【华子】机器学习008-简单线性分类器解决二分类问题 -

(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)

分类问题，就是将数据点按照不同的类别区分开来，所谓人以类聚，物以群分，就是这个道理。以前的【机器学习001-007】都是讲解的回归问题，两者的不同之处在于：回归输出的结果是实数，并且一般是连续的实数值，而分类问题的输出结果是离散的某一个类别或不同类别的概率。

最简单的分类问题是二元分类，将整个样本划分为两个类别，比如将整个人类分为男人和女人（泰国人妖不在考虑范围内，呵呵）。稍微复杂一点的分类问题是多元分类，它将整个样本划分为多个（一般大于两个）不同类别，比如将家禽数据集可以划分为：鸡，鸭，鹅等，将家畜样本划分为：狗，猪，牛，羊等等。

下面从一个最简单的二元分类问题入手，看看二元分类器是如何构建的。

1. 准备数据集

由于二元分类问题比较简单，此处我们自己构建了一些数据点，并将这些数据点按照不同类别放入不同变量中，比如把所有第0类别的数据点都放置到class_0中，把所有第1类别的数据点放入class_1中，如下所示。

```
# 首先准备数据集
# 特征向量
X = np.array([[3,1], [2,5], [1,8], [6,4], [5,2], [3,5], [4,7], [4,-1]]) # 自定义的数据集
# 标记
y = [0, 1, 1, 0, 0, 1, 1, 0]

# 由于标记中只含有两类，故而将特征向量按照标记分割成两部分
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)
```

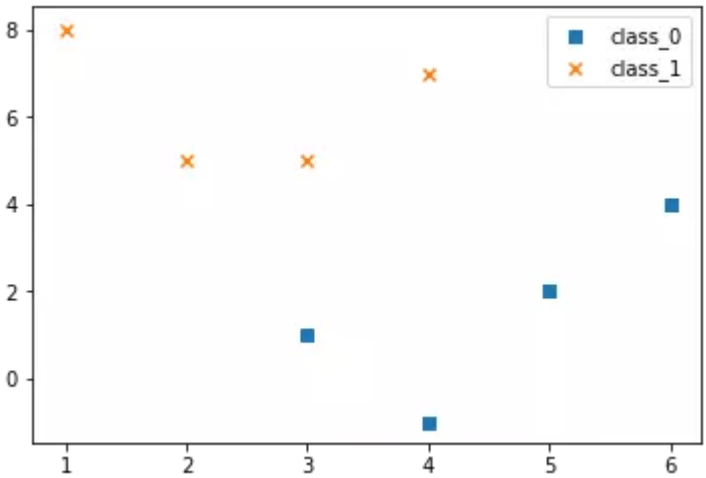
----------输---------出----------

[[3 1] [6 4] [5 2] [4 -1]] [[2 5] [1 8] [3 5] [4 7]]

----------完----------

上面虽然构建了数据点，但是难以直观的看清这个二分类问题的数据点有什么特点，所以为了有更加直观的认识，一般会把数据点的散点图画出来，如下所示：

```
# 在图中画出这两个不同类别的数据集，方便观察不同类别数据的特点
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.legend()
```



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

1，本次研究的二分类问题是极其简单的分类问题，故而构建了8个样本的两个类别的数据点，每个类别有四个点。

2，为了更加直观的查看数据点的分布特点，一般我们要把数据点画在平面上，对数据点的分布情况有一个初步的了解，便于后面我们采用哪种分类器。

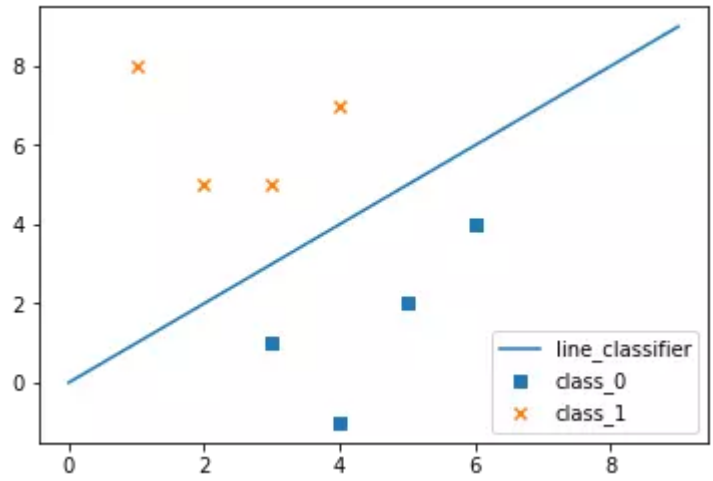
3，本次构建的数据集是由8行2列构成的特征矩阵，即8个样本，每个样本有两个features.

###

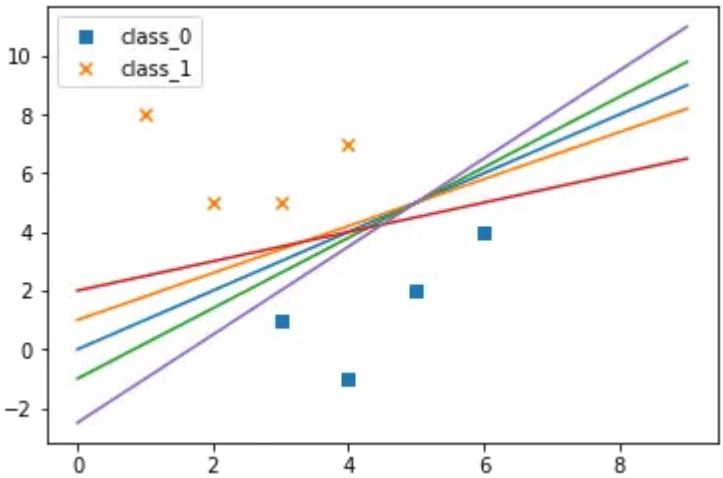
2. 构建简单线性分类器

所谓线性可分问题，是指在平面上可以通过一条直线（或更高维度上的，一个平面）来将所有数据点划分开来的问题，“可以用直线分开”是线性可分问题的本质。相对应的，“不可以用直线分开”便是线性不可分问题的本质，对于线性不可分问题，需要用曲线或曲面来将这些数据分开，对应的就是非线性问题。比如，上面自己定义的数据集可以用简单的直线划分开来，比如可以采用y=x这条直线分开，如下所示：

```
# 从上面图中可以看出，可以画一条直线轻松的将class_0和class_1两个数据点分开
# 其实有很多直线可以起到分类器的效果，此处我们只用最简单的y=x作为演示
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.plot(range(10),range(10),label='line_classifier') # 此处x=range(10), y=x
plt.legend()
```



实际上，可以采用非常多的直线来将本数据集的两个类别区分开来，如下图所示，这些直线是在斜率和截距上稍微调整而来。



那么，这么多直线都可以解决简单分类问题，肯定会有一条最佳直线，能够达到最佳的分类效果。下面，使用sklearn模块中的SGD分类器构建最佳直线分类器。

SGD分类器可以轻松地解决这样的问题：超过10^5的训练样本、超过10^5的features

SGD的优点是：

- 高效
- 容易实现（有许多机会进行代码调优）

SGD的缺点是：

- SGD需要许多超参数：比如正则项参数、迭代数。
- SGD对于特征归一化（feature scaling）是敏感的。

如下代码：

```
# 上面虽然随机的选择了一条直线（y=x）作为分类器，但很多时候我们不知道分类
# 下面构建一个SGD分类器，它使用随机梯度下降法来训练
from sklearn.preprocessing import StandardScaler


ss=StandardScaler() #对数据进行标准化，保证每个维度的特征数据方差为1，均值为0，避免某个特征值过大而成为
```

```
影响分类的主因
X_train=ss.fit_transform(X) # 由于本项目数据集太少，故而全部用来train

# 构建SGD分类器进行训练（只能解决线性模型）
from sklearn.linear_model import SGDClassifier
sgdClassifier=SGDClassifier(random_state=42)
sgdClassifier.fit(X_train,y) # y作为label已经是0,1形式，不需进一步处理

# 使用训练好的SGD分类器对陌生数据进行分类
X_test=np.array([[3,2],[2,3],[2.5,2.4],[2.4,2.5],[5,8],[6.2,5.9]])
X_test=ss.fit_transform(X_test) # test set也要记过同样的处理
test_predicted=sgdClassifier.predict(X_test)
print(test_predicted)
```

----------输---------出----------

[0 1 1 1 1 0]

----------完----------

##########小********结##########**

1，使用sklearn中的SGDClassifier可以对数据集进行简单的线性分类，达到比较好的分类效果。

2，在数据集的特征上，貌似x>y时，数据属于class_0, 而x<y时，数据属于class_1，SGDClassifier模型在测试数据集上也基本能够正确划分，只有在x和y大体相等的关键点处容易出现错误判断。

##########

参考资料:

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