

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**   **苏婉婷**

**学 号 201530612767**

**邮 箱 352191482@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12 月2日**

## 1. 实验题目: 逻辑回归、线性分类与随机梯度下降

## 2. 实验时间：2017年 12 月 2 日

## 3. 报告人:苏婉婷

## 4. 实验目的:

4.1对比理解梯度下降和随机梯度下降的区别与联系。

4.2对比理解逻辑回归和线性分类的区别与联系。

4.3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](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/" \t "https://www.zybuluo.com/chenyaofo/note/_blank)的中的[a9a](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html" \l "a9a" \t "https://www.zybuluo.com/chenyaofo/note/_blank)数据，包含32561 / 16281(testing)个样本，每个样本有123/123 (testing)个属性。

## 6. 实验步骤:

**6.1 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。

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，，和。

7.重复步骤4-6若干次，画出，，和随迭代次数的变化图。

**6.2 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。

4.求得部分样本对Loss函数的梯度。

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，，和

7.重复步骤4-6若干次，画出，，和随迭代次数的变化图。

## 7. 代码内容:

（针对逻辑回归和线性分类分别填写8-11内容）

7.1 逻辑回归：

# -\*- coding: utf-8 -\*-

"""

Created on Wed Dec 13 16:05:39 2017

@author: siye

"""

from sklearn import datasets

from sklearn import model\_selection

from sklearn import linear\_model

import numpy as np

import matplotlib.pyplot as plt

import math

import jupyter

import random

import time

#sigmoid 函数

def sigmoid(Xi,theta):

z=math.exp(Xi.dot(theta.T))

return 1.0/(1+np.exp(-z))

#loss函数： L(Θ)=∑[yi\*log(h(xi))+(1-yi)\*log(1-h(xi))]/(-m)

def loss(X,y,theta):

m=y.shape[0]

loss=0.

for i in range(m):

loss+=(y[i]\*math.log(sigmoid(X[i,:],theta)))+((1-y[i])\*math.log(1-sigmoid(X[i,:],theta)))

loss /= - m

return loss

#求导∂L/∂Θ = (1/m) ∑ (h(Xi) - yi) \* Xi = (1/m) X.T (sigmoid(XΘ)-y)

def gradient(X,y,theta):

m=y.shape[0]

gradient=np.zeros(theta.shape)

for i in range(m):

gradient+=(sigmoid(X[i,:],theta)-y[i])\*(X[i,:])

gradient/=m

return gradient

#使用NAG优化方法更新模型参数

def NAG\_train(X, y, theta, learning\_rate, gamma, iteration):

startTime=time.time() #计算训练时间

test\_loss\_history = np.zeros((iteration, 1))

v = np.zeros(theta.shape)

for iter in range(iteration):

index = random.randint(0, y.shape[0]-10)

theta = theta - gamma \* v

v = gamma \* v - learning\_rate \* gradient(X[index:index+1,:], y[index:index+1], theta)

theta = theta + v

test\_loss\_history[iter] = loss(Xt\_, yt\_, theta)[-1:]

print ('NAG training is completed! Took %fs!'%(time.time()-startTime))

return test\_loss\_history, theta

#使用RMSProp 更新模型参数

def RMSProp\_train(X, y, theta, learning\_rate, gamma, epsilon, iteration):

startTime=time.time()

test\_loss\_history = np.zeros((iteration, 1))

G\_t = 0.

Theta\_gradient = np.zeros(theta.shape)

for iter in range(iteration):

index = random.randint(0, y.shape[0]-10)

Theta\_gradient = gradient(X[index:index+10,:], y[index:index+10], theta)

G\_t = gamma \* G\_t + (1 - gamma) \* Theta\_gradient.dot(Theta\_gradient.T)

theta = theta - (learning\_rate / np.sqrt(G\_t + epsilon)) \* Theta\_gradient

test\_loss\_history[iter] = loss(Xt\_, yt\_, theta)[-1:]

print ('RMSProp training is completed! Took %fs!'%(time.time()-startTime))

return test\_loss\_history, theta

#使用AdaDelta 更新模型

def AdaDelta\_train(X, y, theta, gamma, epsilon, iteration):

startTime=time.time()

test\_loss\_history = np.zeros((iteration, 1))

Theta\_gradient = np.zeros(theta.shape)

G\_t = 0.

delta\_theta = np.zeros(theta.shape)

delta\_t = 0.03

for iter in range(iteration):

index = random.randint(0, y.shape[0]-10)

Theta\_gradient = gradient(X[index:index+10,:], y[index:index+10], theta)

G\_t = gamma \* G\_t + (1 - gamma) \* Theta\_gradient.dot(Theta\_gradient.T)

delta\_theta = - (np.sqrt(delta\_t + epsilon) / np.sqrt(G\_t + epsilon)) \* Theta\_gradient

theta = theta + delta\_theta

delta\_t = gamma \* delta\_t + (1 - gamma) \* (delta\_theta.dot(delta\_theta.T))

test\_loss\_history[iter] = loss(Xt\_, yt\_, theta)[-1:]

print('AdaDelta training is completed! Took %f s!'%(time.time()-startTime))

return test\_loss\_history, theta

#使用Adam 更新模型

def Adam\_train(X, y, theta, learning\_rate, beta1, beta2, epsilon, iteration):

startTime=time.time()

test\_loss\_history = np.zeros((iteration, 1))

Theta\_gradient = np.zeros(theta.shape)

v\_t = 0.

m\_t = np.zeros(theta.shape)

for iter in range(iteration):

index = random.randint(0, y.shape[0]-10)

Theta\_gradient = gradient(X[index:index+10,:], y[index:index+10], theta)

m\_t = beta1 \* m\_t + (1 - beta1) \* Theta\_gradient

v\_t = beta2 \* v\_t + (1 - beta2) \* Theta\_gradient.dot(Theta\_gradient.T)

mt\_estimate = m\_t / (1 - pow(beta1, iter + 1))

vt\_estimate = v\_t / (1 - pow(beta2, iter + 1))

theta = theta - learning\_rate \* mt\_estimate / (np.sqrt(vt\_estimate) + epsilon)

test\_loss\_history[iter] = loss(Xt\_, yt\_, theta)[-1:]

print('Adam training is completed! Took %f s!'%(time.time()-startTime))

return test\_loss\_history, theta

#读取训练集和验证集

X\_train, y\_train = datasets.load\_svmlight\_file("D:/MLExperiment/logisticRegression/a9a.txt")

x\_ = np.array(X\_train.toarray(), np.float32).reshape((-1, 123))

y\_ = np.array(y\_train, np.float32).reshape((-1, 1))

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

if y\_[i,0] == -1.0 : y\_[i,0] = 0.

X\_ = np.hstack([x\_, np.ones((x\_.shape[0], 1))])

X\_test, y\_test = datasets.load\_svmlight\_file("D:/MLExperiment/logisticRegression/a9atest.txt")

xt\_ = np.array(X\_test.toarray(), np.float32).reshape((-1, 122))

yt\_ = np.array(y\_test, np.float32).reshape((-1, 1))

for i in range(yt\_.shape[0]):

if yt\_[i,0] == -1.0 : yt\_[i,0] = 0.

Xt\_ = np.hstack([xt\_, np.zeros((xt\_.shape[0], 1)),np.ones((xt\_.shape[0], 1))])

#设置迭代次数

iteration=500

#超参数设置

be1 = 0.9

be2 = 0.999

ep = 1e-8

t\_nag = np.zeros((1, 124))

t\_rmsprop = np.zeros((1, 124))

t\_adadelta = np.zeros((1, 124))

t\_adam = np.zeros((1, 124))

#NAG

nag\_loss\_history, t\_nag = NAG\_train(X\_, y\_, t\_nag, 0.005, be1, iteration)

#RMSProp

rmsprop\_loss\_history, t\_rmsprop = RMSProp\_train(X\_, y\_, t\_rmsprop, 0.005, be1, ep, iteration)

#Adadelta

adadelta\_loss\_history, t\_adadelta = AdaDelta\_train(X\_, y\_, t\_adadelta, be1, ep, iteration)

#Adam

adam\_loss\_history, t\_adam = Adam\_train(X\_, y\_, t\_adam, 0.005, 0.999, 0.99, ep, iteration)

#绘制graph

plt.plot(nag\_loss\_history, 'g', label='NAG loss')

plt.plot(rmsprop\_loss\_history, 'b', label='RMSProp loss')

plt.plot(adadelta\_loss\_history, 'r', label='AdaDelta loss')

plt.plot(adam\_loss\_history, 'y', label='Adam loss')

plt.legend(loc='upper right')

plt.ylabel('loss');

plt.xlabel('iteration ')

plt.show()

**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:

9.1 逻辑回归：

Loss 函数: 

其中，

导数: 



9.2 线性分类：

loss 函数：

其中，

导数 ：

## 10.实验结果和曲线图:（各种梯度下降方式分别填写此项）

**10.1 逻辑回归**

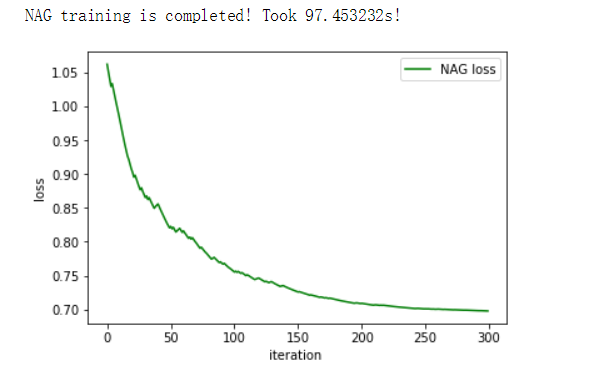
## 10.1.1超参数选择：

1. NAG：η=0.005 =0.9 epoch = 300
2. RMSProp:η=0.005 = 0.9 = 1e-8 epoch =300
3. AdaDelta: = 0.9 = 1e-8 epoch=100

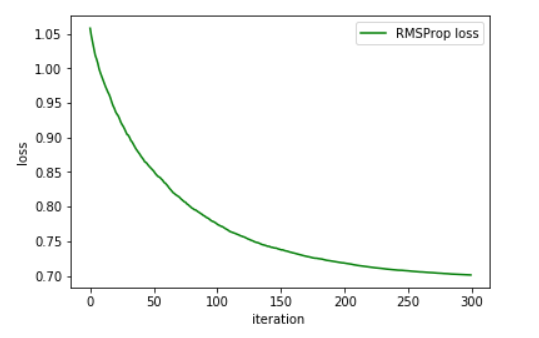
## （4）Adam:η=0.005 = 0.9 = 0.999 = 1e-8 epoch=300

## 10.1.2预测结果（最佳结果）

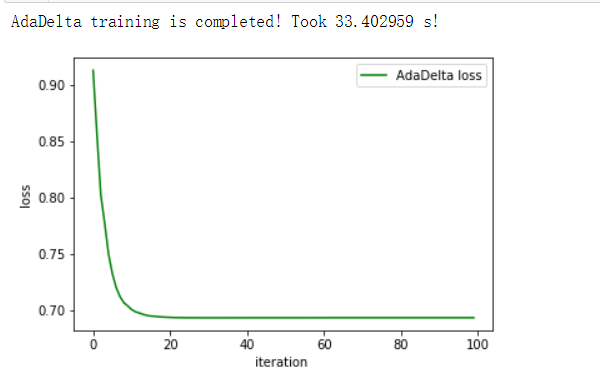
（1）NAG：



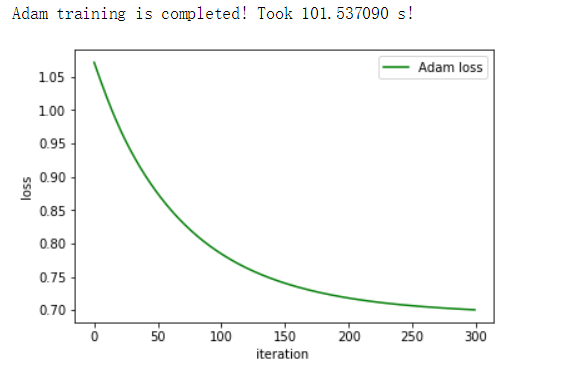
1. RMSProp:



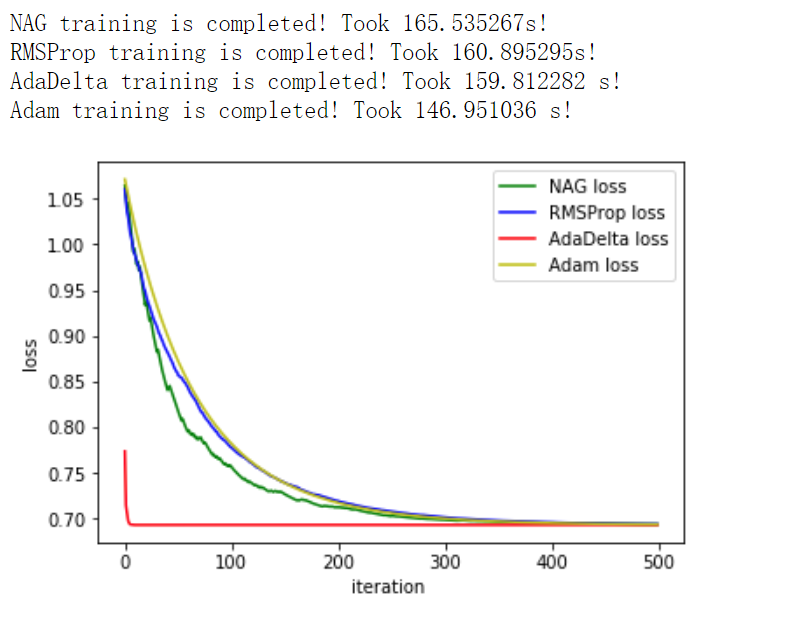
1. AdaDelta：



1. Adam:



**10.1.3 Loss 曲线图：**



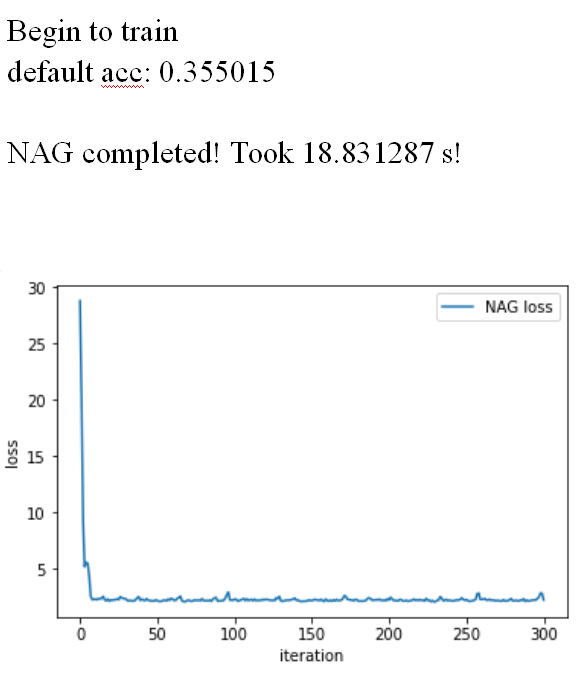
**10.2 线性分类**

## 10.1.1超参数选择

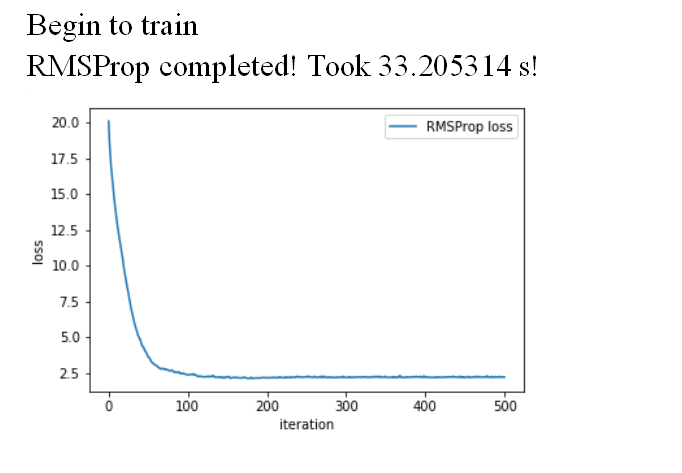
学习率alpha:0.7

## 10.2.2预测结果（最佳结果）

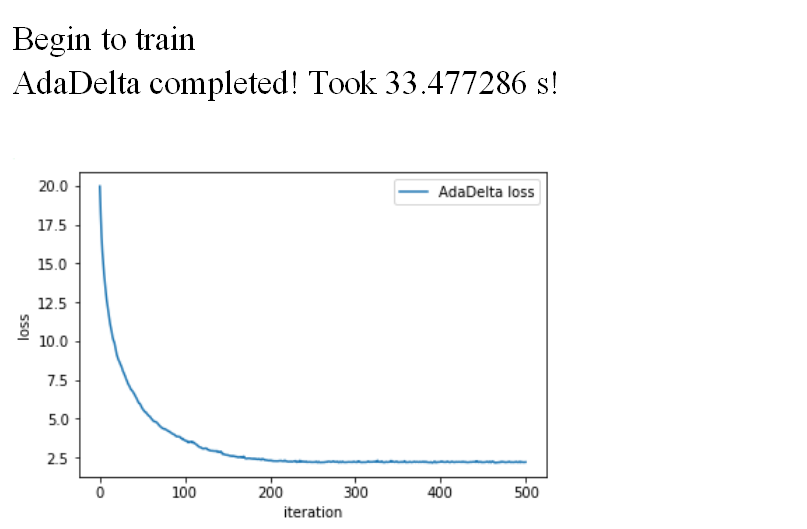
1. NAG:



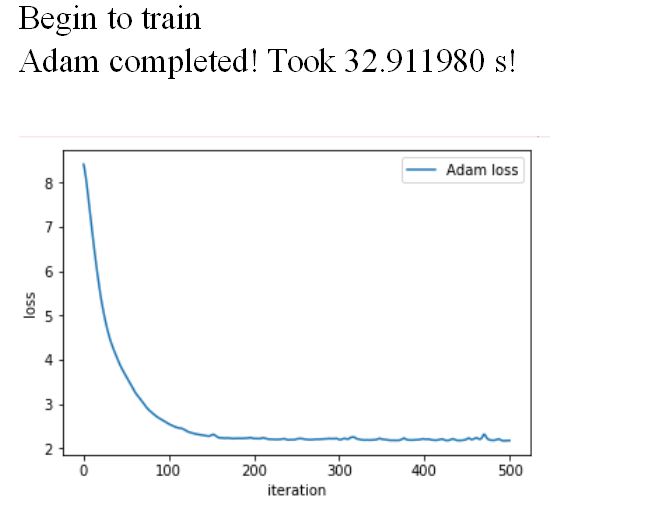
1. RMSProp:



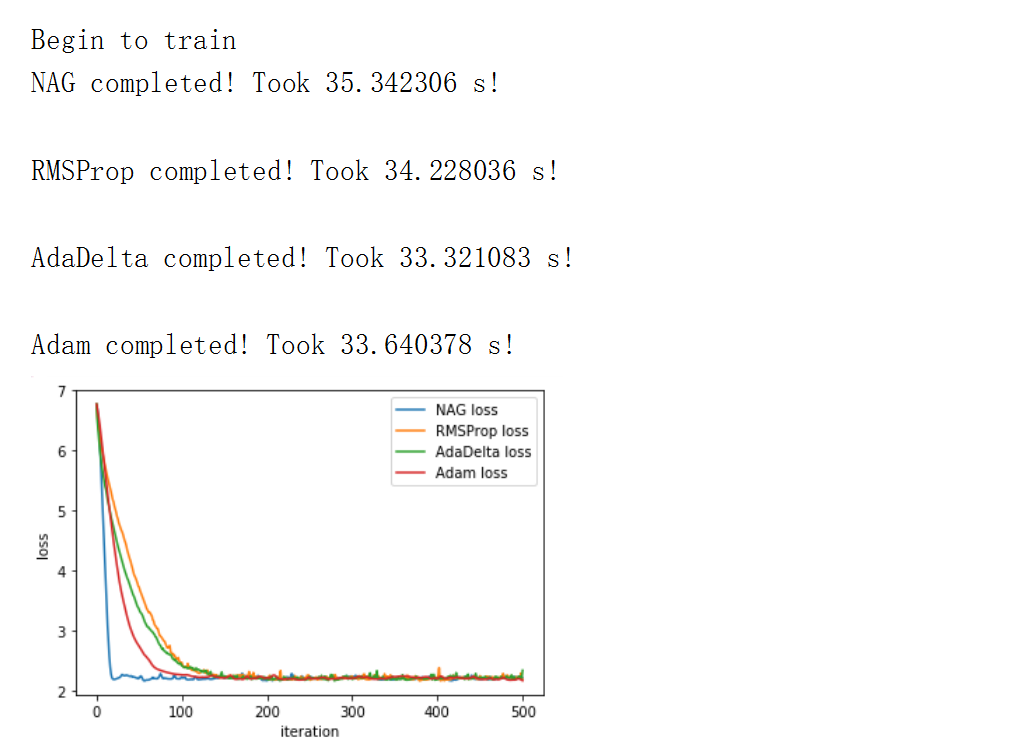
1. AdaDelta:



1. Adam:



10.2.3 Loss 函数



## 11.实验结果分析:

## 12.对比逻辑回归和线性分类的异同点：

逻辑回归和线性回归都属于可以用作分类和预测。logistic回归与多重线性回归最大的区别就在于他们的因变量不同。线性分类器是以超平面为决策边界的分类器. 常见的线性分类器有逻辑回归, SVM, Perceptron。这些分类算法都不是通过线性回归得到自己的分类超平面的。

逻辑回归采用的是logistical loss, svm采用的是hinge loss.SVM的处理方法只考虑了support vector去学习分类器，而逻辑回归通过非线性映射来减小了离分类平面较远的点的权重,提升了与分类最相关的数据点的权重。

## 13.实验总结：