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　　符王朝

学　　号　　201530611470

邮　　箱　　2262458345@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 12 月 2 日

3. 报告人:符王朝

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。

数据集

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

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。
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**。

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**。

8. 代码内容:

逻辑回归

```
#NAG
los = loss(x_val, y_validation, NAGW) / x_val.shape[0]
loss_val.append(los)
g = x.T.dot(sigmoid(np.dot(x, NAGW - gamma * v)) - y) / x.shape[0]
v = gamma * v + learning_rate * g
NAGW = NAGW - v

#RMSProp
los = loss(x_validation, y_validation, RMSPropW) / x_validation.shape[0]
loss_val.append(los)
g = x.T.dot(sigmoid(np.dot(x, RMSPropW)) - y) / x.shape[0]
G = gamma * G + (1 - gamma) * g * g
RMSPropW = RMSPropW - learning_rate / np.sqrt(G + epsilon) * g

#AdaDelta
g = x.T.dot(sigmoid(np.dot(x, AdaDeltaW)) - y) / x.shape[0]
G = gamma * G + (1 - gamma) * g * g
delta_w = - np.sqrt(delta + epsilon) / np.sqrt(G + epsilon) * g
AdaDeltaW = AdaDeltaW + delta_w
delta = gamma * delta + (1 - gamma) * delta_w * delta_w

#Adam
g = x.T.dot(sigmoid(np.dot(x, AdamW)) - y) / x.shape[0]
m = beta * m + (1.0 - beta) * g
G = gamma * G + (1.0 - gamma) * g * g
alpha = learning_rate * np.sqrt(1.0 - gamma**(i + 1)) / (1.0 - beta**(i + 1))
AdamW = AdamW - alpha * m / np.sqrt(G + epsilon)
```

线性分类:

```
#NAG
w_gt, b_gt = gradient(dx, dy, w - gamma * vw, b - gamma * vb, C=gammaC)
vw = gamma * vw + learning_rate * w_gt
w = w - vw
vb = gamma * vb + learning_rate * b_gt
b = b - vb
```

```python
#RMSProp
wg, bg = gradient(dx, dy, w, b, C=gammaC)
wG = gamma * wG + (1 - gamma) * (wg ** 2)
w = w - learnRate / np.sqrt(wG + e) * wg
bG = gamma * bG + (1 - gamma) * (bg ** 2)
b = b - learnRate / np.sqrt(bG + e) * bg

#AdaDeltacA
wG = gamma * wG + (1 - gamma) * (wg ** 2)
wdw = - (np.sqrt(wt + e) / np.sqrt(wG + e)) * wg
w = w + wdw
wt = gamma * wt + (1 - gamma) * (wdw ** 2)
bG = gamma * bG + (1 - gamma) * (bg ** 2)
bdw = - (np.sqrt(bt + e) / np.sqrt(bG + e)) * bg
b = b + bdw
bt = gamma * bt + (1 - gamma) * (bdw ** 2)

#Adam
wm = beta * wm + (1 - beta) * w_gt
wG = gama * wG + (1 - gama) * (w_gt ** 2)
alp = learnRate * np.sqrt(1 - gama ** (i + 1)) / (1 - beta ** (i + 1))
w = w - alp * wm / np.sqrt(wG + e)
bm = beta * bm + (1 - beta) * b_gt
bG = gama * bG + (1 - gama) * (b_gt ** 2)
alp = learnRate * np.sqrt(1 - gama ** (i + 1)) / (1 - beta ** (i + 1))
b = b - alp * bm / np.sqrt(bG + e)
```

（针对逻辑回归和线性分类分别填写 8-11 内容）

9. 模型参数的初始化方法:

全 0 初始化

10. 选择的 loss 函数及其导数:

逻辑回归

$$J(\mathbf{w}) = -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log h_{\mathbf{w}}(\mathbf{x}_i) + (1-y_i)\log\left(1-h_{\mathbf{w}}(\mathbf{x}_i)\right)\right]$$

$$\begin{aligned}\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= -\frac{1}{\partial \mathbf{w}} \cdot \partial\left[y \cdot \log h_{\mathbf{w}}(\mathbf{x}) + (1-y)\log\left(1-h_{\mathbf{w}}(\mathbf{x})\right)\right] \\ &= -y \cdot \frac{1}{h_{\mathbf{w}}(\mathbf{x})} \cdot \frac{\partial h_{\mathbf{w}}(\mathbf{x})}{\partial \mathbf{w}} + (1-y) \cdot \frac{1}{1-h_{\mathbf{w}}(\mathbf{x})} \frac{\partial h_{\mathbf{w}}(\mathbf{x})}{\partial \mathbf{w}}\end{aligned}$$

线性分类：

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

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b) = \mathbf{w} + \frac{C}{n}\sum_{i=1}^{n} g_{\mathbf{w}}(\mathbf{x}_i)$$

$$\nabla_b L(\mathbf{w}, b) = \frac{C}{n}\sum_{i=1}^{n} g_b(\mathbf{x}_i)$$

10.实验结果和曲线图:（各种梯度下降方式分别填写此项）

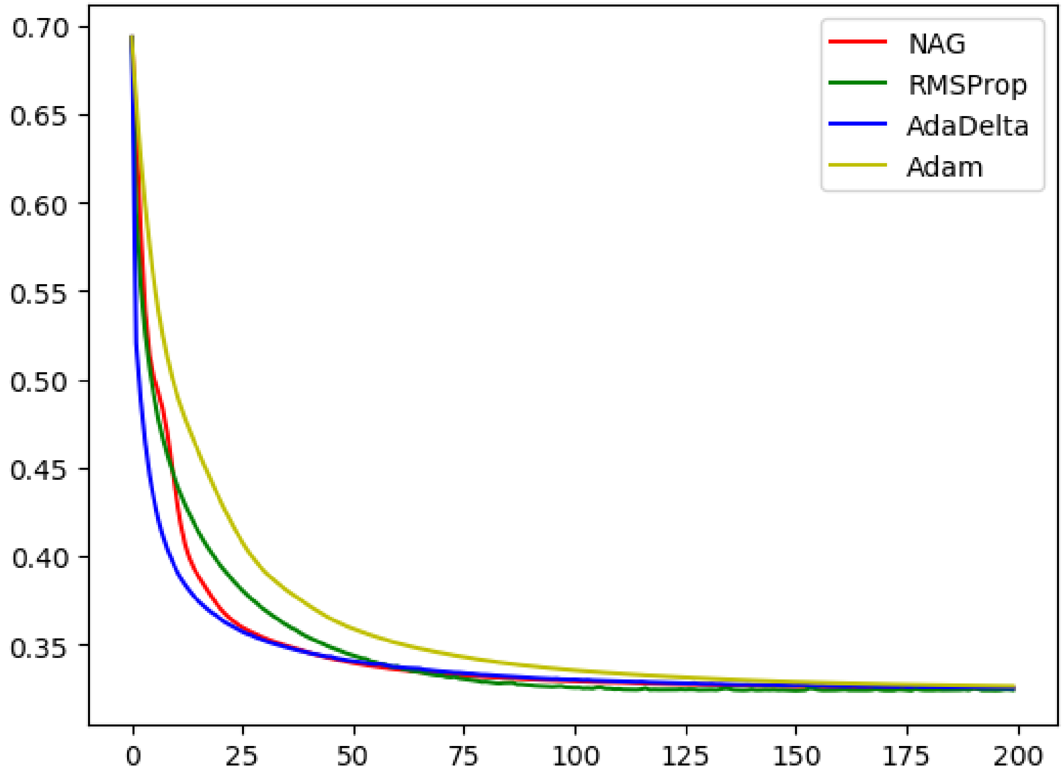
超参数选择：

逻辑回归：

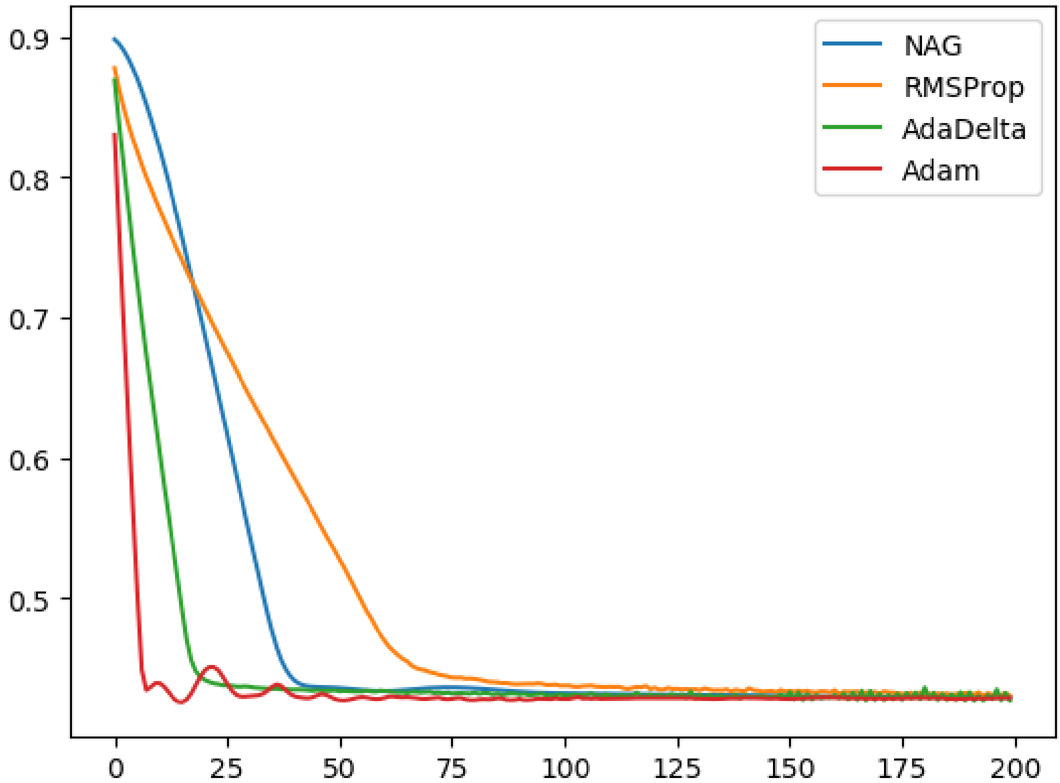
```
#NAG
NAGLearnRate = 0.1
NAGGamma = 0.9

#RMSProp
RMSPropLearnRate = 0.01
RMSPropGamma = 0.9
RMSPropEpsilon = 1e-8

#AdaDelta
#AdaDeltaLearnRate = 0.1
AdaDeltaGamma = 0.95
AdaDeltaEpsilon = 0.001

#Adam
AdamLearnRate = 0.01
AdamGamma = 0.999
AdamEpsilon = 1e-8
AdamBeta = 0.9
#NAG
learn_gate = 0.001
gamma = 0.9
#RMSProp
learn_gate = 0.001
gama = 0.001
e = 1e-9
# AdaDelta
gamma = 0.95
e = 1e-6
# Adam
e = 1e-8
beta = 0.9
gama = 0.9
learnRate = 1e-2
```

loss 曲线图：

逻辑回归


0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
25
50
75
100
125
150
175
200
NAG
RMSProp
AdaDelta
Adam

线性分类


0.9
0.8
0.7
0.6
0.5
0
25
50
75
100
125
150
175
200
NAG
RMSProp
AdaDelta
Adam

11. 实验结果分析:

学习率太低，会导致观察效果不好，Adam 效果在线性分类好在逻辑回归略差，adaDelte、NAG 表现都好,RMSProp 一般

12. 对比逻辑回归和线性分类的异同点：

两种方法都是常见的分类算法,从目标函数来看,区别在于逻辑回归采用的是logistical loss,svm 采用的是 hinge loss.这两个损失函数的目的都是增加对分类影响较大的数据点的权重,减少与分类关系较小的数据点的权重.SVM 的处理方法是只考虑 support vectors,也就是和分类最相关的少数点,去学习分类器.而逻辑回归通过非线性映射,大大减小了离分类平面较远的点的权重,相对提升了与分类最相关的数据点的权重.两者的根本目的都是一样的.此外,根据需要,两个方法都可以增加不同的正则化项,如 l1,l2 等等.所以在很多实验中,两种算法的结果是很接近的.

但是逻辑回归相对来说模型更简单,好理解,实现起来,特别是大规模线性分类时比较方便.而 SVM 的理解和优化相对来说复杂一些.但是 SVM 的理论基础更加牢固,有一套结构化风险最小化的理论基础,虽然一般使用的人不太会去关注.还有很重要的一点,SVM 转化为对偶问题后,分类只需要计算与少数几个支持向量的距离,这个在进行复杂核函数计算时优势很明显,能够大大简化模型和计算

svm 更多的属于非参数模型,而 logistic regression 是参数模型,本质不同.其区别就可以参考参数模型和非参模型的区别就好了.

logic 能做的 svm 能做,但可能在准确率上有问题,svm 能做的 logic 有的做不了

13. 实验总结：

加深了对逻辑回归和线性分类的理解，同时知道了四种优化方法更新模型