f(x)进行了Sigmod运算。

逻辑回归和线性分类的标签y取值都是有限的几个（本实验为2个）离散值，同一个值可以同时对应超空间中一定区域的x的取值。

从实验结果来看，逻辑回归和线性分类的Loss值大小都始终和数据集样本数成正比，对于不同大小的数据集，具有相同“训练程度”的模型产生不匹配的概率是一定的，所以数据集越大，Loss值越大。

逻辑回归是通过预测函数来进行分类，y的标签值是已知的（0，1），而线性分类是通过拟合函数来进行预测，y的值是未知的。

## 实验总结：

本次实验让我更深地了解了通过数据样本训练特定机器学习模型的过程，不同的优化模型让我意识到了机器学习中模型的重要性，我认为充分运用代码的模块化有助于基本函数和较优化模型间的组合，也方便对超参数进行调整。并且让我更加熟悉了python中矩阵的基本运算的相关函数。