scikit-learn 实战之监督学习 (/courses/866)

课程名称：scikit-learn 实战之监督学习

一、实验介绍

1.1 实验内容

监督学习（英语：Supervised learning）是机器学习中最为常见、应用最为广泛的分支之一。本次实验将带你了解监督学习中运用十分广泛的支持向量机，并学会使用 scikit-learn 来构建预测模型，用于解决实际问题。

1.2 实验知识点

- 支持向量机理论基础
- 使用支持向量机进行线性分类
- 使用支持向量机进行非线性分类
- 使用支持向量机进行回归预测

1.3 实验环境

- python3

1.4 适合人群

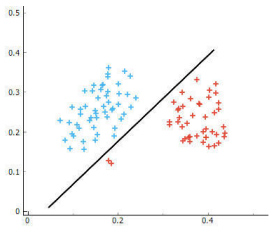
本课程难度为一般，属于初级级别课程，适合具有 Python 基础和线性代数基础，并对机器学习中分类问题感兴趣的用户。

scikit-learn 实战之监督学习 (/courses/866)

二、线性支持向量机

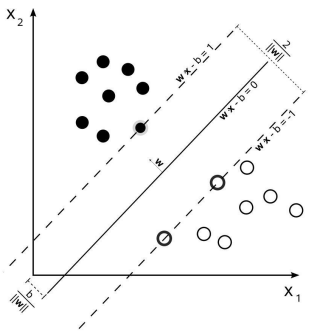
在上一节关于广义线性模型的课程中，我们学习了通过感知机构建一个线性分类器，完成二分类问题。

感知机的学习过程由误分类驱动，即当感知机寻找到没有实例被错误分类时，就确定了分割超平面。这样虽然可以解决一些二分类问题，但是训练出来的模型往往容易出现过拟合。



如上图所示，当感知机在进行分类时，为了照顾左下角的两个红色标记样本，分割线会呈现出如图所示的走向。你应该通过观察就能发现，这条分割线不是特别合理。

于是，Vapnik 于 1963 年提出了支持向量机理论，并将其用于解决线性分类问题。支持向量机也被看成是感知机的延伸。简单来讲，支持向量机就是通过找出一个最大间隔超平面来完成分类。



如图所示，中间的实线是我们找到的分割超平面。这个超平面并不是随手一画，它必须满足两个类别中距离直线最近的样本点，与实线的距离一样且**最大**。这里的最大，也就是上面提

scikit-learn 实战之监督学习 (/courses/866)

到的最大间隔超平面。

许多朋友在一开始接触支持向量机时，对它这个奇怪的名字比较疑惑。其实，支持向量机中的 “支持向量” 指的是上图中，距离分割超平面最近的样本点，即两条虚线上的一个实心点和两个空心点。

接下来我们就来通过 scikit-learn 对上面的红蓝样本数据分别进行线性支持向量机和感知机分类实验。

首先，我们获取上图对应的数据文件。

☞ **示例代码：**

```
!wget http://labfile.oss.aliyuncs.com/courses/866/data.csv
```

☞ **动手练习：**

接下来，我们需要先导入数据文件，这里使用到了 pandas。如果你没有用过也不必担心，我们只是使用了其中导入 csv 文件的一个方法。

☞ **示例代码：**

```
import pandas as pd # 导入 pandas 模块

# 读取 csv 数据文件
df = pd.read_csv("data.csv", header=0)
df.head()
```

☞ **动手练习：**

你可以直接通过 `df.head()` 语句查看一下这个数据集头部，对里面的数据组成初步熟悉一下。

scikit-learn 实战之监督学习 (/courses/866)

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、线性支持向量机

三、非线性支持向量机

四、支持向量机回归

五、实验总结

六、课后习题

	x	y	class
0	0.178681	0.300682	C1
1	0.202033	0.320188	C1
2	0.175568	0.290042	C1
3	0.156886	0.284722	C1
4	0.144432	0.302455	C1

我们可以看到，有两个类别。其中 C1 即表示上面图中的蓝色样本点，C2 对应着红色样本点。

和上一节课的过程相似，下面导入分割模块，将整个数据集划分为训练集和测试集两部分，其中训练集占 70%。

☞ **示例代码：**

```
from sklearn.model_selection import
train_test_split # 导入数据集划分模块

# 读取特征值及目标值
feature = df[["x", "y"]]
target = df["class"]

# 对数据集进行分割
train_feature, test_feature, train_t
arget, test_target = train_test_spli
t(feature, target, test_size=0.33)
```

☞ **动手练习：**

接下来，分别导入感知机和线性支持向量机分类器。

☞ **示例代码：**

scikit-learn 实战之监督学习 (/courses/866)


```
from sklearn.svm import LinearSVC # 导入线性支持向量机分类器
from sklearn.linear_model import Perceptron # 导入感知机分类器

# 构建感知机预测模型
model = Perceptron()
model.fit(train_feature, train_target)

# 构建线性支持向量机分类模型
model2 = LinearSVC()
model2.fit(train_feature, train_target)
```

☞ **动手练习：**

然后，我们使用模型带有的 score 方法看一下分类的准确度。

☞ **示例代码：**

```
# 感知机分类准确度
print(model.score(test_feature,test_target))

# 支持向量机分类准确度
print(model2.score(test_feature,test_target))
```

☞ **动手练习：**

☞ **参考结果：**

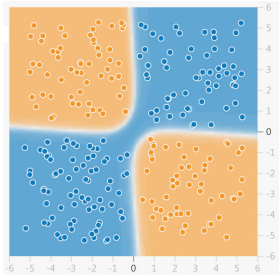
```
0.9696969696969697
1.0
```

可以看出，线性支持向量机全部分类正确，而感知机的准确度为 96%。

三、非线性支持向量机

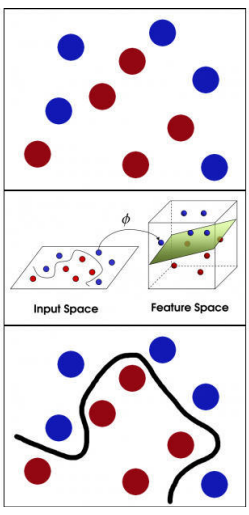
scikit-learn 实战之监督学习 (/courses/866)

通过上面的内容，你应该对线性分类有所了解，并可以使用支持向量机构建一个简单的线性分类器了。而在实际生活中，我们大部分情况面对的却是非线性分类问题，因为实际数据往往都不会让你通过一个水平超平面就能完美分类。



上图展现的就是一个非线性分类问题，而支持向量机就是解决非线性分类的有力武器。那么支持向量机是如何实现非线性分类呢?

这里，支持向量机引入了核函数来解决非线性分类的问题。简单来讲，通过核函数，我们可以将特征向量映射到高维空间中，然后再高维空间中找到最大间隔分割超平面完成分类。而映射到高维空间这一步骤也相当于将非线性分类问题转化为线性分类问题。



如上图所示：

1. 第一张图中，红蓝球无法进行线性分类。
2. 使用核函数将特征映射到高维空间，类似

scikit-learn 实战之监督学习 (/courses/866)

于在桌子上拍一巴掌使小球都飞起来了。

3. 在高维空间完成线性分类后，再将超平面重新投影到原空间。

在将特征映射到高维空间的过程中，我们常常会用到多种核函数，包括：线性核函数、多项式核函数、高斯径向基核函数等。其中，最常用的就算是高斯径向基核函数了，也简称为 RBF 核。

接下来，我们就通过 scikit-learn 来完成一个非线性分类实例。这次，我们选择了 digits 手写数字数据集。digits 数据集无需通过外部下载，可以直接由 scikit-learn 提供的 `datasets.load_digits()` 方法导入。该数据集的详细信息如下：

方法	描述
('images', (1797L, 8L, 8L))	数据集包含 1797 张影像，影像大小为 8x8
('data', (1797L, 64L))	data 将 8x8 像素根据其灰度值转换为矩阵
('target', (1797L,))	记录 1797 张影像各自代表的数字

第一步，导入数据并进行初步观察。

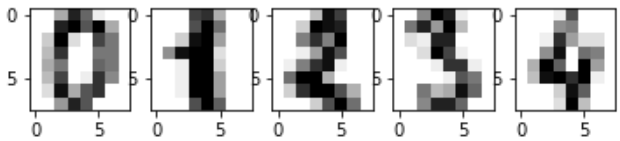
☞ **示例代码：**

scikit-learn 实战之监督学习 (/courses/866)


```
from sklearn import datasets  # 导入数据集模块
%matplotlib inline
import matplotlib.pyplot as plt  # 导入绘图模块

# 载入数据集
digits = datasets.load_digits()

# 绘制数据集前 5 个手写数字的灰度图
for index, image in enumerate(digits.images[:5]):
    plt.subplot(2, 5, index+1)
    plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')

plt.show()
```

☞ **动手练习：**

☞ **参考结果：**



可以用 `digits.target[:5]` 查看前五张手写数字对应的实际标签。

☞ **示例代码：**

```
digits.target[:5]
```

☞ **动手练习：**

☞ **参考结果：**

```
array([0, 1, 2, 3, 4])
```

通常，我们在处理图像问题时，都是将图像的每一个像素转换为灰度值或按比例缩放的灰度值。有了数值，就可以构建和图像像素大小相

scikit-learn 实战之监督学习 (/courses/866)

同的矩阵了。在这里， `digits` 已经预置了每一张图像对应的矩阵，并包含在 `digits.images` 方法中。

我们可以通过 `digits.images[1]` 输出第 1 张手写数字对应的 8x8 矩阵。很方便地，scikit-learn 已经将 8x8 矩阵转换成了方便作为特征变量输入 64x1 的矩阵，并放在了 digits.data 中。你可以使用 `digits.data[1]` 查看。

☞ **示例代码：**

```
digits.data[1]
```

☞ **动手练习：**

☞ **参考结果：**

```
array([ 0.,  0.,  0., 12., 13.,  5.,
0.,  0.,  0.,  0.,  0., 11., 16.,
        9.,  0.,  0.,  0.,  0.,  3.,
15., 16.,  6.,  0.,  0.,  0.,  7.,
       15., 16., 16.,  2.,  0.,  0.,
0.,  0.,  1., 16., 16.,  3.,  0.,
        0.,  0.,  0.,  1., 16., 16.,
6.,  0.,  0.,  0.,  0.,  1., 16.,
       16.,  6.,  0.,  0.,  0.,  0.,
0., 11., 16., 10.,  0.,  0.])
```

如果你连续学习了前面的小节，你应该对接下来的实验步骤比较熟悉了。下面，我们需要划分训练集和测试集，然后针对测试集进行预测并评估预测精准度。

☞ **示例代码：**

scikit-learn 实战之监督学习 (/courses/866)


```
from sklearn.svm import SVC #导入非线性支持向量机分类器
from sklearn.metrics import accuracy_score # 导入评估模块

feature = digits.data # 指定特征
target = digits.target # 指定目标值

# 划分数据集，将其中 70% 划为训练集，另 30% 作为测试集
train_feature, test_feature, train_target, test_target = train_test_split(feature, target, test_size=0.33)

model = SVC() # 建立模型
model.fit(train_feature, train_target) # 模型训练
results = model.predict(test_feature) # 模型预测

scores = accuracy_score(test_target, results) # 评估预测精准度
scores
```

☞ **动手练习：**

☞ **参考结果：**

```
0.4595959595959596
```

最后，模型预测准确度为 44.6%。由于每一次运行时，数据集都会被重新划分，所以你训练的准确度甚至会低于 44.6%。你应该会疑惑，准确度为什么这么低，不是说支持向量机的分类效果很好吗？

不要忘记了，我们在建立模型的时候使用的是默认参数。

☞ **示例代码：**

```
model
```

scikit-learn 实战之监督学习 (/courses/866)

☞ **动手练习：**

☞ **参考结果：**

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

上面输出了默认模型的参数。我们可以看到，该模型的确使用了最常用的 RBF 高斯径向基核函数，这没有问题。问题出在了 gamma 参数，gamma 是核函数的因数，这里选择了 auto 自动。自动即表示　gamma 的取值为 1 / 特征数量，这里为 1/64。

你可以尝试将 gamma 参数的值改的更小一些，比如 0.001。重新建立模型

☞ **示例代码：**

```
model = SVC(gamma = 0.001) # 重新建立模型
model.fit(train_feature, train_target) # 模型训练
results = model.predict(test_feature) # 模型预测

scores = accuracy_score(test_target, results) # 评估预测精准度
scores
```

☞ **动手练习：**

☞ **参考结果：**

```
0.9882154882154882
```

可以看到，这一次的预测准确度已经达到

scikit-learn 实战之监督学习 (/courses/866)

98% 了，结果非常理想。所以说，会用 scikit-learn 建立模型只是机器学习过程中最基础的一步，更加重要的是理解模型的参数，并学会调参使得模型的预测性能更优。

四、支持向量机回归

支持向量机除了可以用于处理分类问题，它也可以用来预测回归问题。其实，许多监督学习方法都有这个特点，包括决策树、K 近邻、随机森林等。在下一节内容中，将会对这些常见的监督学习方法做一个比较。

上面，我们已经使用支持向量机解决了两种不同形式的分类问题，包含线性分类和非线性分类。2002 年，澳大利亚学者 Smola 率先对使用支持向量机来解决回归问题进行了相关探索。这里，我们就不再深究于数学理论，有兴趣的朋友可以自行查找资料学习。总之，支持向量机回归不再受制于线性模型，可以用来解决一些非线性模型的回归问题。同时，支持向量回归无需担心多重共线性问题，对异常点敏感度也较低。总之，好处多多。

接下里，我们选择波士顿房价预测训练集来完成支持向量机回归问题实战。该训练集可以通过 datasets.load_boston() 导入。

☞ **示例代码：**

```
from sklearn import datasets # 导入数据集模块

boston = datasets.load_boston() # 导入波士顿房产数据集

boston.DESCR # 输出数据集介绍文档
```

☞ **动手练习：**

scikit-learn 实战之监督学习 (/courses/866)

我们可以看到对该数据集的一些介绍，但都是英文内容。概括讲来，该数据集有 506 条数据，每条数据包含 13 个特征变量和对应的房屋价格。其中，特征变量包含房屋所在位置的人口比例、交通方便程度、空气质量等。

这一次，我们不再像之前的内容，将数据集划分为 70% 训练集和 30% 测试集。而是采用机器学习中另一种十分常见的测试方式：交叉验证。交叉验证，就是将整个数据集等分为 n 等份，然后使用其中 n-1 等份训练模型，再使用另外的 1 份测试模型，循环验证。

☞ 示例代码：

```
from sklearn.svm import LinearSVR # 导入线性支持向量机回归模块
from sklearn.cross_validation import cross_val_predict # 导入交叉验证模块

feature = boston.data # 数据集特征
target = boston.target # 数据集目标变量

model = LinearSVR() #建立支持向量机回归模型

predictions = cross_val_predict(model, feature, target, cv = 10) #交叉验证
```

☞ 动手练习：

其中，cv = 10 就是指将整个数据集等分为 10 份。一般情况下，我们都设置为 10。

然后，来看一下预测值和真实值之间的关系，这里我们采用 matplotlib 绘制成散点图。

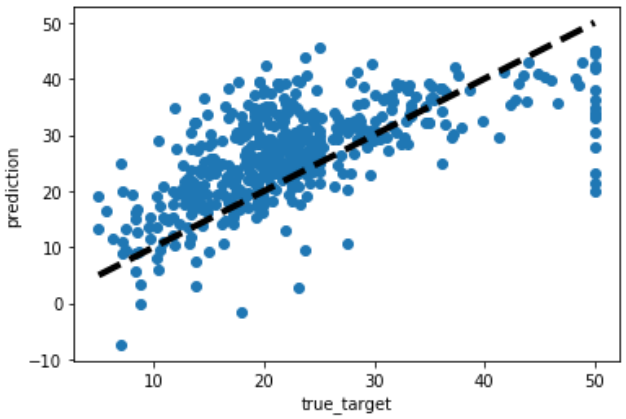
☞ 示例代码：

scikit-learn 实战之监督学习 (/courses/866)


```
plt.scatter(target, predictions) #绘制散点图
plt.plot([target.min(), target.max()], [target.min(), target.max()], 'k--', lw=4) # 绘制 45 度参考线
plt.xlabel("true_target") # X 坐标轴名称
plt.ylabel("prediction") # Y 坐标轴名称

plt.show()
```

☞ **动手练习：**

☞ **参考结果：**



可以看出，预测的效果还是很不错的。大部分都均匀分布在 45 度参考线的左右，即代表真实值与预测值之间的绝对误差较小。

五、实验总结

支持向量机是机器学习中非常实用的模型之一。它理论基础完善，分类结果出色，深受数据科学家的喜欢。希望能通过本次实验，掌握支持向量机的基本原理，并学会使用 scikit-learn 构建一个支持向量机分类模型。

六、课后习题

scikit-learn 实战之监督学习 (/courses/866)

1. 通过官方文档 (http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC)了解 scikit-learn 中支持向量机分类器 sklearn.svm.SVC 中包含的参数，并尝试修改它们查看结果变化。

最有效的学习方式

上一节：广义线性模型用于分类和回归预测 - 实战入门 (/courses/866/labs/3156/document)

下一节：常见的监督学习模型对比评价 (/courses/866/labs/3159/document)