

**《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)

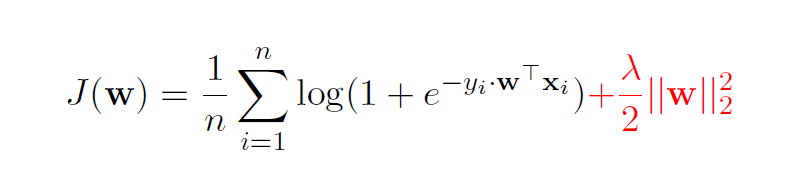
线性分类（暂时只实现NAG方法的线性分类， 只取关键部分）

*#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)

## 8. 模型参数的初始化方法:

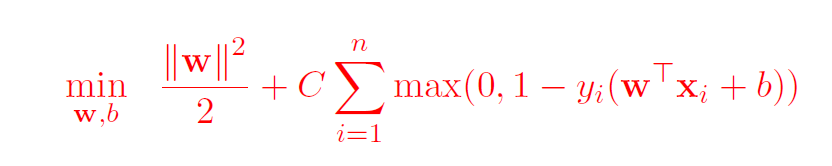
## 9.选择的loss函数及其导数:

逻辑回归loss函数：



导数：

线性分类loss函数：



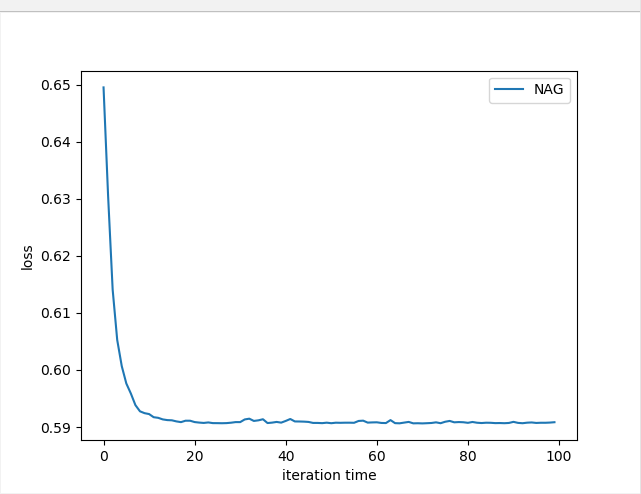
## 实验结果和曲线图:（各种梯度下降方式分别填写此项）

**（NAG方法实现的逻辑回归）**

## 超参数选择：

## 预测结果（最佳结果）：

## loss曲线图：



## 11.实验结果分析:

## 12.对比逻辑回归和线性分类的异同点：

## 13.实验总结：