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　李彩利

学　　号　201530611920

邮　　箱　2561170837@qq.com

指导教师　吴庆耀

提交日期　2017 年 11 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 8 日
3. **报告人:李彩利**
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。

4. **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。将数据集切分为训练集和验证集

5. **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。

7. 重复步骤 4-6 若干次，**画出 Loss 函数值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。
7. 重复步骤 4-6 若干次，**画出** Loss 函数值**和随迭代次数的变化图**。

7. 代码内容:

逻辑回归和随机梯度下降

```
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
from numpy import *
import matplotlib.pyplot as plt
import numpy as np
from numpy import random
import math

def get_data():
    data                                                          =
load_svmlight_file("F:\\BaiduNetdiskDownload\\machinelearning\\a9a.txt")
```

```
        return data[0], data[1]

    def function(x,w):
        z = 0
        x = np.ndarray.tolist(x)
    #       w = np.ndarray.tolist(w)
    #       print(x)
    #       print(w)
    #       print(x[0][0])
    #       print(len(x))
        for i in range(len(x)):
            z += x[0][i] * w[i]
        f = 1/(1+exp(-z))
    #       print f
        return f

    x,y=get_data()
    x_train, x_test, y_train, y_test = train_test_split(x,y, test_size =
0.33,random_state=42)
    #b = random.random() #random
    w = random.random(size=(123,1))
    #print(w)

    count = 300
    lr=0.01
    feature=123
    x_train = x_train.todense()
    x_test = x_test.todense()
    x_train_len=len(x_train)
    x_test_len=len(x_test)
    train_new_loss = []
    test_new_loss = []
    h = function(x_train[0],w)
    #print(h)
    #print(y_train[i] * math.log(function(x_train[i],w)) + (1 - y_train[i]) *
math.log(function(x_train[i],w)) for i in range(x_train_len))
    loss = -1*(sum(((1+y_train[i]) * math.log(function(x_train[i],w)) + (1 - y_train[i])
* math.log(1-function(x_train[i],w))) for i in range(x_train_len)))/x_train_len
    #print(loss)
    for k in range(count):
    #       print(w)
    #       print(x_train_len)
        i = random.randint(x_train_len)
        temp = np.ndarray.tolist((function(x_train[i],w) - y_train[i]) * x_train[i])
```

```
#       print(temp)
        for j in range(feature):
            w[j] = w[j] - lr * temp[0][j]
#       print(w)
#       print(w)
#       print(y_train[i])
        train_loss = -1*(sum(((1+y_train[i]) * math.log(function(x_train[i],w)) + (1 - y_train[i]) * math.log(1-function(x_train[i],w))) for i in range(x_train_len)))/x_train_len
        test_loss = -1*(sum(((1+y_test[i]) * math.log(function(x_test[i],w)) + (1 - y_test[i]) * math.log(1-function(x_test[i],w))) for i in range(x_test_len)))/x_test_len

#       if (loss - train_loss) > 0:
#       loss = train_loss
        train_new_loss.append(train_loss)
        test_new_loss.append(test_loss)
#       else:
#           break
# print(train_new_loss)
# print(test_new_loss)
plt.figure(figsize=(8,6))
plt.xlabel('Stochastic Gradient descent - Iteration times')
plt.ylabel('Loss')
plt.plot(range(count), train_new_loss, 'o-', label=u"Training Set")
plt.plot(range(count), test_new_loss, 'r-', label=u"Testing set")
plt.legend()
plt.grid()
plt.show()
```

线性分类和随机梯度下降：

```
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
from numpy import *
import matplotlib.pyplot as plt
import numpy as np
from numpy import random
import math

def get_data():
        data = load_svmlight_file("F:\\BaiduNetdiskDownload\\machinelearning\\a9a.txt")
        return data[0], data[1]
```

```
def function(x,w,b):
        z = 0
        x = np.ndarray.tolist(x)
#       w = np.ndarray.tolist(w)
#       print(x)
#       print(w)
#       print(x[0][0])
#       print(len(x))
        for i in range(len(x)):
                z += x[0][i] * w[i]
        z += b
#       print f
        return z

x,y=get_data()
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.33,random_state=42)
b = random.random() #random
w = random.random(size=(123,1))
#print(w)

count = 200
lr=0.1
feature=123
x_train = x_train.todense()
x_test = x_test.todense()
x_train_len=len(x_train)
x_test_len=len(x_test)
train_new_loss = []
test_new_loss = []
w2 = math.sqrt(sum((w[i])**2 for i in range(feature)))
#print(h)
#print(y_train[i] * math.log(function(x_train[i],w)) + (1 - y_train[i]) * math.log(function(x_train[i],w)) for i in range(x_train_len))
loss = (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len
#print(loss)
for k in range(count):
#       print(w)
#       print(x_train_len)
# w............
        i = random.randint(x_train_len)
        k = -1*(y_train[i] * function(x_train[i],w,b)<1)*y_train[i]
#         print(k)
```

```
#         print((x_train[i]*y_train[i]))
#       print(y_train[i].type)
#       print(np.float64(max(0,k)).type)
        temp_w = np.ndarray.tolist(k*x_train[i])
        for j in range(feature):
            w[j] -= lr * temp_w[0][j]
#       print(w)
#       print(w)
#       print(y_train[i])
        w2 = math.sqrt(sum((w[i])**2 for i in range(feature)))
        train_loss = (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len
        test_loss = (sum(max(0,1-y_test[i]*function(x_test[i],w,b)) for i in range(x_test_len)))/x_test_len
#       if (loss - train_loss) > 0:
#       loss = train_loss
        train_new_loss.append(train_loss)
        test_new_loss.append(test_loss)
#       else:
#               break
print(train_new_loss)
print(test_new_loss)
plt.figure(figsize=(8,6))
plt.xlabel('Stochastic Gradient descent - Iteration times')
plt.ylabel('Loss')
plt.plot(range(count), train_new_loss, 'o-', label=u"Training Set")
plt.plot(range(count), test_new_loss, 'r-', label=u"Testing set")
plt.legend()
plt.grid()
plt.show()
```

线性分类和 NAG：

```
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
from numpy import *
import matplotlib.pyplot as plt
import numpy as np
from numpy import random
import math

def get_data():
    data = load_svmlight_file("F:\\BaiduNetdiskDownload\\machinelearning\\a9a.txt")
```

```
        return data[0], data[1]

def function(x,w,b):
        z = 0
        x = np.ndarray.tolist(x)
#       w = np.ndarray.tolist(w)
#       print(x)
#       print(w)
#       print(x[0][0])
#       print(len(x))
        for i in range(len(x)):
                z += x[0][i] * w[i]
        z += b
#       print f
        return z

x,y=get_data()
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.33,random_state=42)
b = random.random() #random
w = random.random(size=(123,1))
w1 = w
#print(w)

count = 20
lr=0.00001
v = zeros([123,1])
r = 0.9
feature=123
x_train = x_train.todense()
x_test = x_test.todense()
x_train_len=len(x_train)
x_test_len=len(x_test)
train_new_loss = []
test_new_loss = []
#print(h)
#print(y_train[i] * math.log(function(x_train[i],w)) + (1 - y_train[i]) * math.log(function(x_train[i],w)) for i in range(x_train_len))
loss = (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len
#print(loss)
for k in range(count):
#       v = np.ndarray.tolist(v)
#       print(w)
```

```
#	print(x_train_len)
# w...........
	i = random.randint(x_train_len)
#	print(w[1])
#	print(v[1])
#	print(v[0][1])
	for i in range(feature):
		w1[i] = w[i] - r * v[i]
	k = -1*(y_train[i] * function(x_train[i],w1,b)<1)*y_train[i]
	x = np.ndarray.tolist(x_train[i])
	for j in range(feature):
		v[j] = r * v[j] + lr * k * x[0][j]

#	print(v)
	temp_v = np.ndarray.tolist(v)
#	print(temp_v)
#	temp_w = np.ndarray.tolist(v)
	for j in range(feature):
		w[j] -= temp_v[j]
#	print(w)
#	print(w)
#	print(y_train[i])

	train_loss = (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len
	test_loss = (sum(max(0,1-y_test[i]*function(x_test[i],w,b)) for i in range(x_test_len)))/x_test_len
#	if (loss - train_loss) > 0:
#	loss = train_loss
	train_new_loss.append(train_loss)
	test_new_loss.append(test_loss)
#	else:
#		break
print(train_new_loss)
print(test_new_loss)
plt.figure(figsize=(8,6))
plt.xlabel('Gradient descent - Iteration times')
plt.ylabel('Loss')
plt.plot(range(count), train_new_loss, 'o-', label=u"Training Set")
plt.plot(range(count), test_new_loss, 'r-', label=u"Testing set")
plt.legend()
plt.grid()
plt.show()
```

8. 模型参数的初始化方法:随机初始化

9.选择的 loss 函数及其导数:

逻辑回归

$$J(w) = -\frac{1}{n}\left[\sum_{i=1}^{n}(1+y_i)\log h_w(x_i) + (1-y_i)\log(1-h_w(x_i))\right]$$

$$\frac{\partial J(w)}{\partial w} = (h(x)-y)x$$

线性分类

$$L(f) = \sum_{n} \max(0, 1-\hat{y}\cdot f(x))$$

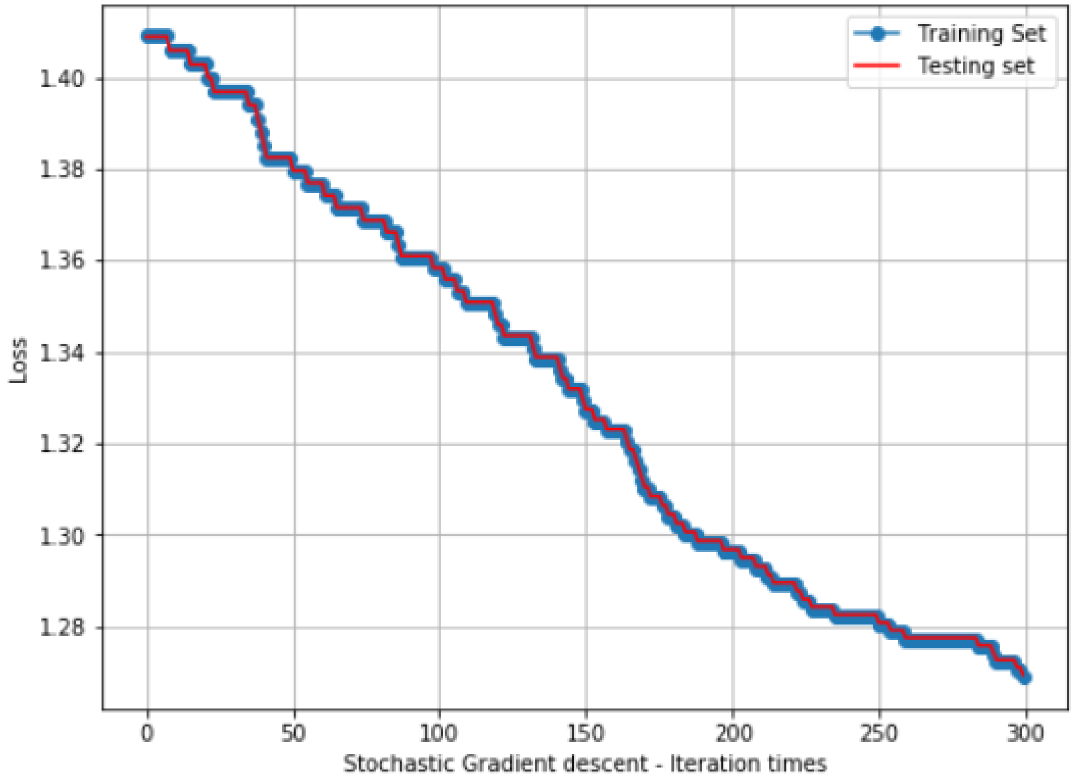
$$\frac{\partial L(f)}{\partial w} = \cancel{\Sigma} -\delta(\hat{y}_n f(x^n) < 1)\hat{y}^n x$$

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：0.1

预测结果（最佳结果）：1.1

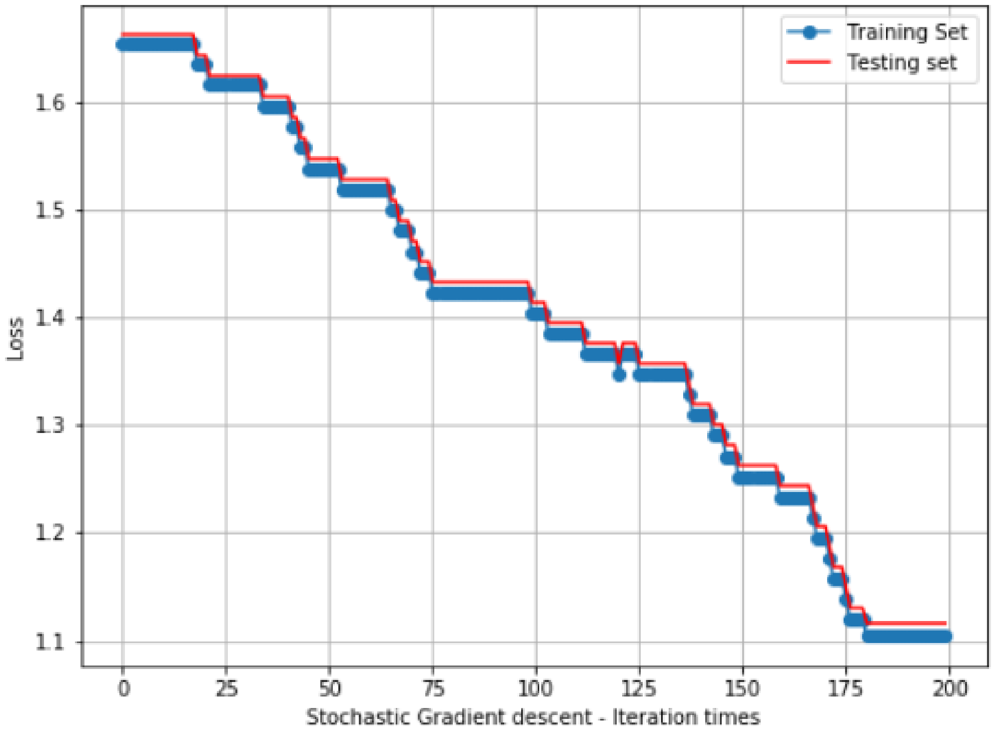
loss 曲线图：



超参数选择：0.01

预测结果（最佳结果）：1.1

loss 曲线图：



11.实验结果分析:

逻辑回归和线性分类的 loss 越来越小，采用 *NAG，RMSProp，AdaDelta* 和 *Adam* 等参数更新法可以加快速度

12.对比逻辑回归和线性分类的异同点：

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. 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.

13.实验总结：

学会了逻辑回归与线性回归、随机梯度下降和梯度下降的区别