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　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

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log\, h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) log\, \left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

导数：

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y\right)x^{(i)}$$

线性分类
Loss

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

导数

$$\nabla f(\beta) = \begin{cases} \mathbf{w}^\top - C\mathbf{y}^\top \mathbf{X} & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0 \\ \mathbf{w}^\top & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0 \end{cases}$$

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

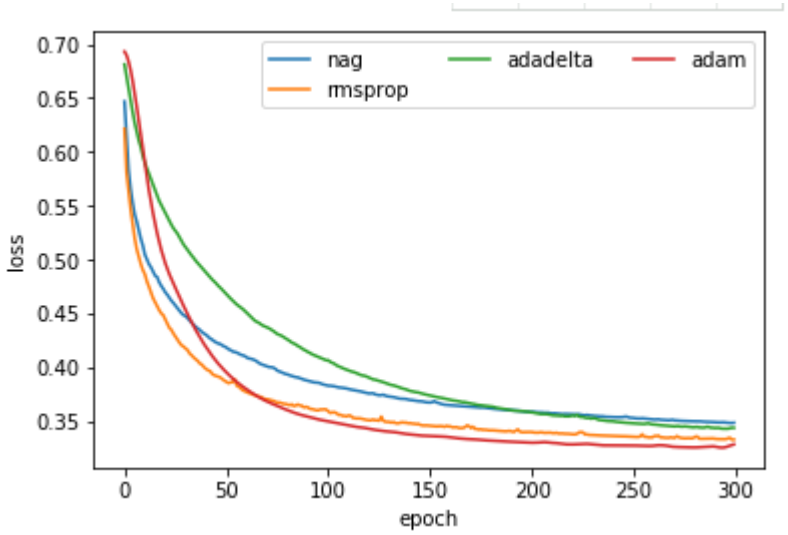
#adam 初始化
beta_ADAM=0.9
gamma_ADAM=0.99
eta_ADAM=1e-4
epsilon_ADAM=1e-8
batch_ADAM=100

预测结果（最佳结果）：

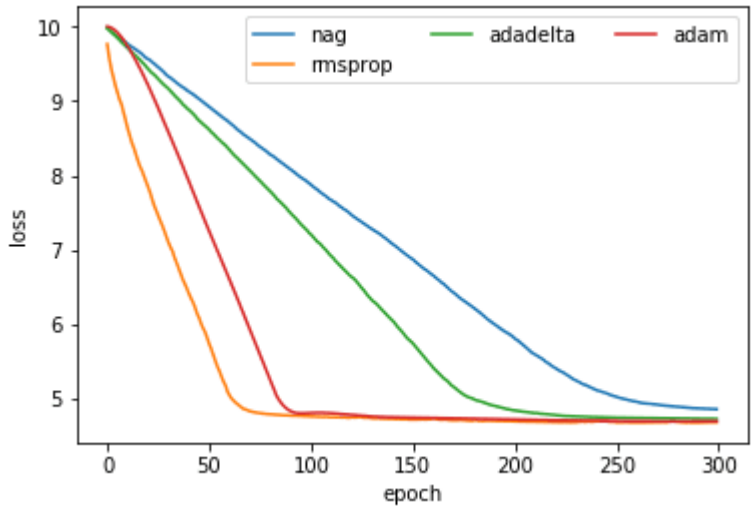
逻辑回归：0.3289
线性分类：4.6868

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。