

**《机器学习》课程实验报告**

**学 院 软件学院**

**专 业 软件工程**

**组 员**  张梓佳

**学 号 201530613726**

**邮 箱 969035716@qq.com**

**指导教师**  **吴庆耀**

**提交日期** **2017年 12月 15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 月 日

## 3. 报告人:张梓佳

## 4. 实验目的: 对比理解梯度下降和随机梯度下降的区别与联系。

## 对比理解逻辑回归和线性分类的区别与联系。

## 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：在本次实验的两次分实验中，均使用LIBSVM Data的a9a.txt 以及 a9a\_testing.txt文件

## 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。

## 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，，和。

## 重复步骤4-6若干次，画出，，和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。
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**。

## 代码内容:

逻辑回归：（暂时只实现NAG方法的逻辑回归， 只取关键部分）

*#初始化参数*w=np.zeros((X\_train.shape[1], 1))  
LAMDA=1  
GAMA=0.9  
learning\_rate=0.1 #定义学习率  
num=200  
iter\_number=100 #定义迭代次数  
threshold=0.5  
shape1=range(0,X\_train.shape[0])  
  
*#实现随机取样***def** select\_function(X, Y, n):  
 random\_num=random.sample(shape1, n)  
 X\_random=np.ones((0, X.shape[1]))  
 Y\_random=np.ones((0, Y.shape[1]))  
  
 **for** i **in** random\_num:  
 X\_random = np.r\_[X\_random, X[i].reshape(1, X.shape[1])]  
 Y\_random= np.r\_[Y\_random, Y[i].reshape(1, Y.shape[1])]  
 **return** X\_random, Y\_random  
  
**def** NAG(X, Y, w):  
 w1 = w - GAMA \* vt1  
 loss\_gradient=np.zeros(w.shape)  
 **for** i **in** range(0, Y.shape[0]):  
 Xi = X[i].reshape(1, X.shape[1]).T  
 Yi = Y[i][0]  
 loss\_gradient = loss\_gradient+ Yi / (1 + math.exp(Yi \* np.dot(w1.T, Xi)[0][0])) \* Xi  
 gradient =( LAMDA \* w1 ) - 1 / num \* loss\_gradient  
 vt = GAMA \* vt1 + learning\_rate \* gradient  
 w = w - vt  
 **return** w  
  
NAG\_loss\_list = []  
**for** i **in** range(0, iter\_number):  
 vt1=np.zeros(w.shape)  
 x\_select, y\_select=select\_function(X\_train, Y\_train, num)  
 w = NAG(x\_select, y\_select, w)  
 loss\_sum=0  
 **for** i **in** range(0, Y\_test.shape[0]):  
 Xi = X\_test[i].reshape(1, X\_test.shape[1]).T  
 Yi = Y\_test[i][0]  
 loss\_sum += math.log(1 + math.exp(-Yi \* np.dot(w.T, Xi)[0][0]))  
  
 NAG\_test\_loss = LAMDA/2 \* np.dot(w.T, w)[0][0] + 1/X\_test.shape[0] \* loss\_sum  
 NAG\_loss\_list.append(NAG\_test\_loss)

线性分类（暂时只实现NAG方法的线性分类， 只取关键部分）

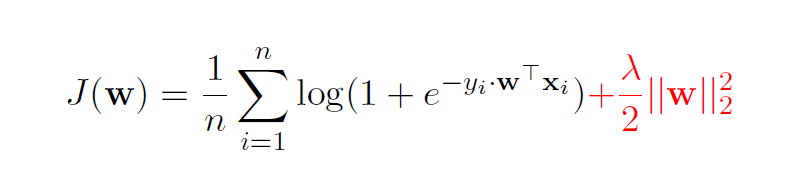
*#随机采样函数***def** select\_function(X, Y, n):  
 random\_num=random.sample(shape1, n)  
 X\_random=np.ones((0, X.shape[1]))  
 Y\_random=np.ones((0, Y.shape[1]))  
  
 **for** i **in** random\_num:  
 X\_random = np.r\_[X\_random, X[i].reshape(1, X.shape[1])]  
 Y\_random= np.r\_[Y\_random, Y[i].reshape(1, Y.shape[1])]  
 **return** X\_random, Y\_random  
  
**def** NAG(X, Y, w):  
  
 w = w - GAMA \* vt1  
 **for** i **in** range(0, num):  
 Xi = X[i].reshape(1, X.shape[1]).T  
 Yi = Y[i][0]  
  
 loss\_gradient=np.zeros(w.shape)  
 **if**(1-Yi\*np.dot(w.T, Xi)[0][0])>=0:  
 loss\_gradient += -Yi\*Xi  
  
 gradient = w + C \* loss\_gradient  
 vt = GAMA \* vt1 + learning\_rate \* gradient  
 w = w - vt  
  
 **return** w  
  
NAG\_loss\_list = []  
**for** i **in** range(0, iter\_number):  
 vt1 = np.zeros(w.shape)  
 x\_train, y\_train = select\_function(X\_train, Y\_train, num)  
 w = NAG(x\_train, y\_train, w)  
 loss\_sum = 0  
 **for** i **in** range(0, Y\_test.shape[0]):  
 Xi = X\_test[i].reshape(1, X\_test.shape[1]).T  
 Yi = Y\_test[i][0]  
  
 WT = 1 - Yi \* np.dot(w.T, Xi)[0][0]  
 **if** ( WT > 0):  
 loss\_sum += WT  
  
 NAG\_test\_loss = 1/2 \* np.dot(w.T, w)[0][0] + C \* 1/Y\_test.shape[0] \* loss\_sum  
 NAG\_loss\_list.append(NAG\_test\_loss)

## 模型参数的初始化方法:

全零初始化

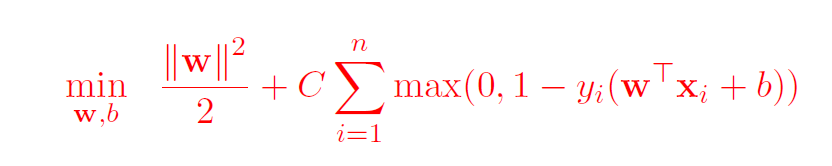
## 9.选择的loss函数及其导数:

逻辑回归loss函数：



导数：

线性分类loss函数：



## 实验结果和曲线图:（各种梯度下降方式分别填写此项）

**（NAG方法实现的逻辑回归）**

## 超参数选择：

GAMA=0.9

## 预测结果（最佳结果）：

(NAG的w)

W=[[ -5.64423145e-02]

[ -3.28699281e-02]

[ -1.45092027e-02]

[ 5.27135944e-03]

[ -1.06964997e-02]

[ -8.63949481e-02]

[ -7.44478249e-03]

[ 7.08725119e-03]

[ 1.60033197e-03]

[ -6.22613229e-03]

[ -6.29363385e-03]

[ -2.03252110e-04]

[ -2.17066525e-04]

[ -2.35476943e-02]

[ -1.87679886e-02]

[ -1.45092883e-02]

[ -2.38747935e-02]

[ -2.85468209e-02]

[ 7.28657767e-03]

[ -3.97903669e-02]

[ -1.21279083e-02]

[ -5.00122246e-02]

[ 7.27202270e-03]

[ -1.63699082e-03]

[ -4.01176044e-03]

[ -4.30860105e-03]

[ -6.06571343e-03]

[ -5.00350217e-03]

[ 7.81064090e-03]

[ -1.59481436e-03]

[ -8.03235077e-03]

[ 4.96299049e-03]

[ -3.29117282e-03]

[ -7.03411687e-04]

[ -4.11274746e-02]

[ -5.00122246e-02]

[ -3.97903669e-02]

[ -5.64875126e-03]

[ 2.73322317e-02]

[ 3.49303838e-02]

[ -3.29237083e-02]

[ -8.83817409e-02]

[ -8.75437526e-03]

[ -1.04149319e-02]

[ -3.92597946e-03]

[ 2.23766419e-04]

[ 1.56997579e-03]

[ -1.50532798e-02]

[ -2.88405175e-02]

[ -1.21393217e-02]

[ 1.59818056e-02]

[ 1.02481892e-02]

[ -1.27565352e-02]

[ -1.17098839e-02]

[ -2.21285882e-02]

[ -9.61346298e-03]

[ -1.04461939e-02]

[ -1.53086612e-03]

[ -1.39651054e-03]

[ -5.99765377e-05]

[ 4.58880633e-03]

[ -4.48394685e-02]

[ 3.23166653e-02]

[ -6.36755992e-02]

[ -9.23918013e-03]

[ -2.83978094e-02]

[ -7.99248224e-02]

[ -3.28122862e-03]

[ -2.94903003e-03]

[ -1.53552946e-03]

[ -2.15559751e-02]

[ -7.73445152e-02]

[ -3.19020704e-02]

[ -1.22944130e-01]

[ 1.36975440e-02]

[ -1.14918375e-01]

[ 5.67178908e-03]

[ -4.51331491e-02]

[ -1.48225651e-02]

[ -5.62124730e-02]

[ 2.23813863e-03]

[ 4.68346304e-03]

[ -9.04736372e-02]

[ -1.16575077e-04]

[ -7.46952441e-04]

[ -1.06522525e-03]

[ -8.96997270e-04]

[ -1.94609630e-04]

[ -2.70827439e-04]

[ 5.75952684e-05]

[ 5.18233929e-04]

[ 2.23919600e-05]

[ -7.92107856e-04]

[ -4.42948293e-04]

[ -1.21961511e-04]

[ 1.66710492e-05]

[ -1.96598596e-04]

[ -1.12012175e-03]

[ -2.20438890e-04]

[ -3.97587963e-04]

[ -7.36147145e-04]

[ -3.67615257e-05]

[ -5.55689090e-03]

[ -3.35815058e-04]

[ -2.94294998e-04]

[ 2.21205691e-04]

[ -6.38266911e-04]

[ -3.13165044e-04]

[ -3.75356750e-04]

[ 9.29185054e-05]

[ -5.03432833e-04]

[ -4.44671660e-04]

[ -4.43389892e-05]

[ -2.16689308e-04]

[ -4.21137685e-04]

[ -6.34039682e-06]

[ -1.17778674e-04]

[ -1.42637408e-04]

[ -1.01299456e-03]

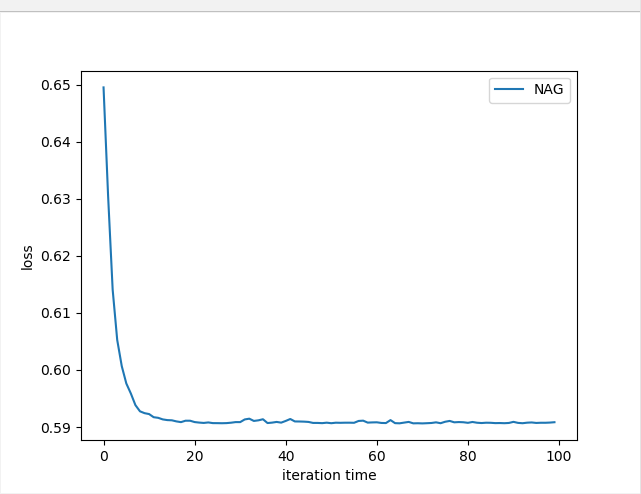
[ -4.32297316e-05]

[ -2.56577086e-04]

[ 1.19619227e-04]

[ 0.00000000e+00]]

## loss曲线图：



## 11.实验结果分析:在逻辑回归的实验中，Loss\_function在最初迭代期间有较大的优化，但是在迭代的后期优化的效果则逐渐不明显

## 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的NAG方法实现w的优化是相同的 并且在最初均是全零初始化vt和w 相同的，线性分类实现中也是将w 和b合并为一起计算

## 13.实验总结：在本次实验中，主要讲NAG方法较好地理解和实现 ，并应用于逻辑回归和线性分类中，并且较于第一次实验优化了梯度下降方法，实现了随机梯度下降方法。但是对于其他三种优化算法，在代码实现上出现了多个暂时没有解决的bug，将在实验后逐步实现代码并优化之。