6.支持向量机

# 1 实验目的

1.了解sklearn. svm模块中三类支持向量机算法（LinearSVM、SVM、NuSVM）；

3.理解sklearn. svm模块中支持向量机分类、回归算法的实现原理和使用方法；

4.了解如何使用sklearn. svm模块中各个常用算法完成机器学习任务。

# 2 实验要求

本次实验后，要求同学们能：

1.掌握支持向量机算法的实现原理；

2.学会使用Scikit-learn库中模型实现常用的支持向量机分类与回归算法；

3.了解sklearn.svm模块中LinearSVM、SVM、NuSVM三种支持向量机算法的不同特点，了解各个超参数的含义。

# 3 实验内容

## 3.1 支持向量机原理概述

支持向量机算法主要有以下几个重要概念：

（1）支持向量和间隔

以鸢尾花数据集花瓣特征分布为例，有无数种分类方式可以用一条直线将数据完美第划分为两类。但是左图的两种分类方式的决策边界离样本过近，如果出现新的样本，很可能无法准确得到结果。相比之下，右侧的实线离两个类别的样本都保持了一个较宽的间隔，出现新样本时可能会得到更好的结果。

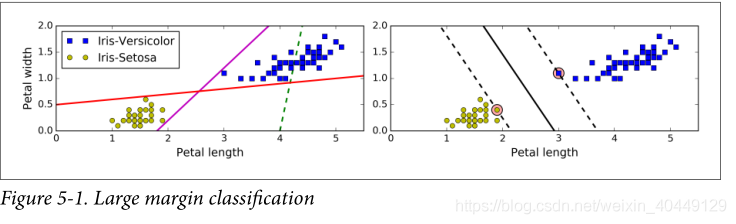


图1 不同分类策略图示

这个“较宽的距离”就是“最大间隔”，两个类别边界上到决策边界最近的样本称作“支持向量”。在支持向量机算法中，只有支持向量会对决策边界产生影响，所以只需考虑支持向量，不用管其他的样本如何分布。

用最大间隔将样本进行分类又可以分为硬间隔分类和软间隔分类。严格地让决策边界一侧全部为一类，另一侧全部为另一类的分类方法叫**硬间隔分类**，用硬间隔分类可能会导致过拟合，而且有的异常值已经融入另一类中很难被区分出来。为了解决这些问题，可以用**软间隔分类**，容忍一定的样本被错误地分到决策边界的另一侧，从而让算法具有更好的泛化能力。

在sklearn库中提供的LinearSVM、SVM模型中，用参数C限制软间隔的程度，而NuSVM使用参数nu进行限制。

（2）高维映射和核函数

对于线性不可分的问题，在高维空间往往就可以变得可以分类。就像一桌子大大小小的石头混在一起，难以将它们分成两类，但是武侠小说中的内功高手往桌子上狠狠一拍，大石头飞得低，小石头飞得高，这时只需要拿一个够大木板隔在中间，就很轻松地将满桌子石头按质量大小分成了两类。

所以对于线性不可分的问题，我们可以通过高维映射，将样本映射到高维空间再去分类。但是映射到高维空间也存在着很多麻烦，比如难以找到合适的高维空间，难以在高维空间上进行计算。

核函数又称“核技巧”，核函数的产生就是为了简化样本映射到高维空间后的计算，完全不需要知道映射到了什么样的高维空间，就可以进行计算。主要有以下几种常见核函数：

线性核函数：𝐾(𝑥,𝑧)=𝑥∙𝑧

多项式核函数：𝐾(𝑥,𝑧)=（𝛾𝑥∙𝑧+𝑟)𝑑

高斯核函数：𝐾(𝑥,𝑧)=𝑒 (−𝛾||𝑥−𝑧||2)

sigmoid核函数：𝐾(𝑥,𝑧)=𝑡𝑎𝑛ℎ（𝛾𝑥∙𝑧+𝑟)

（其中“ ∙ ”表示求内积运算）

支持向量机算法功能强大且全面，可以用于分类、回归、异常检测等任务，是机器学习领域最受欢迎的算法，特别适用于中小型复杂数据集的分类问题。

Scikit-learn库提供三类支持向量机算法LinearSVM、SVM、NuSVM，每一类算法都分别有可以用于分类和回归问题的模型。

简单来说，**LinearSVM**算法只支持线性核函数，算法效率更高，但是对线性不可分问题不适用。**SVM**算法支持线性核函数（linear）、多项式核函数（poly）、高斯核函数（rbf）、sigmoid核函数（sigmoid）等，这些核函数可以计算低维不可分数据到高维空间的映射，从而实现分类。**NuSVM**算法在数学原理上与SVM算法等价，可以理解为对SVM的重新参数化，用参数nu来度量损失。

下面将对Scikit-learn库中常用的支持向量机算法进行简单介绍。

## 3.2 支持向量机分类算法

### 3.2.1 支持向量机处理多分类问题原理

支持向量机原本是用于二分类问题，但是对于多分类问题，也可以通过一定的分类策略实现（LinearSVC用参数multi\_class指定，SVC和NuSVC用参数decision\_function\_shape指定）。

LinearSVC常用OvR“one vs rest”策略，即训练时分别把每一类看作一个正类，其他类全体看作一个负类，这样k个类别构造k个支持向量机。分类时，将待分类样本归入具有最大分类函数值的那一类。

SvC与NuSvC常用OvO“one vs one”策略，即训练时对任意两类之间都构造一个支持向量机，k个类别需要构建k（k-1）/2个支持向量机。分类时，分别用这k（k-1）/2个支持向量机分类器对待分类样本进行分类，最终将得票最多的类别作为待分类样本的类别。

OvR相对简单，但分类效果相对略差，而OvO分类相对精确，但是分类速度没有OvR快。

### 3.2.2 实验

（1）数据预处理

使用Scikit-learn自带的手写数字数据集进行实验，该数据集一共有1797个样本数据，每个样本点都是一个8×8的手写数字图像，即，每个样本有64维特征。而每个特征点上的数据是数字图像在该位置上的灰度（0~255）。

由于支持向量机算法对非标准的数据敏感，需要先对手写数字集进行归一化处理。数据预处理代码如下：

import matplotlib.pyplot as plt

from sklearn.svm import SVC

from sklearn.datasets import load\_digits

from sklearn.preprocessing import MinMaxScaler

from sklearn.model\_selection import train\_test\_split

# 导入数字图像数据集

digits = load\_digits()

data = digits.data

target = digits.target

images = digits.images

# 划分数据集

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.2,

random\_state=1

)

# 在对数据预处理前，先保存下测试集的数字图像和实际值

test\_images = X\_test.reshape(360, 8, 8)

# 归一化

MMS = MinMaxScaler()

MMS.fit(X\_train)

X\_train = MMS.transform(X\_train)

X\_test = MMS.transform(X\_test)

（2）创建算法模型

使用sklearn.svm模块的SVC类可以创建一个支持向量机分类模型实例，其构造器的重要参数及含义如下：

**C：用于指定正则化参数，C越大正则化程度越低，对错误的样本容忍度越小，泛化能力越小，越容易出现过拟合。C默认为1，必须取一个正值。**

**kernel：用于指定核函数，可以选择“linear”表示选用线性核函数；“poly”表示选用多项式核函数；“rbf”表示选用高斯核函数；“sigmoid”表示选用sigmoid核函数；也可以传入自定义核函数。默认使用高斯核“rbf”。**

**degree：多项式核函数的参数d。默认为3，如果没有用多项式核函数，则忽略此参数。**

**gamma：高斯核函数、多项式核函数、sigmoid核函数的参数𝛾。默认为“scale”** 使用**1 / (n\_features \* X.var())作为gamma的值；也可以选用：“auto”，使用**1 / n\_features作为gamma的值。

**coef0：多项式核函数和sigmoid核函数中的偏置量r，默认为0。**

**probability：设置是否输出样本划入每一个类别的概率，默认为False。如果设置为True，则训练模型时会用5折交叉验证计算样本属于各个类别的概率如果想将SVC模型作为一个软投票法的集成学习器中的弱学习器，则此参数必须设置为True。（集成学习的相关知识会在第8、9两个章节中讲解）**

**cache\_size：指定缓存大小，如果训练样本过大，可以通过指定更大的缓存大小来提升模型的训练速度。**

**class\_weight：可以指定样本各类别的的权重，主要是为了防止训练集某些类别的样本过多，导致训练的决策树过于偏向这些类别。可以以字典或字典列表的形式自定义；也可以选择“banlance”自动根据根据标签值，将权值调整为类别频率的反比（即，样本量少的类别有更高的权值）；默认值为“None”表示不设置权值。**

**decision\_function\_shape：用于设置分类决策方式**

注：与SVC相比，

创建LinearSVC模型实例时，①由于只支持线性核函数，所以没有kernel参数，从而也没有degree、gamma、coef等参数；②penalty参数可以设置惩罚项的范数，默认用L2正则化防止过拟合，也可以选择L1正则化一定程度上防止过拟合的同时，进行特征选择；③decision\_function\_shape参数变为multi\_class，用于设置多分类问题中的分类决策，默认为OvR策略；④LinearSVC的计算量不大，所以不需要设置cache\_size参数。

创建NuSVC模型实例时，没有用惩罚系数C，而是改用参数nu来表示训练集训练的错误率的上限，或者支持向量的百分比下限。取值在[0,1]区间上，默认为0.5。

本实验创建一个使用高斯核函数的支持向量机分类算法模型，实验代码如下：

# 创建模型

model = SVC(kernel="rbf")

（3）训练并测试

SVC算法模型实例提供常用的fit()、predict()、score()、get\_params()、set\_params()方法，这里不再赘述。需要注意的是，SVC相关算法实例一般不提供predict\_proba()方法，即不能预测样本属于某一类别的概率，除非创建实例时将参数**probability设置为True，但这会使算法效率降低。**

利用数据对模型进行训练，然后输出模型评分。取前四条手写数字数据，输出它们的预测值和实际值，进行比较。实验代码如下：

# 训练模型

model.fit(X\_train, Y\_train)

# 评分

print("Train Score：\n", model.score(X\_train, Y\_train))

print("Test Score:\n", model.score(X\_test, Y\_test))

# 测试

for i in range(4):

print("\n第%d个测试数据" % (i + 1))

print("图像为：")

plt.matshow(test\_images[i])

plt.show()

print("识别结果为：", model.predict(X\_test[i].reshape(1, -1))[0])

print("实际值为：", Y\_test[i])

（4）查看训练后的参数

可以用parameter\_（参数名加下划线）的形式直接访问训练得到的以下参数：

**support\_：查看支持向量的索引序号。**

**support\_vectors\_：查看支持向量的全部信息。**

**n\_support\_：查看每个类别的支持向量个数。**

**coef\_：查看线性核函数中的参数。**

**fit\_status\_：查看是否完成拟合，返回0表示拟合完成，1表示未完成拟合。**

**classes\_：分类器要划分的类别标签。**

**class\_weight\_：查看每一类的权重。**

**shape\_fit\_：查看训练数据的数量、特征维数。**

**实验代码如下：**

print("class\_weight\_", model.class\_weight\_)

print("classes\_", model.classes\_)

#print("coef\_", model.coef\_) # coef\_是线性核函数的参数

print("intercept\_\n", model.intercept\_)

print("fit\_status\_", model.fit\_status\_)

#print("support\_", model.support\_)

#print("support\_vectors\_", model.support\_vectors\_)

print("n\_support\_", model.n\_support\_)

print("shape\_fit\_", model.shape\_fit\_)

### 3.2.3 实验结果

（1）模型评分

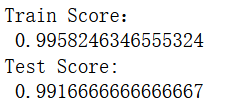
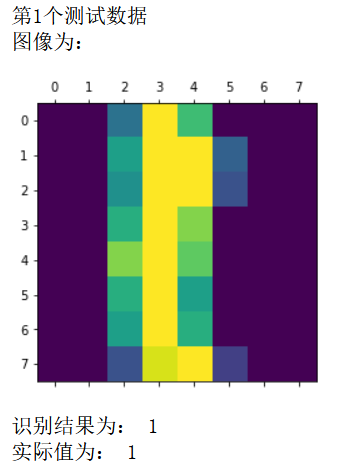
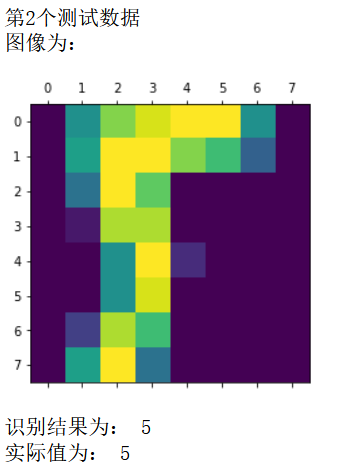


图2 模型评分

（2）预测结果

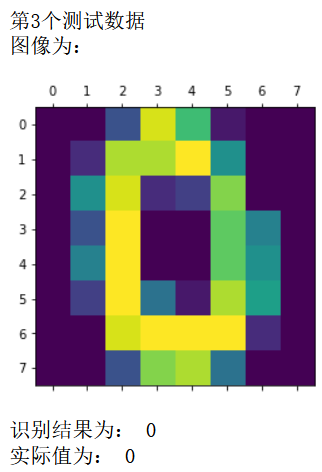
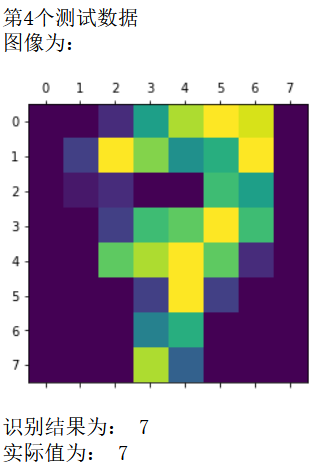
 

图3 预测结果

（3）查看参数

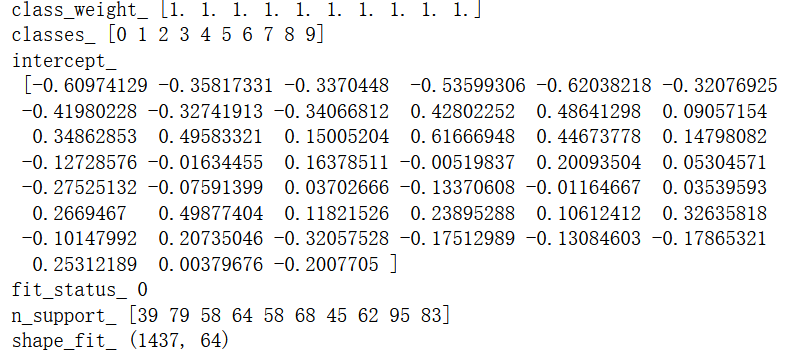


图4 训练出的参数

## 3.3 支持向量机回归算法

### 3.3.1 支持向量机处理回归问题原理

支持向量机分类算法可以推广到回归问题中，称为支持向量机回归算法。

支持向量机分类算法是将距离决策边界最近的样本作为支持向量，让决策边界与支持向量的间隔尽可能大。而支持向量机回归算法与分类算法不同，是将距离决策边界最远的点作为支持向量，让该点到决策边界竖直方向的偏差yi – f（xi）小于ε，形成一个ε管道。

支持向量机回归的主要目标就是让样本尽可能多地分布在这个ε管道内部，并且让这个管道尽可能水平（管道尽可能窄）。

### 3.3.2 实验

（1）数据预处理

本实验使用Scikit-learn自带的糖尿病数据集进行实验，该数据集一共有442个样本数据，每个样本有10个特征维度分别代表糖尿病病人的十个不同生理指标。而样本标签为该病人一年之后患病程度的定量指标。

加载数据集，划分为训练集和测试集，并对数据集进行标准化。

from sklearn.svm import SVR

from sklearn.datasets import load\_diabetes

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

# 加载数据集

diabetes\_data = load\_diabetes()

data = diabetes\_data.data

target = diabetes\_data.target

# 划分数据集

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.2,

random\_state=1

)

# 标准化

STD = StandardScaler()

STD.fit(X\_train)

X\_train = STD.transform(X\_train)

X\_test = STD.transform(X\_test)

（2）创建算法模型实例

用sklearn.svm模块的SVR类可以创建一个支持向量机回归算法模型实例，其构造器的重要参数及含义如下（与支持向量机分类算法构造函数相同的参数不再赘述）：

**epsilon：设置ε管道的宽度值，即|yi - f(xi)|**

**下面创建一个sigmoid核函数，用1 / (n\_features \* X.var())作为sigmoid核函数中参数γ的值，正则化程度为9的支持向量机回归算法模型。**

model = SVR(kernel="sigmoid", gamma="scale", C=9)

（3）训练模型并评分

用训练数据对创建出的模型进行训练，并在测试集上进行测试，代码如下：

model.fit(X\_train, Y\_train)

print("Train Score:", model.score(X\_train, Y\_train))

print("Test Score:", model.score(X\_test, Y\_test))

### 3.3.3 实验结果

使用给定参数创建支持向量机回归算法模型，在糖尿病数据集上评分如下。



图5 支持向量机回归算法模型评分

# 4.总结

训练支持向量机算法模型之前，要对数据进行标准化。

在特征数非常多的情况下，或者样本数远小于特征数的时候，使用线性支持向量机效果就已经很好，而且线性支持向量机运行效率也较快，并且只需要选择惩罚系数C即可。

如果线性拟合不好，一般推荐使用默认的高斯核‘rbf’。这时主要需要对惩罚系数C和核函数参数𝛾进行调参，通过多次实验找到最佳的参数。理论上使用核函数的支持向量机算法效果会比线性支持向量机好，但是这需要建立在大量的调参之上。