【华子】机器学习009-用逻辑回归分类器解决多分类问题 -

(本文所使用的Python库和版本号: Python 3.5, Numpy 1.14, scikit-learn 0.19, matplotlib 2.2)

前面的**[机器学习008]**已经讲解了用简单线性分类器解决二分类问题，但是对于多分类问题，我们该怎么办了?

此处介绍一种用于解决多分类问题的分类器：逻辑回归。虽然名称中含有回归二字，但逻辑回归不仅可以用来做回归分析，也可以用来做分类问题。逻辑回归是机器学习领域比较常用的算法，用于估计样本所属类别的可能性，关于逻辑回归的更深层次的公式推导，可以参看https://blog.csdn.net/devotion987/article/details/78343834。

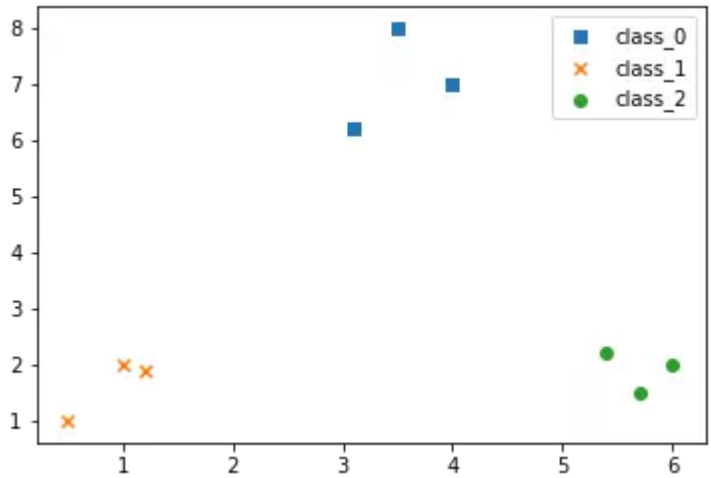
1. 准备数据集

此处我们自己构建了一些简单的数据样本作为数据集，首先我们要分析该数据集，做到对数据集的特性了然如胸。

```
# 首先准备数据集
# 特征向量
X =np.array([[4, 7], [3.5, 8], [3.1, 6.2], [0.5, 1], [1, 2],
             [1.2, 1.9], [6, 2], [5.7, 1.5], [5.4, 2.2]]) # 自定义的数据集
# 标记
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2]) #  三个类别

# 按照类别将数据点画到散点图中
class_0=np.array([feature for (feature,label) in zip(X,y) if label==0])
# print(class_0) # 确保没有问题
class_1=np.array([feature for (feature,label) in zip(X,y) if label==1])
# print(class_1)
class_2=np.array([feature for (feature,label) in zip(X,y) if label==2])
# print(class_2)

# 绘图
plt.figure()
plt.scatter(class_0[:,0],class_0[:,1],marker='s',label='class_0')
plt.scatter(class_1[:,0],class_1[:,1],marker='x',label='class_1')
plt.scatter(class_2[:,0],class_2[:,1],marker='o',label='class_2')
plt.legend()
```



##########################小********结##################################**

1，通过将数据集的y label可以看出，整个数据集有三个类别，每个类别的数据点都聚集到一块，这个可以从散点图中看出，故而此处是典型的多分类问题。

2，此处数据集的样本数比较少（每个类别三个样本），且特征向量只有两个，并且从散点图中可以看出，数据集各个类别都区分的比较开，故而相对比较容易分类。

###

2. 构建逻辑回归分类器

逻辑回归分类器的构建非常简单，如下代码所示，首先我们用该分类器的默认参数做一下分类试试。

```
# 构建逻辑回归分类器
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state=37) # 先用默认的参数
classifier.fit(X, y) # 对国际回归分类器进行训练
```

虽然此处我们构建了逻辑回归分类器， 并且用我们的数据集进行了训练，但训练的效果该怎么查看了？此时我们也没有测试集，所以暂时的，我们将该分类器在训练集上的分类效果画到图中，给出一个直观的分类效果。为了在图中看到分类效果，需要定义一个专门绘制分类器效果展示的函数，如下。

```
# 将分类器绘制到图中
def plot_classifier(classifier, X, y):
    x_min, x_max = min(X[:, 0]) - 1.0, max(X[:, 0]) + 1.0 # 计算图中坐标的范围
    y_min, y_max = min(X[:, 1]) - 1.0, max(X[:, 1]) + 1.0
    step_size = 0.01 # 设置step size
    x_values, y_values = np.meshgrid(np.arange(x_min, x_max, step_size), np.arange(y_min, y_max,
step_size))
    # 构建网格数据
    mesh_output = classifier.predict(np.c_[x_values.ravel(), y_values.ravel()])
    mesh_output = mesh_output.reshape(x_values.shape)
    plt.figure()

    plt.pcolormesh(x_values, y_values, mesh_output, cmap=plt.cm.gray)
```

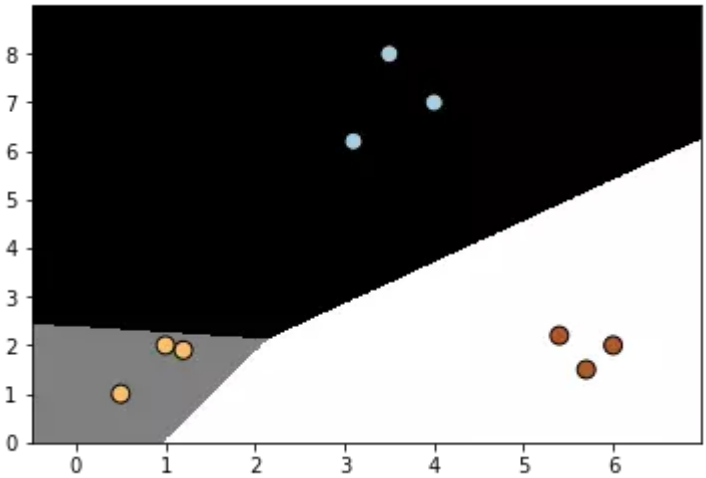
```
    plt.scatter(X[:, 0], X[:, 1], c=y, s=80, edgecolors='black', linewidth=1,
cmap=plt.cm.Paired)
    # specify the boundaries of the figure
    plt.xlim(x_values.min(), x_values.max())
    plt.ylim(y_values.min(), y_values.max())

    # specify the ticks on the X and Y axes
    plt.xticks((np.arange(int(min(X[:, 0])-1), int(max(X[:, 0])+1), 1.0)))
    plt.yticks((np.arange(int(min(X[:, 1])-1), int(max(X[:, 1])+1), 1.0)))

    plt.show()
```

然后直接调用该绘图函数，查看该逻辑回归分类器在训练集上的分类效果。

```
plot_classifier(classifier, X, y)
```



##########################小********结##################################**

1，使用sklearn模块中的LogisticRegression函数可以轻松的定义和训练一个逻辑回归分类器模型。

2，由于此处采用分类器的默认参数，而不是最适合参数，故而得到的分类效果并不是最佳，比如从图中可以看出，虽然该分类模型能够将三个类别区分开来，但是其模型明显还可以继续优化。

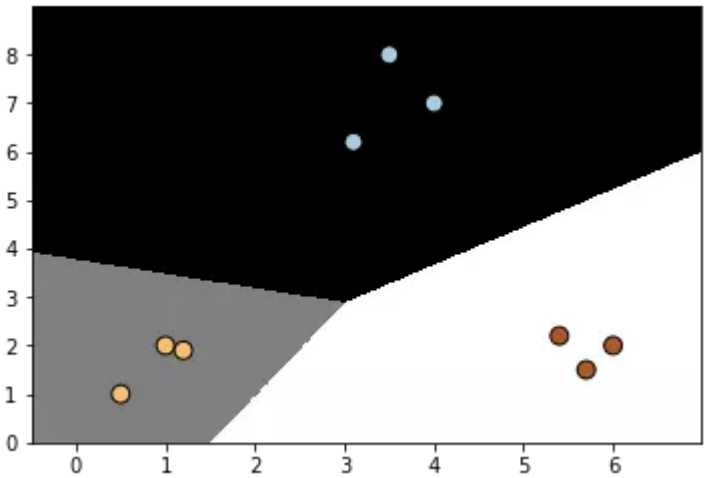
###

3. 对分类模型的优化

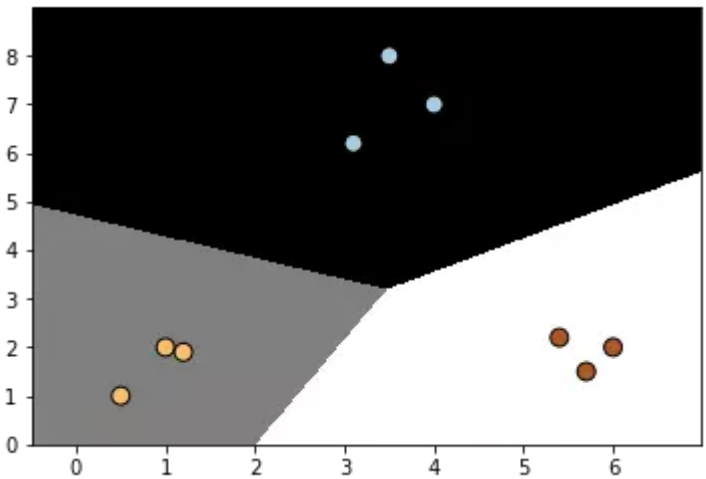
逻辑回归分类器有两个最重要的参数：solver和C，其中参数solver用于设置求解系统方程的算法类型，参数C表示对分类错误的惩罚值，故而C越大，表明该模型对分类错误的惩罚越大，即越不能接受分类发生错误。

此处，作为抛砖引玉，可以优化C值对分类效果的影响，如下，我们随机选择几种C值，然后将分类结果图画出来，凭借直观感受来判断哪一个比较好。当然，更科学的做法是，使用测试集结合各种评估指标来综合评价那个参数组合下的模型最好。

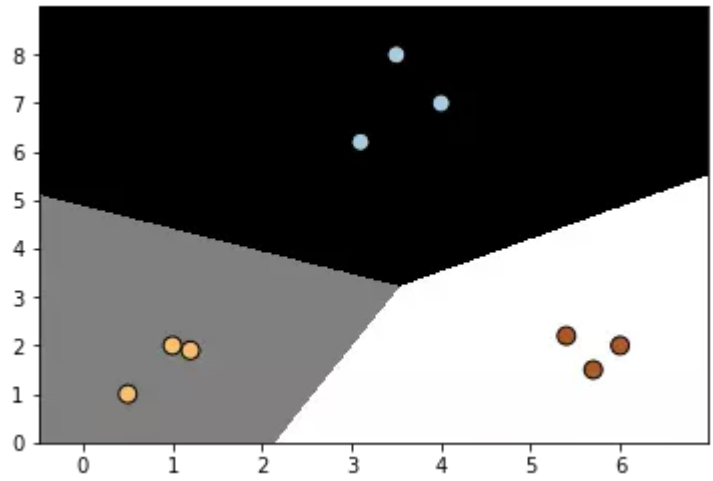
```
# 优化模型中的参数C
for c in [1,5,20,50,100,200,500]:
    classifier = LogisticRegression(C=c,random_state=37)
    classifier.fit(X, y)
    plot_classifier(classifier, X, y)
# 貌似C越多，分类的效果越好。
```



C=5时逻辑回归分类器的分类效果



C=100时逻辑回归分类器的分类效果



C=500时逻辑回归分类器的分类效果

##########################小********结##################################**

1，对模型进行优化是一项体力活，也是最能考验机器学习技术功底的工作，此处作为抛砖引玉，我们仅仅优化了逻辑回归分类器的一个参数。

2，逻辑回归分类器的C值越大，得到的分类器模型就越在两个数据集中间区分开来，这也符合我们的预期，那么，是否有必要在一开始时就设置非常大的C值?

###

参考资料:

1, Python机器学习经典实例，Prateek Joshi著，陶俊杰，陈小莉译