華南理工大學

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　温志全

学　　号　201530612996

邮　　箱　957918462@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：温志全

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

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。

7. 代码内容:

（针对逻辑回归和线性分类分别填写 8-11 内容）

逻辑回归与随机梯度下降:

```
from sklearn.datasets import load_svmlight_file
import numpy as np
import math
import matplotlib.pylab as plt

data_train = load_svmlight_file('./a9a')
data_test = load_svmlight_file('./a9a.t')


init_w = np.zeros(shape=[124,1])
data_train_x = data_train[0].todense()
data_test_x = data_test[0].todense()

data_test_x = np.hstack((data_test_x,np.zeros(shape=[16281,1])))
data_train_x = np.hstack((data_train_x,np.ones(shape=[32561,1])))
data_test_x = np.hstack((data_test_x,np.ones(shape=[16281,1])))#将 b 放进 w 里，x 相应增加一列全为 1

#正则化系数
lamda = 0.7
#动量系数
gama = 0.9

index = data_train_x.shape[0]
#取样
def samples(X,Y,n):
    samples_index = np.random.randint(0,index, n)#随机选取样本
    X_samples = np.ones((0, 124))
    y_samples = np.ones((0, 1))

    for i in samples_index:
        X_samples = np.r_[X_samples, X[i].reshape(1, X.shape[1])]
        y_samples = np.r_[y_samples, Y[i].reshape(1,1)]

    return X_samples, y_samples
```

```
NAG_loss_list = []
#NAG start
def NAG(X,Y,iter_num):
    w=init_w
    v = np.zeros(shape=[124, 1])
    w_ = w - gama * v
    learning_rate = 0.009
    # 动量
    for i in range(iter_num):#最外层循环
        l = 0
        g_ = 0
        for j in range(data_test_x.shape[0]):#求 loss 的累加部分
            l += math.log(1 + math.exp(-data_test[1][j] * np.dot(w.T,
                data_test_x[j].T).tolist()[0][0]))
        for k in range(X.shape[0]):#求 gradient 累加部分
            g_ +=
        Y[k]/(1+math.exp(Y[k]*np.dot(w_.T,X[k].T).tolist()[0][0]))*X[
        k]
        loss = l/data_test_x.shape[0] + lamda/2 *
        np.dot(w.T,w).tolist()[0][0]   #总的 loss
        gradient = lamda*w_ - g_.T/iter_num  #总的梯度
        v = gama*v + learning_rate*gradient  # 动量更新
        w = w - v  #w 跟新
        w_ = w - gama*v  #求导部分的 w_更新
        NAG_loss_list.append(loss)

    print(NAG_loss_list)
    plt.plot(np.arange(0, iter_num), NAG_loss_list, label=u'NAG')
    plt.legend()
    plt.show()
NAG(samples(data_train_x,data_train[1],50)[0],samples(data_train_x,da
ta_train[1],50)[1],100)

RMSProp_loss_list = []
def RMSProp(X,Y,iter_num):
        #根号部分的参数
    EPSILON = 1e-8
    # 学习率
    learning_rate = 0.008
    G = np.zeros(shape=[124, 1])
    w = init_w
    for i in range(iter_num):
        l = 0
        g_ = 0
```

```
        for j in range(data_test_x.shape[0]):  #求 loss 累加的部分
            l += np.log(1 + np.exp(-data_test[1][j] * np.dot(w.T,
        data_test_x[j].T).tolist()[0][0]))
        for k in range(X.shape[0]):   # 求 gradient 累加的部分
            g_ += Y[k] / (1 + math.exp(Y[k] * np.dot(w.T,
        X[k].T).tolist()[0][0])) * X[k]
        loss = l / data_test_x.shape[0] + lamda / 2 * np.dot(w.T,
        w).tolist()[0][0]   #总的损失
        gradient = lamda * w - g_.T / iter_num  #总的梯度
        G = gama*G + (1 - gama)* np.array(gradient) * np.array(gradient)
        w = w - np.multiply((learning_rate/np.sqrt(G + EPSILON)) ,
        gradient)
        RMSProp_loss_list.append(loss)
    print(RMSProp_loss_list)
    plt.plot(np.arange(0, iter_num), RMSProp_loss_list,
        label=u'RMSProp')
    plt.legend()
    plt.show()

RMSProp(samples(data_train_x,data_train[1],50)[0],samples(data_train_
x,data_train[1],50)[1],50)
```

线性分类与随机梯度下降

```
   from sklearn.datasets import load_svmlight_file
import numpy as np
import math
import matplotlib.pylab as plt

data_train = load_svmlight_file('./a9a')
data_test = load_svmlight_file('./a9a.t')

init_w = np.zeros(shape=[124,1])
data_train_x = data_train[0].todense()
data_test_x = data_test[0].todense()

data_test_x = np.hstack((data_test_x,np.zeros(shape=[16281,1])))
data_train_x = np.hstack((data_train_x,np.ones(shape=[32561,1])))
data_test_x = np.hstack((data_test_x,np.ones(shape=[16281,1])))#将 b
放进 w 里，x 相应增加一列全为 1

C = 0.1
learning_rate = 0.01
```

```
index = data_train_x.shape[0]
#取样
def samples(X,Y,n):
    samples_index = np.random.randint(0,index, n)
    X_samples = np.ones((0, 124))
    y_samples = np.ones((0, 1))

    for i in samples_index:
        X_samples = np.r_[X_samples, X[i].reshape(1, X.shape[1])]
        y_samples = np.r_[y_samples, Y[i].reshape(1,1)]

    return X_samples, y_samples

NAG_loss_list = []
def NAG(X,Y,iter_num):
    gama = 0.9
    v = np.zeros(shape=[124,1])
    w = init_w
    w_ = w - gama*v

    for i in range(iter_num):
        g_ = 0
        l = 0
        for k in range(data_test_x.shape[0]):
            l += max(0,1 -
(data_test[1][k]*np.dot(w.T,data_test_x[k].T)).tolist()[0][0])
        for j in range(X.shape[0]):
            if 1 - (Y[j] * np.dot(w_.T,X[j].T)).tolist()[0][0] >= 0:
                g_ += - Y[j]*X[j]
        gradient = w_ + C * g_.T
        loss = 1/2 * np.dot(w.T,w).tolist()[0][0] + C*l
        v = gama*v + learning_rate*gradient
        w = w - v
        w_ = w - gama * v
        NAG_loss_list.append(loss)

    print(NAG_loss_list)
    plt.plot(np.arange(0, iter_num), NAG_loss_list, label=u'NAG')
    plt.legend()
    plt.show()

#NAG(samples(data_train_x,data_train[1],50)[0],samples(data_train_x,d
ata_train[1],50)[1],50)
```

```
RMSProp_loss_list = []
def RMSProp(X,Y,iter_num):
    #根号里面的参数
    EPSILON = 1e-8
    # 学习率
    gama = 0.9
    learning_rate = 0.008
    G = np.zeros(shape=[124, 1])
    w = init_w
    for i in range(iter_num):
        l = 0
        g_ = 0
        for j in range(data_test_x.shape[0]):
            l += max(0, 1 - (data_test[1][j] * np.dot(w.T,
data_test_x[j].T)).tolist()[0][0])
        for k in range(X.shape[0]):
            if 1 - (Y[k] * np.dot(w.T,X[k].T)).tolist()[0][0] >= 0:
                g_ += - Y[k]*X[k]
        gradient = w + C * g_.T
        loss = 1 / 2 * np.dot(w.T, w).tolist()[0][0] + C * l
        G = gama * G + (1 - gama) * np.multiply(gradient,gradient)
        w = w - np.multiply((learning_rate / np.sqrt(G + EPSILON)),
gradient)
        RMSProp_loss_list.append(loss)
    print(RMSProp_loss_list)
    plt.plot(np.arange(0, iter_num), RMSProp_loss_list,
label=u'RMSProp')
    plt.legend()
    plt.show()
RMSProp(samples(data_train_x,data_train[1],50)[0],samples(data_train_
x,data_train[1],50)[1],50)
```

8. 模型参数的初始化方法:

全 0 初始化

9. 选择的 loss 函数及其导数:

线性分类与随机梯度下降

Hinge_loss:

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

Gradient:

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

逻辑回归与随机梯度下降:

Loss:

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

Gradient:

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

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：

逻辑回归与随机梯度下降:

NAG Learning_rate : 0.009 ;gama :0.9 ;
RMSProp Learning_rate : 0.009;gama:0.9

线性分类与随机梯度下降

NAG : C = 0.1, learn_rate=0.01, gama:0.9
RMSProp Learning_rate = 0.008, gama = 0.9, EPSILON = 1e-8

预测结果（最佳结果）：

线性分类与随机梯度下降

NAG：

```
[[ 0.0291019 ]
 [ 0.16084804]
 [-0.2695342 ]
 [-0.03947725]
 [-0.02778485]
 [-0.15771512]
 [ 0.21105021]
 [-0.03190016]
 [ 0.08281527]
 [ 0.05237764]
 [-0.17719903]
 [ 0.        ]
 [ 0.        ]
 [-0.05413564]
 [ 0.20994214]
 [ 0.0156821 ]
 [-0.23631521]
 [-0.08201974]
 [-0.14132924]
 [ 0.09937063]
 [-0.08932523]
 [-0.14172237]
 [ 0.        ]
 [ 0.10552511]
 [ 0.        ]
 [-0.04462638]
 [ 0.        ]
 [ 0.03608666]
```

```
[ 0.03568889]
[ 0.        ]
[ 0.06884244]
[ 0.        ]
[-0.07535686]
[ 0.        ]
[-0.10437937]
[-0.14172237]
[ 0.09937063]
[ 0.10552511]
[-0.10564035]
[-0.0934999 ]
[ 0.16328864]
[-0.15217858]
[-0.00239991]
[-0.06394983]
[ 0.00189322]
[ 0.        ]
[ 0.        ]
[ 0.14577179]
[-0.08609225]
[ 0.0874467 ]
[-0.03648387]
[ 0.03568889]
[ 0.00189322]
[ 0.00189322]
[-0.1164772 ]
[-0.06450906]
[-0.01858822]
```

```
[ 0.        ]
[ 0.02888561]
[ 0.        ]
[-0.06783696]
[ 0.31181692]
[-0.02566293]
[-0.16582647]
[-0.05394724]
[-0.14538967]
[-0.10303893]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[-0.04380743]
[-0.11178342]
[-0.03506294]
[-0.0528699 ]
[-0.09397646]
[ 0.00376953]
[-0.15061589]
[ 0.13726216]
[-0.07531103]
[-0.05792999]
[-0.05342322]
[-0.09744428]
[ 0.02544081]
[ 0.        ]
```

```
[-0.09882353]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[-0.07346364]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
```

RMSProp：

```
[[ -6.09090385e-02]
 [ -1.36589186e-02]
 [ -1.75888592e-01]
 [    2.95574985e-01]
 [ -1.46598501e-01]
 [ -1.24332807e-01]
 [ -3.75829851e-02]
 [    1.00000000e-01]
 [ -7.21848475e-02]
 [ -1.32816244e-19]
 [    0.00000000e+00]
 [    0.00000000e+00]
```

[0.00000000e+00]
[-5.79570797e-02]
[2.55468008e-02]
[-6.33982956e-02]
[5.20657646e-03]
[-1.03423561e-01]
[1.59762274e-02]
[-5.75459135e-02]
[-6.25374802e-02]
[-8.22403640e-02]
[5.50660029e-02]
[-4.19497527e-14]
[0.00000000e+00]
[0.00000000e+00]
[-1.01585821e-02]
[-2.59326843e-02]
[1.00000000e-01]
[0.00000000e+00]
[3.58446962e-03]
[-7.84409807e-02]
[0.00000000e+00]
[0.00000000e+00]
[-9.99843735e-02]
[-8.22403640e-02]
[-5.75459135e-02]
[-4.19497527e-14]
[5.54327016e-02]
[-2.73028023e-02]
[4.25806262e-02]
[-1.75698473e-01]
[1.00000000e-01]
[-9.77758118e-03]
[-1.06007605e-02]
[0.00000000e+00]
[3.90406194e-02]
[-1.18008236e-05]
[8.55660909e-02]
[-4.31455945e-02]
[-3.94039055e-02]
[-8.35832035e-02]
[-4.60076147e-02]
[1.68988606e-02]
[4.47731064e-02]
[0.00000000e+00]

[-1.55889278e-01]
[0.00000000e+00]
[-3.99303512e-02]
[0.00000000e+00]
[4.55700992e-02]
[-4.51658134e-02]
[-5.07115727e-02]
[-7.02315398e-02]
[-8.48788530e-02]
[3.61157574e-02]
[-5.33271662e-02]
[-1.39404500e-01]
[0.00000000e+00]
[0.00000000e+00]
[-1.27594612e-02]
[-7.44987607e-02]
[-7.70422421e-02]
[-8.13558082e-02]
[-6.17726252e-02]
[-1.07423256e-01]
[1.82027730e-02]
[3.49712929e-03]
[-1.15486311e-01]
[-1.18334614e-01]
[-2.82176451e-02]
[1.28903792e-01]
[-8.23906011e-02]
[0.00000000e+00]
[0.00000000e+00]
[0.00000000e+00]
[0.00000000e+00]
[0.00000000e+00]
[0.00000000e+00]
[0.00000000e+00]
[0.00000000e+00]
[0.00000000e+00]
[2.00811487e-20]
[-1.39849300e-02]
[0.00000000e+00]
[0.00000000e+00]
[0.00000000e+00]
[-3.98265097e-02]
[0.00000000e+00]
[0.00000000e+00]

```
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[  0.00000000e+00]
[ -9.92548272e-02]]
```

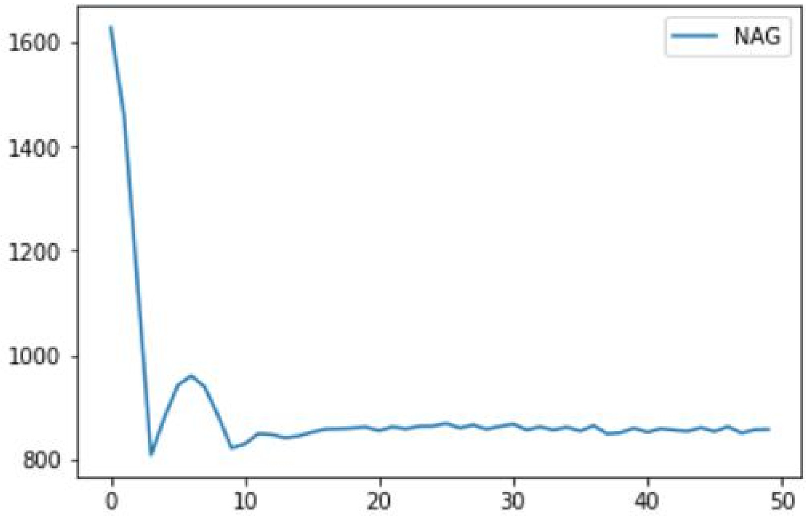
逻辑回归与随机梯度下降

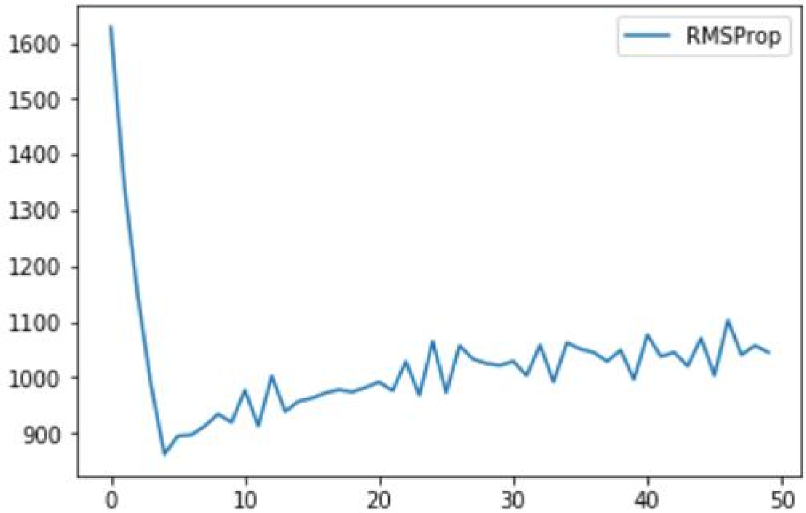
NAG：

w

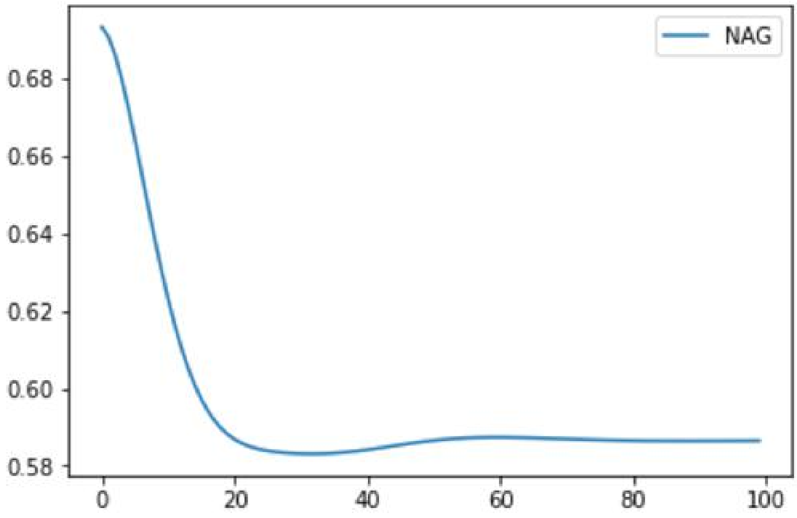
loss 曲线图：

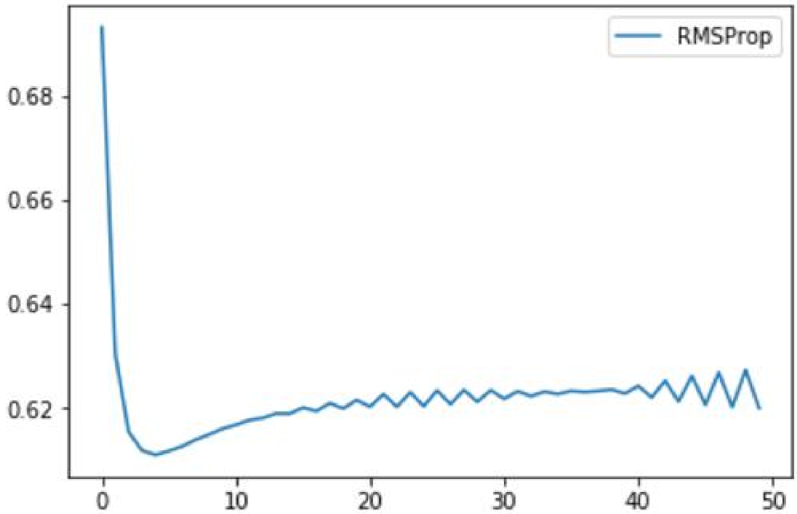
线性分类与随机梯度下降





逻辑回归与随机梯度下降





11.实验结果分析:

线性分类与随机梯度下降

由图可知，经过训练，随着梯度的下降，test loss 在初始是下降的很快，直线下降，但到达一定值后就不怎么下降了，并出现波动情况的情况，可见，loss 已经降到相对最低点了，达到局部最优了

逻辑回归与随机梯度下降

由图可知，经过训练，随着梯度的下降，test loss 在初始是下降的很快，但有一定的平缓，但到达一定值后就不怎么下降了，并出现波动情况的情况，可见，loss 已经降到相对最低点了，达到局部最优了

12.对比逻辑回归和线性分类的异同点：

同：都是使用线性模型加上随机梯度下降的方法进行优化模型，使模型达到局部最优化

异：loss 函数表达方式不一样，线性分类用了 hinge loss ，而逻辑回归用了不通的 loss，同时，梯度的表达方式也不一样，线性分类梯度求导方法通过判断是否分类成功有不同的表示，而逻辑回归是通过 sigmo 函数表示的

13.实验总结：

本次实验我每个实验都只写了两个 loss 函数，但是后面两个损失函数对我来说有一定难度，我尝试过了，但是并没有什么好的效果，可能我对后面两个函数的理解比较浅显，不太懂，因为没弄好的代码会报错，所以并没有加进来