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　邓泽帅

学　　号　201530611425

邮　　箱　1321730442@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：　邓泽帅

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:

本次实验代码及画图均在 jupyter 上完成。

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}，$L_{RMSprop}$，$L_{AdaDelta}$ 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。

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}，$L_{RMSprop}$，$L_{AdaDelta}$ 和 L_{Adam}。

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

7. 代码内容:

逻辑回归（红色部分的为更新代码）

```
if optimizer == "NAG":
            grad = LogisticRegression.gradient(X_train, y_train, theta+
momemtum*Velocity)
            Velocity = momemtum*Velocity - learning_rate*grad
            theta += Velocity
        elif optimizer == "RMSprop":
            grad = LogisticRegression.gradient(X_train, y_train, theta)
            Velocity = decay_rate*Velocity + (1-decay_rate)*(grad**2)
            theta -= learning_rate*grad/(np.sqrt(epsilon) + Velocity)
        elif optimizer == "Adadelta":
            grad = LogisticRegression.gradient(X_train, y_train, theta)
            Velocity = decay_rate*Velocity + (1-decay_rate)*(grad**2)
            step_update =
-(np.sqrt(update_accumulate+epsilon))*grad/(np.sqrt(Velocity+epsilon))
            theta +=  step_update
            update_accumulate = decay_rate*update_accumulate +
(1-decay_rate)*(step_update**2)
        elif optimizer == "Adam":
            grad = LogisticRegression.gradient(X_train, y_train, theta)
            S = beta1*S + (1-beta1)*grad
            Velocity = beta2*Velocity + (1-beta2)*(grad**2)
            S_t = S/(1 - (beta1**episode))
            Velocity_t = Velocity/(1- (beta2**episode))
            step_update = - learning_rate * S_t/ (np.sqrt(Velocity_t) + epsilon)
            theta += step_update
```

线性分类

```
    if optimizer == "NAG":
            #train with NAG optimizer
            grad = LinearClassification.gradient(X_train, y_train, theta+
momemtum*Velocity, C)
            Velocity = momemtum*Velocity - learning_rate*grad
            theta += Velocity
        elif optimizer == "RMSprop":
```

```
            #train with RMSprop optimizer
            grad = LinearClassification.gradient(X_train, y_train, theta, C)
            Velocity = decay_rate*Velocity + (1-decay_rate)*(grad**2)
            theta -= learning_rate*grad/(np.sqrt(epsilon) + Velocity)
        elif optimizer == "Adadelta":
            #train with Adadelta optimizer
            grad = LinearClassification.gradient(X_train, y_train, theta, C)
            Velocity = decay_rate*Velocity + (1-decay_rate)*(grad**2)
            step_update =
-(np.sqrt(update_accumulate+epsilon))*grad/(np.sqrt(Velocity+epsilon))
            # step_update = - (np.sqrt(Velocity) + epsilon) * grad / (np.sqrt(Velocity)
+ epsilon)
            update_accumulate = decay_rate*update_accumulate +
(1-decay_rate)*(step_update**2)
            theta +=  step_update
        elif optimizer == "Adam":
            #train with Adam optimizer
            grad = LinearClassification.gradient(X_train, y_train, theta, C)
            S = beta1*S + (1-beta1)*grad
            Velocity = beta2*Velocity + (1-beta2)*(grad**2)
            S_t = S/(1 - (beta1**episode))
            Velocity_t = Velocity/(1- (beta2**episode))
            step_update = - learning_rate * S_t/ (np.sqrt(Velocity_t) + epsilon)
            theta += step_update
```

8. 模型参数的初始化方法:

逻辑回归：随机初始化
线性分类：随机初始化

9.选择的 loss 函数及其导数:

逻辑回归：

Loss 函数：

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

导数（一个样本）：

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \left(h_{\mathbf{w}}(\mathbf{x}) - y\right)\mathbf{x}$$

线性分类：

Loss 函数：

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

导数（所有样本）：

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

10.实验结果和曲线图:

超参数选择：

逻辑回归：

1、NAG

'learning_rate': 1e-2, 'epoch': 1000, 'mini_batch': 1024, 'momentum': 0.9

2、RMSprop

'learning_rate': 1e-3, 'epoch': 1000, 'mini_batch': 1024, 'decay_rate': 0.9, "epsilon":1e-7

3、AdaDelta

'learning_rate': 1e-3, 'epoch': 2000, 'mini_batch': 1024, 'decay_rate': 0.9, "epsilon":1e-7

4、Adam

'learning_rate': 1e-2, 'epoch': 2000, 'mini_batch': 1024, 'beta1': 0.9, 'beta1': 0.99, "epsilon":1e-7

线性分类

1、NAG

'learning_rate':1e-5, 'C':9e-1, 'epoch':2000, 'mini_batch': 128, 'momemtum': 0.9

2、RMSprop

'learning_rate':1e-3, 'C':9e-3, 'epoch':2000, 'mini_batch': 128, 'decay_rate': 0.9, 'epsilon':1e-8

3、AdaDelta

'learning_rate':1e-4, 'C':9e-1, 'epoch':2000, 'mini_batch': 128, 'decay_rate': 0.9, 'epsilon':1e-8

4、Adam

'learning_rate':1e-3, 'C':9e-3, 'epoch':2000, 'mini_batch': 128, 'epsilon':1e-8，'beta1':0.9, 'beta2':0.99

预测结果（最佳结果）：

逻辑回归：

1、NAG

lower test loss: 0.296892291059

best classification accuracy: 0.8779296875

2、RMSprop

lower test loss: 0.279613835006

best classification accuracy: 0.8857421875

3、AdaDelta

lower test loss: 0.279025162599

best classification accuracy: 0.890625

4、Adam

lower test loss: 0.274318107181

best classification accuracy: 0.8837890625

线性分类

1、NAG

lower loss: 0.242785857476

classification accuracy: 0.953125

2、RMSprop

lower loss: 0.00287358746064

classification accuracy: 0.8984375

3、AdaDelta

lower loss: 0.176022103304

classification accuracy: 0.953125

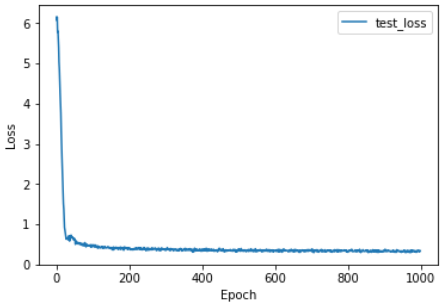
4、Adam

lower loss: 0.195718822408

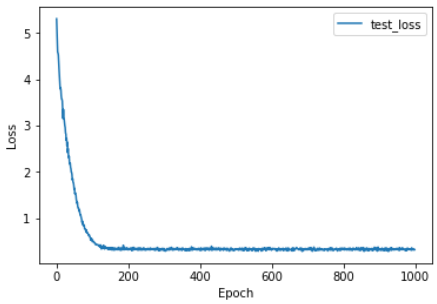
classification accuracy: 0.9296875

loss 曲线图：

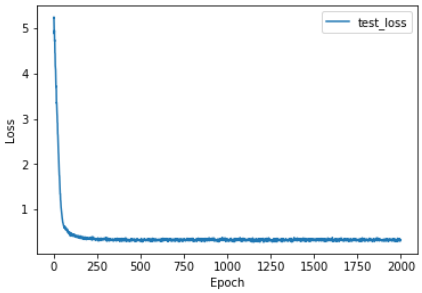
逻辑回归



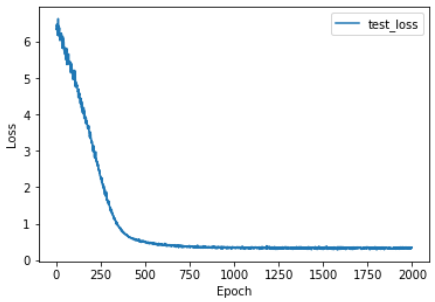
NAG loss 曲线



RMSprop loss 曲线

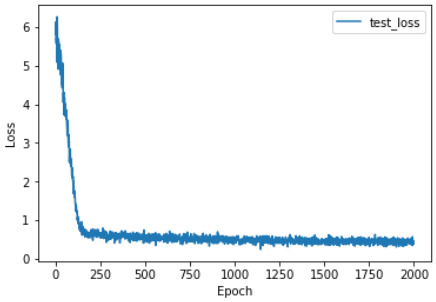


Adadelta loss 曲线

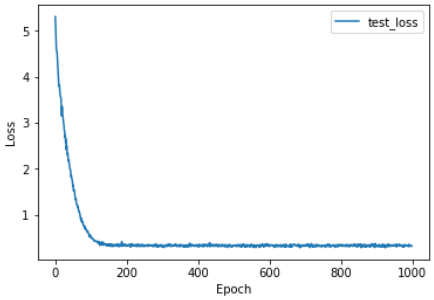


Adam loss 曲线

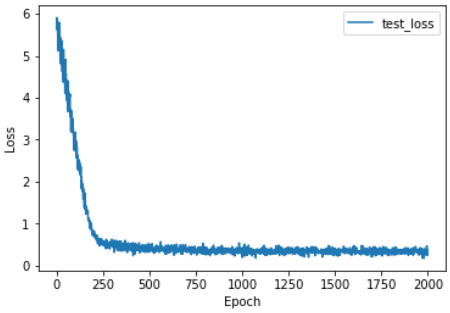
线性分类



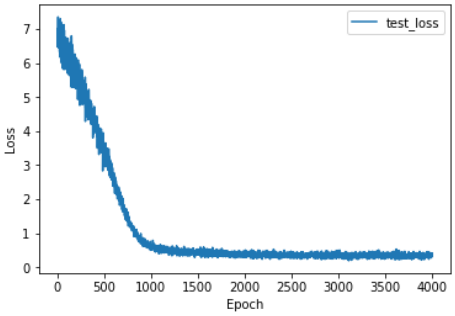
NAG loss 曲线



RMSprop loss 曲线



Adadelta loss 曲线



Adam loss 曲线

11.实验结果分析:

逻辑回归与线性分类都是用了随机梯度下降以及四种优化算法，由于随机梯度下降算法的随机特性，loss 下降的不够平滑，但是总体的趋势是下降的，这说明了随机梯度下架能够代替梯度下降算法，另外四中优化算法也能提高随机梯度下降的性能

12.对比逻辑回归和线性分类的异同点：

相同点：

逻辑回归和线性分类都是处理分类问题，在实验中也都是处理二分类问题，并且都使用了随机梯度下降的算法，以及四种优化算法

不同点：

逻辑分类以及线性分类使用的阈值不同， 并且分类使用的方法

也不同，逻辑分类使用 sigmod 函数分类，而线性分类只是使用一条直线划分

13.实验总结：

通过这次实验，不进熟悉了一种新的算法，逻辑回归，而且还实现了随机梯度下降在大量样本数据的情况下，能够加快模型的训练速度，并且性能并无太大损失，同时，通过四种优化算法进一步提高模型的性能