华南理工大学

South China University of Technology

《机器学习》课程实验二报告

学　　院　软件学院

专　　业　软件工程

组　　员　陈佳欣

学　　号　201530611180

邮　　箱　244821405@qq.com

指导教师　吴庆耀

提交日期　2017 年 12 月 16 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: 陈佳欣

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：

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 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}，$L_{RMS\mathrm{Prop}}$，L_{AdaDelta} 和 L_{Adam}。
7.重复步骤 4-6 若干次，画出 L_{NAG}，$L_{RMS\mathrm{Prop}}$，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

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 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}， $L_{RMS\,\mathrm{Prop}}$， L_{AdaDelta} 和 L_{Adam}。

7.重复步骤 4-6 若干次，画出 L_{NAG}， $L_{RMS\,\mathrm{Prop}}$， L_{AdaDelta} 和 L_{Adam} 随迭代次数的变化图。

7. 代码内容:

逻辑回归：

```
def compute_loss(w, x, y):
    n = x.shape[0]
    total = 0
    for z in range(n):
        total += np.log(1 + np.exp(np.sum(-y[z]*x[z]*w)))
    loss = lamb * np.sum(np.square(w)) / 2 + total/n
    print(loss)
    return loss

compute_gradient:
for i in range(iteration):
        num = 0
        w_gradient = np.zeros((x_train.shape[1],1))

        # parts of samples
        pad = np.random.randint(1, 5)
        for j in range(0, x_train.shape[0], pad):
            num += 1
            temp = (1 + np.exp(np.sum(y_train[j] * x_train[j] * w)))
            w_gradient += np.sum(y_train[j] / temp) * x_train[j].T

        dw = learning_rate * lamb * w - learning_rate * w_gradient / num
```

线性分类：

```
def compute_loss(w, b, x, y):
    n = x.shape[1]
    total = 0
    for z in range(x.shape[0]):
        if np.sum((1 - y[z] * (x[z] * w + b[z]))) > 0:
            total += np.sum((1 - y[z] * (x[z] * w + b[z])))
    loss = np.sum(np.square(w)) / (2*n) + C * total
    print(loss)
    return loss
```

```
compute_gradient:
for i in range(iteration):
        w_gradient = np.zeros((x_train.shape[1],1))
        b_gradient = 0

        # parts of samples
        pad = np.random.randint(1, 5)
        for j in range(0, x_train.shape[0], pad):
            if np.sum((1 - y_train[j] * (x_train[j] * w + b[j]))) > 0:
                w_gradient += x_train[j].T * (-1 * y_train[j])
                b_gradient += -y_train[j]
            else:
                w_gradient += 0
                b_gradient += 0

        dw = w + C * w_gradient
        db = C * b_gradient
```

8. 模型参数的初始化方法:

逻辑回归：

init_w = np.ones((x_train.shape[1],1))

线性分类：

init_b = np.zeros((x_train.shape[0],1))

init_w = np.ones((x_train.shape[1],1))

9.选择的 loss 函数及其导数:

逻辑回归：

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\log(1+e^{-y_i\cdot\mathbf{w}^\top\mathbf{x}_i}) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

$$\mathbf{w}' \to \mathbf{w} - \eta\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = (1-\eta\lambda)\mathbf{w} + \eta\frac{1}{n}\sum_{i=1}^{n}\frac{y_i\mathbf{x}_i}{1+e^{y_i\cdot\mathbf{w}^\top\mathbf{x}_i}}$$

线性回归：

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

$$g_{\mathbf{w}}(\mathbf{x}_i) = \begin{cases} -y_i\mathbf{x}_i & 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b) >= 0 \\ 0 & 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b) < 0 \end{cases}$$

Let $g_b(\mathbf{x}_i) = \frac{\partial(\max(0, 1-y_i(\mathbf{w}^\top\mathbf{x}_i+b)))}{\partial b}$

$$g_b(\mathbf{x}_i) = \begin{cases} -y_i & 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b) >= 0 \\ 0 & 1 - y_i(\mathbf{w}^\top\mathbf{x}_i + b) < 0 \end{cases}$$

$$\frac{\partial f(\mathbf{w}, b)}{\mathbf{w}} = \mathbf{w} + C\sum_{i=1}^{N} g_{\mathbf{w}}(\mathbf{x}_i)$$

$$\frac{\partial f(\mathbf{w}, b)}{b} = C\sum_{i=1}^{N} g_b(\mathbf{x}_i)$$

10.实验结果和曲线图:

超参数选择：

逻辑回归：

epoch = 100，learning_rate = 0.1， λ = 0.01

线性分类：

epoch= 50，learning_rate = 0.01， C = 0.01

NAG：

v=0, μ =0.9

RMSprop：

decay_rate =0.9， eps = 1e-6，cache_w = 0

Adam：

eps=1e-8，beta1=0.9, beta2=0.999

AdaDelta：

eps = 1e-8，decay_rate = 0.9)

预测结果（最佳结果）：

逻辑回归：

$L_{NAG} = 0.585176993867$， $L_{RMS\,Prop} = 0.58516947776$， $L_{Adam} = 0.719562461566$

$L_{AdaDelta} = 0.587877465183$

线性分类：

$L_{NAG} = 79.3893856324$， $L_{RMS\,Prop}$ =106.880695729，

$L_{AdaDelta}$ =198.749823057， $L_{Adam} = 302.293927047$

loss 曲线图：

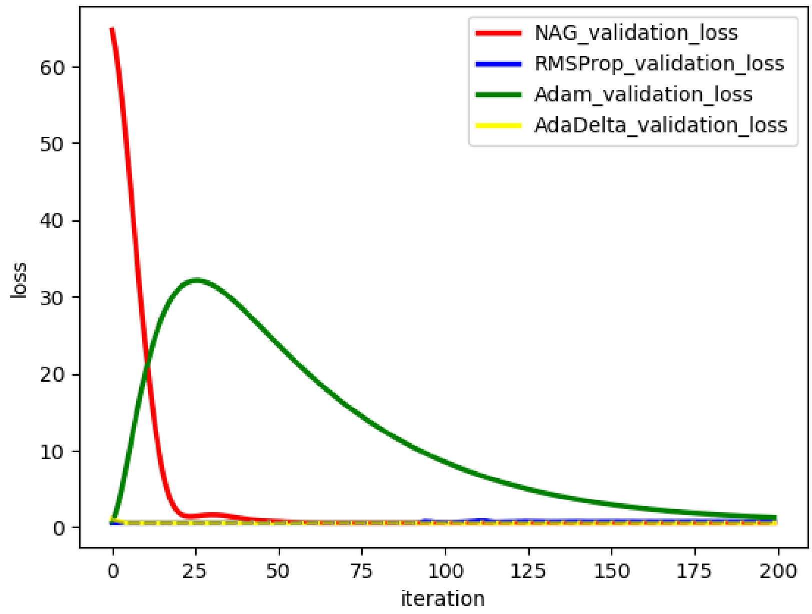

图 1.逻辑回归

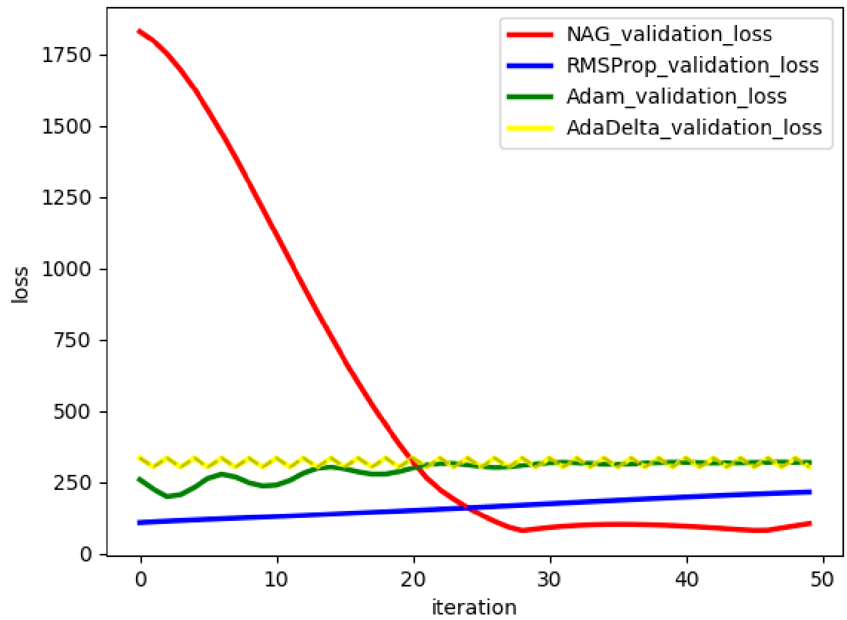

图 2.线性分类

11.实验结果分析:

逻辑回归的收敛速度很快，且四种优化算法最终的误差也较小。总体比较，NAG 在前二十几轮的下降速度很快，后面平缓收敛。Adam 前期不稳定，有较大波动，随后逐渐收敛。RMSProp 最开始就已经很接近最小值，然后缓慢趋近最小值。AdaDelta 与 RMSProp 的效果差不多。

12.对比逻辑回归和线性分类的异同点：

SVM：支持向量机 SVM(Support Vector Machine）作为一种可训练的机器学习方法，依靠小样本学习后的模型参数进行导航星提取，可以得到分布均匀且恒星数量大为减少的导航星表

logistic：logistic 回归(Logistic regression) 与多重线性回归实际上有很多相同之处，最大的区别就在于他们的因变量不同，其他的基本都差不多，正是因为如此，这两种回归可以归于同一个家族，即广义线性模型（generalized linear model）

13.实验总结：

在本次实验的过程中，自己动手去各种论文以及权威网站找了这四种优化算法的公式，对这四种优化算法的了解更加深刻，学到了许多课件上没有的知识。