【华子】机器学习014-用SVM构建非线性分类模型 -

(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)

支持向量机（Support Vector Machine，SVM）是一种常见的判别方法，其基本模型是在特征空间上找到最佳的分离超平面，使得数据集上的正负样本间隔最大。SVM是用来解决二分类问题的有监督学习算法，其可以解决线性问题，也可以通过引入核函数的方法来解决非线性问题。

支持向量：对线性可分的情形,位于间隔边界上的样本点

超平面：在几何体中，超平面是一维小于其环境空间的子空间。 如果空间是3维的，那么它的超平面是二维平面，而如果空间是二维的，则其超平面是一维线。 该概念可以用于定义子空间维度概念的任何一般空间。能使支持向量和超平面最小距离的最大值； （最佳分类等价于，地主让管家给两个儿子分地，只要让两家之间一样多就可以了。那根红线让两家距离之和离分界线最远就可以了）

1554874344834

SVM是Support Vector Machines（支持向量机）的缩写，可以用来做分类和回归。

SVC是SVM的一种Type，是用来的做分类的，SVR是SVM的另一种Type，是用来的做回归的。

SVM的两个参数 C 和 gamma:
C是惩罚系数，即对误差的宽容度。c越高，说明越不能容忍出现误差,容易过拟合。C越小，容易欠拟合。C过大或过小，泛化能力变差 ;
gamma是选择RBF函数作为kernel后，该函数自带的一个参数。隐含地决定了数据映射到新的特征空间后的分布，gamma越大，支持向量越少，gamma值越小，支持向量越多。支持向量的个数影响训练与预测的速度。默认是'auto'，则会选择1/n_features

1. 准备数据集

首先来加载和查看数据集的一些特性。如下代码加载数据集并查看其基本信息

```
import numpy  as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline

# 准备数据集
data_path='./014-data_multivar.csv'
df=pd.read_csv(data_path,header=None)
print(df.info()) # 查看数据信息，确保没有错误
dataset_X,dataset_y=df.iloc[:,:-1],df.iloc[:,-1]
dataset_X=dataset_X.values
dataset_y=dataset_y.values
```

----------**输**---------**出**----------

RangeIndex: 300 entries, 0 to 299 Data columns (total 3 columns): 0 300 non-null float64 1 300 non-null float64 2 300 non-null int64 dtypes: float64(2), int64(1) memory usage: 7.1 KB None

----------**完**----------

从上面的df.info()函数的结果可以看出，这个数据集有两个特征属性（0,1列，连续的float类型），一个标记（2列，离散的int型，只有两个类别），每一列都没有缺失值。然后来看看这个数据集中数据点的分布情况，如下图所示：

```
# 数据集可视化
def visual_2D_dataset(dataset_X,dataset_y):
    '''将二维数据集dataset_X和对应的类别dataset_y显示在散点图中'''
    assert dataset_X.shape[1]==2,'only support dataset with 2 features'
    plt.figure()
    classes=list(set(dataset_y))
    markers=['.',',','o','v','^','<','>','1','2','3','4','8'
             ,'s','p','*','h','H','+','x','D','d','|']
    colors=['b','c','g','k','m','w','r','y']
    for class_id in classes:
        one_class=np.array([feature for (feature,label) in
                    zip(dataset_X,dataset_y) if label==class_id])
        plt.scatter(one_class[:,0],one_class[:,1],marker=np.random.choice(markers,1)[0],
                    c=np.random.choice(colors,1)[0],label='class_'+str(class_id))
    plt.legend()

visual_2D_dataset(dataset_X,dataset_y)
```

img

2. 用SVM构建线性分类器

这个数据集是一个线性不可分类型，需要用非线性分类器来解决。所以，此处，我就用线性分类器来拟合一下，看看会有什么样的“不好”的结果。

```
# 从数据集的分布就可以看出，这个数据集不可能用直线分开
# 为了验证我们的判断，下面还是使用SVM来构建线性分类器将其分类

# 将整个数据集划分为train set和test set
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y=train_test_split(
    dataset_X,dataset_y,test_size=0.25,random_state=42)

# 使用线性核函数初始化一个SVM对象。
from sklearn.svm import SVC
classifier=SVC(kernel='linear') # 构建线性分类器
classifier.fit(train_X,train_y)
```

----------**输--------出**----------

SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)

----------**完**----------

然后查看一下这个训练好的SVM线性分类器在训练集和测试集上的表现，如下为在训练集上的性能报告：

```
# 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()
```

```
# 模型在训练集上的性能报告:
from sklearn.metrics import classification_report
plot_classifier(classifier,train_X,train_y)  # 分类器在训练集上的分类效果
target_names = ['Class-0', 'Class-1']
y_pred=classifier.predict(train_X)
print(classification_report(train_y, y_pred, target_names=target_names))
```

img

----------**输**--------**出**----------

precision recall f1-score support

Class-0 0.60 0.96 0.74 114 Class-1 0.89 0.35 0.50 111

avg / total 0.74 0.66 0.62 225

----------**完**----------

很明显，从分类效果图和性能报告中，都可以看出这个模型很差，差到姥姥家了。。。 所以，更不用说，在测试集上的表现了，当然是一个差字了得。。。

```
# 分类器在测试集上的分类效果
plot_classifier(classifier,test_X,test_y)

target_names = ['Class-0', 'Class-1']
y_pred=classifier.predict(test_X)
print(classification_report(test_y, y_pred, target_names=target_names))
```

img

----------输--------出----------

precision recall f1-score support

Class-0 0.57 1.00 0.73 36 Class-1 1.00 0.31 0.47 39

avg / total 0.79 0.64 0.59 75

----------完----------

3. 用SVM构建非线性分类器

很明显，用线性分类器解决不了这个数据集的判别问题，所以我们就上马非线性分类器吧。

使用SVM构建非线性分类器主要是考虑使用不同的核函数，此处指讲述两种核函数：多项式核函数和径向基函数。

径向：指在径向平面内通过轴心线的方向，如图；

img

径向基函数是某种沿径向对称的标量函数，通常定义为样本到数据中心之间径向距离（通常是欧氏距离）的单调函数。RBF核是一种常用的核函数。它是支持向量机分类中最为常用的核函数。

20171115203524364

```
# 故而我们使用SVM建立非线性分类器，看看其分类效果
# 使用SVM建立非线性分类器主要是使用不同的核函数
# 第一种：使用多项式核函数：
classifier_poly=SVC(kernel='poly',degree=3) # 三次多项式方程
classifier_poly.fit(train_X,train_y)

# 在训练集上的表现为：
plot_classifier(classifier_poly,train_X,train_y)
# 打印模型报告
target_names = ['Class-0', 'Class-1']
y_pred=classifier_poly.predict(train_X)
print(classification_report(train_y, y_pred, target_names=target_names))
```

img

----------输--------出----------

precision recall f1-score support

Class-0 0.92 0.85 0.89 114 Class-1 0.86 0.93 0.89 111

avg / total 0.89 0.89 0.89 225

----------完----------

```
# 第二种：使用径向基函数建立非线性分类器
classifier_rbf=SVC(kernel='rbf')
classifier_rbf.fit(train_X,train_y)

# 在训练集上的表现为：（见上）
```

img

----------**输**---------**出**----------

precision recall f1-score support

Class-0 0.96 0.96 0.96 114 Class-1 0.96 0.95 0.96 111

avg / total 0.96 0.96 0.96 225

----------**完**----------

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

1. 用SVM构建非线性分类器很简单，只要使用不同的核函数就可以。

2. 径向基函数rbf的分类效果要比多项式核函数的效果更好一些。

##########

参考资料:

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