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　陈宇坤

学　　号　201530611326

邮　　箱　972640770@qq.com

指导教师　吴庆耀

提交日期　2017 年　12 月　2 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:陈宇坤

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。请自行下载训练集和验证集。

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:

逻辑回归：

```python
# define objective function
def sigmoid_function(x):
    return 1 / (1 + np.exp(-x))


# define calculate gradient function
def calculate_gradient(given_x, given_y, omega):
    hw = sigmoid_function(np.dot(given_x, omega))
    a = hw - given_y
    gradient_ = np.dot(given_x.T, a) / given_y.size
    return gradient_


# define loss function
def loss_function(given_x, given_y, omega):
    hw = sigmoid_function(np.dot(given_x, omega))
    loss = -np.mean(given_y * np.log(hw) + (1 - given_y) * np.log(1 - hw))
    return loss
```

```python
index = random.randint(0, y_train.size - batch_number)

SGD_gradient = calculate_gradient(X_train[index:index + batch_number], y_train[index:index + batch_number], SGD_theta)

# update parameters
SGD_theta = SGD_theta - learning_rate * SGD_gradient

SGD_loss.append(loss_function(X_test, y_test, SGD_theta))
```

```python
# NAG
NAG_gradient=calculate_gradient(X_train[index:index + batch_number], y_train[index:index + batch_number],NAG_theta+Alpha*v_theta)
v_theta=Alpha*v_theta- learning_rate * NAG_gradient
NAG_theta+=v_theta
NAG_loss.append(loss_function(X_test, y_test, NAG_theta))

# RMSProp
RMSProp_gradient=calculate_gradient(X_train[index:index + batch_number], y_train[index:index + batch_number],RMSProp_theta)
r_theta=decay_rate*r_theta+(1-decay_rate)*(RMSProp_gradient**2)
RMSProp_theta = RMSProp_theta-learning_rate*RMSProp_gradient/(np.sqrt(r_theta+delta))
RMSProp_loss.append(loss_function(X_test, y_test, RMSProp_theta))
```

```
    # Adadelta
    Adadelta_gradient=calculate_gradient(X_train[index:index + batch_number], y_train[index:index + batch_number],AdaDelta_theta)
    o_theta = decay_rate * o_theta + (1 - decay_rate) * (Adadelta_gradient ** 2)
    if i != 0:
        s_theta = decay_rate * s_theta + (1 - decay_rate) * Delta_theta ** 2

    Delta_theta = np.sqrt(s_theta) / (np.sqrt(o_theta + delta)) * Adadelta_gradient
    AdaDelta_theta=Delta_theta
    Adadelta_loss.append(loss_function(X_test, y_test, AdaDelta_theta))

    # Adam
    Adam_gradient=calculate_gradient(X_train[index:index + batch_number], y_train[index:index + batch_number],Adam_theta)
    rho_1=decay_rate*rho_1+(1-decay_rate)*Adam_gradient
    rho_2=decay_rate2*rho_2+(1-decay_rate2)*(Adam_gradient**2)
    rho_1_ = rho_1 / (1 - decay_rate ** (i + 1))
    rho_2_ = rho_2/(1-decay_rate2**(i+1))
    Adam_theta=Adam_theta-rho_1_*learning_rate/(np.sqrt(rho_2_+delta))
    Adam_loss.append(loss_function(X_test, y_test, Adam_theta))
```

线性分类：

```
# define calculate gradient function
def calculate_gradient(given_x, given_y, theta):
    Gradient_theta = np.zeros(124)
    for curr_x,curr_y in zip(given_x,given_y):
        if (1 - curr_y * np.dot(curr_x,theta)) >= 0:
            Gradient_theta += C * (-curr_y * curr_x)

        Gradient_theta += theta
    return Gradient_theta


# define loss function
def loss_function(given_x, given_y, omega):
    loss = np.mean(np.maximum(0, 1 - given_y * np.dot(given_x, omega))) + np.dot(omega, omega) * C / 2
    return loss
```

四种梯度下降方式的实现与逻辑回归相同。

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：

$$\mathrm{L(w)} = -\frac{1}{n}[\sum_{i=1}^{n} y_i \log h_{\boldsymbol{w}}(\boldsymbol{x}_i) + (1-y_i)\log(1-h_{\boldsymbol{w}}(\boldsymbol{x}_i))]$$

其中$\mathrm{h_w}(x) = \frac{1}{1+e^{-w^T x}}$

导数：

$$\frac{\partial \mathrm{L}}{\partial \mathbf{w}} = \frac{1}{n}\sum_{i=1}^{n}(h_{\boldsymbol{w}}(\boldsymbol{x_i}) - \boldsymbol{y})\,\boldsymbol{x_i}$$

线性分类：

$$\mathrm{L}(\mathbf{w}) = \frac{||\boldsymbol{w}||^2}{2} + \sum_{i=1}^{n}\left(\max\left(0,1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i)\right)\right)$$

导数：

$$g_{\boldsymbol{w}}(\boldsymbol{x}_i) = \begin{cases} -y_i\boldsymbol{x}_i, & 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 0 \\ 0, & 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) < 0 \end{cases}$$

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

10.实验结果和曲线图:

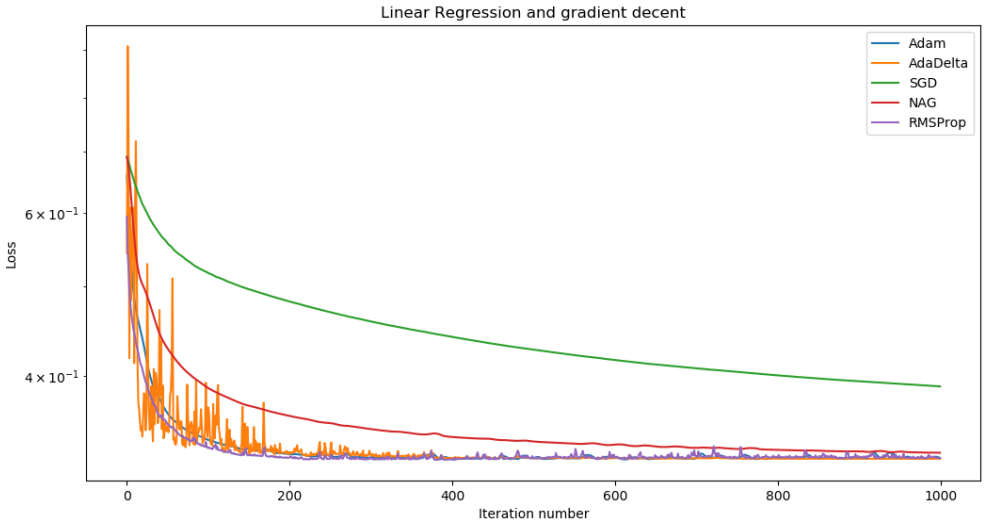
超参数选择：

四种梯度下降共用的超参数：学习率：逻辑回归：0.01　　线性分类：0.0001
迭代次数：1000　　批量数量：100
线性分类的阈值为 0.1 常数 C 为 0.5

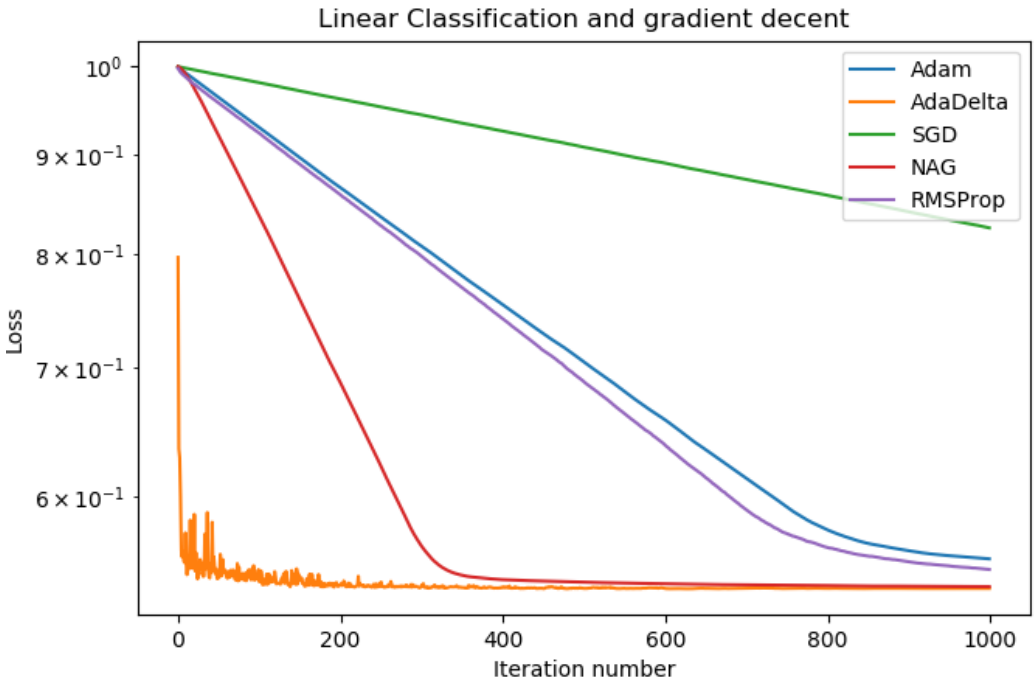
NAG 中更新动量的参数 Alpha 为 0.9
RMSProp 中的衰减速率 decay_rate 为 0.9
Adadelta 的衰减速率 decay_rate 为 0.9，防止分母为零的参数 delta 为 10^(-8)
Adam 的两个衰减速率分别为 0.9 和 0.999，delta 也为 10^(-8)

预测结果（最佳结果）：

逻辑回归：



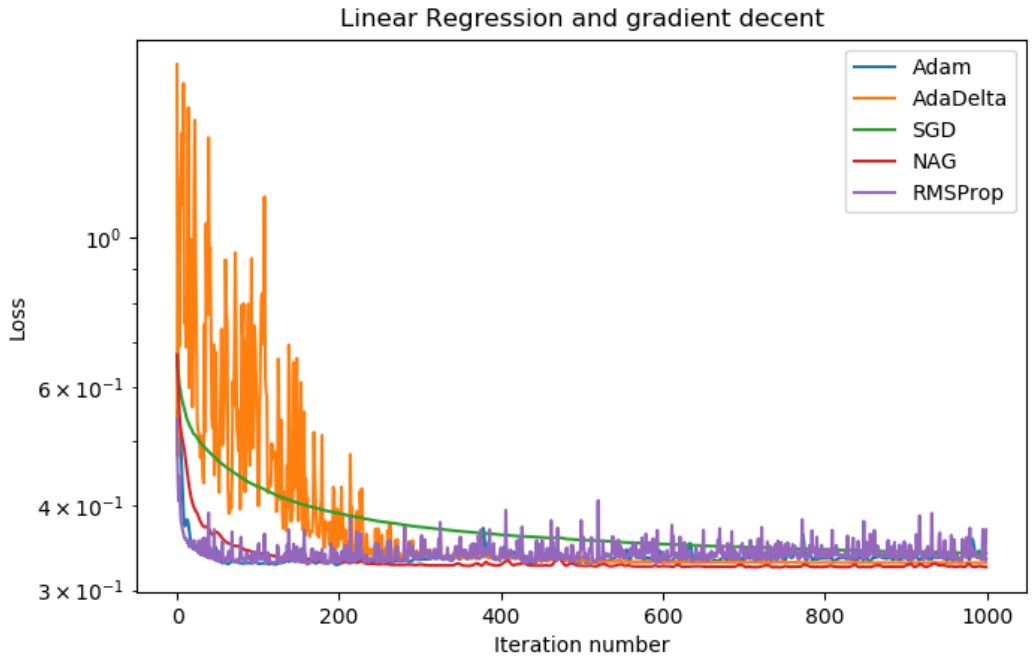
线性分类：



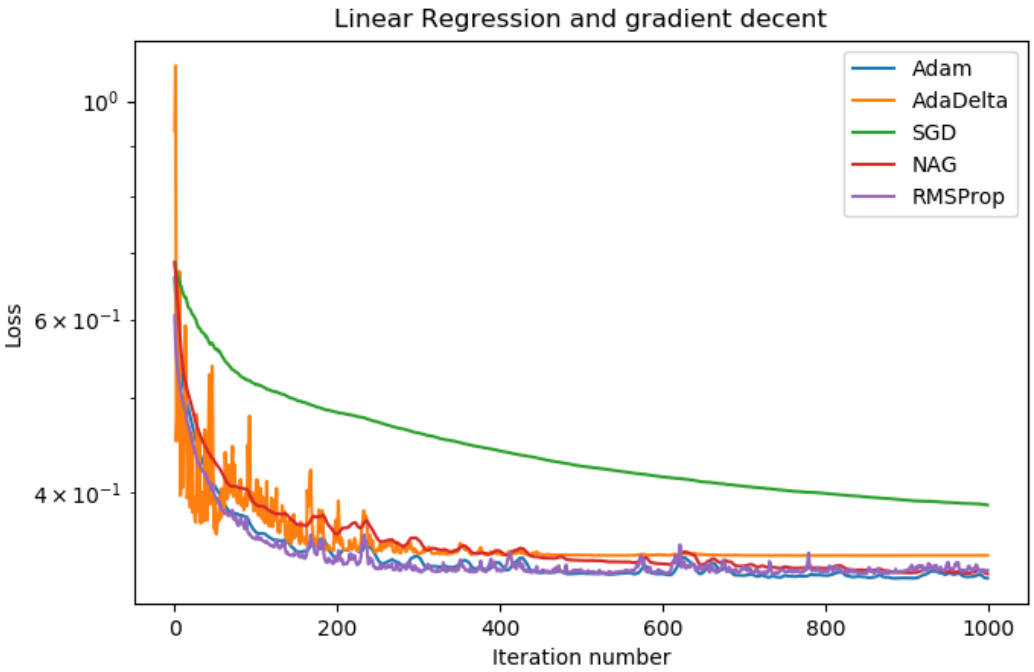
学习率=0.0001

loss 曲线图：

逻辑回归：

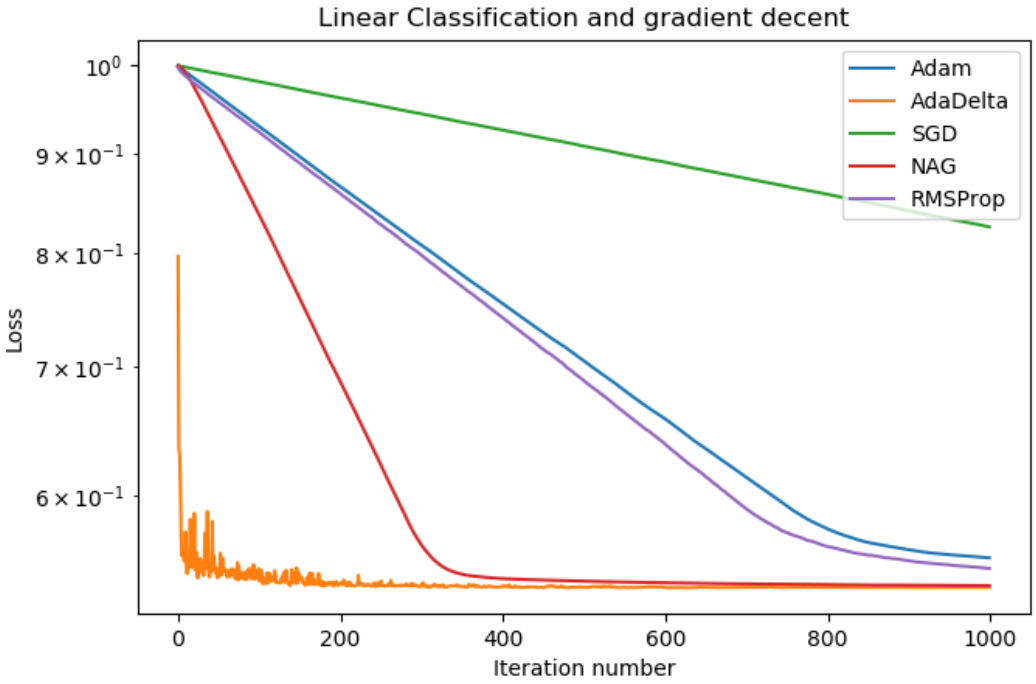


学习率=0.05

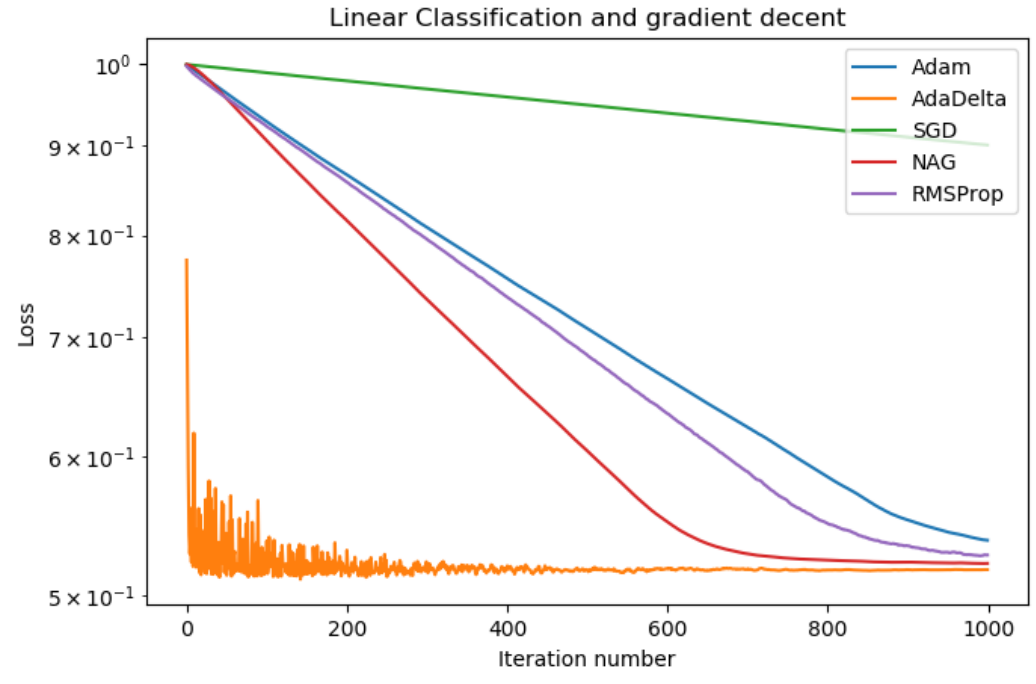


批量数：10

线性分类：



学习率=0.0002　C=1.2



学习率=0.0001　C=0.5

11.实验结果分析:

从调参的过程中可以看出学习率越大，Loss 曲线的波动也越大，批量数越大，Loss 曲线越光滑，但是收敛速度回稍微下降。在随机梯度下降的几种优化算法中，AdaDleta 在两个实验中表现都较好，只是波动幅度会很大，而且可能会提前收敛。Adam 与 RMSProp 对梯度的更新没有 AdaDelta 那么激进，故比较平稳，在逻辑回归中表现很好，但是在线性分类中收敛较慢，可能是线性分类的梯度更新本来就是比较平缓的。

12.对比逻辑回归和线性分类的异同点：

逻辑回归与线性分类本质上都是分类器，但两者也有区别。逻辑回归的主体还是回归，回归对象是 sigmoid 函数, 它将输入映射为一个处于 0 到 1 之间的小数. 得到这个 0 到 1 之间的小数之后人为将其解读成概率, 然后根据事先设定的阈值进行分类.而这里用 SVM 实现的线性分类器是以超平面为决策边界进行分类。

13.实验总结：

通过这次实验，我了解并尝试去实现随机梯度下降，并用 4 种不同的方法对其进行优化，了解到了各种优化方法的优劣，同时对两种不同分类器的本质也有了更深入的了解。