第3章

MATLAB机器学习工具箱

源码

在MATLAB近几年的版本中，推出了一个新的产品功能，即统计和机器学习工具箱(Statistics and Machine Learning Toolbox)。这个工具箱具有很多功能，并且在不断地完善中。具体来说它包含如下一些子模块：探索性数据分析、数据降维、机器学习、回归和方差分析、概率分布拟合及假设检验等功能模块。如果读者希望快速地了解部署一个机器学习应用，那么MATLAB提供的工具箱将会是一个不错的选择。基于这个原因，本章将着重介绍MATLAB统计和机器学习工具箱中的机器学习模块。

3.1 工具箱简介

机器学习算法使用计算方法直接从数据中“学习”信息，不把预定方程假设为模型。在第1章中罗列了各种不同的机器学习算法，归纳起来，按照解决问题的性质，可以分为分类(回归)、聚类和强化学习问题。相应地，在Statistics and Machine Learning Toolbox中提供用于执行受监督和无/非监督机器学习的方法。分类算法使用户可以将一个分类应变量建模为一个或多个预测元的函数。Statistics and Machine Learning Toolbox提供了涵盖多种参数化和非参数化分类算法的应用程序和函数，如logistic回归、朴素贝叶斯、k近邻、SVM等[1]。研究者可以直接利用MATLAB提供的这些算法的函数接口，通过编写脚本程序来使用这些算法。更直观地，MATLAB提供了一个GUI形式的分类学习应用程序，它使得研究者能够以窗口菜单的形式构建一个机器学习应用。本章将着重介绍这个应用的使用方法。

分类学习器应用程序(Classification Learner App)提供了一个机器学习应用常用的操作，如交互式探查数据、选择特征、指定交叉验证方案、训练模型和评估结果。分类学习

器应用程序用于使用监督式机器学习来训练模型对数据进行分类，使用它可以执行常见任务，例如，导入数据和指定交叉验证方案；探索数据和选择特征；使用多种分类算法训练模型；比较和评估模型；在计算机视觉和信号处理等应用场合中共享训练过的模型。

除此之外，分类学习器集成了多种可视化方式来方便用户选择模型，进行模型评估和比较。训练好的模型也可以直接导入 MATLAB 的工作空间，来对新的数据预测，也可以直接生成代码，方便和其他应用集成。

在 MATLAB 统计和机器学习工具箱中当然也实现了很多聚类算法，聚类算法通过根据相似度测量对数据分组来发现数据集中的规律。可用的算法包括 k-均值、k-中心点、分层聚类、高斯混合模型和隐马尔可夫模型。当不知道聚类的数量时，可以使用聚类评估技术根据特定指标确定数据中存在的聚类数量。只是聚类算法还没有对应的 GUI 应用。对于回归算法，MATLAB 2017 版本也推出了回归学习器，它的操作流程和界面与分类学习器很类似，感兴趣的读者可安装 2017 版本进行实践操作和学习。

3.2　分类学习器基本操作流程

对于构建机器学习应用，通常包括五部分，分别是数据导入、数据的探索和特征选择、训练模型、比较模型和输出模型。分类学习器也在不同的窗口中实现了这些功能。

首先，为了启动分类学习器，可以通过直接在命令行窗口中输入“classification Learner”，或者在 MATLAB 的菜单栏中选择“应用程序”选项卡下的分类学习器应用 Classification Learner，如图 3.1 所示。

图 3.1　启动分类学习器

此时 MATLAB 将弹出一个空白的分类器窗口，如图 3.2 所示。

该窗口就是进行分类操作最核心的一个窗口。但是可以看到大部分的菜单和按钮都是灰色的，这是因为还没有选择数据。对于机器学习应用来说，数据就好像是机器的燃料，没有燃料，机器自然动不起来，所以为了进行下一步工作，首先需要输入数据。

导入数据的方法分为两种方式：一种是单击 CLASSIFICATION LEARNER 选项卡下 FILE 组中的 New Session 下拉按钮，然后选择 From Workspace，如图 3.3 所示，其含

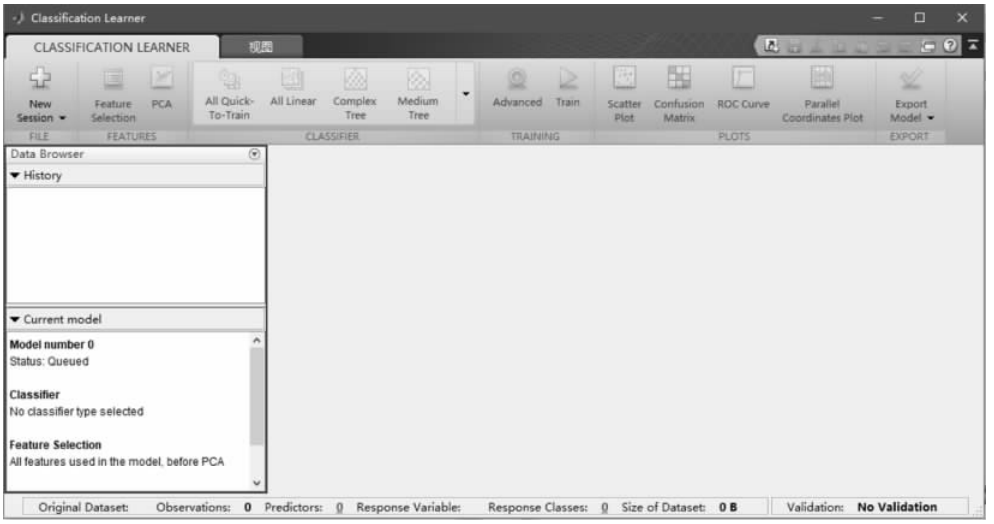

图 3.2　空白的分类器窗口

义是导入 MATLAB 工作空间的函数数据；另一种则是选择 From File，如图 3.3 所示，其含义是通过数据文件导入数据，如. xls、. xlsx、. xlsm、. txt、. csv 等格式数据文件。

图 3.3　导入数据

数据导入后，则进入数据处理窗口界面，如图 3.4 所示。图 3.4 中的数据是在命令行窗口中调用 fisheriris 数据集而产生的，其具体操作是在 MATLAB 命令行窗口中输入：fishertable＝readtable('fisheriris. csv')；并按 Enter 键生成的数据变量，此时使用上述的第一种导入数据的方法，即可生成图 3.4。

该窗口主要目的是用来设置训练数据的相关属性、标签及设置验证集。可以看到窗口主要分成 3 个部分，其依次对应 3 个步骤。在 Step1 中主要功能是选择数据集，且设置数据集矩阵中的行作为一个变量，还是将列作为变量；在 Step2 需要向算法声明哪些维

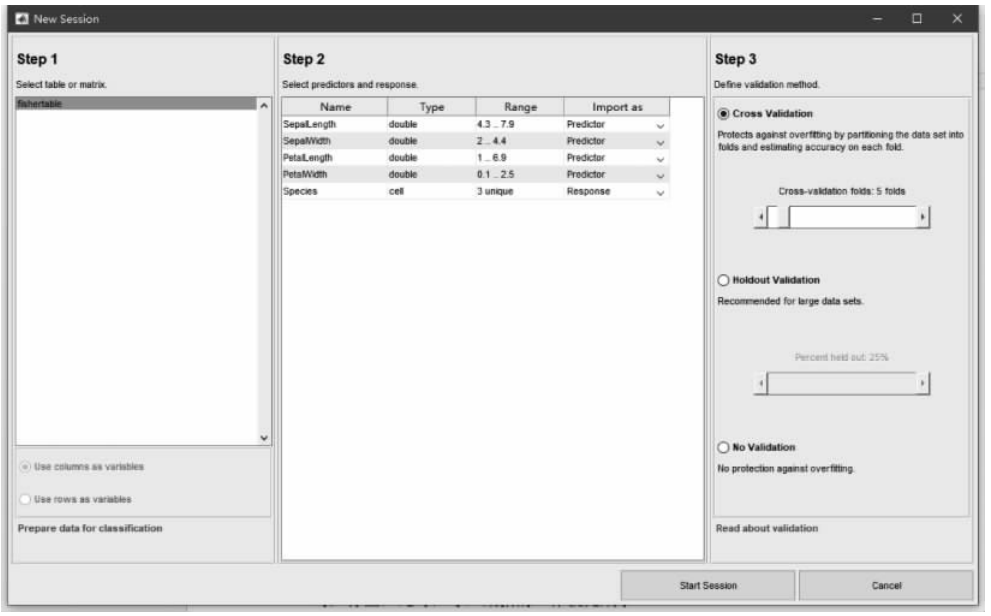

图 3.4 数据处理窗口

度是输入量，哪些是输出量，对于分类问题来说就是选择哪几个变量作为属性值，哪个变量作为标签。这个声明可以通过把变量导入为 Predictor 还是 Response 完成，其中，Predictor 对应输入，Response 对应标签，此时，导入数据的工作就完成了。

为了优化算法中的一些超参数，需要一定的手段来评估不同参数的表现。其中，验证集的目的是用来对算法的超参数调优。在模型计算过程中，测试集数据只能使用一次，不能用测试集数据调优，因为其会导致算法对测试集过拟合，将会导致模型在测试集上有较好的结果，但是实际效果不能得到保证。所以可以知道验证集应该是训练集的一部分。但是当训练集数量较少（因此验证集的数量更少）时，将用到交叉验证法。所谓 k-fold validation 就是把训练集均分成 k 份，其中 $k-1$ 份用来训练，1 份用来验证。然后循环取其中 $k-1$ 份来训练，其中 1 份来验证，最后取所有 k 次验证结果的平均值作为算法验证结果。在窗口的 Step3 中可以允许用户设定这个 k 值。导入数据并设置好交叉验证后，单击 Start Session 按钮就弹出如图 3.5 所示的分类器窗口。

可看出图 3.5 所示的窗口和图 3.2 所示的窗口是一样的，只是灰色的部分被激活了。这个激活的窗口包含了训练一个机器学习应用的核心要素，其大体分六部分。最上边的菜单栏提供了特征选择、算法选择、模型训练、可视化绘图及输出等操作。特征选择模块用于选择输入特征，即选择属性特征，一方面可通过单击该模块中的 Feature Selection 按钮进行人工选择，也可以利用 PCA 的方法自动选择。分类算法模块是各种算法的一个仓库。单击其下拉按钮会出现如图 3.6 所示下拉列表。

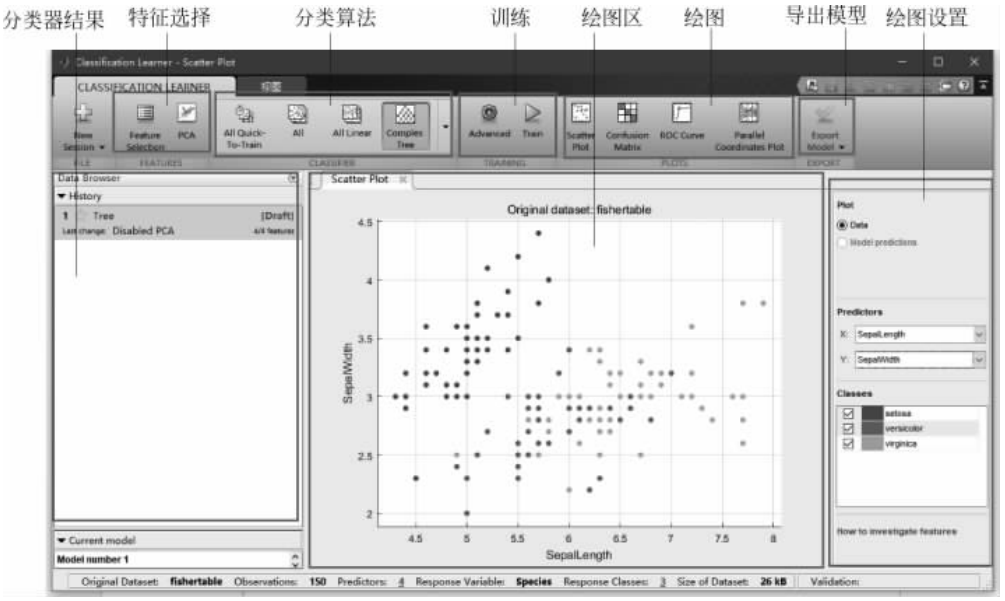

图 3.5　分类器窗口

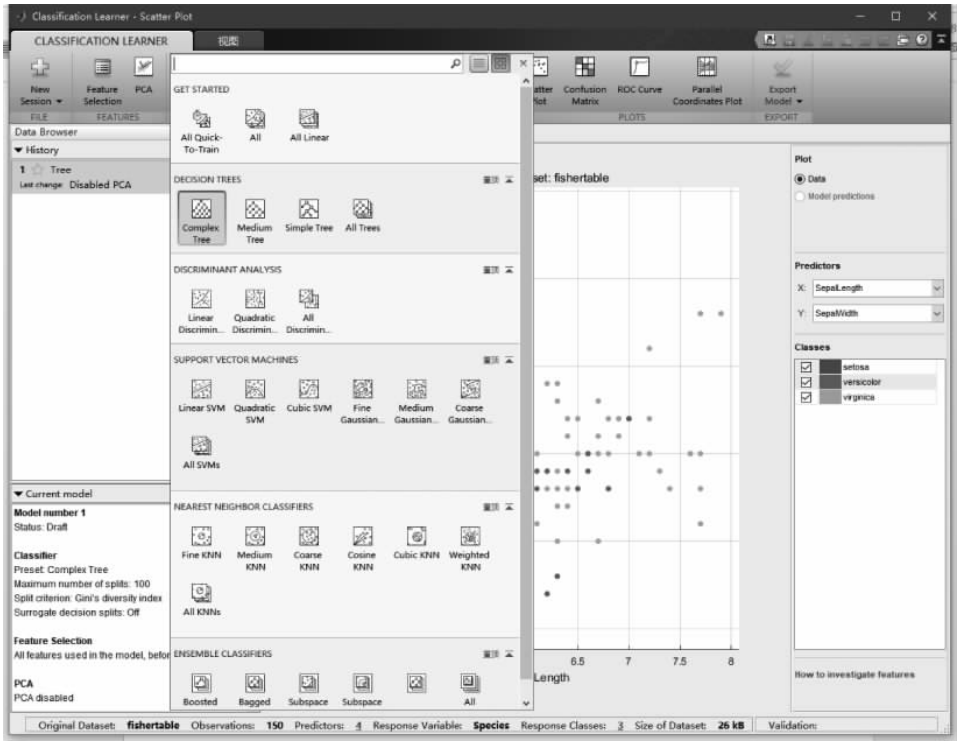

图 3.6　多种分类器

可以看到，分类学习器将可以应用的算法分成了四大类，分别是决策树类、判别分析类、支持向量机类、神经网络类。注意，在 GET STARTED 中的 All 和 AllLinear 并不是一个算法，而是一个快捷操作，它能够快速地对数据应用分类学习器中的所有模型或所有线性模型。在做了特征选择后，下一步则需要选择一个合适的分类算法。有了算法，下一步就是训练了。在 TRAINING 组中，除了简单地单击 Train 进行训练外，还可以通过 Advanced 按钮设置一些训练的参数，如对于决策树类算法来说可以设置最大的叶子结点数及确定决策边界的准则等。

当用户选择一个训练模型后，它就会出现在窗口的右侧方框中。用户可以选择多个模型，一次训练。得到的每个分类器的分类精度会显示在方框中。其中分类效果最好的模型会以方框突出显示。History 窗格显示当前模型的一些详细信息。

训练完毕后，用户不仅仅得到一个分类的精度值，同时用户还可以通过各种图形来直观地观测当前模型的表现。当然，图示化训练数据对于数据探索，发现数据的模式也是大有裨益的。绘图的选项在 PLOTS 组中，可以绘制的图形包括训练数据的散点图、confusion matrix、ROC 曲线等。关于这些不同图形的含义，读者可参考机器学习相关书籍，另外，在窗口的最右边可以设置绘图参数。

通过可视化窗口界面，用户可以再次调整模型，如进一步的特征提取，或者设置更加合理的超参数值，来改善模型的表现。那么，一旦得到了一个满意的模型，该如何利用它呢？分类学习器提供了两种模式：一种方法是用户导出训练好的模型到工作空间中，这个时候用户将发现变量空间中多了一个结构体变量，这个变量含有一个用于预测的成员函数，此时，用户将可以用它来做预测。但是如果用户希望更改模型，或者把它集成到其他的应用中，则需要用到另外一种方法，也是最通用的方法，即以代码的形式使用。MATLAB 可以直接通过 GUI 形式的窗口生成代码。单击图 3.5 所示的分类器窗口中的 Export Model 按钮，操作如图 3.7 所示。

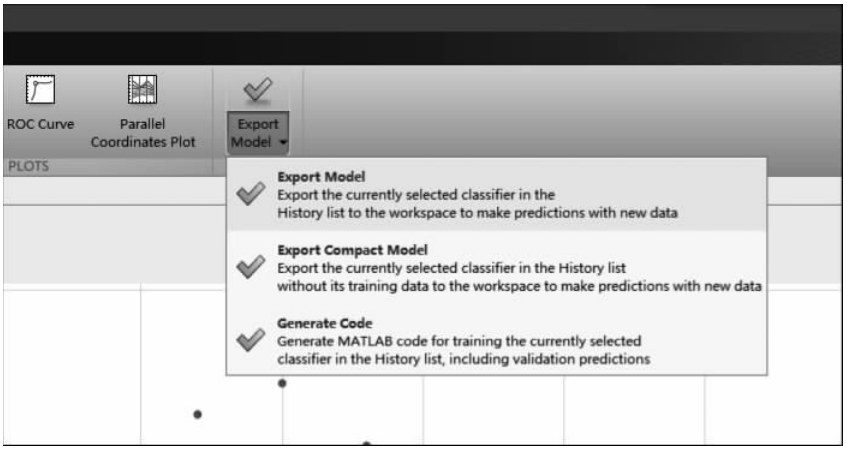

图 3.7 导出训练模型

因为特征选择和算法选择对于一个机器学习应用至关重要，下面将会更详细地介绍如何使用分类学习器 APP 做特征选择和分类器选择。

3.3 分类学习器算法优化与选择

使用工具箱的方法并不是智能的，同样需要对数据进行预处理，并根据经验和分析选择合适的数据特征。另外，机器学习各类算法也具有不同的特点，使用者应在不断的实践中了解各类机器学习算法，在特定的应用场合应选择适合数据自身的算法。

3.3.1 特征选择

在分类学习器中可以通过对原始数据做散点图来分析是否需要或排除某个特征。选择不同特征作为坐标轴，如果数据某一类别数据很好地分开，说明这个特征是有用的。如果某个特征对于分类没有任何作用，则可以考虑把它排除。在 fisheriris 数据集中选择 X 和 Y 分别为 PetalLength 和 PetalWidth，可以看到 setosa 类能够被很好地分开，如图 3.8 所示。

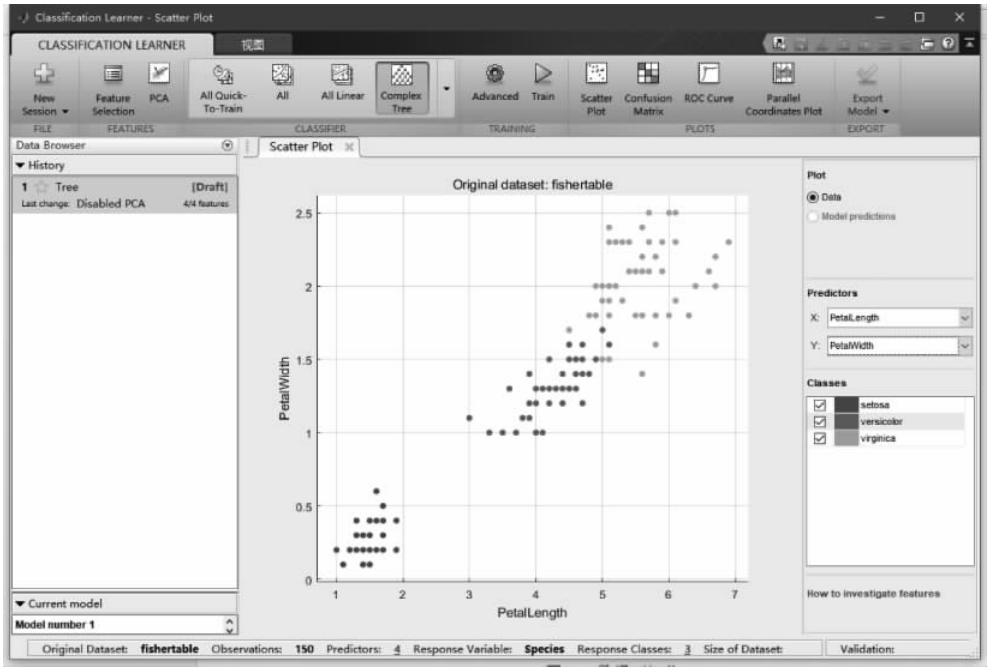

图 3.8 数据可视化

在分类学习器窗口中，也可以通过勾选某个特征，观察分类器的表现。如果删除某个特征后可以提高模型的性能，那么应该排除这个特征，尤其是当收集该数据比较昂贵和困难的时候。具体操作为单击 FEATURES 组中的 Feature Selection 按钮，将出现如图 3.9 所示的窗口，用户可以取消选中特征名称后的复选框，从而排除该特征。

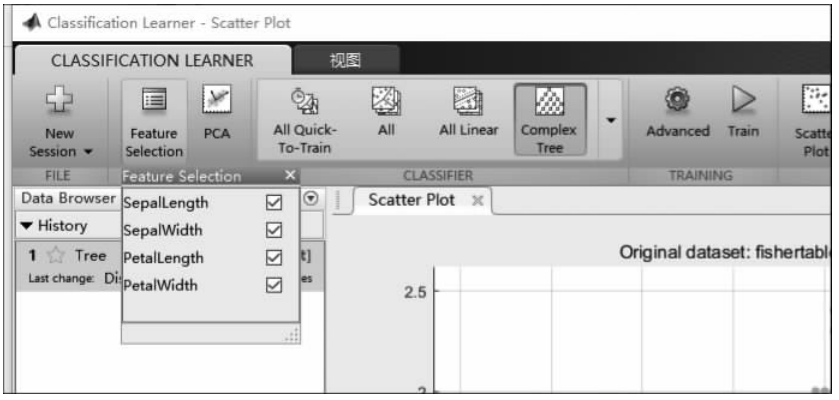

图 3.9 特征选择菜单

利用主成分分析来降低特征空间的维度，有助于防止过拟合。PCA 能够消除数据中的冗余信息，产生一个新的变量集，该变量集称为主成分。在工具箱中使用 PCA 包括以下步骤：(1)在分类器窗口的 FEATURES 组中单击 PCA 按钮；(2)在 Advanced PCA Options 下拉列表框中选中 Enable PCA 复选框，并设置相应参数(一般采用默认)。至此，用户已完成了对 PCA 的设置，之后，当用户单击 Train 的时候，PCA 会首先对数据做变换和处理，然后进行模型训练。

在分类学习器中也可以利用平行坐标图(Parallel Coordinates Plot)来选择特征，具体操作为单击分类学习器窗口的 PLOTS 组中的 Parallel Coordinates Plot 按钮，生成平行坐标图，如图 3.10 所示。

在平行坐标图中其实它就是把每个特征列在一个一维的轴上画出来，然后把每个记录(一个样本点)依次连接起来，最后用不同的颜色表示不同的类别，错分的类别用虚线表示。如果某个特征具有很好的区分度，那么在坐标轴上就会出现明显的聚类现象。从图 3.10 可以看出 PetalWidth 和 PetalLength 特征具有很好的分类效果。

3.3.2 选择分类器算法

各分类器算法有各自的特点，依赖于具体的需求，如速度、存储、灵活性、可解释性等，会有不同的选择。如果对于一个数据没有特别深刻的了解，或者特别适合的模型，最开始用户可以选择 All Quick-To-Train，这个选项会用所有高效的模型对数据训练，能够快速

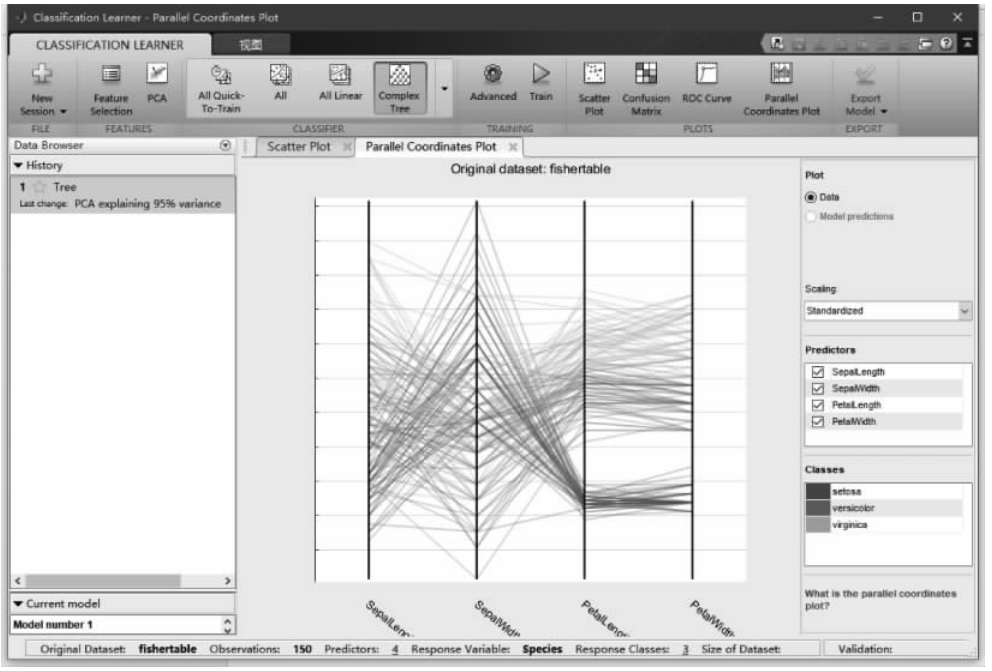

图 3.10 平行坐标图

地得到不同分类器的表现。但是一般来说,不同类型算法有不同的特点,有大概的了解也会有利于用户选择分类器算法。表 3.1 中针对不同类型分类器算法进行了对比。

表 3.1 不同分类器算法特性对比

分类器种类	预 测 速 度	内存需求量	解 释 性
决策树	快	小	容易
判别分析	快	小(线性)、大(其他)	容易
逻辑回归	快	中等	容易
支持向量机	中等(线性核函数)、慢(其他)	中等(线性)、大(非线性)	容易(线性)、较难(非线性)
k 近邻分类	中等	中等	较难
集成分类	取决于集成用的算法	取决于集成用的算法	较难

3.4 工具箱分类学习实例

本节将通过实例来阐述如何使用分类学习器 APP。所述的实例是基于 fisheriris 数据集。中文名为安德森鸢尾花卉数据集(Anderson's Iris Data Set),也称鸢尾花卉数据集

(Iris Flower Data Set)或费雪鸢尾花卉数据集(Fisher's Iris Data Set),是一类多重变量分析的数据集。其数据集包含了 150 个样本,都属于鸢尾属下的 3 个亚属,分别是山鸢尾、变色鸢尾和维吉尼亚鸢尾,如图 3.11 所示。4 个特征被用作样本的定量分析,它们分别是花萼和花瓣的长度和宽度。因此算法的任务就是基于这 4 个特征,利用不同的分类算法来分辨它们到底属于哪个亚属[2]。

图 3.11 几种鸢尾花卉

下面介绍利用 Classification Learner 实现分类算法的步骤。

(1) 在 MATLAB 命令行窗口中输入命令,进行 fisheriris 数据集的加载,命令为"fishertable=readtable('fisheriris.csv');"。

(2) 对 fisheriris 数据集的四项特征的数据进行归一化处理,相应的方法在第 1 章中已经介绍了较为经典的 3 种方法,MATLAB 中编程实现如下。

min-max 标准化(Norminmax.m 文件):

```
%% min-max 归一化方法
clear all;
close all;
clc;
fishertable = readtable('fisheriris.csv');   % 导入样本数据,样本数据为 Table 型
```

```
SepalLengthMat = fishertable. SepalLength;      % 取 Table 型数据的 SepalLength 的属性值,
                                                % 转化为矩阵形式
NorSepalLength = mapminmax(SepalLengthMat');    % 对 SepalLength 数据进行归一化处理
NorSepalLength = NorSepalLength';               % 行矩阵变为列矩阵
SepalWidthMat = fishertable. SepalWidth;        % 取 Table 型数据的 SepalWidth 的属性值,转
                                                % 化为矩阵形式
NorSepalWidth = mapminmax(SepalWidthMat');      % 对 SepalWidth 数据进行归一化处理
NorSepalWidth = NorSepalWidth';                 % 行矩阵变为列矩阵
PetalLengthMat = fishertable. PetalLength;      % 取 Table 型数据的 PetalLength 的属性值,
                                                % 转化为矩阵形式
NorPetalLength = mapminmax(PetalLengthMat');    % 对 PetalLength 数据进行归一化处理
NorPetalLength = NorPetalLength';               % 行矩阵变为列矩阵
PetalWidthMat = fishertable. PetalWidth;        % 取 Table 型数据的 PetalWidth 的属性值,转
                                                % 化为矩阵形式
NorPetalWidth = mapminmax(PetalWidthMat');      % 对 PetalWidth 数据进行归一化处理
NorPetalWidth = NorPetalWidth';                 % 行矩阵变为列矩阵
Norfishertable = table( NorSepalLength, NorSepalWidth, NorPetalLength, NorPetalWidth,
fishertable.Species);                           % 将归一化处理后的数据转化为 Table 型数
                                                % 据,且添加上已有的标签
```

Z-score 标准化(Norzscore. m 文件):

```
%% Z - score 归一化方法
clear all;
close all;
clc;
fishertable = readtable('fisheriris.csv');      % 导入样本数据,样本数据为 Table 型
SepalLengthMat = fishertable.SepalLength;       % 取 Table 型数据的 SepalLength 的属性值,
                                                % 转化为矩阵形式
NorSepalLength = zscore (SepalLengthMat');      % 对 SepalLength 数据进行归一化处理
NorSepalLength = NorSepalLength';               % 行矩阵变为列矩阵
SepalWidthMat = fishertable.SepalWidth;         % 取 Table 型数据的 SepalWidth 的属性值,转
                                                % 化为矩阵形式
NorSepalWidth = zscore (SepalWidthMat');        % 对 SepalWidth 数据进行归一化处理
NorSepalWidth = NorSepalWidth';                 % 行矩阵变为列矩阵
PetalLengthMat = fishertable.PetalLength;       % 取 Table 型数据的 PetalLength 的属性值,
                                                % 转化为矩阵形式
NorPetalLength = zscore (PetalLengthMat');      % 对 PetalLength 数据进行归一化处理
```

```
NorPetalLength = NorPetalLength';                    % 行矩阵变为列矩阵
PetalWidthMat = fishertable.PetalWidth;              % 取 Table 型数据的 PetalWidth 的属性值,转
                                                     % 化为矩阵形式
NorPetalWidth = zscore (PetalWidthMat');             % 对 PetalWidth 数据进行归一化处理
NorPetalWidth = NorPetalWidth';                      % 行矩阵变为列矩阵
Norfishertable = table ( NorSepalLength, NorSepalWidth, NorPetalLength, NorPetalWidth,
fishertable.Species);    % 将归一化处理后的数据转化为 Table 型数据,且添加上已有的标签
```

小数点定标标准化(NotBit.m 文件):

```
%% 小数点定标归一化方法
clear all;
close all;
clc;
fishertable = readtable('fisheriris.csv');      % 导入样本数据,样本数据为 Table 型
SepalLengthMat = fishertable.SepalLength;       % 取 Table 型数据的 SepalLength 的属性值,
                                                % 转化为矩阵形式
Bit = floor(log10(max(abs(SepalLengthMat)))) + 1; % 计算出属性值中绝对值最大的数据的位数
if Bit~= 0
    NorSepalLength = (SepalLengthMat')/(Bit * 10); % 小数点定标归一化
    NorSepalLength = NorSepalLength';           % 行矩阵变为列矩阵
else
    NorSepalLength = SepalLengthMat;            % 假如位数为 0,则原数据即为处理后的数据
end
SepalWidthMat = fishertable.SepalWidth;         % 取 Table 型数据的 SepalWidth 的属性值,转
                                                % 化为矩阵形式
Bit = floor(log10(max(abs(SepalWidthMat)))) + 1;  % 计算出属性值中绝对值最大的数据的位数
if Bit~= 0
    NorSepalWidth = (SepalWidthMat')/(Bit * 10);     % 小数点定标归一化
    NorSepalWidth = NorSepalWidth';             % 行矩阵变为列矩阵
else
    NorSepalWidth = SepalWidthMat;              % 假如位数为 0,则原数据即为处理后的数据
end
PetalLengthMat = fishertable. PetalLength;      % 取 Table 型数据的 SepalLength 的属性值,
                                                % 转化为矩阵形式
Bit = floor(log10(max(abs(PetalLengthMat)))) + 1; % 计算出属性值中绝对值最大的数据的位数
if Bit~= 0
    NorPetalLength = (PetalLengthMat')/(Bit * 10); % 小数点定标归一化
```

```
    NorPetalLength = NorPetalLength';              % 行矩阵变为列矩阵
else
    NorPetalLength = PetalLengthMat;               % 假如位数为 0,则原数据即为处理后的数据
end
PetalWidthMat = fishertable.PetalWidth;            % 取 Table 型数据的 SepalLength 的属性值,
                                                   % 转化为矩阵形式
Bit = floor(log10(max(abs(PetalWidthMat)))) + 1;   % 计算出属性值中绝对值最大的数据的位数
if Bit~= 0
    NorPetalWidth = (PetalWidthMat')/(Bit * 10);   % 小数点定标归一化
    NorPetalWidth = NorPetalWidth';                % 行矩阵变为列矩阵
else
    NorPetalWidth = PetalWidthMat;                 % 假如位数为 0,则原数据即为处理后的数据
end
Norfishertable = table( NorSepalLength, NorSepalWidth, NorPetalLength, NorPetalWidth,
fishertable.Species);   % 将归一化处理后的数据转化为 Table 型数据,且添加上已有的标签
```

(3) 上述 3 种归一化方法中,在本实例中选择第一种方法进行处理。然后,在 MATLAB 的"应用程序"选项卡中选择 Classification Learner。

(4) 经过上一步骤,会出现一个分类学习器窗口(见图 3.2)。在分类学习器窗口中单击 New Session 按钮,此时将出现一个用于数据处理的窗口。依照 3.3 节介绍的方法进行设置,分别设置 Predictors 和 Response 及一些关于交叉验证的参数,本实例采用默认设置。之后单击该窗口右下角的 Start Session 按钮。此时,分类学习器创建了一个数据的散点图,如图 3.12 所示,与 3.3 节中的图 3.8 对比,可发现散点的分布形式没有发生变化,仅仅是坐标轴的范围发生了改变,这就是对数据进行归一化后的数据结果。用户可以通过选择不同的特征来绘图,从而观察哪些变量能够很好地区分数据。值得注意的是,通过其他方式进行归一化处理,散点的分布形式会有一定的变化。本实例中样本属性特征为 4,且各特征对于训练都有作用,因此不进行删减特征处理及 PCA 主成分分析。

为了应用判别分析算法,单击 CLASSIFICATION LEARNER 选项卡 CLASSIFIER 组中的下拉按钮,选择 All 选项,然后单击 TRAINING 组中的 Train 按钮,此时,就可以开始训练模型了,经过 2～3 分钟的训练后,窗口如图 3.13 所示。同样,也可以在 CLASSIFIER 组中的下拉列表中单独选择决策树算法、SVM、logistic 回归及集成方法。这里为了方便,这里直接选择 All 选项,它会对训练数据应用所有的可用的分类器。

在图 3.13 中的左侧可以看到总共应用了 22 个分类算法,最好的分辨率达到了 96.7%的识别率,相应的识别算法包括 ComplexTree、Medium Tree、Simple Tree、Quadratic SVM、Medium Gaussian SVM。需要具体了解训练算法的相关参数及训练时间,可单击相应算法,则在其左下角的 Current model 窗格中显示。

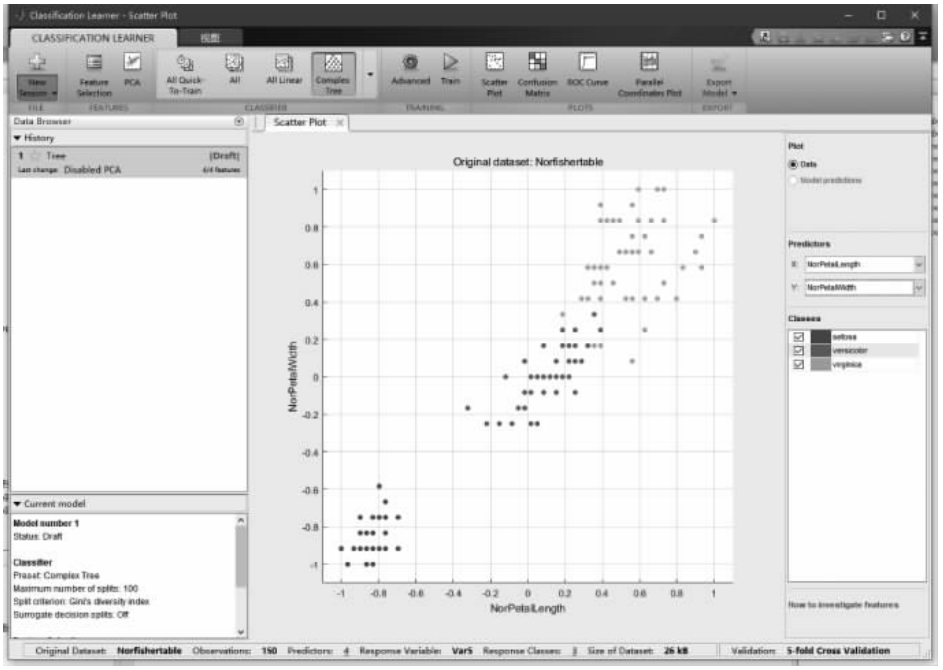

图 3.12　归一化后的数据可视化

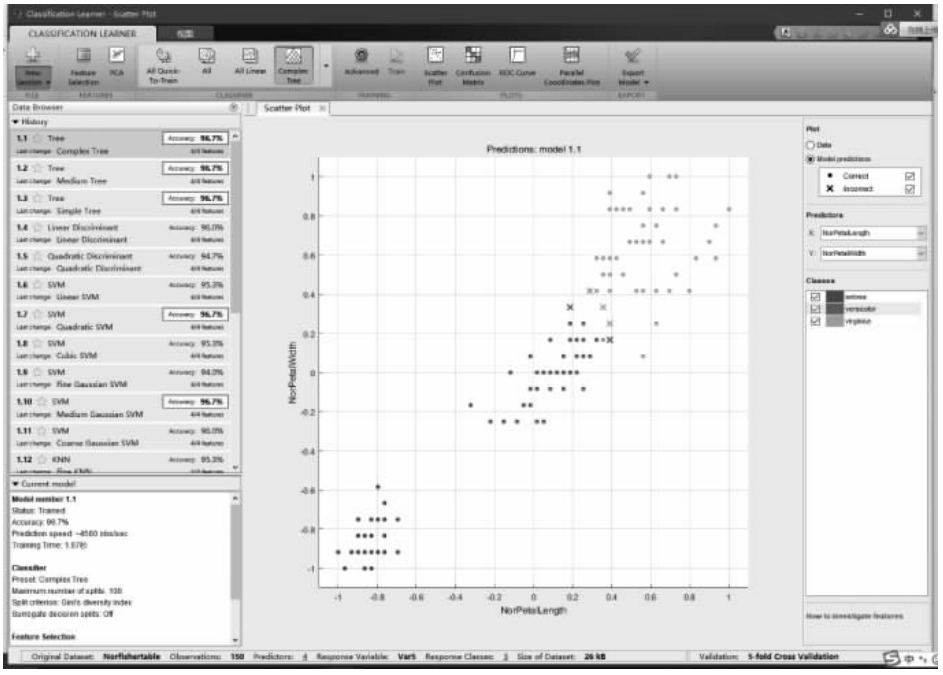

图 3.13　各个分类器在 fisheriris 数据集上的表现

另外，在图 3.13 的散点显示图中，可发现显示为“X”的点，这些点表示预测错误的点，单击图中的散点，可显示出其具体的数据信息。

为了观察每个类预测的准确率，可以单击 PLOTS 组中的 Confusion Matrix 按钮，如图 3.14 所示。也可以单击 ROC Curve 按钮观察 ROC 曲线，如图 3.15 所示。

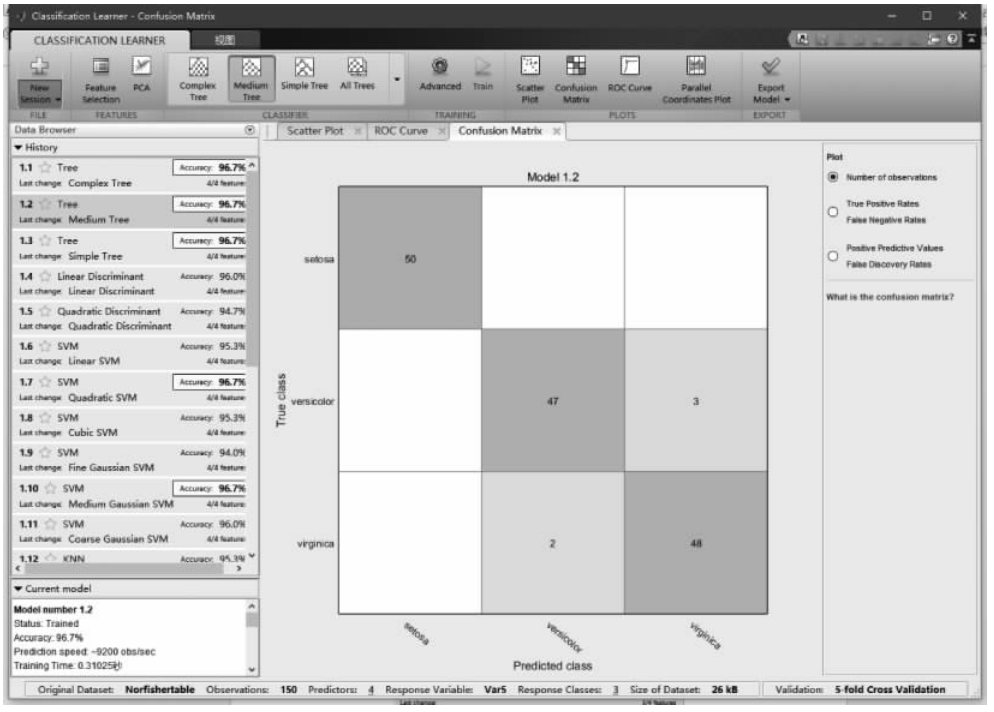

图 3.14　Confusion Matrix 图

最后，为了导出模型，在 Export Model 下拉列表中可以选择不同的导出方式，参见 3.2 节所述。本实例中采用导入工作空间的方法，即选择 Export Model 下拉列表中的 Export Compact Model 选项，此时出现对话框如图 3.16 所示，要求输入导出模型的名称。采用默认值，单击“确定”按钮。此时在 MATLAB 的工作空间中便显示了一个名为 trainedClassifier 的结构体数据，即为训练好的模型。

为了利用训练好的数据模型进行新样本的预测，需要调用和使用它的成员函数×××.predictFcn，其中×××为模型名称，本实例中使用格式如下：

```
yfit = trainedClassifier.predictFcn (T),
```

其中，T 是新的预测数据样本，它的形式和数据类型必须与训练中的数据保持一致，且不包含标签。另外，值得注意的是，需要对样本数据采用同样的归一化方法进行处理，

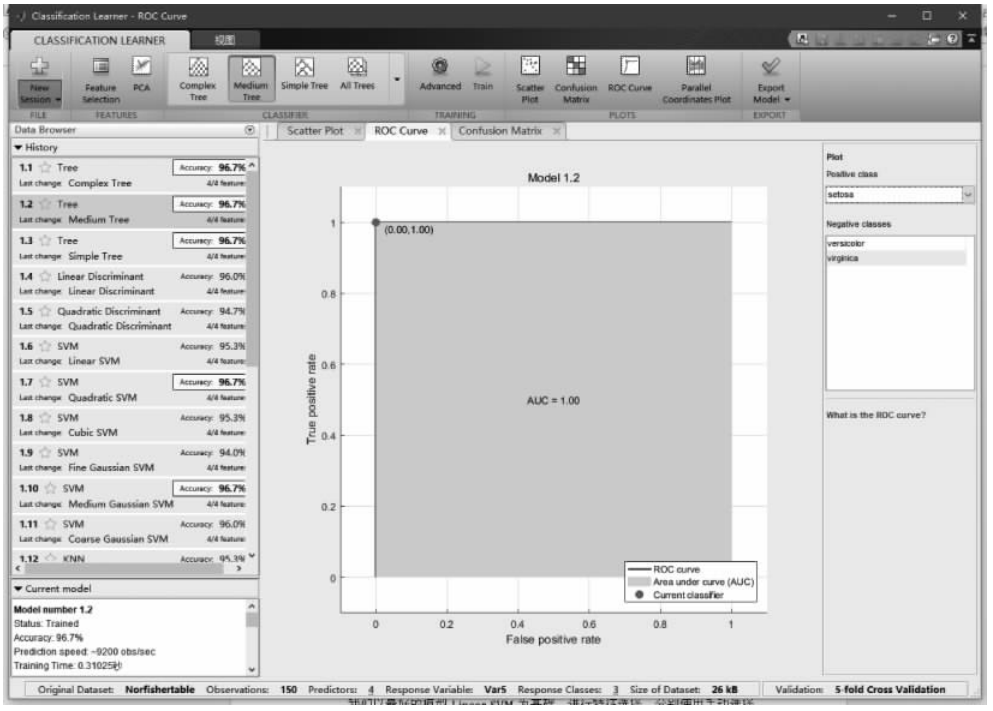

图 3.15　ROC 曲线

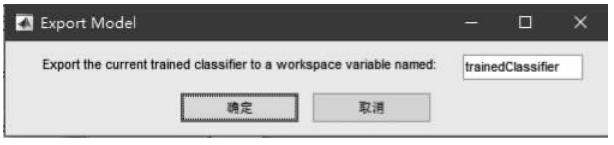

图 3.16　命名模型名称

之后才能进行预测。

参 考 文 献

[1] https://www.mathworks.com/products/statistics.html.

[2] https://zh.wikipedia.org/wiki/%E5%AE%89%E5%BE%B7%E6%A3%AE%E9%B8%A2%E5%B0%BE%E8%8A%B1%E5%8D%89%E6%95%B0%E6%8D%AE%E9%9B%86.