Lab8_homework

2018 年 11 月 3 日

1 实验目的

sigmoid 函数和 Logistic 回归分类器

2 实验环境

- windows 10 64 位
- anaconda3
- jupyter notebook

3 实验步骤

3.1 logistic 回归实现

导入必要的库:

In [10]:
```
from numpy import *
import regres
```

构建数据集和标签向量:

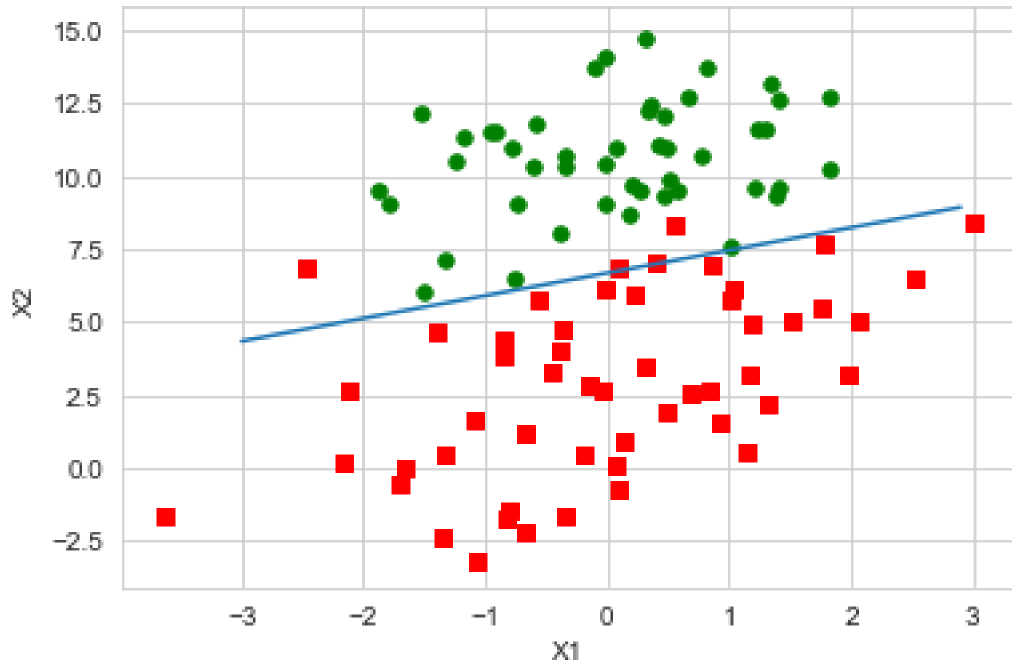
In [11]:
```
dataArr, labelMat = regres.loadDataSet()
```

画出决策边界:

In [12]:
```
weights = regres.gradAscent(dataArr, labelMat)
weights
```

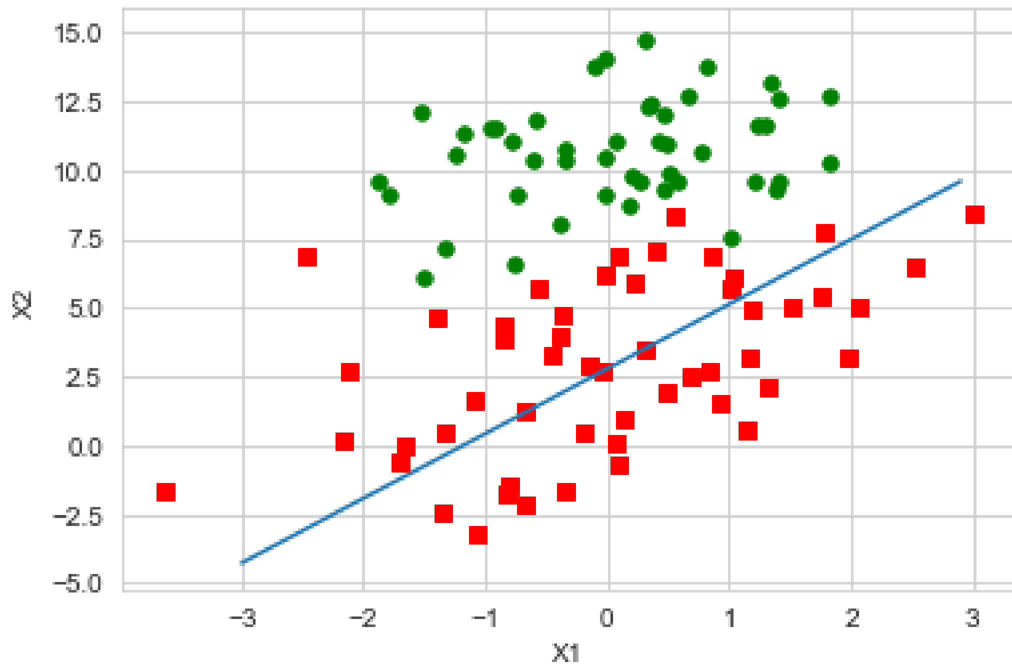
Out[12]:
```
matrix([[ 4.12414349],
        [ 0.48007329],
        [-0.6168482 ]])
```

```
In [13]: weights = regres.plotBestFit(weights.getA())
```



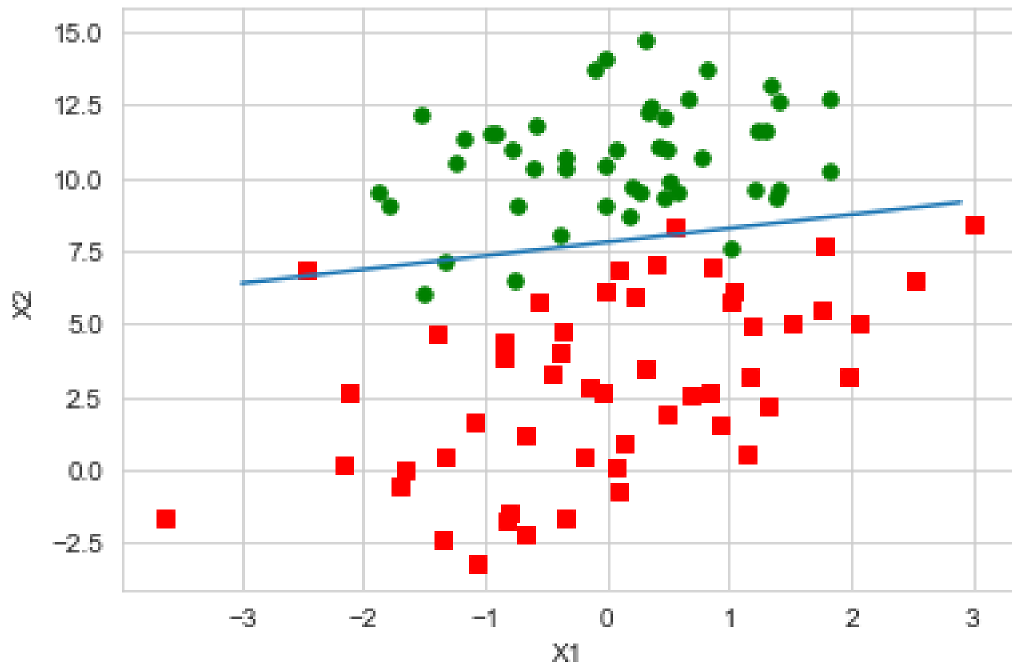
使用随机梯度法训练:

```
In [14]: dataArr,labelMat = regres.loadDataSet()
         weights = regres.stocGradAscent0(array(dataArr), labelMat)
         regres.plotBestFit(weights)
```



改进后的随机梯度法:

```
In [15]: dataArr,labelMat = regres.loadDataSet()
         weights = regres.stocGradAscent1(array(dataArr), labelMat,500)
         regres.plotBestFit(weights)
```



4 从疝气症预测病马的死亡率

```
In [7]: regres.multiTest()
```

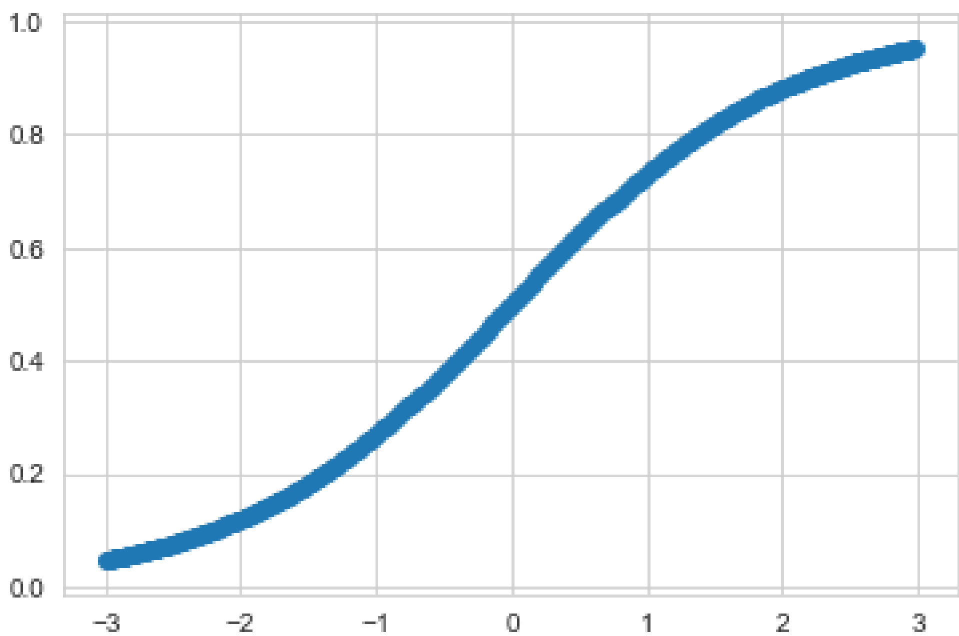
```
the error rate of this test is: 0.343284
the error rate of this test is: 0.328358
the error rate of this test is: 0.313433
the error rate of this test is: 0.283582
the error rate of this test is: 0.373134
the error rate of this test is: 0.283582
the error rate of this test is: 0.358209
the error rate of this test is: 0.373134
the error rate of this test is: 0.402985
the error rate of this test is: 0.343284
after 10 iterations,the average error rate is: 0.340299
```

5 操作练习

实现 sigmoid 函数, 并画出其函数形状

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
def sigmoid(inX):
    return 1.0/(1.0+exp(-inX))
x = np.arange(-3,3,0.02)
y = list(map(sigmoid, x))
plt.scatter(x, y)
```

Out[8]:
```
<matplotlib.collections.PathCollection at 0x2a12d0b1470>
```



实验比较三种梯度下函数的异同

In [ ]:
```
# 绘制回归系数和迭代次数的关系
def plotWeights(weights_array1,weights_array2, weights_array3):
    # 设置汉字格式
```

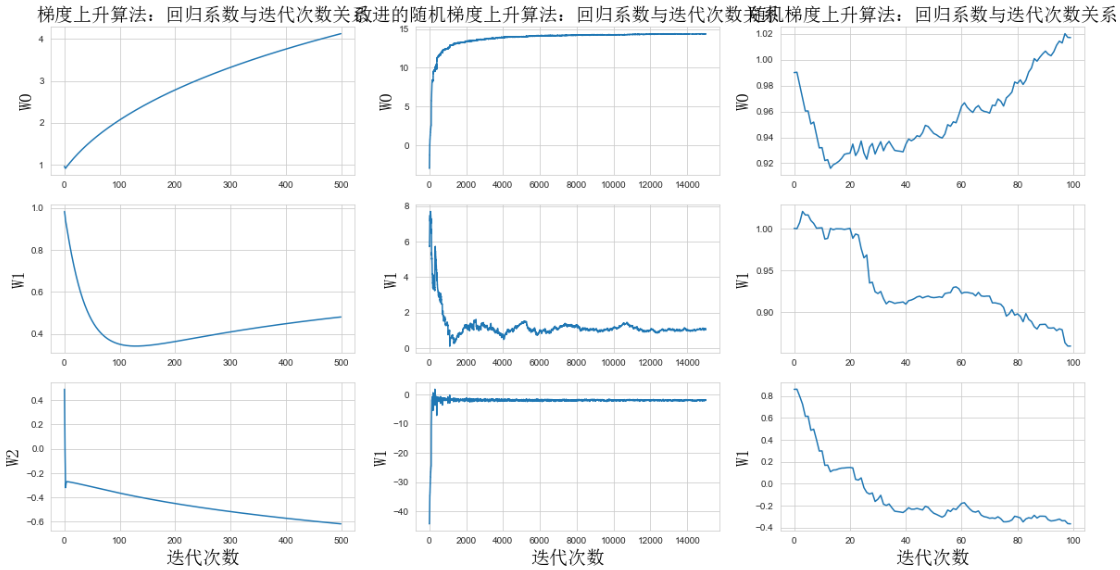
```python
font = FontProperties(fname=r"c:\windows\fonts\simsun.ttc", size=14)
# 将 fig 画布分隔成 3 行 3 列，不共享 x 轴和 y 轴,fig 画布的大小为 (20, 10)
# 当 nrow=3,nclos=3 时，代表 fig 画布被分为九个区域,axs[0][0] 表示第一行第一列
fig, axs = plt.subplots(nrows=3, ncols=3,sharex=False, sharey=False, figsize=(20,1
x1 = arange(0, len(weights_array1), 1)
# 绘制 w0 与迭代次数的关系
axs[0][0].plot(x1,weights_array1[:,0])
axs0_title_text = axs[0][0].set_title(u'梯度上升算法：回归系数与迭代次数关系',FontPr
axs0_ylabel_text = axs[0][0].set_ylabel(u'W0',FontProperties=font)
plt.setp(axs0_title_text, size=20, weight='bold', color='black')
plt.setp(axs0_ylabel_text, size=20, weight='bold', color='black')
# 绘制 w1 与迭代次数的关系
axs[1][0].plot(x1,weights_array1[:,1])
axs1_ylabel_text = axs[1][0].set_ylabel(u'W1',FontProperties=font)
plt.setp(axs1_ylabel_text, size=20, weight='bold', color='black')
# 绘制 w2 与迭代次数的关系
axs[2][0].plot(x1,weights_array1[:,2])
axs2_xlabel_text = axs[2][0].set_xlabel(u'迭代次数',FontProperties=font)
axs2_ylabel_text = axs[2][0].set_ylabel(u'W2',FontProperties=font)
plt.setp(axs2_xlabel_text, size=20, weight='bold', color='black')
plt.setp(axs2_ylabel_text, size=20, weight='bold', color='black')

x2 = arange(0, len(weights_array2), 1)
# 绘制 w0 与迭代次数的关系
axs[0][1].plot(x2,weights_array2[:,0])
axs0_title_text = axs[0][1].set_title(u'改进的随机梯度上升算法：回归系数与迭代次数关
axs0_ylabel_text = axs[0][1].set_ylabel(u'W0',FontProperties=font)
plt.setp(axs0_title_text, size=20, weight='bold', color='black')
plt.setp(axs0_ylabel_text, size=20, weight='bold', color='black')
# 绘制 w1 与迭代次数的关系
axs[1][1].plot(x2,weights_array2[:,1])
axs1_ylabel_text = axs[1][1].set_ylabel(u'W1',FontProperties=font)
plt.setp(axs1_ylabel_text, size=20, weight='bold', color='black')
# 绘制 w2 与迭代次数的关系
axs[2][1].plot(x2,weights_array2[:,2])
axs2_xlabel_text = axs[2][1].set_xlabel(u'迭代次数',FontProperties=font)
axs2_ylabel_text = axs[2][1].set_ylabel(u'W1',FontProperties=font)
```

```
    plt.setp(axs2_xlabel_text, size=20, weight='bold', color='black')
    plt.setp(axs2_ylabel_text, size=20, weight='bold', color='black')

    x2 = arange(0, len(weights_array3), 1)
    # 绘制 w0 与迭代次数的关系
    axs[0][2].plot(x2,weights_array3[:,0])
    axs0_title_text = axs[0][2].set_title(u'随机梯度上升算法：回归系数与迭代次数关系',Fo
    axs0_ylabel_text = axs[0][2].set_ylabel(u'W0',FontProperties=font)
    plt.setp(axs0_title_text, size=20, weight='bold', color='black')
    plt.setp(axs0_ylabel_text, size=20, weight='bold', color='black')
    # 绘制 w1 与迭代次数的关系
    axs[1][2].plot(x2,weights_array3[:,1])
    axs1_ylabel_text = axs[1][2].set_ylabel(u'W1',FontProperties=font)
    plt.setp(axs1_ylabel_text, size=20, weight='bold', color='black')
    # 绘制 w2 与迭代次数的关系
    axs[2][2].plot(x2,weights_array3[:,2])
    axs2_xlabel_text = axs[2][2].set_xlabel(u'迭代次数',FontProperties=font)
    axs2_ylabel_text = axs[2][2].set_ylabel(u'W1',FontProperties=font)
    plt.setp(axs2_xlabel_text, size=20, weight='bold', color='black')
    plt.setp(axs2_ylabel_text, size=20, weight='bold', color='black')
    plt.show()
```

In [9]:
```
import regres_paint
dataMat, labelMat = regres_paint.loadDataSet()
# 对梯度法的函数做出一些修改，使其能够返回每次迭代后的 weights 值组成的 array
weights1,weights_array1 = regres_paint.stocGradAscent1(array(dataMat), labelMat)
weights2,weights_array2 = regres_paint.gradAscent(dataMat, labelMat)
weights3,weights_array3 = regres_paint.stocGradAscent0(array(dataMat), labelMat)
regres_paint.plotWeights(weights_array2, weights_array1,weights_array3)
```



使用朴素贝叶斯分类器对该数据集进行分类

In [17]:
```python
from sklearn.naive_bayes import GaussianNB
frTrain = open('horseColicTraining.txt')
frTest = open('horseColicTest.txt')
trainingSet = []; trainingLabels = []
for line in frTrain.readlines():
        currLine = line.strip().split('\t')
        lineArr = []
        for i in range(21):
            lineArr.append(float(currLine[i]))
        trainingSet.append(lineArr)
        trainingLabels.append(float(currLine[21]))
testingSet = []; testingLabels = []
for line in frTest.readlines():
        currLine = line.strip().split('\t')
        lineArr =[]
        for i in range(21):
            lineArr.append(float(currLine[i]))
        testingSet.append(lineArr)
        testingLabels.append(float(currLine[21]))
classifier = GaussianNB().fit(trainingSet, trainingLabels)
```

```
test_accuracy = classifier.score(testingSet, testingLabels)
test_accuracy
```

Out[17]: 0.6716417910447762

利用 matplotlib 画出决策边界, 即训练后的 sigmoid 函数曲线

In [9]:
```
import regres
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')

frTrain = open('horseColicTraining.txt')# 打开训练集文件
frTest = open('horseColicTest.txt')      # 打开测试集文件
# 将训练集数据和对应标签存放到 trainingSet 和 trainingLabels 中
trainingSet = []; trainingLabels = []
for line in frTrain.readlines():
    currLine = line.strip().split('\t')
    lineArr =[]
    for i in range(21):
        lineArr.append(float(currLine[i]))
    trainingSet.append(lineArr)
    trainingLabels.append(float(currLine[21]))
    # 训练算法, 得到回归系数 trainWeights
trainWeights = regres.stocGradAscent1(array(trainingSet), trainingLabels, 1000)
testingSet = []; testingLabels = []
for line in frTest.readlines():
    currLine = line.strip().split('\t')
    lineArr = []
    for i in range(21):
        lineArr.append(float(currLine[i]))
    testingSet.append(lineArr)
    testingLabels.append(float(currLine[21]))
probs = []
for i in range(len(testingSet)):
    prob = regres.sigmoid(sum(testingSet[i] * trainWeights))
    probs.append(prob)
probs = array(probs); testingLabels = array(testingLabels)
```

```python
positive_list = []; negative_list = []
for i in range(len(testingSet)):
    if testingLabels[i] == 1:
        positive_list.append(sum(testingSet[i] * trainWeights))
    else:
        negative_list.append(sum(testingSet[i] * trainWeights))
plt.scatter(positive_list, probs[where(testingLabels == 1)], label='positive')
plt.scatter(negative_list, probs[where(testingLabels == 0)], label='negative')
plt.legend(['positive', 'negative'])
plt.show()
```