

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  **刘志杰**

**学 号 201530612392**

**邮 箱 791638657@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12月 14 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:刘志杰

## 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。

## 5. 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。

## 6. 实验步骤:

逻辑回归与随机梯度下降

读取实验训练集和验证集。

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。

选择Loss函数及对其求导，过程详见课件ppt。

求得部分样本对Loss函数的梯度G。

使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta和Adam）。

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，L\_NAG，L\_RMSProp,L\_AdaDelta和L\_Adam。

重复步骤4-6若干次，画出L\_NAG，L\_RMSProp,L\_AdaDelta和L\_Adam随迭代次数的变化图。

线性分类与随机梯度下降

读取实验训练集和验证集。

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。

选择Loss函数及对其求导，过程详见课件ppt。

求得部分样本对Loss函数的梯度。

使用不同的优化方法更新模型参数（NAG，RMSProp，AdaDelta和Adam）。

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L\_NAG，L\_RMSProp,L\_AdaDelta和L\_Adam。

重复步骤4-6若干次，画出L\_NAG，L\_RMSProp,L\_AdaDelta和L\_Adam随迭代次数的变化图。

## 7. 代码内容:

逻辑回归部分：

# training nag

for epoch in range(300):

random.seed()

i=randint(0,n-1-batch\_NAG)

g=gradient(x\_train[i:i+batch\_NAG].reshape((batch\_NAG,m)),y\_train[i:i+batch\_NAG].reshape((batch\_NAG,1)),W\_NAG-gamma\_NAG\*v)

v=gamma\_NAG\*v+eta\_NAG\*g

W\_NAG=W\_NAG-v

l\_test=loss(x\_test,y\_test,W\_NAG)

L\_NAG.append(l\_test)

print("L\_NAG")

print(L\_NAG)

# training rmsprop

for epoch in range(300):

random.seed()

i=randint(0,n-1-batch\_RMS)

g=gradient(x\_train[i:i+batch\_RMS].reshape((batch\_RMS,m)),y\_train[i:i+batch\_RMS].reshape((batch\_RMS,1)),W\_RMS)

G=gamma\_RMS\*G+(1-gamma\_RMS)\*(g\*g)

W\_RMS=W\_RMS-eta\_RMS/np.sqrt(G+epsilon\_RMS)\*g

l\_test=loss(x\_test,y\_test,W\_RMS)

L\_RMSProp.append(l\_test)

print("L\_RMSProp")

print(L\_RMSProp)

# training adadelta

for epoch in range(300):

random.seed()

i=randint(0,n-1-batch\_ADA)

g=gradient(x\_train[i:i+batch\_ADA].reshape((batch\_ADA,m)),y\_train[i:i+batch\_ADA].reshape((batch\_ADA,1)),W\_ADA)

GG=gamma\_ADA\*GG+(1-gamma\_ADA)\*g\*g

dw=-np.sqrt(dt+epsilon\_ADA)/np.sqrt(GG+epsilon\_ADA)\*g

W\_ADA=W\_ADA+dw

dt=gamma\_ADA\*dt+(1-gamma\_ADA)\*dw\*dw

l\_test=loss(x\_test,y\_test,W\_ADA)

L\_AdaDelta.append(l\_test)

print("L\_AdaDelta")

print(L\_AdaDelta)

# training adam

for epoch in range(300):

i=randint(0,n-1-batch\_ADAM)

g=gradient(x\_train[i:i+batch\_ADAM].reshape((batch\_ADAM,m)),y\_train[i:i+batch\_ADAM].reshape((batch\_ADAM,1)),W\_ADAM)

M=beta\_ADAM\*M+(1-beta\_ADAM)\*g

G=gamma\_ADAM\*G+(1-gamma\_ADAM)\*g\*g

alpha=eta\_ADAM\*np.sqrt(1-math.pow(gamma\_ADAM,epoch))/(1-beta\_ADAM)

W\_ADAM=W\_ADAM-alpha\*M/np.sqrt(G+epsilon\_ADAM)

l\_test=loss(x\_test,y\_test,W\_ADAM)

L\_Adam.append(l\_test)

print("L\_Adam")

print(L\_Adam)

线性分类部分：

#NAG Training

for epoch in range(300):

random.seed()

i=random.randint(0,n-1-batch\_NAG)

g=gradient(x\_train[i:i+batch\_NAG].reshape((batch\_NAG,m)),y\_train[i:i+batch\_NAG].reshape((batch\_NAG,1)),W\_NAG-gamma\_NAG\*v,C)

v=gamma\_NAG\*v+eta\_NAG\*g

W\_NAG=W\_NAG-v

l\_test=loss(x\_test,y\_test,W\_NAG,C)

L\_NAG.append(l\_test)

print("L\_NAG")

print(L\_NAG)

#RMS Training

for epoch in range(300):

random.seed()

i=random.randint(0,n-1-batch\_RMS)

g=gradient(x\_train[i:i+batch\_RMS].reshape((batch\_RMS,m)),y\_train[i:i+batch\_RMS].reshape((batch\_RMS,1)),W\_RMS,C)

G=gamma\_RMS\*G+(1-gamma\_RMS)\*(g\*g)

W\_RMS=W\_RMS-eta\_RMS/np.sqrt(G+epsilon\_RMS)\*g

l\_test=loss(x\_test,y\_test,W\_RMS,C)

L\_RMSProp.append(l\_test)

print("L\_RMSProp")

print(L\_RMSProp)

#Adadelta training

for epoch in range(300):

random.seed()

i=random.randint(0,n-1-batch\_ADA)

g=gradient(x\_train[i:i+batch\_ADA].reshape((batch\_ADA,m)),y\_train[i:i+batch\_ADA].reshape((batch\_ADA,1)),W\_ADA,C)

G=gamma\_ADA\*G+(1-gamma\_ADA)\*g\*g

dw=-np.sqrt(dt+epsilon\_ADA)/np.sqrt(G+epsilon\_ADA)\*g

W\_ADA=W\_ADA+dw

dt=gamma\_ADA\*dt+(1-gamma\_ADA)\*dw\*dw

l\_test=loss(x\_test,y\_test,W\_ADA,C)

L\_AdaDelta.append(l\_test)

print("L\_AdaDelta")

print(L\_AdaDelta)

#adam training

for epoch in range(300):

i=random.randint(0,n-1-batch\_ADAM)

g=gradient(x\_train[i:i+batch\_ADAM].reshape((batch\_ADAM,m)),y\_train[i:i+batch\_ADAM].reshape((batch\_ADAM,1)),W\_ADAM,C)

M=beta\_ADAM\*M+(1-beta\_ADAM)\*g

G=gamma\_ADAM\*G+(1-gamma\_ADAM)\*g\*g

alpha=eta\_ADAM\*np.sqrt(1-math.pow(gamma\_ADAM,epoch))/(1-beta\_ADAM)

W\_ADAM=W\_ADAM-alpha\*M/np.sqrt(G+epsilon\_ADAM)

l\_test=loss(x\_test,y\_test,W\_ADAM,C)

L\_Adam.append(l\_test)

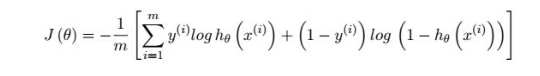
print("L\_Adam")

print(L\_Adam)

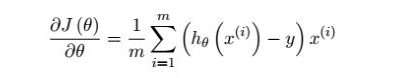
## 8. 模型参数的初始化方法:全零初始化

## 9.选择的loss函数及其导数:

逻辑回归：

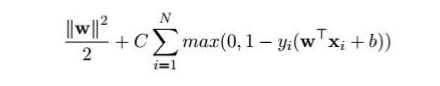
Loss

导数：



线性分类

Loss



导数

C:\Users\555\AppData\Roaming\Tencent\Users\791638657\QQ\WinTemp\RichOle\KW$_]LPSW~39ZW}O$0R@YWO.png

## 10.实验结果和曲线图:（各种梯度下降方式分别填写此项）

## 超参数选择：

逻辑回归：

#初始化NAG

eta\_NAG=0.1

gamma\_NAG=0.1

batch\_NAG=100

v=0

#初始化RMS

eta\_RMS=0.01

gamma\_RMS=0.9

epsilon\_RMS=1e-3

batch\_RMS=100

#初始化AdaDelta

gamma\_ADA=0.9

epsilon\_ADA=1e-6

batch\_ADA=100

#初始化adam

beta\_ADAM=0.9

gamma\_ADAM=0.99

eta\_ADAM=0.001

epsilon\_ADAM=1e-8

batch\_ADAM=100

线性分类：

#NAG初始化

eta\_NAG=1e-6

gamma\_NAG=1e-5

batch\_NAG=100

#RMSProp初始化

eta\_RMS=1e-3

gamma\_RMS=0.9

epsilon\_RMS=1e-6

batch\_RMS=100

#AdaDelta 初始化

gamma\_ADA=0.95

epsilon\_ADA=1e-8

batch\_ADA=100

#adam 初始化

beta\_ADAM=0.9

gamma\_ADAM=0.99

eta\_ADAM=1e-4

epsilon\_ADAM=1e-8

batch\_ADAM=100

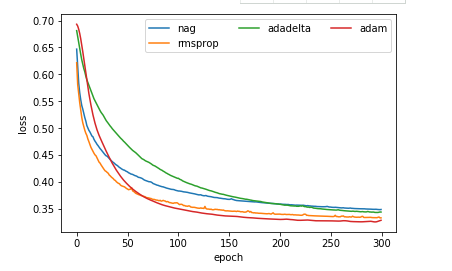
## 预测结果（最佳结果）：

逻辑回归：0.3289

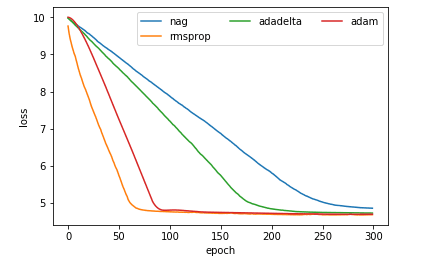
线性分类：4.6868

## loss曲线图：

逻辑回归



线性分类：



## 11.实验结果分析:使用不同的优化算法优化的时候，不同优化算法下降的速度不一样，优化逻辑回归时，adam与rmsprop可能会下降的比较快，优化线性分类时候，也是一样道理，可能是因为其学习率的可调性，不过如果将学习率调太高的时候，其loss值也可能反向增大。

## 12.对比逻辑回归和线性分类的异同点：

逻辑回归是一种广义线性回归，其y=wx+b, 其中w和b是待求参数, 逻辑回归的因变量可以是二分类的，也可以是多分类的，但是二分类的更为常用，所以逻辑回归是一个分类器, 不是真回归，线性分类就是以超平面为决策边界的分类器. 常见的线性分类有逻辑回归与SVM等, 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.

## 13.实验总结：

这次实验，相比第一次实验难度提升了比较多，这次实验的公式也比较繁多，有些参数一开始都难以理解，导致迟迟不敢下手，通过这次实验，我慢慢学会了着手比较不同优化算法对于不同问题的优劣，通过调节不同算法的参数以及收敛速度，减少算法运行的时间成本，进一步加深了对逻辑回归与线性分类的学习。