华南理工大学

South China University of Technology

《机器学习》课程实验报告

学　　院　软件学院

专　　业　软件工程

组　　员

学　　号　201530651780

邮　　箱　599585056@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 14 日

3. 报告人:詹李钦

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 的原理并在就大数据上实践。

5. 数据集以及数据分析：

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

逻辑回归：

6. 实验步骤:

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L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}。
7. 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:

RMSProp：

```
learningRate=0.01
predictRate=0.9
bias=0.9
times=1
def training():
    global W
    global G
    index=random.randint(0,length_train-1)
    X_train_index=X_train[index]
    X_train_index.shape=(-1,1)
    element=(X_train_index.T).dot(W)
    descentLeft=getSimo(element[0][0])-y_train[index][0]
    D=X_train_index*descentLeft
    D2=D*D
    G=G*predictRate+D2*(1-predictRate)
    Rate=G
    for i in range(124):
        Rate[i]=getRate(G[i],learningRate)
    W-=Rate*D
```

NAG：

```
learningRate=0.002
predictRate=0.9
bias=0.9
times=1
def training():
    global W
    global V
    W=W-V*predictRate
    index=random.randint(0,length_train-1)
    X_train_index=X_train[index]
    X_train_index.shape=(-1,1)
    element=(X_train_index.T).dot(W)
    descentLeft=getSimo(element[0][0])-y_train[index][0]
    D=X_train_index*descentLeft
    V=predictRate*V+learningRate*D
    W-=V
```

Adam：

```
learningRate=0.002
predictRate=0.999
bias=0.9
times=1
def training():
    global W
    global G
    global T
    global M
    global learningRate
    global times
    A=learningRate*math.sqrt(predictRate**times)/(1-(bias**times))
    index=random.randint(0,length_train-1)
    X_train_index=X_train[index]
    X_train_index.shape=(-1,1)
    element=(X_train_index.T).dot(W)
    descentLeft=getSimo(element[0][0])-y_train[index][0]
    D=X_train_index*descentLeft
    D2=D*D
    G=G*predictRate+D2*(1-predictRate)
    M=M*bias+(1-bias)*D
    Rate=G
    for i in range(124):
        Rate[i]=getRateAdaDelta(G[i],A)
    W-=Rate*M
    times+=1
```

AdaDelta：

```
learningRate=0.02
predictRate=0.95
bias=0.9
times=1
def training():
    global W
    global G
    global T
    index=random.randint(0,length_train-1)
    X_train_index=X_train[index]
    X_train_index.shape=(-1,1)
    element=(X_train_index.T).dot(W)
    descentLeft=getSimo(element[0][0])-y_train[index][0]
    D=X_train_index*descentLeft
    D2=D*D
    G=G*predictRate+D2*(1-predictRate)
    Rate=G
    for i in range(124):
        Rate[i]=getRateAdaDelta(G[i],T[i])
    W-=Rate*D
    T=predictRate*T+(1-predictRate)*(Rate*D*(Rate*D))
```

8.模型参数的初始化方法:

逻辑回归模型初始化，采用全零初始化。

9.选择的 loss 函数及其导数:

Loss:

$$\sum_n -\left[\hat{y}^n ln f_{w,b}(x^n) + (1-\hat{y}^n) ln\left(1 - f_{w,b}(x^n)\right)\right]$$

Descent:

$$-\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$$

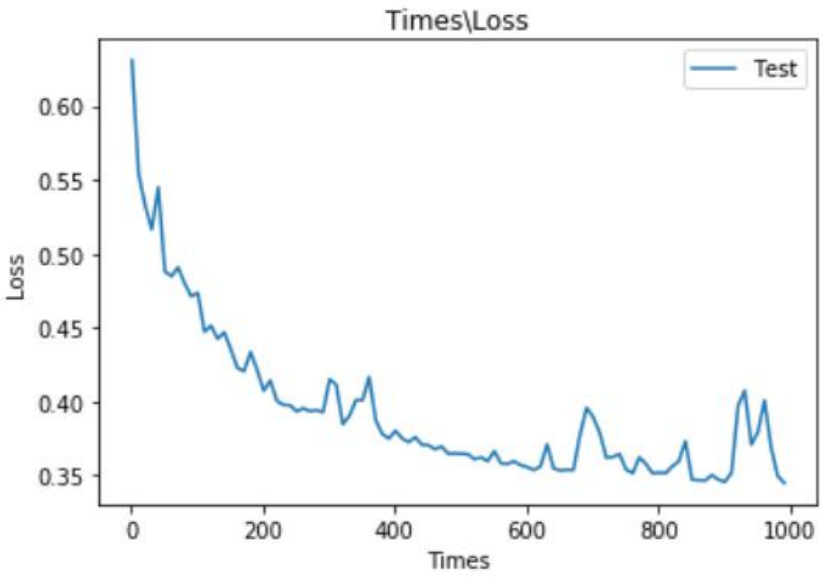
10. 实验结果和曲线图:（各种梯度下降方式分别填写此项）

RMSProp：

超参数选择：predictRate=0.9 learningRate=0.01

预测结果（最佳结果）：0.8347

loss 曲线图：



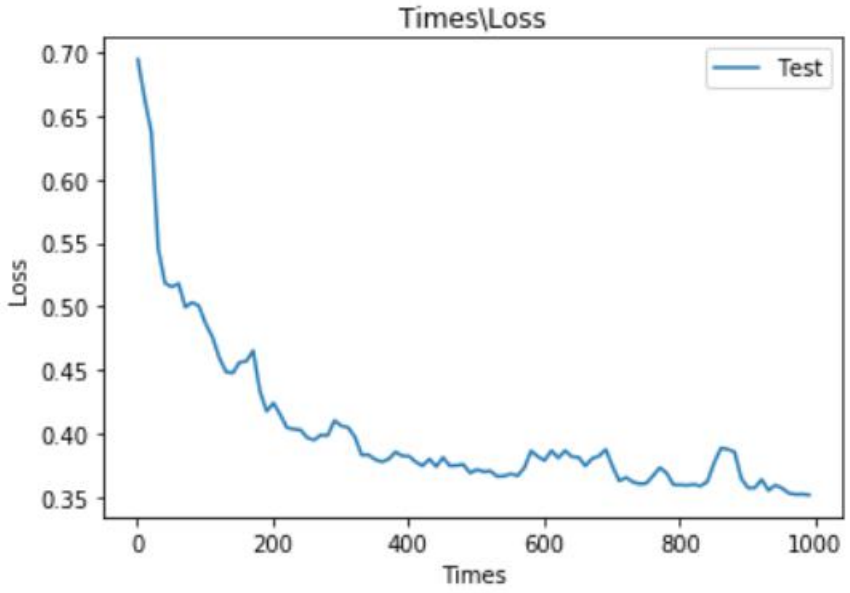
0.8347767336158712

NAG：

超参数选择：predictRate=0.9 learningRate=0.002

预测结果（最佳结果）：0.8345

loss 曲线图：



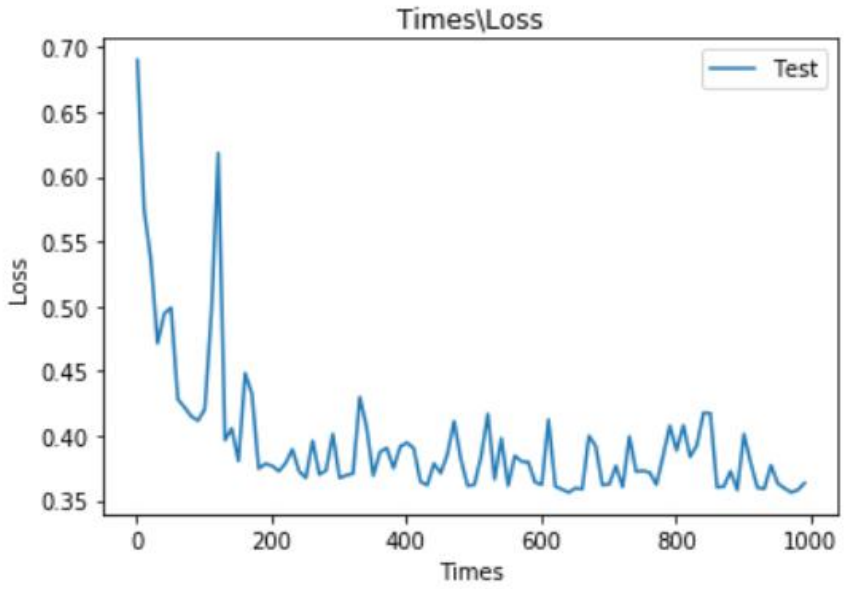
0.8345310484613967

Adam：

超参数选择：predictRate=0.999 learningRate=0.002

预测结果（最佳结果）：0.8312

loss 曲线图：



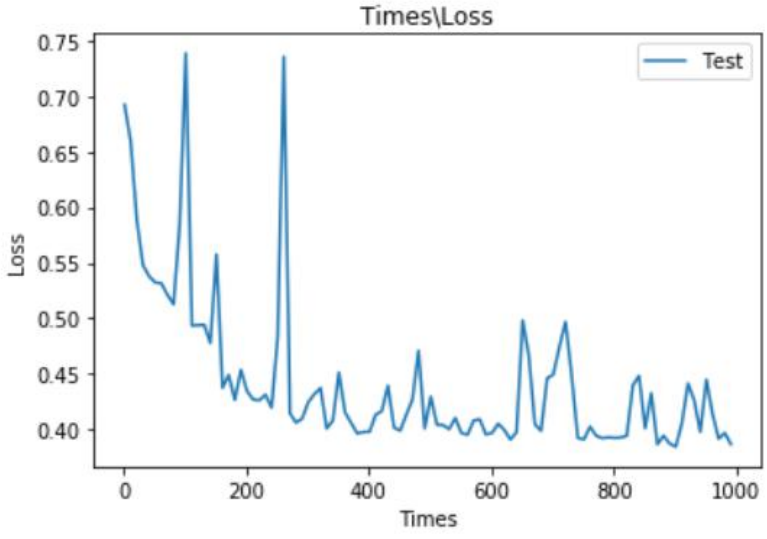
0.8311528775873718

AdaDelta：

超参数选择：predictRate=0.95

预测结果（最佳结果）：0.8161

loss 曲线图：



0.8191143050181193

11.实验结果分析:

认为当样例相对某个分类的概率大于 0.5，即可属于哪个分类。几种优化方法选用不同的超参数，在梯度下降的速度方面差距明显，但是最终的准确率并没有太大的变化

线性分类：

6.实验步骤:

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L_{NAG}，$L_{RMSProp}$，$L_{AdaDelta}$和L_{Adam}。
7. 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:

NAG：

```
learningRate=0.006
predictRate=0.95
bias=0.9
times=1
C=5
def training():
    global W
    W=W-V*predictRate
    index=random.randint(0,length_train-1)
    X_train_index=X_train[index]
    X_train_index.shape=(-1,1)
    element=(X_train_index.T).dot(W)
    descentLeft=1-element[0][0]*y_train[index][0]
    if descentLeft>0:
        D=W-y_train[index][0]* X_train_index*C
        V=predictRate*V+learningRate*D
        W-=V
        return 1
    else:
        return 0
```

RMSProp：

```
learningRate=0.005
predictRate=0.9
bias=0.9
times=1
C=1
def training():
    global W
    global G
    index=random.randint(0,length_train-1)
    X_train_index=X_train[index]
    X_train_index.shape=(-1,1)
    element=(X_train_index.T).dot(W)
    descentLeft=1-element[0][0]*y_train[index][0]
    if descentLeft>0:
        D=W-y_train[index][0]* X_train_index*C
        D2=D*D
        G=G*predictRate+D2*(1-predictRate)
        Rate=G
        for i in range(124):
            Rate[i]=getRate(G[i],learningRate)
        W-=Rate*D
        return 1
    else:
        return 0
```

RMSProp：

```
learningRate=0.005
predictRate=0.9
bias=0.9
times=1
C=1
def training():
    global W
    global G
    index=random.randint(0,length_train-1)
    X_train_index=X_train[index]
    X_train_index.shape=(-1,1)
    element=(X_train_index.T).dot(W)
    descentLeft=1-element[0][0]*y_train[index][0]
    if descentLeft>0:
        D=W-y_train[index][0]* X_train_index*C
        D2=D*D
        G=G*predictRate+D2*(1-predictRate)
        Rate=G
        for i in range(124):
            Rate[i]=getRate(G[i],learningRate)
        W-=Rate*D
        return 1
    else:
        return 0
```

AdaDelta：

```
learningRate=0.002
predictRate=0.95
bias=0.9
times=1
C=2
def training():
    global W
    global G
    global T
    index=random.randint(0,length_train-1)
    X_train_index=X_train[index]
    X_train_index.shape=(-1,1)
    element=(X_train_index.T).dot(W)
    descentLeft=1-element[0][0]*y_train[index][0]
    if descentLeft>0:
        D=W-y_train[index][0]* X_train_index*C
        D2=D*D
        G=G*predictRate+D2*(1-predictRate)
        Rate=G
        for i in range(124):
            Rate[i]=getRateAdaDelta(G[i],T[i])
        W-=Rate*D
        T=predictRate*T+(1-predictRate)*(Rate*D*(Rate*D))
        return 1
    else:
        return 0
```

8. 模型参数的初始化方法:

支持向量机模型参数初始化，采用全零初始化。

9.选择的 loss 函数及其导数:

Loss:

$$\frac{\|\mathbf{w}\|^2}{2}+C\sum_{i=1}^{N}max(0,1-y_i(\mathbf{w}^\top\mathbf{x}_i+b))$$

Descent:

$$\mathbf{w}^\top-C\mathbf{y}^\top\mathbf{X}$$

10.实验结果和曲线图:（各种梯度下降方式分别填写此项）

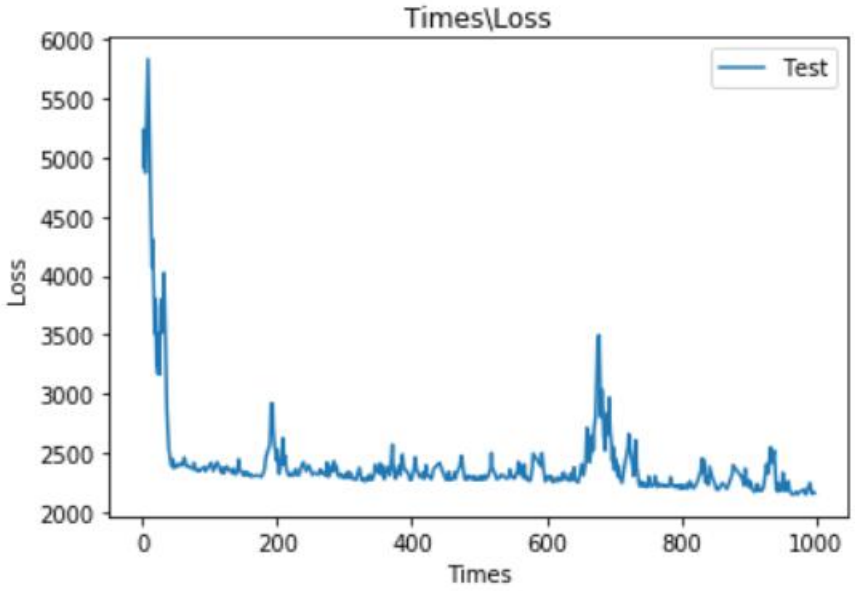
NAG：

超参数选择：predictRate=0.95 learningRate=0.006

预测结果（最佳结果）：0.8114

loss 曲线图：

503



0.8114366439407898

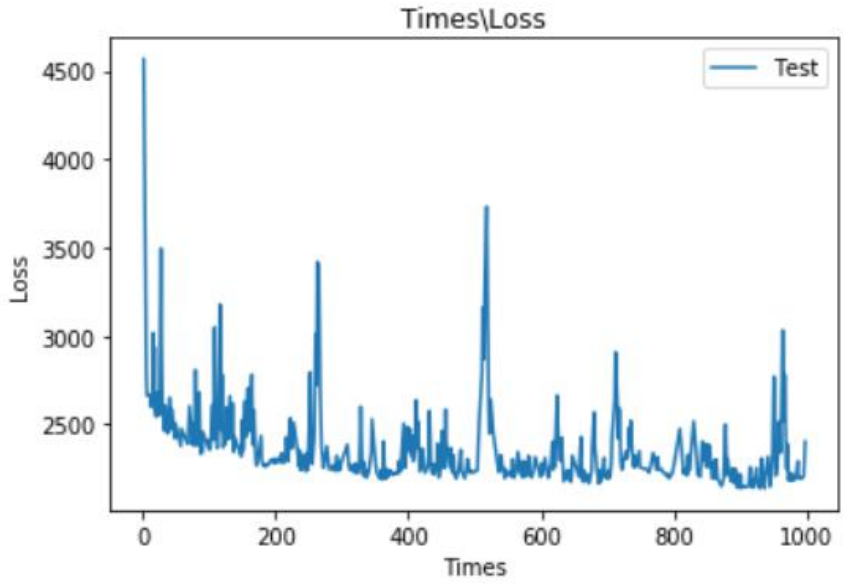
RMSProp：

超参数选择：predictRate=0.9 learningRate=0.005

预测结果（最佳结果）：0.8294

loss 曲线图：

513



0.82943308150605

RMSProp：

超参数选择：predictRate=0.9 learningRate=0.005

预测结果（最佳结果）：0.8294

loss 曲线图：

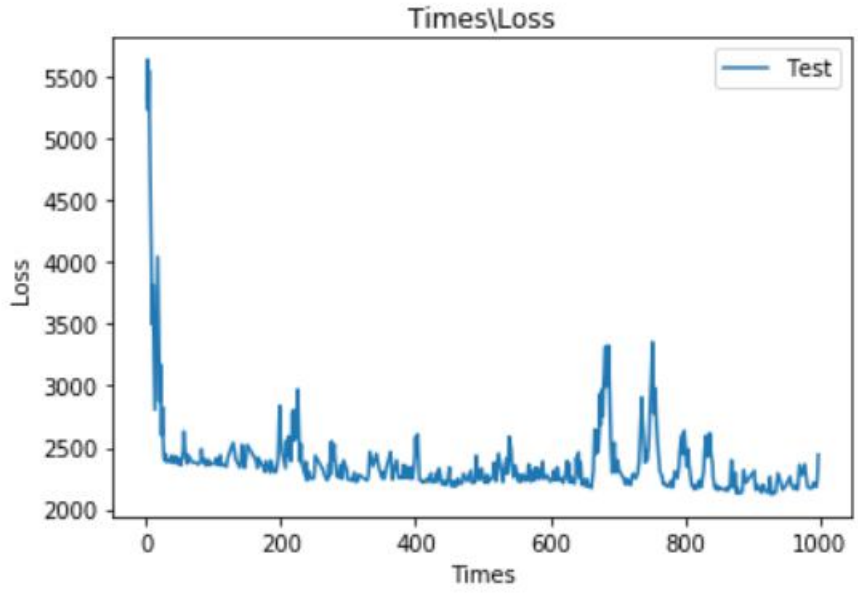
AdaDelta：

超参数选择：predictRate=0.95 C=2

预测结果（最佳结果）：0.8331

loss 曲线图：

501



0.8330569375345495

AdaDelta：

超参数选择：predictRate=0.95 C=2

预测结果（最佳结果）：0.8331

loss 曲线图：

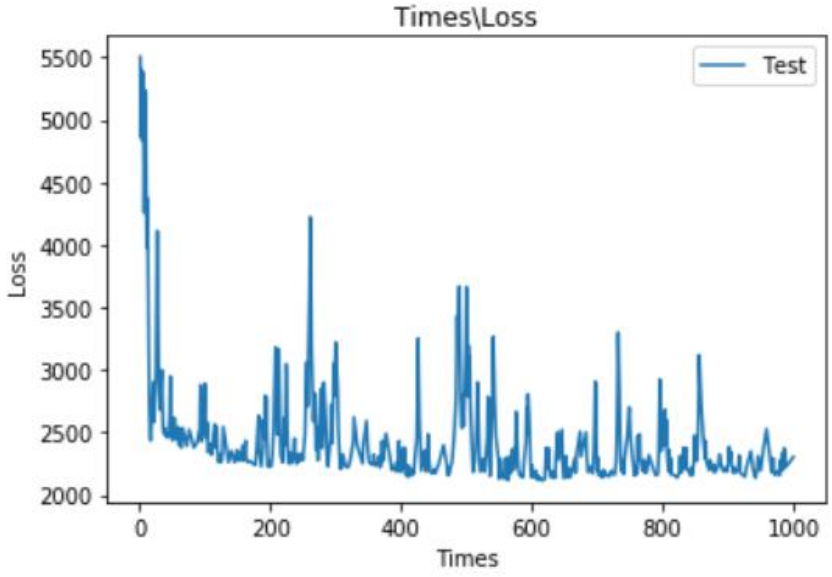
Adam：

超参数选择：predictRate=0.999 C=4 learningRate=0.005

预测结果（最佳结果）：0.8291

loss 曲线图：

517



0.8291873963515755

12.实验结果分析:

由于支持向量机模型，只有少数的样例对模型有影响，所以需要提前判断样例是否为有效样例。几种优化方法选用不同的超参数，在梯度下降的速度方面差距明显，但是最终的准确率并没有太大的变化

13.对比逻辑回归和线性分类的异同点：

相同点：都通过模型和模型参数判断数据，并选择一定的阈值进行分类。

不同点：逻辑回归生成的是分类的概率，根据概率判断更有可能是哪个分类，线性分类通过判断样例落在超平面的那一侧来判断属于哪个分类

13.实验总结：

这次实验数据量比上个实验大了不少，适合采用随机梯度下降，能够大大减少数据处理的时间，同时试验了不少梯度下降的优化方法。并且对调参有了进一步的理解。