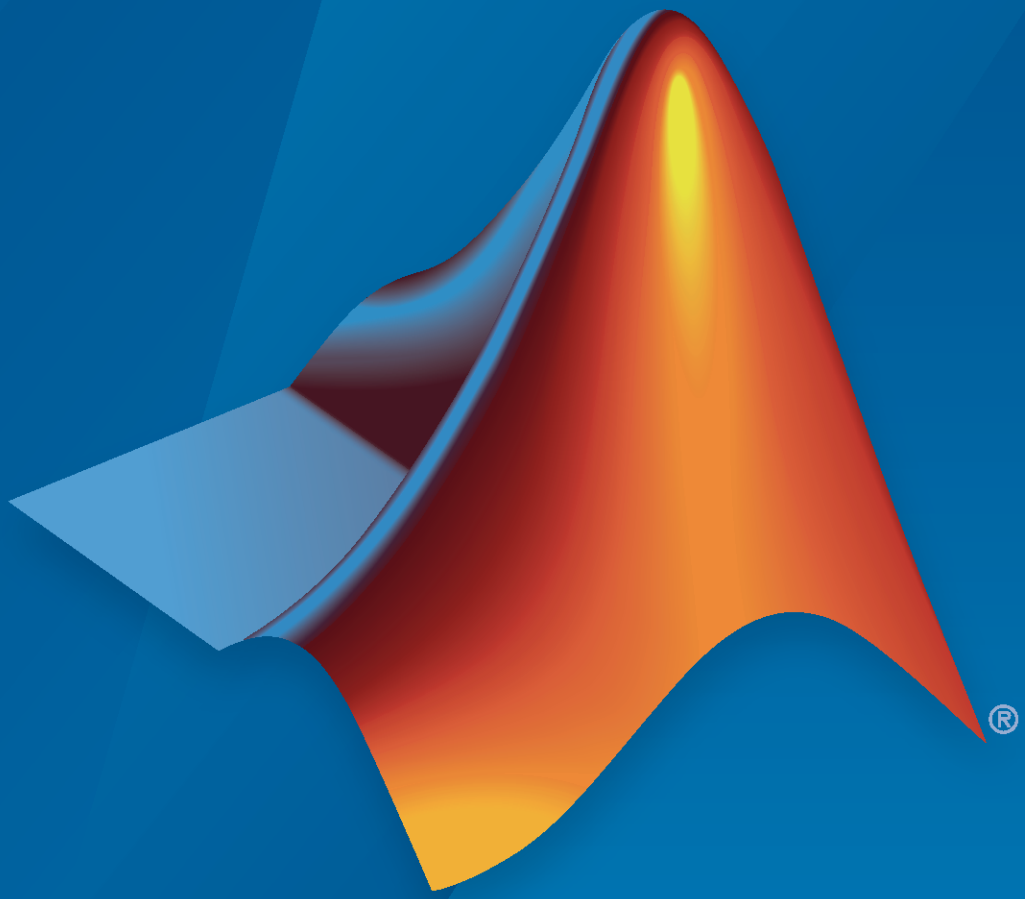
Statistics and Machine Learning Toolbox™
用户指南



MATLAB®

R2021a

MathWorks®

如何联系 MathWorks

最新动态: **www.mathworks.com**

销售和服务: **www.mathworks.com/sales_and_services**

用户社区: **www.mathworks.com/matlabcentral**

技术支持: **www.mathworks.com/support/contact_us**

电话: 010-59827000

迈斯沃克软件 (北京) 有限公司
北京市朝阳区望京东园四区 6 号楼
北望金辉大厦 16 层 1604

Statistics and Machine Learning Toolbox™ 用户指南

修订历史记录

1993 年 9 月	第一次印刷	版本 1.0
1996 年 3 月	第二次印刷	版本 2.0
1997 年 1 月	第三次印刷	版本 2.11
2000 年 11 月	第四次印刷	3.0 版（版本 12）中的修订内容
2001 年 5 月	第五次印刷	轻微修改版
2002 年 7 月	第六次印刷	4.0 版（版本 13）中的修订内容
2003 年 2 月	仅限在线版本	4.1 版（版本 13.0.1）中的修订内容
2004 年 6 月	第七次印刷	5.0 版（版本 14）中的修订内容
2004 年 10 月	仅限在线版本	5.0.1 版（版本 14SP1）中的修订内容
2005 年 3 月	仅限在线版本	5.0.2 版（版本 14SP2）中的修订内容
2005 年 9 月	仅限在线版本	5.1 版（版本 14SP3）中的修订内容
2006 年 3 月	仅限在线版本	版本 5.2（版本 2006a）中的修订内容
2006 年 9 月	仅限在线版本	版本 5.3（版本 2006b）中的修订内容
2007 年 3 月	第八次印刷	版本 6.0（版本 2007a）中的修订内容
2007 年 9 月	第九次印刷	版本 6.1（版本 2007b）中的修订内容
2008 年 3 月	仅限在线版本	版本 6.2（版本 2008a）中的修订内容
2008 年 10 月	仅限在线版本	版本 7.0（版本 2008b）中的修订内容
2009 年 3 月	仅限在线版本	版本 7.1（版本 2009a）中的修订内容
2009 年 9 月	仅限在线版本	版本 7.2（版本 2009b）中的修订内容
2010 年 3 月	仅限在线版本	版本 7.3（版本 2010a）中的修订内容
2010 年 9 月	仅限在线版本	版本 7.4（版本 2010b）中的修订内容
2011 年 4 月	仅限在线版本	版本 7.5（版本 2011a）中的修订内容
2011 年 9 月	仅限在线版本	版本 7.6（版本 2011b）中的修订内容
2012 年 3 月	仅限在线版本	版本 8.0（版本 2012a）中的修订内容
2012 年 9 月	仅限在线版本	版本 8.1（版本 2012b）中的修订内容
2013 年 3 月	仅限在线版本	版本 8.2（版本 2013a）中的修订内容
2013 年 9 月	仅限在线版本	版本 8.3（版本 2013b）中的修订内容
2014 年 3 月	仅限在线版本	版本 9.0（版本 2014a）中的修订内容
2014 年 10 月	仅限在线版本	版本 9.1（版本 2014b）中的修订内容
2015 年 3 月	仅限在线版本	版本 10.0（版本 2015a）中的修订内容
2015 年 9 月	仅限在线版本	版本 10.1（版本 2015b）中的修订内容
2016 年 3 月	仅限在线版本	版本 10.2（版本 2016a）中的修订内容
2016 年 9 月	仅限在线版本	版本 11（版本 2016b）中的修订内容
2017 年 3 月	仅限在线版本	版本 11.1（版本 2017a）中的修订内容
2017 年 9 月	仅限在线版本	版本 11.2（版本 2017b）中的修订内容
2018 年 3 月	仅限在线版本	版本 11.3（版本 2018a）中的修订内容
2018 年 9 月	仅限在线版本	版本 11.4（版本 2018b）中的修订内容
2019 年 3 月	仅限在线版本	版本 11.5（版本 2019a）中的修订内容
2019 年 9 月	仅限在线版本	版本 11.6（版本 2019b）中的修订内容
2020 年 3 月	仅限在线版本	版本 11.7（版本 2020a）中的修订内容
2020 年 9 月	仅限在线版本	版本 12.0（版本 2020b）中的修订内容
2021 年 3 月	仅限在线版本	版本 12.1（版本 2021a）中的修订内容

目录

13 非线性回归

14 生存分析

15 多元方法

16 聚类分析

17 参数化分类

18 非参数化有监督学习

参考书目

C

快速入门

组织数据

访问数据集数组变量中的数据

此示例说明如何使用数据集数组变量及其数据。

按名称访问变量。

您可以通过使用变量（列）名称和点索引来访问变量数据或选择变量子集。加载样本数据集数组。显示 **hospital** 中变量的名称。

```
load hospital
hospital.Properties.VarNames(:)
```

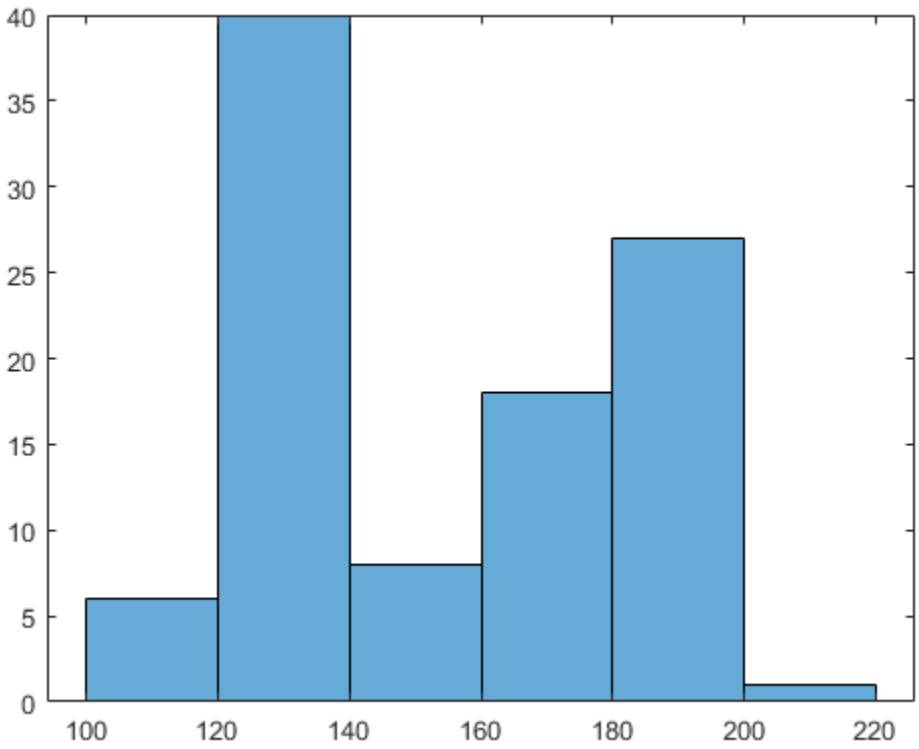
```
ans = 7x1 cell
    {'LastName'    }
    {'Sex'         }
    {'Age'         }
    {'Weight'      }
    {'Smoker'      }
    {'BloodPressure'}
    {'Trials'      }
```

数据集数组有 7 个变量（列）和 100 个观测值（行）。您可以在工作区窗口中双击 **hospital** 以在变量编辑器中查看数据集数组。

绘制直方图。

绘制变量 **Weight** 中数据的直方图。

```
figure
histogram(hospital.Weight)
```

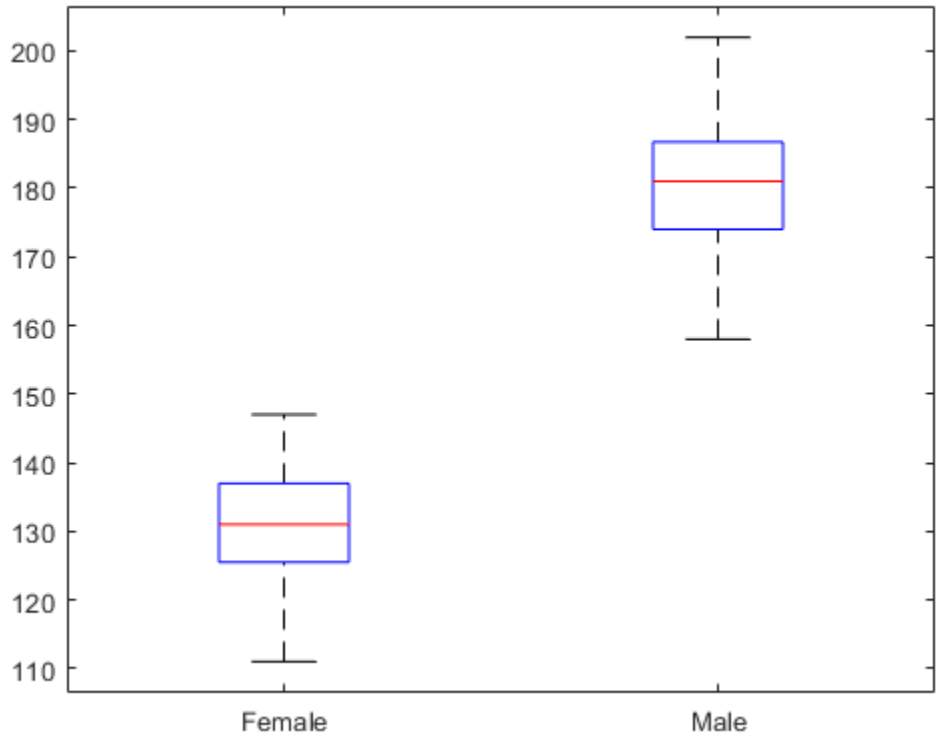


直方图显示体重呈双峰分布。

绘制按类别分组的数据。

绘制按 **Sex** 中的值分组（男性和女性）的 **Weight** 的箱线图。也就是说，使用变量 **Sex** 作为分组变量。

```
figure
boxplot(hospital.Weight,hospital.Sex)
```



箱线图表明性别是体重呈双峰分布的原因。

选择一个变量子集。

创建一个新数据集数组，其中仅包含变量 **LastName**、**Sex** 和 **Weight**。您可以通过名称或列号访问变量。

```
ds1 = hospital(:,{'LastName','Sex','Weight'});
ds2 = hospital(:,[1,2,4]);
```

数据集数组 **ds1** 和 **ds2** 是等同的。在对数据集数组进行索引时，使用括号 **( )** 可保留数据类型；也就是说，基于数据集数组的子集创建一个数据集数组。您还可以使用变量编辑器基于变量和观测值的子集创建一个新数据集数组。

转换变量数据类型。

将变量 **Smoker** 的数据类型从逻辑值转换为名义值，标签为 **No** 和 **Yes**。

```
hospital.Smoker = nominal(hospital.Smoker,{'No','Yes'});
class(hospital.Smoker)
```

```
ans =
'nominal'
```

探查数据。

显示 **Smoker** 的前 10 个元素。

```
hospital.Smoker(1:10)

ans = 10x1 nominal
    Yes
    No
    No
    No
    No
    No
    Yes
    No
    No
    No
```

如果要更改名义数组中的水平标签，请使用 **setlabels**。

添加变量。

变量 **BloodPressure** 是 100×2 数组。第一列对应于收缩压，第二列对应于舒张压。将此数组分成两个新变量 **SysPressure** 和 **DiaPressure**。

```
hospital.SysPressure = hospital.BloodPressure(:,1);
hospital.DiaPressure = hospital.BloodPressure(:,2);
hospital.Properties.VarNames(:)

ans = 9x1 cell
    {'LastName'    }
    {'Sex'         }
    {'Age'         }
    {'Weight'      }
    {'Smoker'      }
    {'BloodPressure'}
    {'Trials'      }
    {'SysPressure' }
    {'DiaPressure' }
```

数据集数组 **hospital** 有两个新变量。

按名称搜索变量。

使用 **regexp** 查找 **hospital** 中变量名称包含 **'Pressure'** 的变量。创建只包含这些变量的新数据集数组。

```
bp = regexp(hospital.Properties.VarNames,'Pressure');
bpIdx = cellfun(@isempty,bp);
bpData = hospital(:,~bpIdx);
bpData.Properties.VarNames(:)

ans = 3x1 cell
    {'BloodPressure'}
    {'SysPressure' }
    {'DiaPressure' }
```

新数据集数组 **bpData** 仅包含血压变量。

删除变量。

从数据集数组 **hospital** 中删除变量 **BloodPressure**。

```
hospital.BloodPressure = [];
hospital.Properties.VarNames(:)

ans = 8x1 cell
    {'LastName'   }
    {'Sex'        }
    {'Age'        }
    {'Weight'     }
    {'Smoker'     }
    {'Trials'     }
    {'SysPressure'}
    {'DiaPressure'}
```

变量 **BloodPressure** 不再在数据集数组中。

另请参阅

dataset

相关示例

- “Add and Delete Variables”
- “Calculations on Dataset Arrays”
- “Dataset Arrays in the Variables Editor”
- “Index and Search Dataset Arrays”

详细信息

- “Dataset Arrays”
- “Grouping Variables”

3

描述性统计量

4

统计可视化

可视化多变量数据

此示例说明如何使用各种统计图可视化多变量数据。许多统计分析只涉及两个变量：预测变量和响应变量。利用二维散点图、二元直方图、箱线图等可以轻松地可视化这些数据。还可以利用三维散点图或带有第三个变量（比如采用颜色编码）的二维散点图来可视化三元数据。但是，许多数据集涉及大量变量，使直接可视化变得更加困难。此示例将探讨在 MATLAB® 中使用 Statistics and Machine Learning Toolbox™ 可视化高维数据的一些方法。

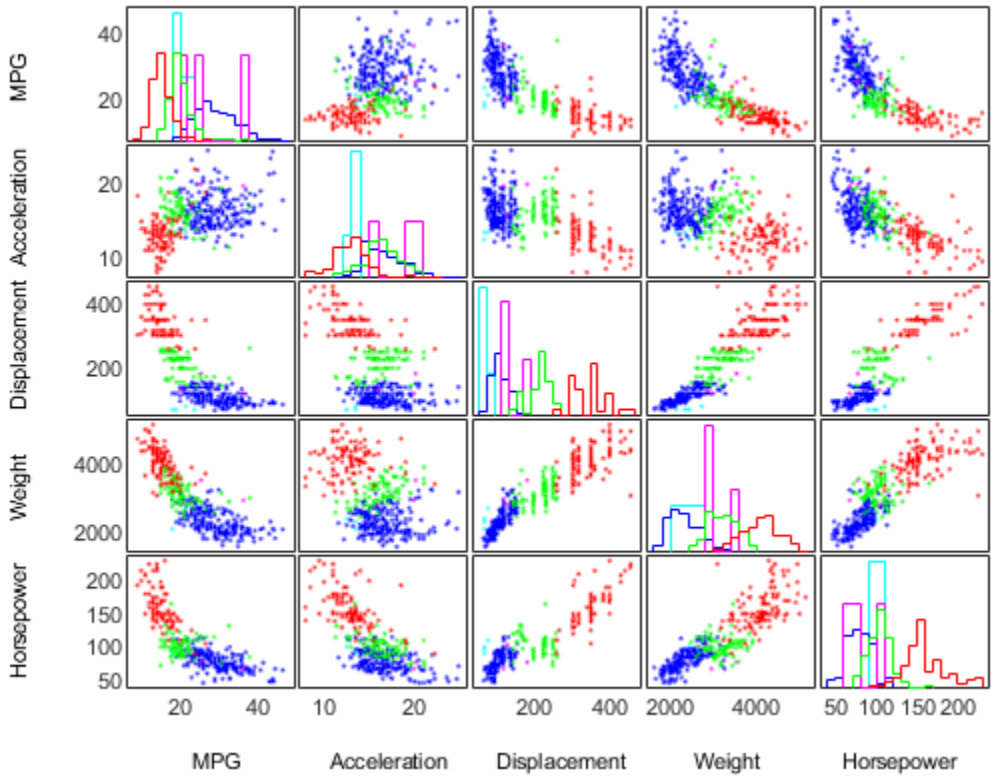
在此示例中，我们将使用 **carbig** 数据集，该数据集包含 20 世纪 70 和 80 年代约 400 辆汽车的各种测量变量。我们将使用燃油效率（每加仑汽油可行驶的英里数，以 MPG 为单位）、加速度（从 0MPH 到 60MPH 需要的时间，以秒为单位）、发动机排量（以立方英寸为单位）、重量和马力等值来说明多元可视化。我们将按照气缸数量对观测值进行分组。

```
load carbig
X = [MPG,Acceleration,Displacement,Weight,Horsepower];
varNames = {'MPG'; 'Acceleration'; 'Displacement'; 'Weight'; 'Horsepower'};
```

散点图矩阵

通过低维子空间查看切片是从一定程度上解决二维或三维限制的一种方法。例如，我们可以使用 **gplotmatrix** 函数显示一个阵列，其中包含五个变量两两之间的二元散点图，以及表示每个变量自身的一元直方图。

```
figure
gplotmatrix(X,[],Cylinders,['c' 'b' 'm' 'g' 'r'],[],[],false);
text([.08 .24 .43 .66 .83], repmat(-.1,1,5), varNames, 'FontSize',8);
text(repmat(-.12,1,5), [.86 .62 .41 .25 .02], varNames, 'FontSize',8, 'Rotation',90);
```

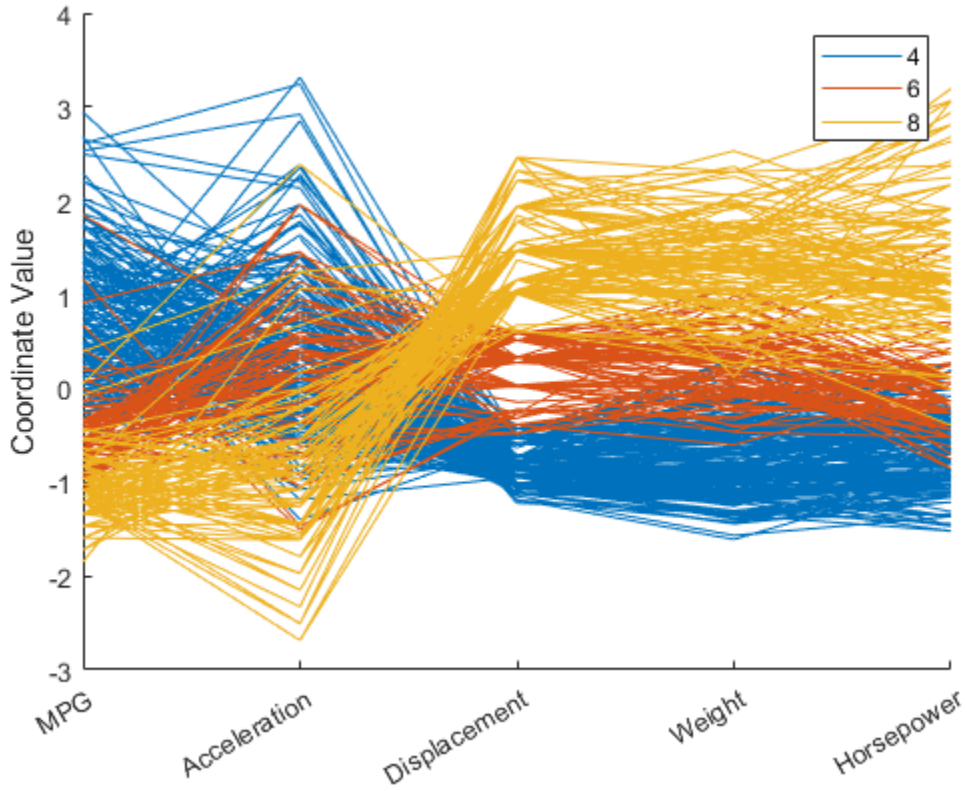


每个散点图中用不同颜色表示不同的气缸数量：蓝色表示 4 缸、绿色表示 6 缸、红色表示 8 缸。还有少数 5 缸汽车，以及视为 3 缸的转子发动机汽车。从这组图中可以轻松找出变量对之间的关系模式。但是，可能有一些重要的高维模式在此图中不容易看出。

平行坐标图

散点图矩阵仅显示二元关系。但是，也可以通过其他替代方法将所有变量显示在一起，便于您研究变量之间的高维关系。最简单的多元图就是平行坐标图。在此图中，坐标轴全部水平排列，而不是像通常的笛卡尔图那样使用正交坐标轴。每个观测项在图中表示为一条由几个线段连接起来的连接线。例如，我们可以绘制所有 4 缸、6 缸或 8 缸汽车的图，然后按组对观测项着色。

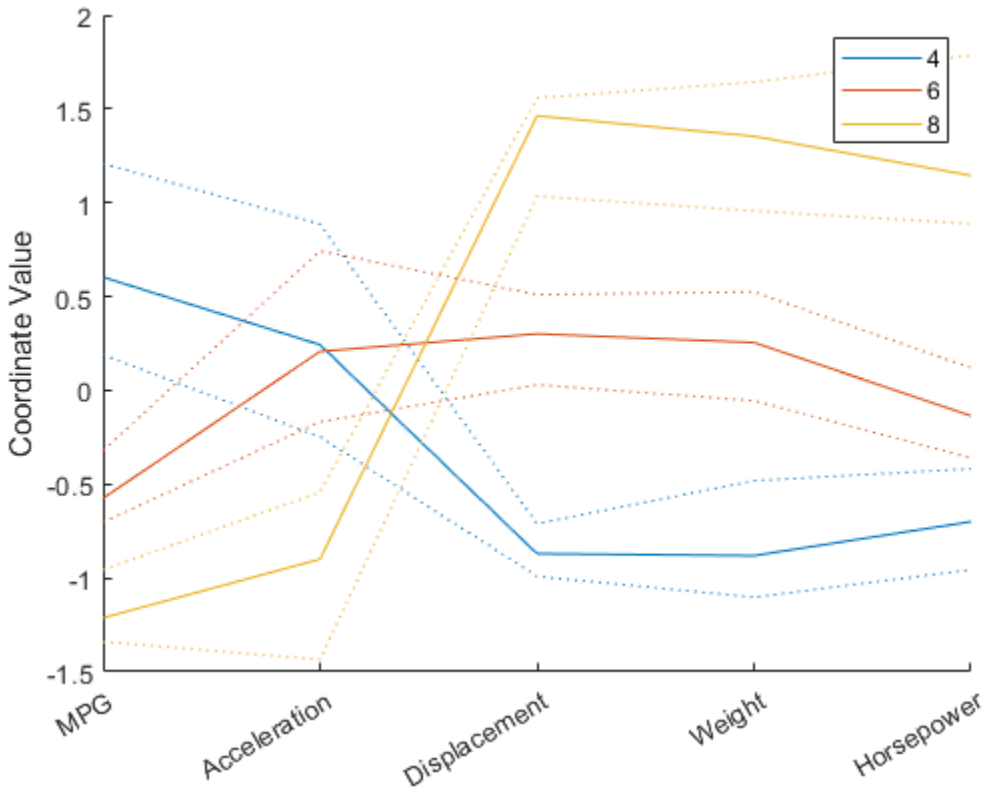
```
Cyl468 = ismember(Cylinders,[4 6 8]);
parallelcoords(X(Cyl468,:), 'group',Cylinders(Cyl468), ...
        'standardize','on', 'labels',varNames)
```



在此图中，水平方向表示坐标轴，垂直方向表示数据。每个观测项由五个变量的测量值组成，每个测量值由对应的线条在每个坐标轴上的高度表示。由于五个变量的范围相差很大，所以此图使用标准化值进行绘制，每个变量都标准化为具有零均值和零单位方差。通过颜色区分，可以从图中看到 8 缸汽车通常具有较低的 MPG 和加速度值，以及较高的排量、重量和马力值。

即使用不同颜色对组进行了区分，在查看包含大量观测值的平行坐标图时，依然不够清晰直观。我们还可以绘制只显示每个组的中位数和四分位数（25% 个点和 75% 个点）的平行坐标图。这样可使组之间的典型差异和相似性更容易区分。另一方面，最有趣的可能是每个组的离群值，而此图中根本未显示。

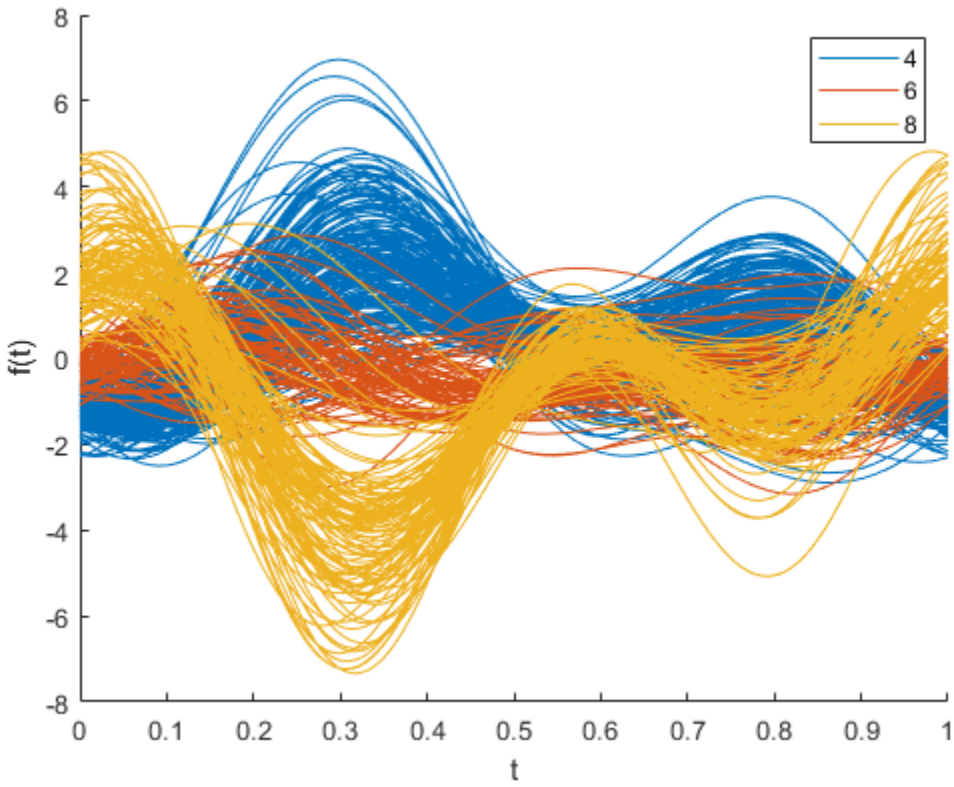
```
parallelcoords(X(Cyl468,:), 'group',Cylinders(Cyl468), ...
        'standardize','on', 'labels',varNames, 'quantile',.25)
```



Andrews 图

另一种类似的多元可视化是 Andrews 图。此图将每个观测值表示为区间 [0,1] 内的平滑函数。

```
andrewsplot(X(Cyl468,:), 'group',Cylinders(Cyl468), 'standardize','on')
```



每个函数均为一个傅里叶级数，系数等于对应的观测值。在此示例中，级数有五个项：一个常项、两个正弦项（周期为 1 和 1/2）和两个与之相似的余弦项。在 Andrews 图中，前三个项对函数形状的影响是最明显的，因此前三个变量的模式往往是最容易识别的。

当 t = 0 时，各组之间存在明显差异，这表明第一个变量 MPG 是 4 缸、6 缸和 8 缸汽车之间的区别特征之一。更有趣的是大约在 t = 1/3 时三个组之间的差异。将此值插入到 Andrews 图函数公式中，我们得到一组系数，这些系数定义区分各组的变量的线性组合。

```
t1 = 1/3;
[1/sqrt(2) sin(2*pi*t1) cos(2*pi*t1) sin(4*pi*t1) cos(4*pi*t1)]
```

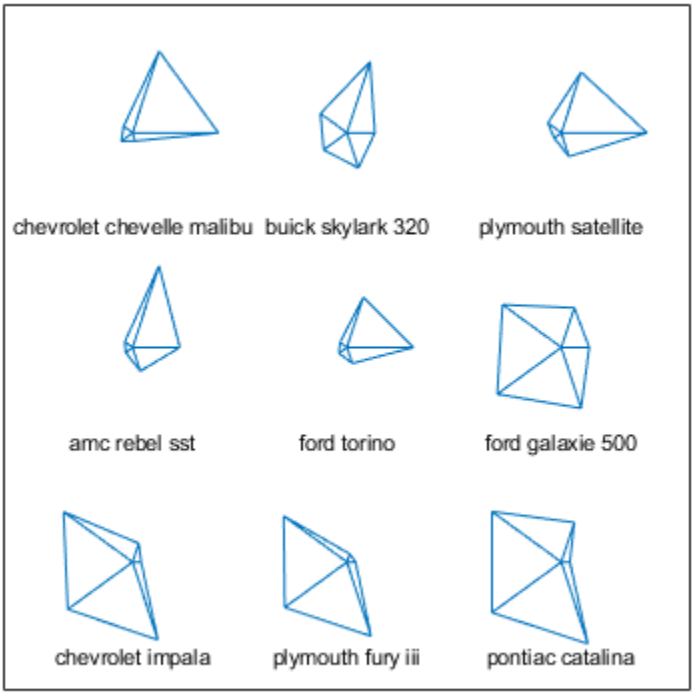
```
ans =

   0.7071    0.8660   -0.5000   -0.8660   -0.5000
```

从这些系数中我们可以看出，区分 4 缸汽车与 8 缸汽车的一个方法是前者具有较高的 MPG 和加速度值，以及较低的排量、马力，特别是重量值，而后者正好相反。这与我们从平行坐标图中得出的结论相同。

图形符号图

可视化多变量数据的另一种方法是使用“图形符号”来表示维度。函数 **glyphplot** 支持两种图形符号：星座图和 Chernoff 脸谱图。例如，下面是汽车数据中前 9 个型号的星座图。星座的每一条边代表一个变量，边长与代表观测值的变量值成正比。

```
h = glyphplot(X(1:9,:), 'glyph','star', 'varLabels',varNames, 'obslabels',Model(1:9,:));
set(h(:,3),'FontSize',8);
```



在实时 MATLAB 图窗窗口中，此图允许使用数据游标交互式探查数据值。例如，点击代表 Ford Torino 的星座中右侧的点，将显示其 MPG 值为 17。

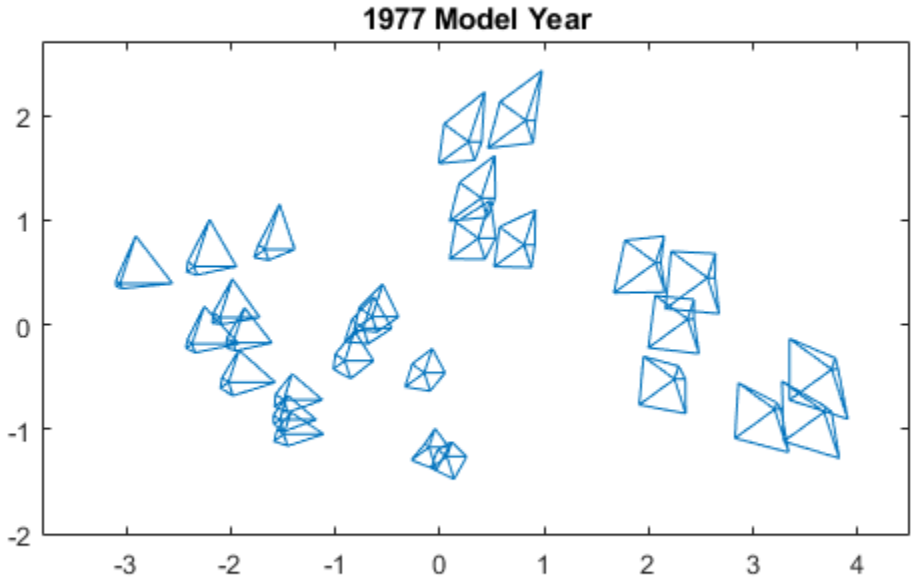
图形符号图和多维尺度分析

在网格上无序绘制星座所得的图可能令人困惑，因为相邻的星座可能看起来完全不同。因此，您可能难以从图中看出明显的模式。将多维尺度分析 (MDS) 与图形符号图结合起来通常很有用。举例说明，我们首先选择 1977 年的所有汽车，然后使用 **zscore** 函数将五个变量都标准化为具有零均值和零单位方差。然后我们计算这些标准化观测值中的欧几里德距离作为相异度的度量。此选择在实际应用中可能过于简单化，但此处只是为了方便说明。

```
models77 = find((Model_Year==77));
dissimilarity = pdist(zscore(X(models77,:)));
```

最后，我们使用 **mdscale** 创建一组二维位置（点间距近似于原始高维数据之间的相异度），并使用这些位置绘制图形符号图。此二维图中的距离或许只能粗略地重现数据，但对于这类图而言已经够用。

```
Y = mdscale(dissimilarity,2);
glyphplot(X(models77,:), 'glyph','star', 'centers',Y, ...
      'varLabels',varNames, 'obslabels',Model(models77,:), 'radius',.5);
title('1977 Model Year');
```

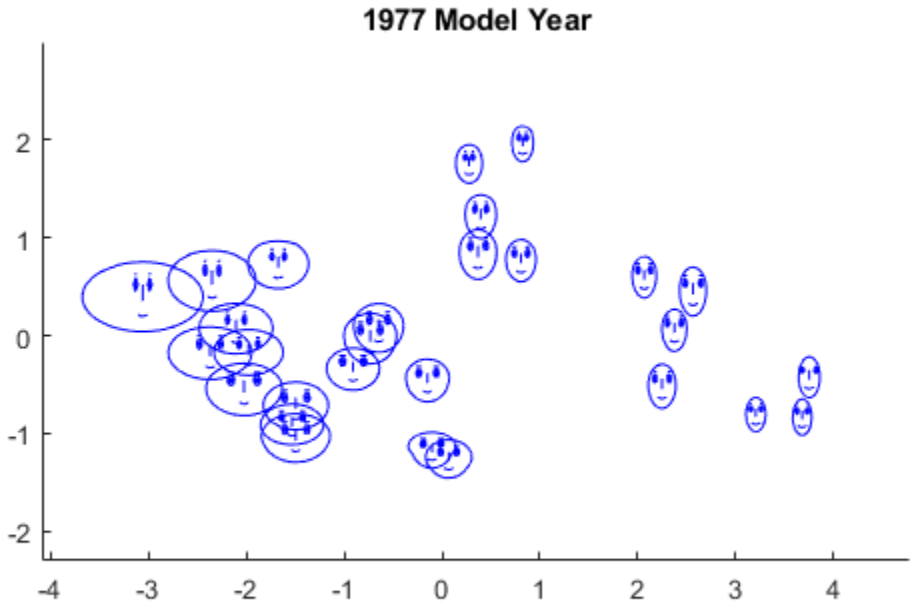


在此图中，我们使用了 MDS 作为降维方法来创建二维图。这通常可能导致信息丢失，但通过绘制图形符号图，我们已经融入了数据中的所有高维信息。使用 MDS 是为了使数据的变化呈现某种规律性，从而更容易看到图形符号中的模式。

与前面的图一样，此图也可以在实时图窗窗口中进行交互式探查。

另一种图形符号是 Chernoff 脸谱图。此图形符号将每个观测值的数据值编码为面部特性，例如面孔的大小、面孔的形状、眼睛的位置等。

```
glyphplot(X(models77,:), 'glyph','face', 'centers',Y, ...
      'varLabels',varNames, 'obslabels',Model(models77,:));
title('1977 Model Year');
```



此处，面孔大小和前额/下巴相对大小这两个最为明显的特征对应 MPG 和加速度，前额形状和下巴形状则分别对应排量和重量。两眼间距对应马力。值得注意的是，前额宽下巴窄的面孔很少，前额窄下巴宽的面孔也很少，这表明排量和重量这两个变量正线性相关。这也是我们在散点图矩阵中所看到的。

何种关系最为明显取决于特征与变量的对应关系，而在 **glyphplot** 中，您可以很方便地更改所选的对应关系。

close

概率分布

使用 copula 仿真相关随机变量

此示例说明当变量之间存在复杂关系时，或者当各个变量来自不同分布时，如何使用 copula 从多元分布中生成数据。

要运行融合了随机输入或噪声的仿真，MATLAB® 是理想的工具。Statistics and Machine Learning Toolbox™ 提供了可用来根据许多常见的一元分布创建随机数据序列的函数，还包括一些可从多元分布（例如多元正态分布和多元 t 分布）生成随机数据的函数。但是，并没有一种内置的方法可以为所有边缘分布生成多元分布，或者在变量来自不同分布的情况下生成多元分布。

近来，copulas 在仿真模型中越来越常用。copula 是一种函数，它描述变量之间的相关性，可用来创建分布，以对相关的多变量数据进行建模。使用 copula，数据分析师可以通过指定边缘一元分布，并选择一个特定的 copula 来描述变量间的相关结构，从而构建出一个多元分布。copula 也可以用于二元分布以及更高维的分布。在此示例中，我们将讨论如何使用 Statistics and Machine Learning Toolbox，在 MATLAB 中使用 copula 生成相关的多元随机数据。

仿真输入之间的相关性

蒙特卡罗模拟的设计决策之一是为随机输入选择概率分布。为每个变量选择一种分布往往很简单，但确定输入之间应该存在什么样的相关性却可能不那么简单。理想情况下，仿真的输入数据应反映要建模的实际数量之间已知的相关性。然而，判断仿真中的任何相关性时可以依据的信息可能很少或根本没有，在这种情况下，最好的做法是尝试不同的可能性，以确定模型的敏感性。

但是，当输入数据的分布不是标准多元分布时，很难真正生成具有相关性的随机输入。而且，有些标准多元分布只能对非常有限的几种相关性进行建模。我们可以始终将输入视为各自独立，这很简便，但并不总是合理的，有可能导致错误的结论。

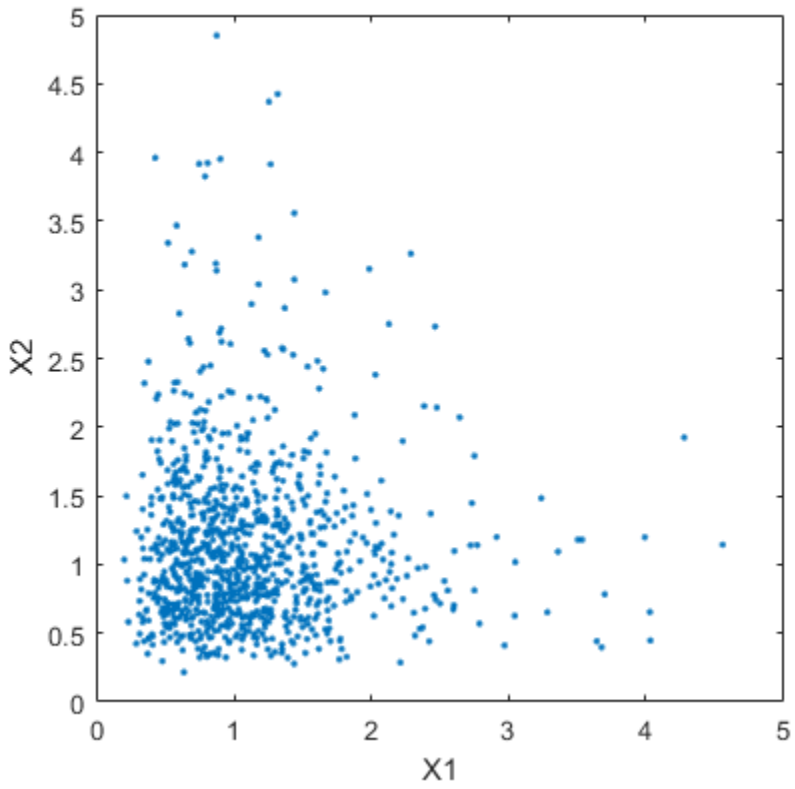
例如，在有关金融风险的蒙特卡罗模拟中，可能有一些表示各种保险损失来源的随机输入。这些输入可以建模为对数正态随机变量。我们会想了解两两输入间的相关性对仿真结果有何影响。确实，从实际数据中可能已经知道，相同的随机条件可对两个损失来源都产生影响，如果在仿真中忽略这一点，可能会导致错误的结论。

独立的对数正态随机变量的仿真非常繁琐。最简单的方法是使用 **lognrnd** 函数。此处，我们将使用 **mvnrnd** 函数生成 n 对独立的正态随机变量，然后计算它们的幂。请注意，这里使用的协方差矩阵是对角矩阵，即 Z 的各列之间是独立的。

```
n = 1000;
sigma = .5;
SigmaInd = sigma.^2 .* [1 0; 0 1]
```

```
SigmaInd =

    0.2500         0
         0    0.2500
```

```
ZInd = mvnrnd([0 0], SigmaInd, n);
XInd = exp(ZInd);
plot(XInd(:,1),XInd(:,2),'.');
axis equal;
axis([0 5 0 5]);
xlabel('X1');
ylabel('X2');
```



生成具有相关性的二元对数正态随机变量也很简单，可使用非零项不在对角线上的协方差矩阵。

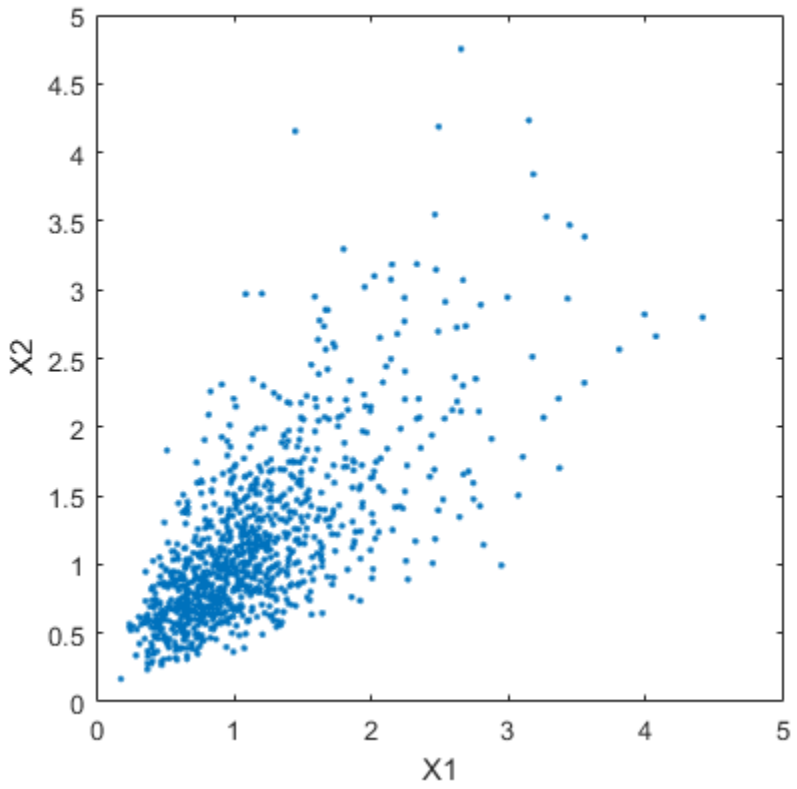
```
rho = .7;
SigmaDep = sigma.^2 .* [1 rho; rho 1]
```

```
SigmaDep =

    0.2500    0.1750
    0.1750    0.2500
```

```
ZDep = mvnrnd([0 0], SigmaDep, n);
XDep = exp(ZDep);
```

第二个散点图说明这两种二元分布之间的差异。

```
plot(XDep(:,1),XDep(:,2),'.');
axis equal;
axis([0 5 0 5]);
xlabel('X1');
ylabel('X2');
```



很明显，第二个数据集中较大的 X1 值更倾向于与较大的 X2 值关联，小值之间也是如此。这种相关性由基础二元正态分布的相关参数 rho 决定。从仿真中得出的结论可能很大程度上取决于生成的 X1 和 X2 是否存在相关性。

本示例中的二元对数正态分布是一个简单的解决方法，它当然可以泛化应用于更高维度以及边缘分布为其他对数正态分布的情况。也有一些多元分布，例如，多元 t 分布和 Dirichlet 分布，它们分别用于仿真相关的 t 随机变量和 beta 随机变量。但是，简单的多元分布并不多，而且仅适用于边缘分布都为同一族分布（甚至是完全相同的分布）的情形。在很多情况下，这可能会成为限制。

构造相关二元分布的更通用方法

尽管上面创建二元对数正态分布的构造很简单，但它说明了一种更普遍适用的方法。首先，我们从二元正态分布中生成值对。两两变量间存在统计相关性，而且每个变量都具有正态边缘分布。接下来，分别对每个变量进行变换（指数函数），将边缘分布变成对数正态分布。变换后的变量仍具有统计相关性。

如果能够找到一种合适的变换，就可以将此方法泛化，从而为其他边缘分布生成相关二元随机向量。实际上，确实存在构造这种变换的一般方法，尽管不像求幂那样简单。

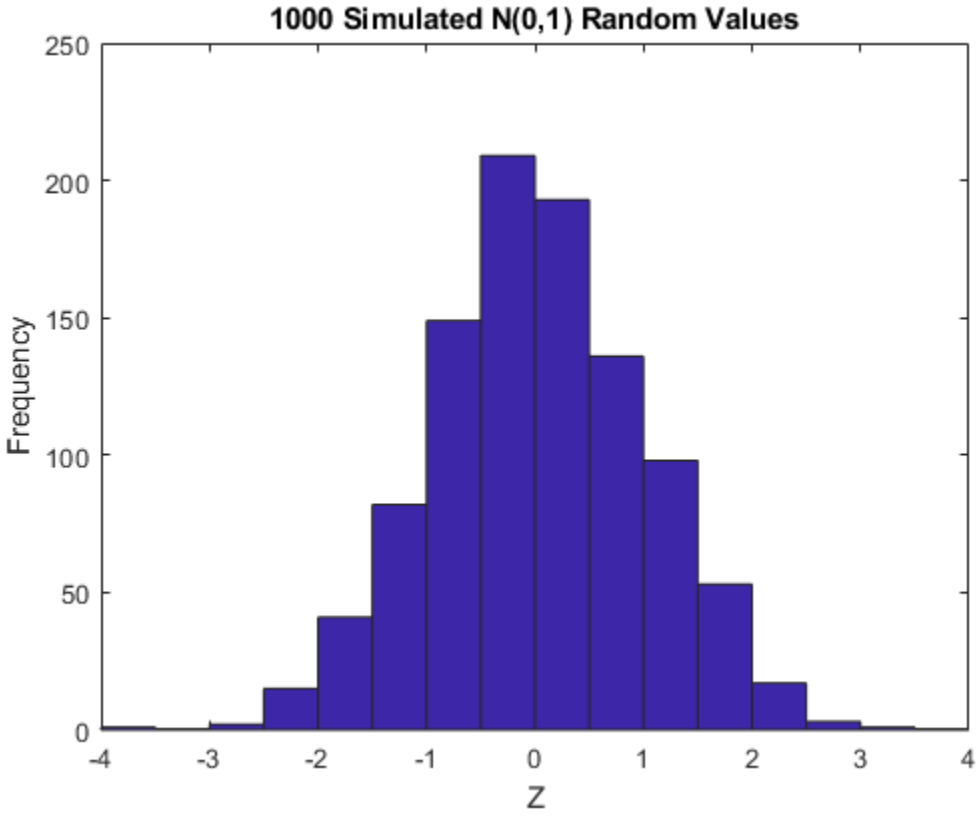
根据定义，将正态 CDF（这里用 PHI 表示）应用于标准正态随机变量将生成在区间 [0,1] 内均匀分布的随机变量。为说明这一点，假设 Z 具有标准正态分布，则 U = PHI(Z) 时，CDF 为

$$\Pr\{U <= u0\} = \Pr\{PHI(Z) <= u0\} = \Pr\{Z <= PHI^{(-1)}(u0)\} = u0,$$

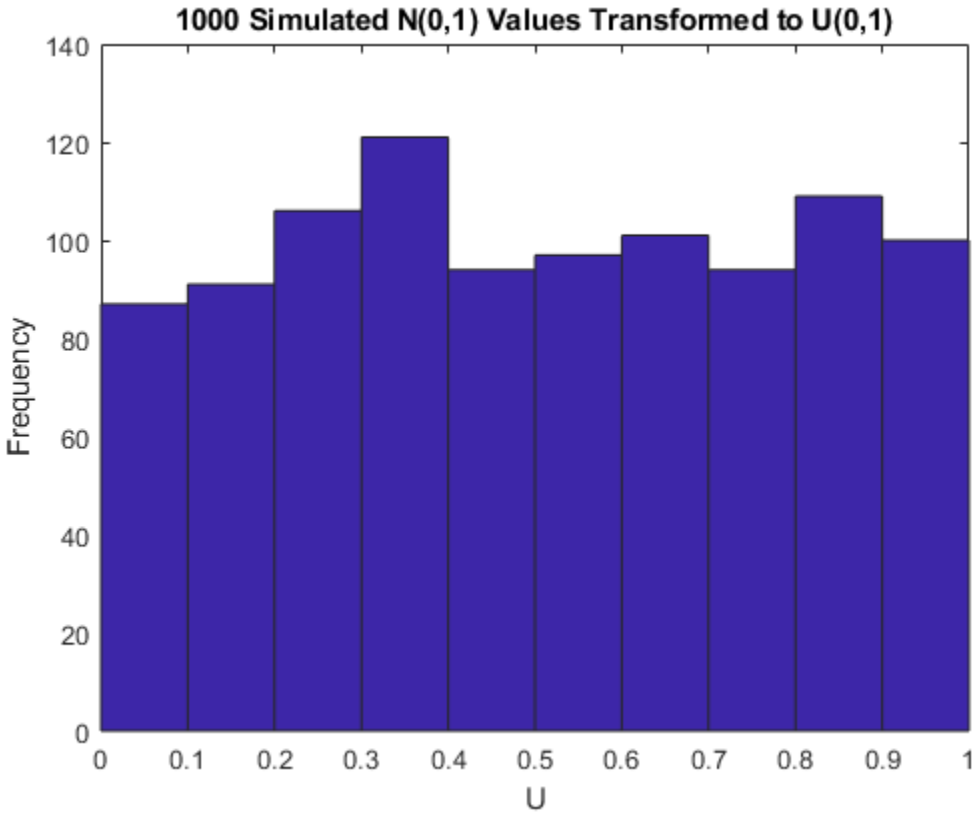
而这就是 U(0,1) 随机变量的 CDF。通过为仿真的正态值和仿真的变换值绘制的直方图证实了这一点。

```
n = 1000;
z = normrnd(0,1,n,1);
```

```
hist(z,-3.75:.5:3.75);
xlim([-4 4]);
title('1000 Simulated N(0,1) Random Values');
xlabel('Z');
ylabel('Frequency');
```

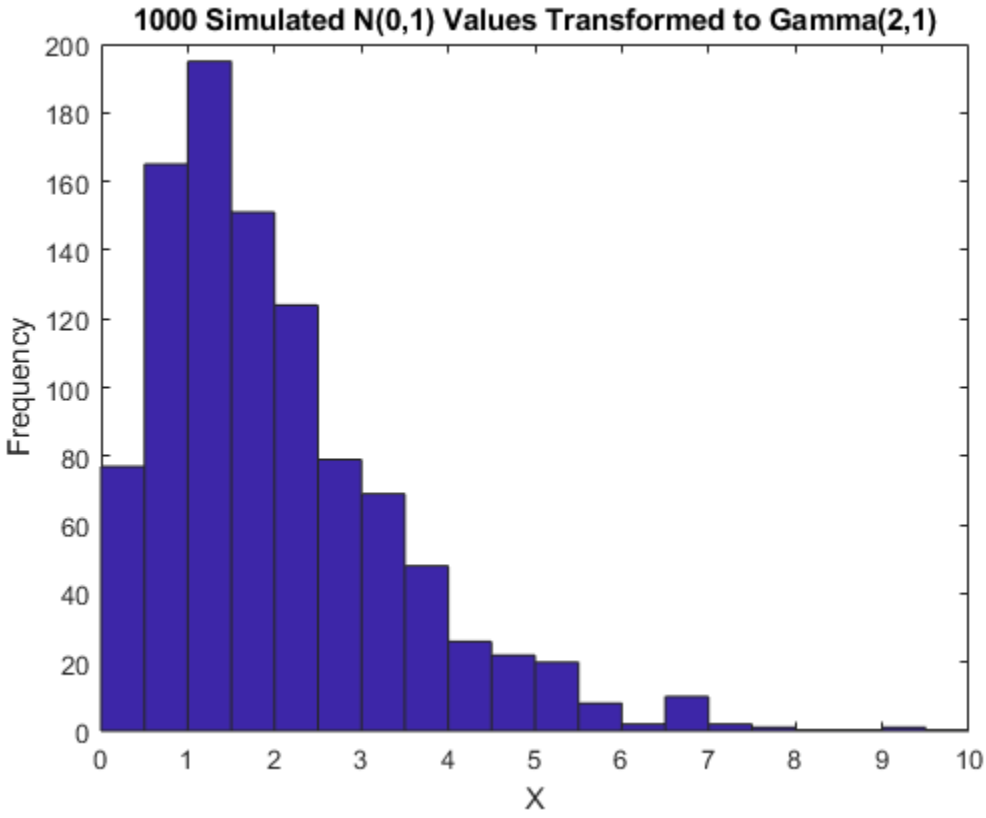



```
u = normcdf(z);
hist(u,.05:.1:.95);
title('1000 Simulated N(0,1) Values Transformed to U(0,1)');
xlabel('U');
ylabel('Frequency');
```



现在，借用一元随机数生成理论，将任意分布 F 的逆 CDF 应用于 U(0,1) 随机变量，可得到分布正好为 F 的随机变量。这就是逆变换法。其证明过程与上述正向情形的证明过程基本相反。另一个直方图说明了变换为 gamma 分布的结果。

```
x = gaminv(u,2,1);
hist(x,.25:.5:9.75);
title('1000 Simulated N(0,1) Values Transformed to Gamma(2,1)');
xlabel('X');
ylabel('Frequency');
```



这个两步变换方法可以应用于标准二元正态分布的每个变量，从而生成具有任意边缘分布的相关随机变量。由于变换分别作用于每个成分，因此生成的两个随机变量甚至不需要具有相同的边缘分布。变换定义为

```
Z = [Z1 Z2] ~ N([0 0],[1 rho; rho 1])
U = [PHI(Z1) PHI(Z2)]
X = [G1(U1) G2(U2)]
```

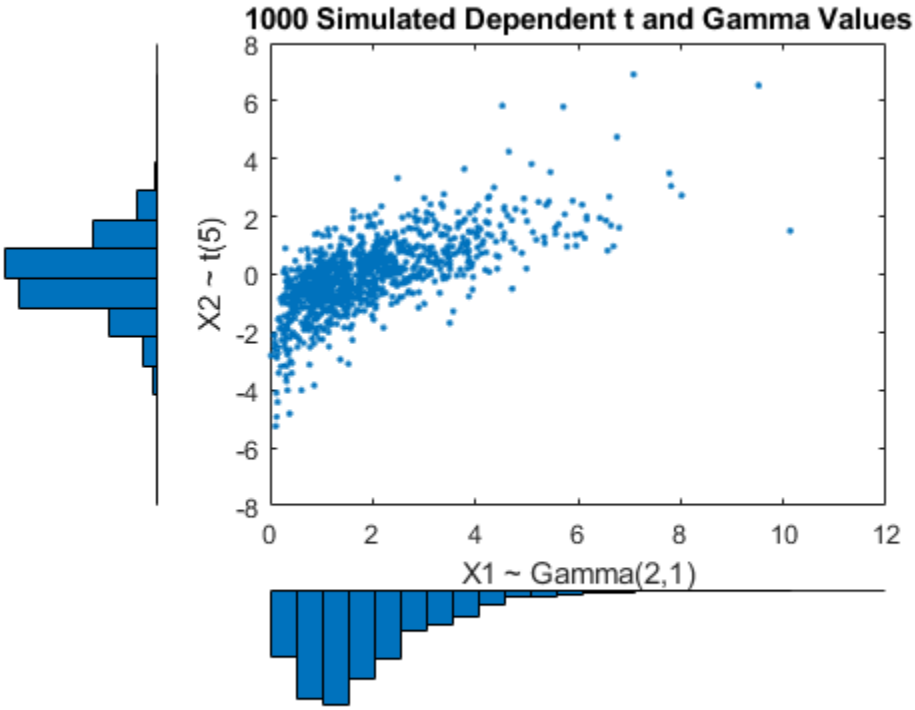
其中 G1 和 G2 是两个可能不相同的分布的逆 CDF。例如，我们可以从具有 t(5) 和 Gamma(2,1) 边缘分布的二元分布生成随机向量。

```
n = 1000;
rho = .7;
Z = mvnrnd([0 0], [1 rho; rho 1], n);
U = normcdf(Z);
X = [gaminv(U(:,1),2,1) tinv(U(:,2),5)];
```

在下图中，直方图显示在散点图的左侧和下方，以显示边缘分布和相关性。

```
[n1,ctr1] = hist(X(:,1),20);
[n2,ctr2] = hist(X(:,2),20);
subplot(2,2,2);
plot(X(:,1),X(:,2),'.');
axis([0 12 -8 8]);
h1 = gca;
title('1000 Simulated Dependent t and Gamma Values');
xlabel('X1 ~ Gamma(2,1)');
```

```
ylabel('X2 ~ t(5)');
subplot(2,2,4);
bar(ctr1,-n1,1);
axis([0 12 -max(n1)*1.1 0]);
axis('off');
h2 = gca;
subplot(2,2,1);
barh(ctr2,-n2,1);
axis([-max(n2)*1.1 0 -8 8]);
axis('off');
h3 = gca;
h1.Position = [0.35 0.35 0.55 0.55];
h2.Position = [.35 .1 .55 .15];
h3.Position = [.1 .35 .15 .55];
colormap([.8 .8 1]);
```



秩相关系数

此构造中 X1 和 X2 之间的相关性由基础二元正态分布的相关参数 rho 决定。但是，X1 和 X2 的线性相关系数并不完全等于 rho。例如，在原来的对数正态分布中，该相关系数的闭式解为：

```
cor(X1,X2) = (exp(rho.*sigma.^2) - 1) ./ (exp(sigma.^2) - 1)
```

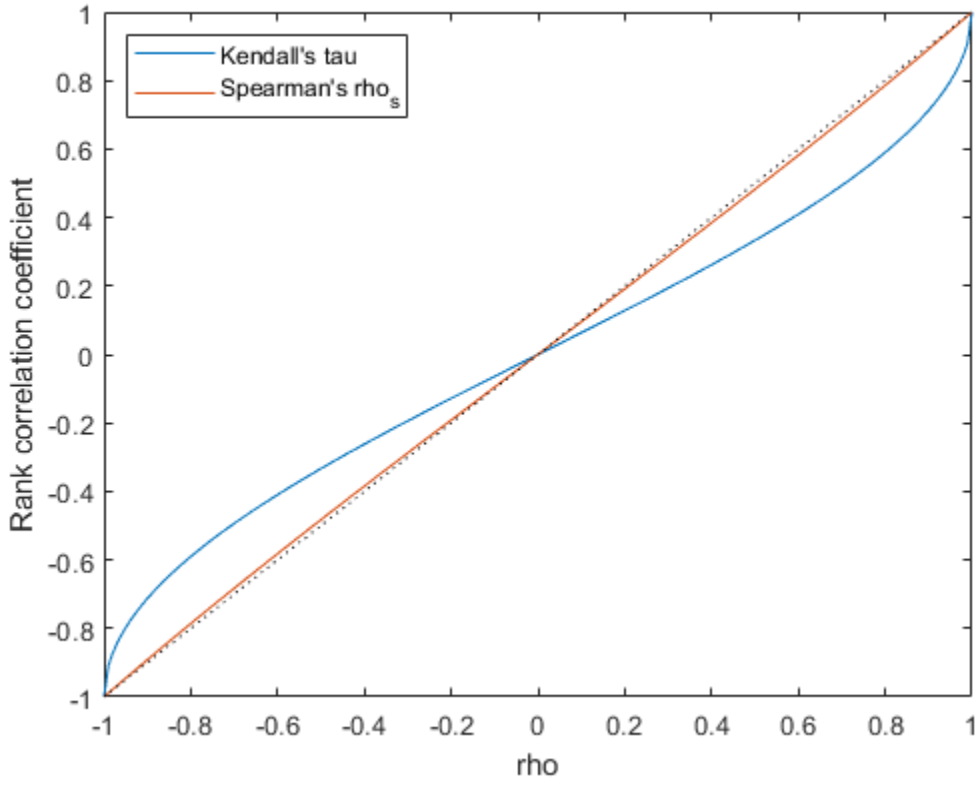
它严格小于 rho，除非 rho 刚好为 1。但更普遍的情形是，例如在上面的 gamma/t 分布中，X1 和 X2 之间的线性相关性很难或无法用 rho 表示，但可以通过仿真来说明存在同样的效应。

这是因为线性相关系数表达的是随机变量之间的线性相关性，当对这些随机变量应用非线性变换后，将不会保留线性相关性。在这种情况下，秩相关系数（如 Kendall 的 tau 或 Spearman 的 rho）更合适。

大体上来讲，这些秩相关衡量的是一个随机变量的大值或小值与另一个随机变量的大值或小值之间的相关程度。然而，与线性相关系数不同的是，秩相关系数只从秩的角度衡量相关性。因此，秩相关在任何单调变换下都会保留。前面所述的变换方法也会保留秩相关。因此，知道二元正态分布 Z 的秩相关就可以准确地确定最后变换的随机变量 X 的秩相关。尽管仍然需要使用 rho 来参数化基础的二元正态分布，但 Kendall tau 或 Spearman rho 在描述随机变量之间的相关性时更有用，因为它们不会随着选择的边缘分布不同而改变。

事实证明，对于二元正态分布，Kendall tau 或 Spearman rho 与线性相关系数 rho 之间存在简单的一一对应关系：

```
tau   = (2/pi)*arcsin(rho)     or   rho = sin(tau*pi/2)
rho_s = (6/pi)*arcsin(rho/2)   or   rho = 2*sin(rho_s*pi/6)

subplot(1,1,1);
rho = -1:.01:1;
tau = 2.*asin(rho)./pi;
rho_s = 6.*asin(rho./2)./pi;
plot(rho,tau,'-', rho,rho_s,'-', [-1 1],[-1 1],'k:');
axis([-1 1 -1 1]);
xlabel('rho');
ylabel('Rank correlation coefficient');
legend('Kendall''s tau', 'Spearman''s rho_s', 'location','northwest');
```



因此，通过为 Z1 和 Z2 之间的线性相关选择正确的 rho 参数值，很容易获得所需的 X1 和 X2 之间的秩相关系数，而不管它们的边缘分布是什么。

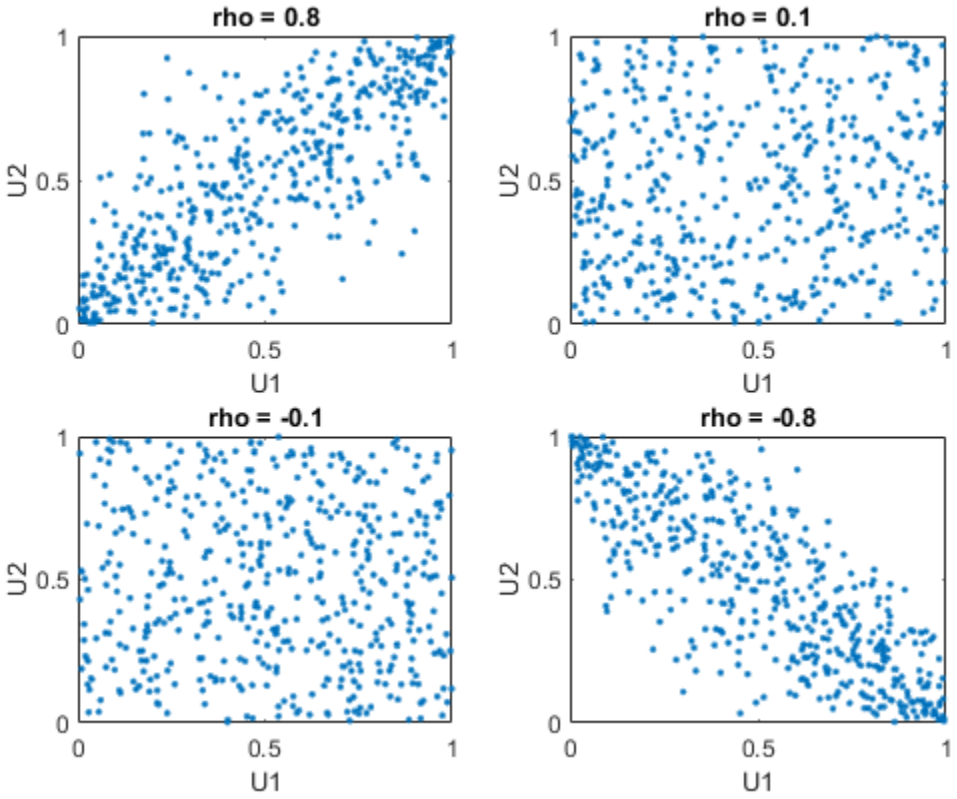
请注意，对于多元正态分布，Spearman 秩相关系数几乎与线性相关系数完全相同。但是，一旦我们变换为最终的随机变量，上面的情形便不再成立了。

copula

上述构造的第一步是定义一个 copula，具体来说，就是高斯 copula。二元 copula 就是两个随机变量的概率分布，每个随机变量的边缘分布都是均匀的。这两个变量之间的关系可以是完全独立、确定性相关（例如，U2 = U1）或者介于二者之间的任何状态。二元高斯 copula 族通过线性相关矩阵 Rho = [1 rho; rho 1] 进行参数化。当 rho 接近 +/- 1 时，U1 和 U2 接近线性相关；当 rho 接近 0 时，U1 和 U2 接近完全独立。

不同 rho 水平的仿真随机值散点图说明了高斯 copula 的不同可能性的范围：

```
n = 500;
Z = mvnrnd([0 0], [1 .8; .8 1], n);
U = normcdf(Z,0,1);
subplot(2,2,1);
plot(U(:,1),U(:,2),'.');
title('rho = 0.8');
xlabel('U1');
ylabel('U2');
Z = mvnrnd([0 0], [1 .1; .1 1], n);
U = normcdf(Z,0,1);
subplot(2,2,2);
plot(U(:,1),U(:,2),'.');
title('rho = 0.1');
xlabel('U1');
ylabel('U2');
Z = mvnrnd([0 0], [1 -.1; -.1 1], n);
U = normcdf(Z,0,1);
subplot(2,2,3);
plot(U(:,1),U(:,2),'.');
title('rho = -0.1');
xlabel('U1');
ylabel('U2');
Z = mvnrnd([0 0], [1 -.8; -.8 1], n);
U = normcdf(Z,0,1);
subplot(2,2,4);
plot(U(:,1),U(:,2),'.');
title('rho = -0.8');
xlabel('U1');
ylabel('U2');
```



U1 和 U2 之间的相关性完全独立于 X1 = G(U1) 和 X2 = G(U2) 的边缘分布。可以为 X1 和 X2 指定任何边缘分布，而它们的秩相关都将保持不变。这是 copula 的主要优势之一，即它们允许分别指定相关性和边缘分布。

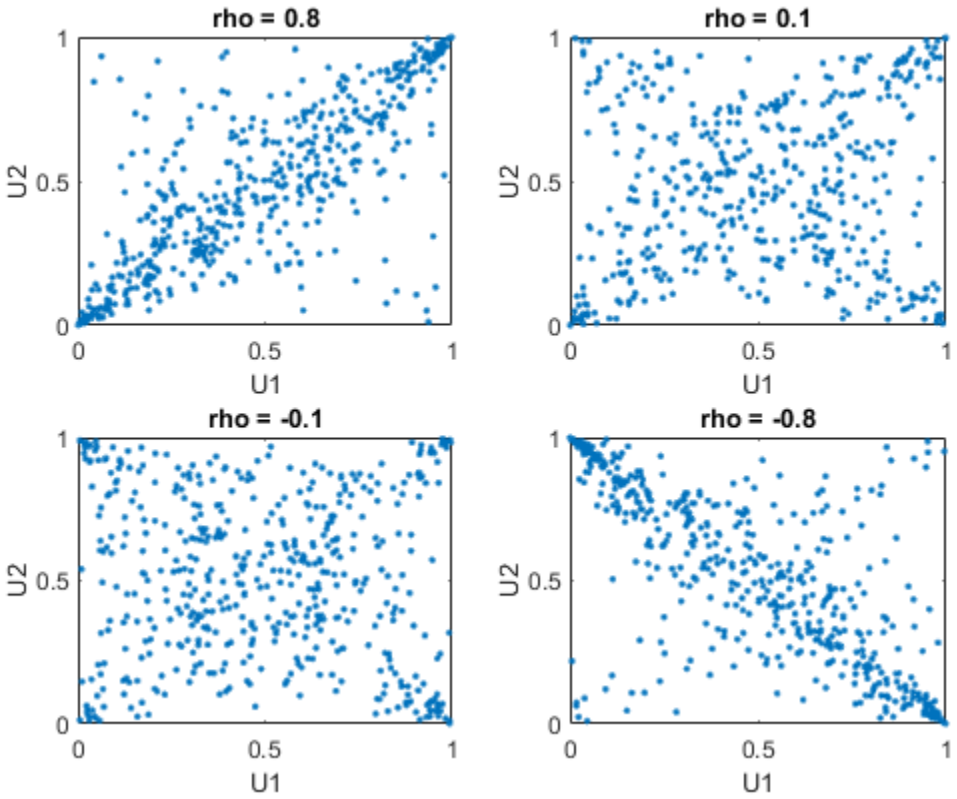
t copula

基于二元 t 分布，使用对应的 t CDF 进行变换，可以构造一个不同的 copula 族。二元 t 分布使用线性相关矩阵 Rho 和自由度 nu 进行参数化。因此，假设多元 t 分布分别具有 1 个和 5 个自由度，我们可以把它们分别叫做 t(1) copula 和 t(5) copula。

不同 rho 水平时的仿真随机值散点图说明了 t(1) copula 的不同可能性的范围：

```
n = 500;
nu = 1;
T = mvtrnd([1 .8; .8 1], nu, n);
U = tcdf(T,nu);
subplot(2,2,1);
plot(U(:,1),U(:,2),'.');
title('rho = 0.8');
xlabel('U1');
ylabel('U2');
T = mvtrnd([1 .1; .1 1], nu, n);
U = tcdf(T,nu);
subplot(2,2,2);
plot(U(:,1),U(:,2),'.');
title('rho = 0.1');
xlabel('U1');
```

```
ylabel('U2');
T = mvtrnd([1 -.1; -.1 1], nu, n);
U = tcdf(T,nu);
subplot(2,2,3);
plot(U(:,1),U(:,2),'.');
title('rho = -0.1');
xlabel('U1');
ylabel('U2');
T = mvtrnd([1 -.8; -.8 1], nu, n);
U = tcdf(T,nu);
subplot(2,2,4);
plot(U(:,1),U(:,2),'.');
title('rho = -0.8');
xlabel('U1');
ylabel('U2');
```

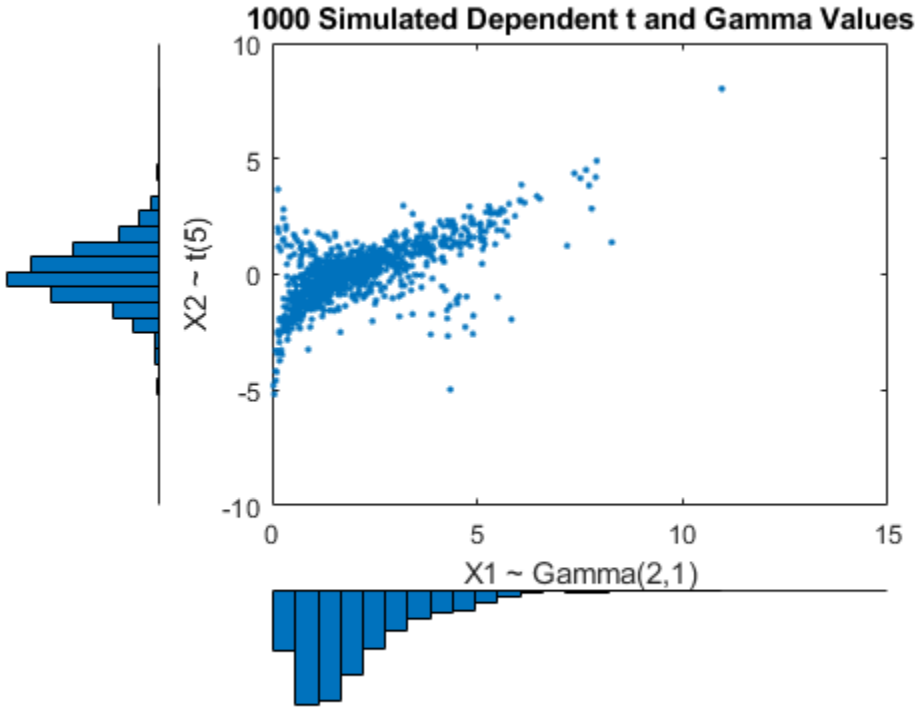


对于 U1 和 U2，t copula 具有均匀的边缘分布，就像高斯 copula 一样。t copula 中各成分之间的秩相关 tau 或 rho_s 也是与高斯 copula 一样的 rho 函数。然而，正如这些图所示，t(1) copula 与高斯 copula 差别很大，即使当它们的成分具有相同的秩相关时也是如此。这种差别是它们的相关性结构所导致的。毫不奇怪，随着自由度参数 nu 变大，t(nu) copula 接近对应的高斯 copula。

与高斯 copula 一样，可以对 t copula 应用任何边缘分布。例如，使用具有 1 个自由度的 t copula，我们可以再次从具有 Gam(2,1) 和 t(5) 边缘的二元分布中生成随机向量：

```
subplot(1,1,1);
n = 1000;
rho = .7;
```

```
nu = 1;
T = mvtrnd([1 rho; rho 1], nu, n);
U = tcdf(T,nu);
X = [gaminv(U(:,1),2,1) tinv(U(:,2),5)];

[n1,ctr1] = hist(X(:,1),20);
[n2,ctr2] = hist(X(:,2),20);
subplot(2,2,2);
plot(X(:,1),X(:,2),'.');
axis([0 15 -10 10]);
h1 = gca;
title('1000 Simulated Dependent t and Gamma Values');
xlabel('X1 ~ Gamma(2,1)');
ylabel('X2 ~ t(5)');
subplot(2,2,4);
bar(ctr1,-n1,1);
axis([0 15 -max(n1)*1.1 0]);
axis('off');
h2 = gca;
subplot(2,2,1);
barh(ctr2,-n2,1);
axis([-max(n2)*1.1 0 -10 10]);
axis('off');
h3 = gca;
h1.Position = [0.35 0.35 0.55 0.55];
h2.Position = [.35 .1 .55 .15];
h3.Position = [.1 .35 .15 .55];
colormap([.8 .8 1]);
```

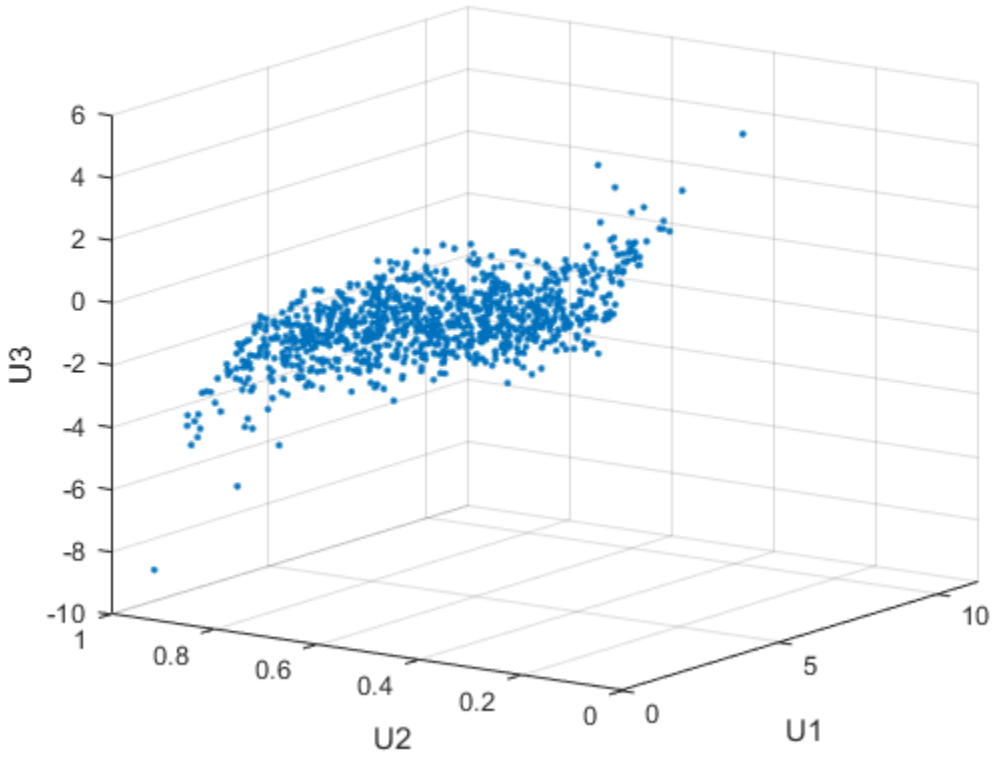


与之前基于高斯 copula 构造的二元 gamma/t 分布相比，这里基于 t(1) copula 构造的分布的变量之间具有相同的边缘分布和相同的秩相关，但相关性结构非常不同。这说明一个事实，即多元分布并不是仅由边缘分布或者仅由相关性定义的。在实际应用中，可以基于实际观测数据选择特定的 copula，也可以使用不同的 copula 来确定仿真结果对输入分布的敏感性。

高阶 copula

高斯 copula 和 t copula 被称为椭圆形 copula。将椭圆形 copula 泛化应用于更多的维数很容易。例如，我们可以按如下所述使用高斯 copula，来仿真具有 Gamma(2,1)、Beta(2,2) 和 t(5) 边缘分布的三元分布的数据。

```
subplot(1,1,1);
n = 1000;
Rho = [1 .4 .2; .4 1 -.8; .2 -.8 1];
Z = mvnrnd([0 0 0], Rho, n);
U = normcdf(Z,0,1);
X = [gaminv(U(:,1),2,1) betainv(U(:,2),2,2) tinv(U(:,3),5)];
plot3(X(:,1),X(:,2),X(:,3),'.');
grid on;
view([-55, 15]);
xlabel('U1');
ylabel('U2');
zlabel('U3');
```



请注意，对于此处使用的相关矩阵 Rho 中的每个条目，线性相关参数 rho 与 Kendall tau 等相关系数间的既定关系都成立。我们可以验证，数据的样本秩相关系数近似等于理论值。

```
tauTheoretical = 2.*asin(Rho)./pi
```

```
tauTheoretical =

    1.0000    0.2620    0.1282
    0.2620    1.0000   -0.5903
    0.1282   -0.5903    1.0000
```

```
tauSample = corr(X, 'type','Kendall')
```

```
tauSample =

    1.0000    0.2655    0.1060
    0.2655    1.0000   -0.6076
    0.1060   -0.6076    1.0000
```

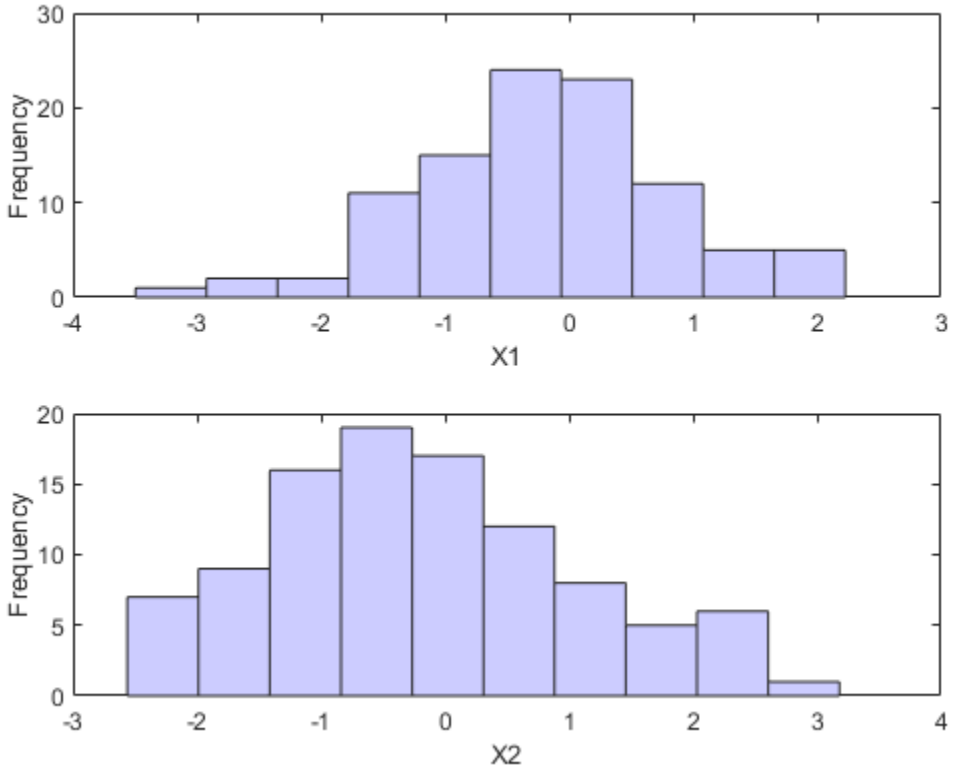
copula 和经验边缘分布

我们已看到，要使用 copula 来仿真相关多变量数据，我们需要指定

1) the copula family (and any shape parameters),
2) the rank correlations among variables, and
3) the marginal distributions for each variable

假设我们有两个股票收益数据集，我们想要使用分布与这两个数据集相同的输入来运行蒙特卡罗模拟。

```
load stockreturns
nobs = size(stocks,1);
subplot(2,1,1);
hist(stocks(:,1),10);
xlabel('X1');
ylabel('Frequency');
subplot(2,1,2);
hist(stocks(:,2),10);
xlabel('X2');
ylabel('Frequency');
```



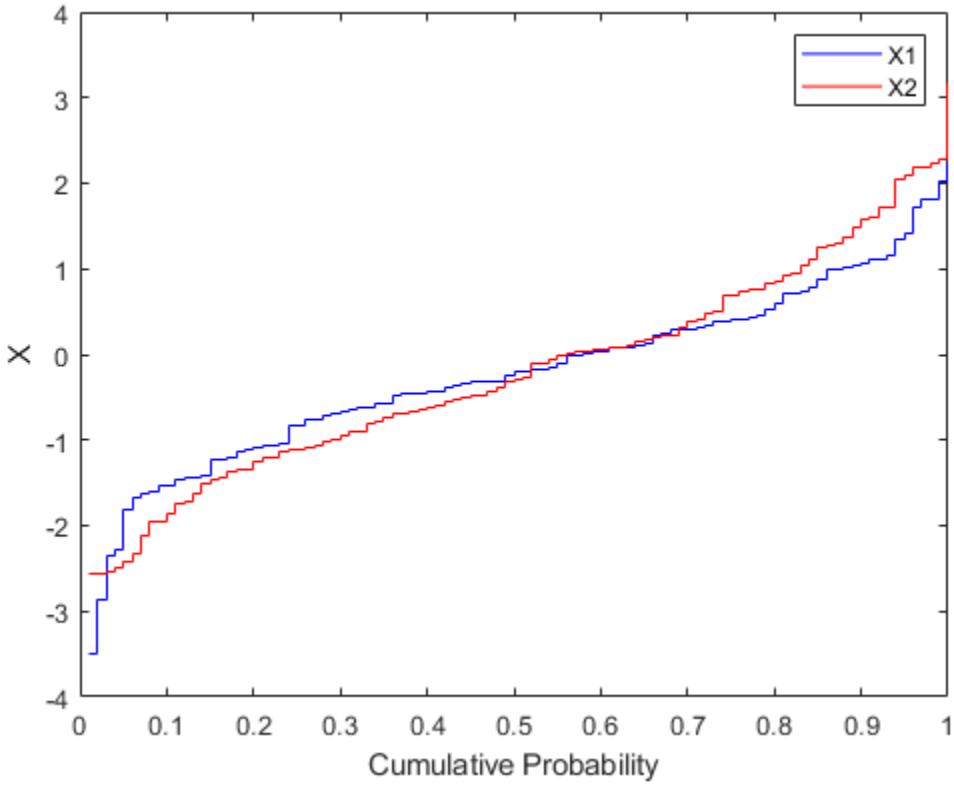
（两个数据向量的长度相同，但这并不重要。）

我们可以分别对每个数据集进行参数化模型拟合，并使用这些估计值作为我们的边缘分布。但是，参数化模型可能不够灵活。所以，我们可以使用经验模型来计算边缘分布。我们只需要一种计算逆 CDF 的方法。

这些数据集的经验逆 CDF 就是一个阶跃函数，阶跃点在 1/nobs、2/nobs 等值处。1.阶跃高度是简单的有序数据。

```
invCDF1 = sort(stocks(:,1));
n1 = length(stocks(:,1));
invCDF2 = sort(stocks(:,2));
```

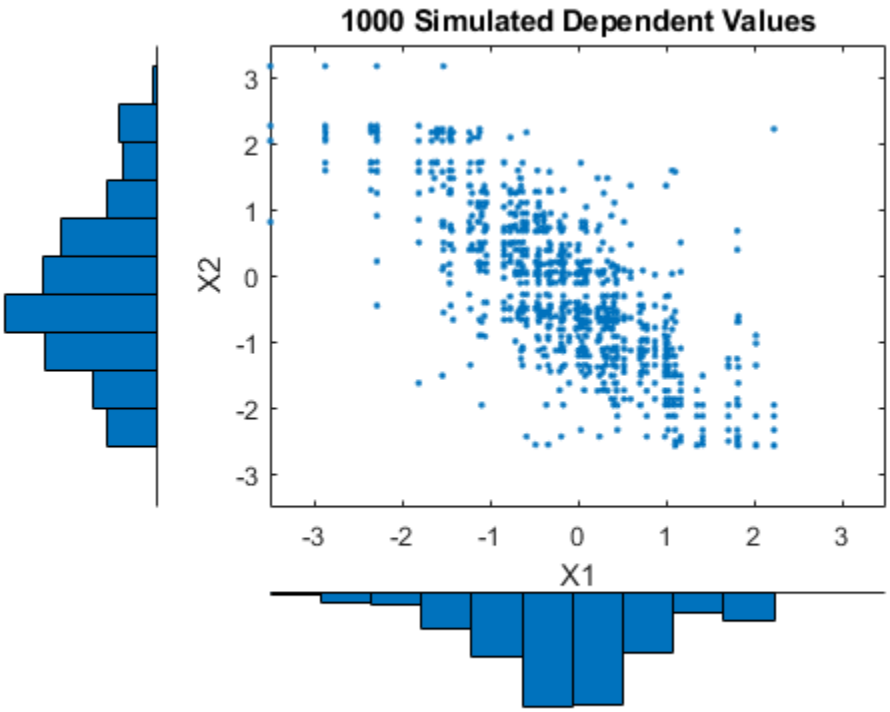
```
n2 = length(stocks(:,2));
subplot(1,1,1);
stairs((1:nobs)/nobs, invCDF1,'b');
hold on;
stairs((1:nobs)/nobs, invCDF2,'r');
hold off;
legend('X1','X2');
xlabel('Cumulative Probability');
ylabel('X');
```



对于仿真，我们可能希望尝试不同的 copula 和相关性。这里，我们将使用一个二元 t(5) copula，并指定一个非常大的负相关参数。

```
n = 1000;
rho = -.8;
nu = 5;
T = mvtrnd([1 rho; rho 1], nu, n);
U = tcdf(T,nu);
X = [invCDF1(ceil(n1*U(:,1))) invCDF2(ceil(n2*U(:,2)))];

[n1,ctr1] = hist(X(:,1),10);
[n2,ctr2] = hist(X(:,2),10);
subplot(2,2,2);
plot(X(:,1),X(:,2),'.');
axis([-3.5 3.5 -3.5 3.5]);
h1 = gca;
title('1000 Simulated Dependent Values');
xlabel('X1');
```

```
ylabel('X2');
subplot(2,2,4);
bar(ctr1,-n1,1);
axis([-3.5 3.5 -max(n1)*1.1 0]);
axis('off');
h2 = gca;
subplot(2,2,1);
barh(ctr2,-n2,1);
axis([-max(n2)*1.1 0 -3.5 3.5]);
axis('off');
h3 = gca;
h1.Position = [0.35 0.35 0.55 0.55];
h2.Position = [.35 .1 .55 .15];
h3.Position = [.1 .35 .15 .55];
colormap([.8 .8 1]);
```



仿真数据的边缘直方图与原始数据的边缘直方图非常接近，而且随着我们仿真的值对组增多，二者会变得完全相同。请注意，这些值是从原始数据中抽取的，由于每个数据集中只有 100 个观测值，因此仿真数据看上去有些“离散”。要解决此问题，可在最后仿真的值中增加少量可能呈正态分布的随机变异。这就相当于使用经过平滑的经验逆 CDF。

利用广义帕累托分布对尾数据建模

此示例说明如何通过最大似然估计对尾数据进行广义帕累托分布拟合。

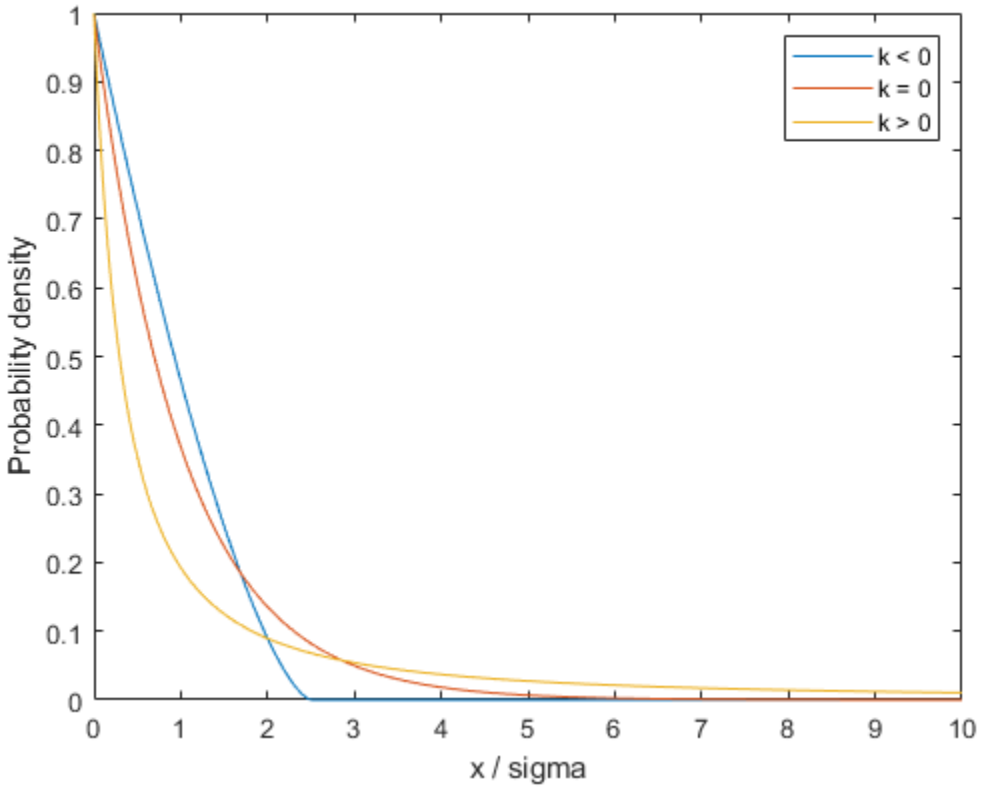
对数据进行参数化分布拟合有时会导致模型与数据在高密度区域拟合很好，但在低密度区域拟合很差。对于单峰分布，例如正态分布或者 Student t 分布，这些低密度区域称为分布的“尾部”。模型可能在尾部拟合不佳的原因之一在于，尾部的数据显然较少，不足以作为选择模型的依据，因此通常会基于模型拟合众数附近数据的能力来选择模型。另一个原因可能在于，实际数据的分布通常要比常见的参数化模型更复杂。

但是，在许多应用中，尾部数据拟合是主要问题所在。广义帕累托分布 (GP) 应运而生，它能够基于理论参数对各种分布的尾部进行建模。一种使用到 GP 的分布拟合方法是在观测值密集的区域使用非参数化拟合（例如经验累积分布函数），而在数据尾部使用 GP 拟合。

广义帕累托分布

广义帕累托 (GP) 分布是一种右偏态分布，使用形状参数 k 和尺度参数 sigma 进行参数化。k 也称为“尾部指数”参数，可以为正值、零或负值。

```
x = linspace(0,10,1000);
plot(x,gppdf(x,-.4,1),'-', x,gppdf(x,0,1),'-', x,gppdf(x,2,1),'-');
xlabel('x / sigma');
ylabel('Probability density');
legend({'k < 0' 'k = 0' 'k > 0'});
```



请注意，当 k < 0 时，GP 高于上限 -(1/k) 的概率为零。当 k >= 0 时，GP 没有上限。此外，GP 还常常与第三个参数即阈值参数结合使用，该阈值参数使下限偏离零。我们这里不需要用到该参数。

GP 分布是指数分布 (k = 0) 和帕累托分布 (k>0) 的广义化。GP 将这两个分布包括在更大的族中，因此可以实现连续的形状范围。

仿真超阈值数据

GP 分布可从超阈值的角度来定义。基于右尾下降到零的概率分布（例如正态分布），我们可以从该分布独立地抽取随机值。如果我们固定一个阈值，排除所有低于该阈值的值，并将未排除的值减去阈值，那么得出的结果就是超出值。超出值的分布近似于 GP。同样，我们可以在分布的左尾设置一个阈值，并忽略高于该阈值的所有值。阈值必须足够远离原始分布的尾部，这样，逼近才会合理。

原始分布决定生成的 GP 分布的形状参数 k。尾部下降为多项式的分布（例如 Student t 分布）得到的形状参数为正值。尾部呈指数级减小的分布（例如正态分布）对应的形状参数为零。具有有限尾部的分布（例如 beta 分布）对应的形状参数为负值。

GP 分布的实际应用包括对股市收益极值进行建模以及对特大洪水进行建模。对于本示例，我们将使用从具有 5 个自由度的 Student t 分布生成的仿真数据。我们将从 t 分布的 2000 个观测值中提取最大的 5% 的值，然后减去 95% 分位数以获得超阈值。

```
rng(3,'twister');
x = trnd(5,2000,1);
q = quantile(x,.95);
y = x(x>q) - q;
n = numel(y)
```

```
n =

   100
```

使用最大似然进行分布拟合

GP 分布定义为 0 < sigma，-Inf < k < Inf。然而，当 k < -1/2 时，最大似然估计结果的解释存在问题。幸运的是，这些情况对应于 beta 或三角等分布的尾部拟合，所以这里不会出现问题。

```
paramEsts = gpfit(y);
kHat      = paramEsts(1)   % Tail index parameter
sigmaHat  = paramEsts(2)   % Scale parameter
```

```
kHat =

    0.0987


sigmaHat =

    0.7156
```

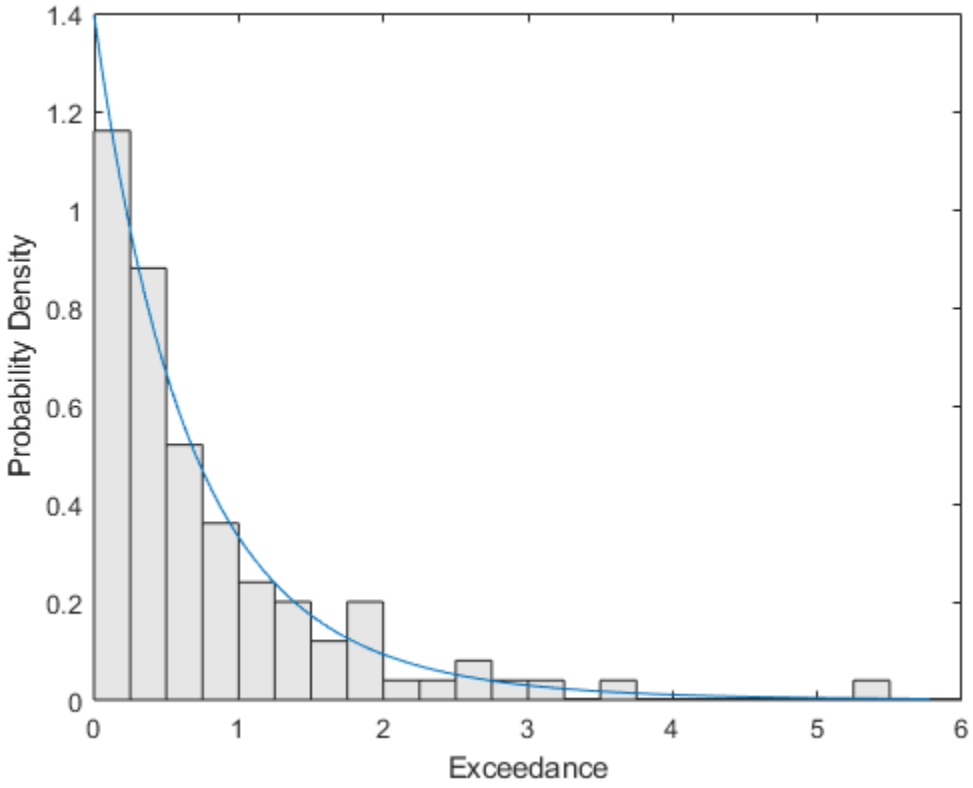
正如所预料的那样，由于仿真数据是使用 t 分布生成的，因此 k 的估计值为正。

通过图形检查拟合效果

为了直观地评估拟合效果，我们将绘制尾部数据的缩放直方图，然后叠加我们估计的 GP 的密度函数。因为是缩放直方图，所以条形高度乘以宽度的总和为 1。

```
bins = 0:.25:7;
h = bar(bins,histc(y,bins)/(length(y)*.25),'histc');
```

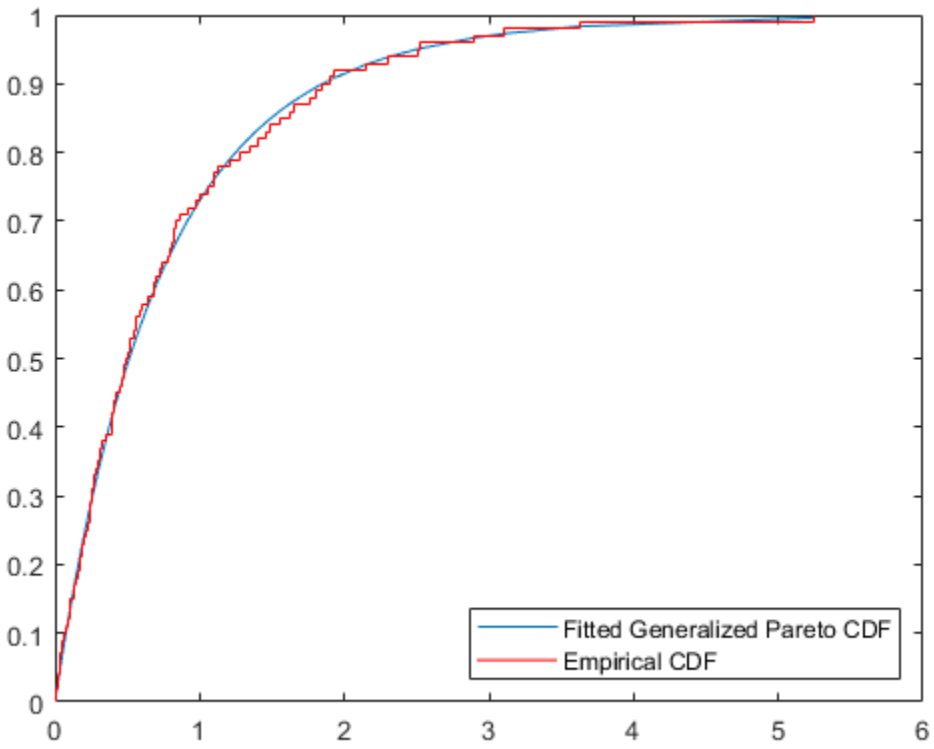
```
h.FaceColor = [.9 .9 .9];
ygrid = linspace(0,1.1*max(y),100);
line(ygrid,gppdf(ygrid,kHat,sigmaHat));
xlim([0,6]);
xlabel('Exceedance');
ylabel('Probability Density');
```



我们使用了比较小的 bin 宽度，因此直方图中存在大量噪声。即便如此，拟合后的密度也符合数据的形状，因此 GP 模型看起来是不错的选择。

我们还可以将经验 CDF 与拟合的 CDF 进行比较。

```
[F,yi] = ecdf(y);
plot(yi,gpcdf(yi,kHat,sigmaHat),'-');
hold on;
stairs(yi,F,'r');
hold off;
legend('Fitted Generalized Pareto CDF','Empirical CDF','location','southeast');
```



计算参数估计值的标准误差

为了量化估计值的精度，我们将使用从最大似然估计量的渐近协方差矩阵计算的标准误差。函数 **gplike** 计算协方差矩阵的数值近似值（作为其第二个输出）。我们也可以使用两个输出实参调用 **gpfit**，它应该返回形参的置信区间。

```
[nll,acov] = gplike(paramEsts, y);
stdErr = sqrt(diag(acov))
```

```
stdErr =

    0.1158
    0.1093
```

这些标准误差表明 k 估计值的相对精度远低于 sigma 的相对精度（sigma 的标准误差与估计值本身相似）。形状参数通常难以估计。必须要记住的是，计算这些标准误差的前提是假设 GP 模型正确，并且我们有足够的数据保证对协方差矩阵的渐近逼近是合理的。

检查渐近正态性假设

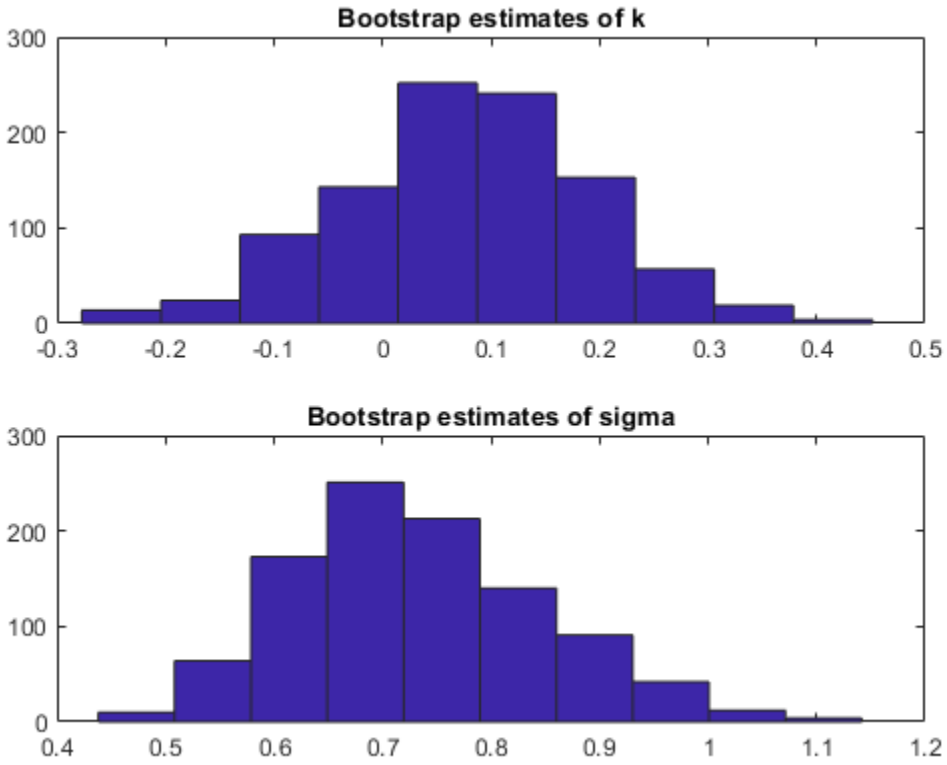
对标准误差的解释通常基于如下假设：如果可以对来自相同数据源的数据多次重复相同的拟合，那么参数的最大似然估计值将大致遵循正态分布。例如，置信区间通常基于这一假设。

然而，正态逼近可能好，也可能不好。为了评估它在此示例中的良好程度，我们可以使用自助仿真。我们通过重抽样从数据中生成 1000 个副本数据集，然后对每个数据集进行 GP 分布拟合，并保存所有副本估计值。

```
replEsts = bootstrp(1000,@gpfit,y);
```

要大致检查参数估计量的抽样分布，我们可以查看自助复制数据的直方图。

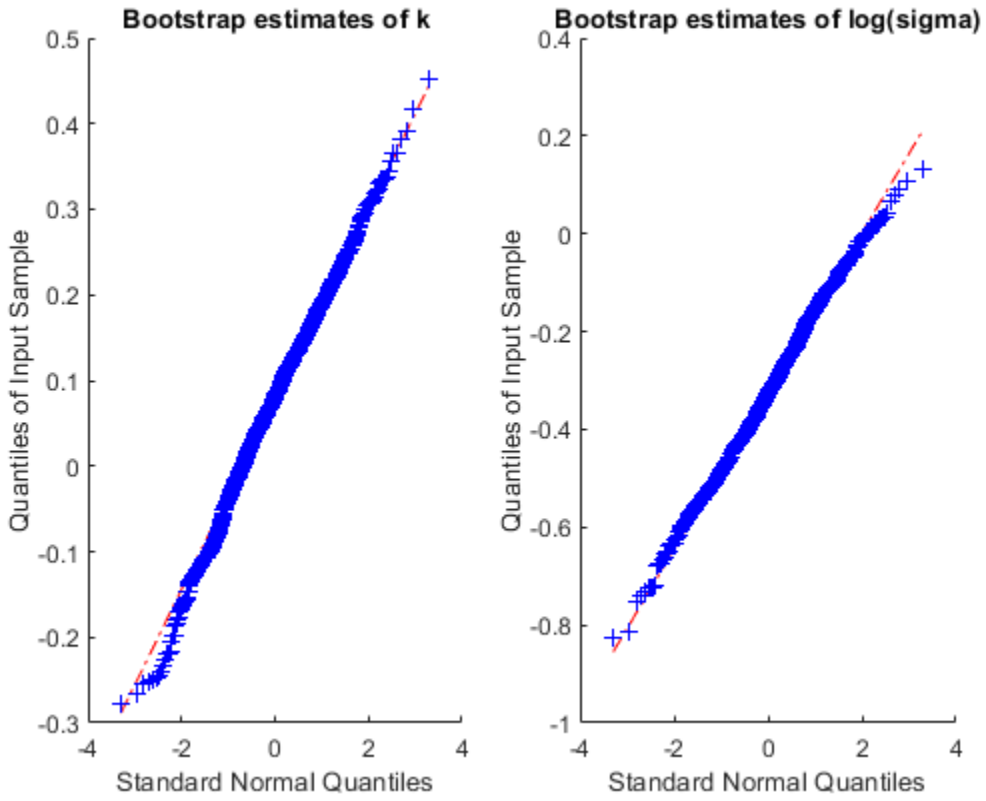
```
subplot(2,1,1);
hist(replEsts(:,1));
title('Bootstrap estimates of k');
subplot(2,1,2);
hist(replEsts(:,2));
title('Bootstrap estimates of sigma');
```



使用参数变换

k 的自助估计值的直方图看起来只有一点不对称，而 sigma 估计值的直方图明显向右偏斜。为了纠正这一偏斜，常见的做法是基于对数尺度来估计参数及其标准误差，因为基于对数尺度的正态逼近可能更为合理。在评估正态性方面，Q-Q 图比直方图更好，因为非正态性的数据会表现为落在一条近似直线之外的点。我们来检验一下，看看对数变换对 sigma 是否合适。

```
subplot(1,2,1);
qqplot(replEsts(:,1));
title('Bootstrap estimates of k');
subplot(1,2,2);
qqplot(log(replEsts(:,2)));
title('Bootstrap estimates of log(sigma)');
```



k 和 log(sigma) 的自助估计值显示为大致接近正态。基于非对数尺度的 sigma 估计值 Q-Q 图将印证我们已经在直方图中看到的偏斜。因此，先在假设正态的情况下计算 log(sigma) 的置信区间，然后取幂，将该置信区间变换回 sigma 的原始尺度，这样构造 sigma 的置信区间更为合理。

事实上，这正是函数 **gpfit** 在幕后所做的工作。

```
[paramEsts,paramCI] = gpfit(y);

kHat
kCI  = paramCI(:,1)
```

```
kHat =

    0.0987


kCI =

   -0.1283
    0.3258
```

```
sigmaHat
sigmaCI  = paramCI(:,2)
```

```
sigmaHat =
```

```
    0.7156

sigmaCI =

    0.5305
    0.9654
```

请注意，虽然 k 的 95% 的置信区间围绕最大似然估计都是对称的，但 sigma 的置信区间并不是这样。那是因为它是通过变换 log(sigma) 的对称 CI 创建的。

曲线拟合和分布拟合

此示例说明如何执行曲线拟合和分布拟合，并讨论每种方法适用的情况。

在曲线拟合与分布拟合之间进行选择

曲线拟合和分布拟合是不同类型的数据分析。

- 当您要将某个响应变量建模为预测变量的函数时，请使用曲线拟合。
- 当您要为单一变量的概率分布建模时，请使用分布拟合。

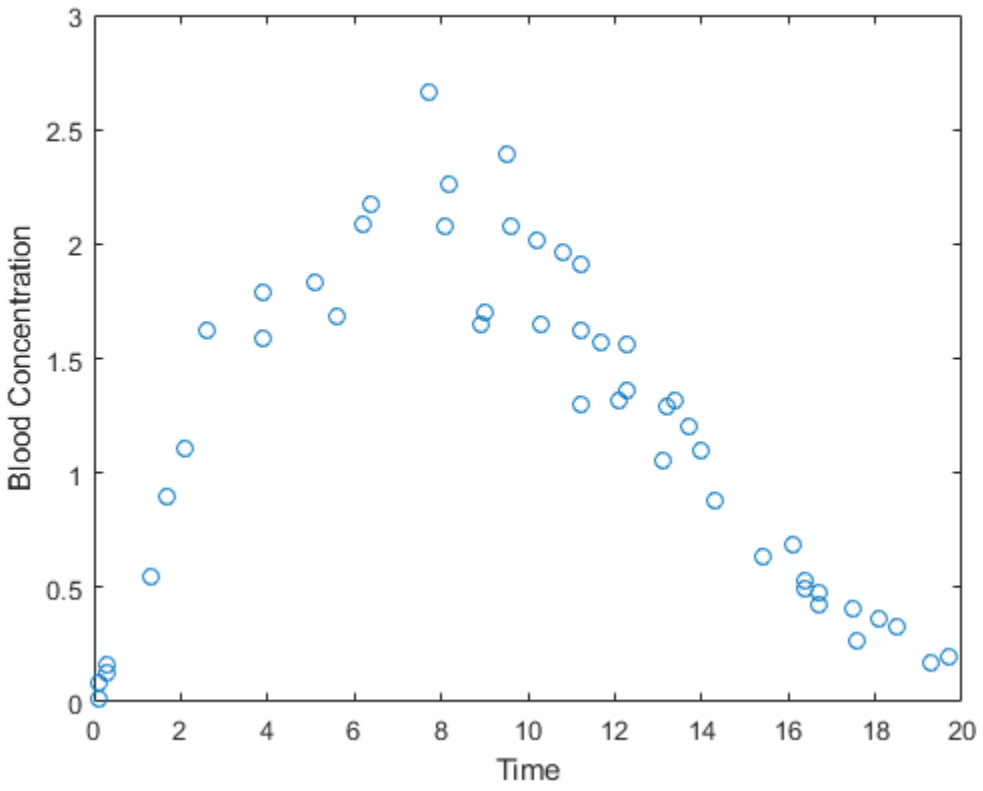
曲线拟合

在以下试验数据中，预测变量为 **time**，即服用药物之后的时间。响应变量为 **conc**，即血液中的药物浓度。假设只有响应数据 **conc** 受试验误差的影响。

```
time = [ 0.1   0.1   0.3   0.3   1.3   1.7   2.1   2.6   3.9   3.9 ...
         5.1   5.6   6.2   6.4   7.7   8.1   8.2   8.9   9.0   9.5 ...
         9.6  10.2  10.3  10.8  11.2  11.2  11.2  11.7  12.1  12.3 ...
        12.3  13.1  13.2  13.4  13.7  14.0  14.3  15.4  16.1  16.1 ...
        16.4  16.4  16.7  16.7  17.5  17.6  18.1  18.5  19.3  19.7]';
conc = [0.01  0.08  0.13  0.16  0.55  0.90  1.11  1.62  1.79  1.59 ...
        1.83  1.68  2.09  2.17  2.66  2.08  2.26  1.65  1.70  2.39 ...
        2.08  2.02  1.65  1.96  1.91  1.30  1.62  1.57  1.32  1.56 ...
        1.36  1.05  1.29  1.32  1.20  1.10  0.88  0.63  0.69  0.69 ...
        0.49  0.53  0.42  0.48  0.41  0.27  0.36  0.33  0.17  0.20]';
```

假设您要将血液浓度建模为时间的函数。绘制 **conc** 对 **time** 的图。

```
plot(time,conc,'o');
xlabel('Time');
ylabel('Blood Concentration');
```



假设 **conc** 作为 **time** 的函数，遵循双参数 Weibull 曲线。Weibull 曲线的形式和参数如下

$$y = c(x/a)^{(b-1)}e^{-(x/a)^b},$$

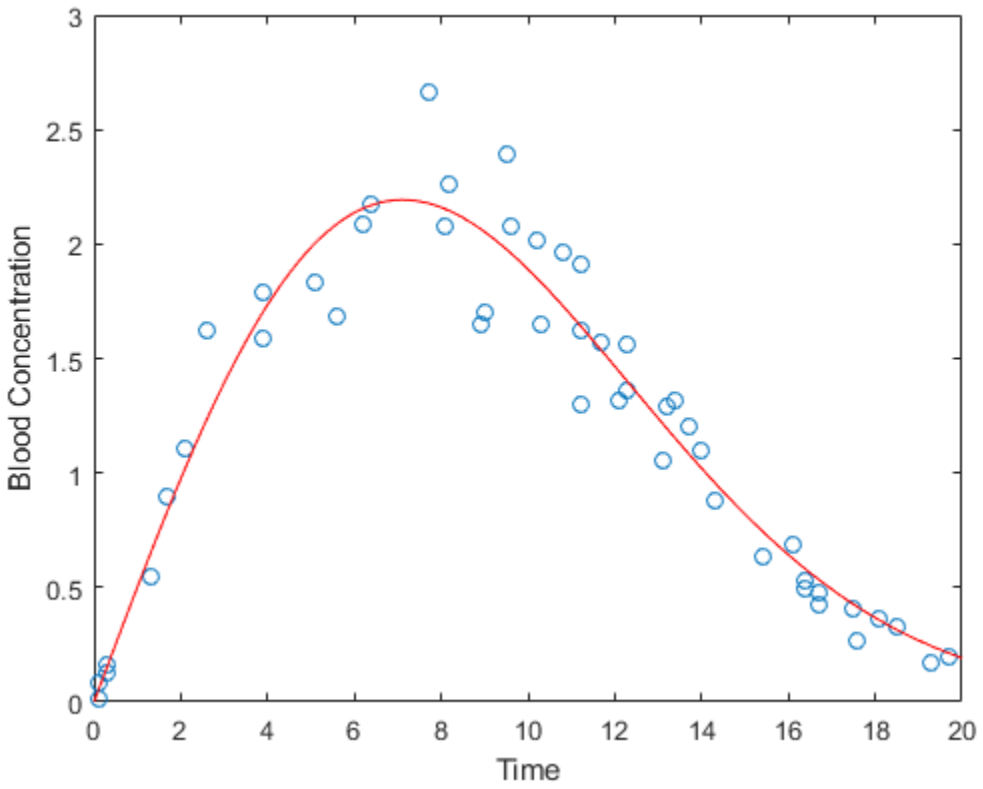
其中，a 是水平缩放，b 是形状参数，c 是垂直缩放。

使用非线性最小二乘来拟合 Weibull 模型。

```
modelFun =  @(p,x) p(3) .* (x./p(1)).^(p(2)-1) .* exp(-(x./p(1)).^p(2));
startingVals = [10 2 5];
nlModel = fitnlm(time,conc,modelFun,startingVals);
```

在数据上绘制 Weibull 曲线。

```
xgrid = linspace(0,20,100)';
line(xgrid,predict(nlModel,xgrid),'Color','r');
```



拟合的 Weibull 模型存在问题。**fitnlm** 假设试验误差为加性误差，并且来自具有常量方差的对称分布。但散点图显示，误差方差与曲线高度成正比。而且，加性对称误差意味着可能出现负的血液浓度测量值。

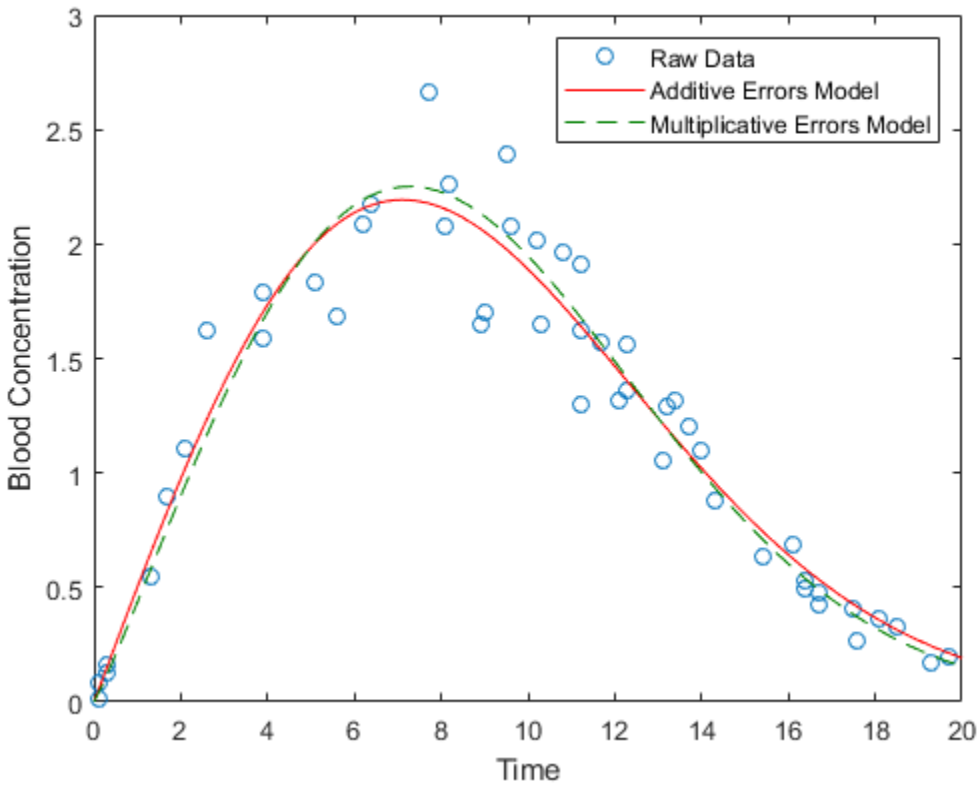
更加现实的假设是，乘性误差在对数尺度上对称。依据该假设，通过取两侧的对数，对数据进行 Weibull 曲线拟合。使用非线性最小二乘来拟合曲线：

$$\log(y) = \log(c) + (b-1)\log(x/a) - (x/a)^b .$$

```
nlModel2 = fitnlm(time,log(conc),@(p,x) log(modelFun(p,x)),startingVals);
```

将新曲线添加到现有绘图。

```
line(xgrid,exp(predict(nlModel2,xgrid)),'Color',[0 .5 0],'LineStyle','--');
legend({'Raw Data','Additive Errors Model','Multiplicative Errors Model'});
```



模型对象 **nlModel2** 包含精度估计。最佳做法是检查模型的拟合优度。例如，在对数尺度上绘制残差图，以检验假设乘性误差具有常量方差是否正确。

在此示例中，使用乘性误差模型对模型预测的影响很小。有关模型类型影响较大的示例，请参阅 “Pitfalls in Fitting Nonlinear Models by Transforming to Linearity”。

曲线拟合函数

- Statistics and Machine Learning Toolbox™ 包含以下用于拟合模型的函数：用于非线性最小二乘模型的 **fitnlm**、用于广义线性模型的 **fitglm**、用于高斯过程回归模型的 **fitrgp** 和用于支持向量机回归模型的 **fitrsvm**。
- Curve Fitting Toolbox™ 提供了命令行和图形工具，可以简化曲线拟合中的任务。例如，该工具箱自动为各种模型选择起始系数值，并提供稳健和非参数拟合方法。
- Optimization Toolbox™ 包含用于执行复杂类型的曲线拟合分析的函数，例如分析具有系数约束的模型。
- MATLAB® 函数 **polyfit** 用于拟合多项式模型，而 MATLAB 函数 **fminsearch** 适用于其他种类的曲线拟合。

分布拟合

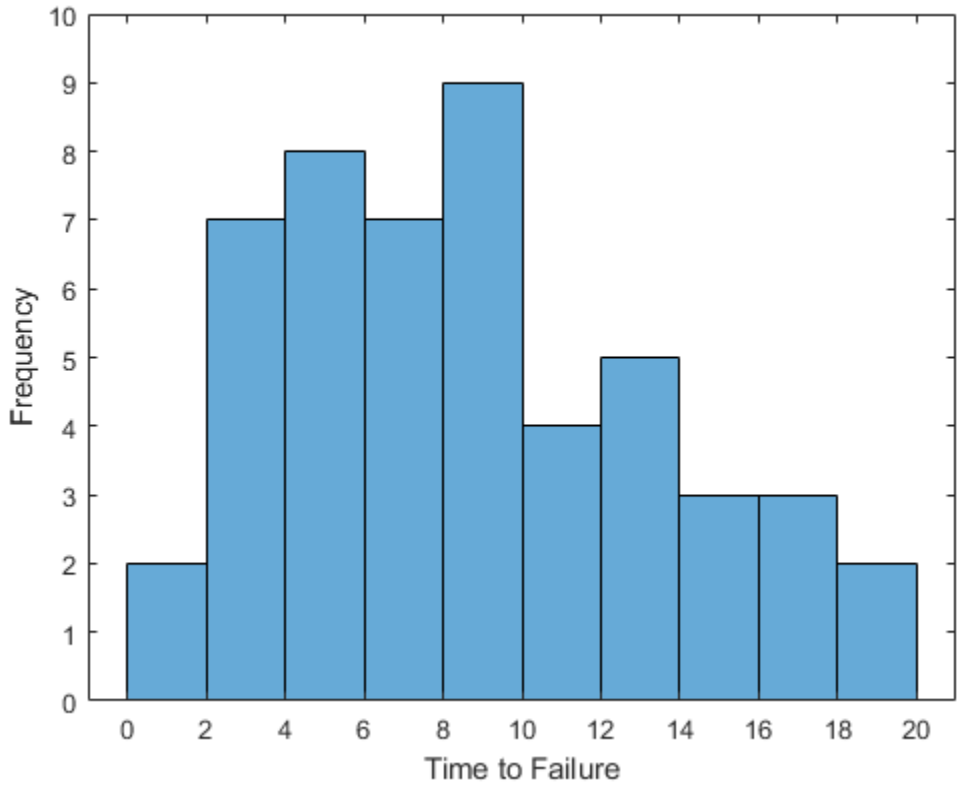
假设您要对电子元件使用寿命的分布进行建模。变量 **life** 用于测量 50 个完全相同的电子元件的失效时间。

```
life = [ 6.2 16.1 16.3 19.0 12.2  8.1  8.8  5.9  7.3  8.2 ...
      16.1 12.8  9.8 11.3  5.1 10.8  6.7  1.2  8.3  2.3 ...
```

```
4.3  2.9 14.8  4.6  3.1 13.6 14.5  5.2  5.7  6.5 ...
5.3  6.4  3.5 11.4  9.3 12.4 18.3 15.9  4.0 10.4 ...
8.7  3.0 12.1  3.9  6.5  3.4  8.5  0.9  9.9  7.9]';
```

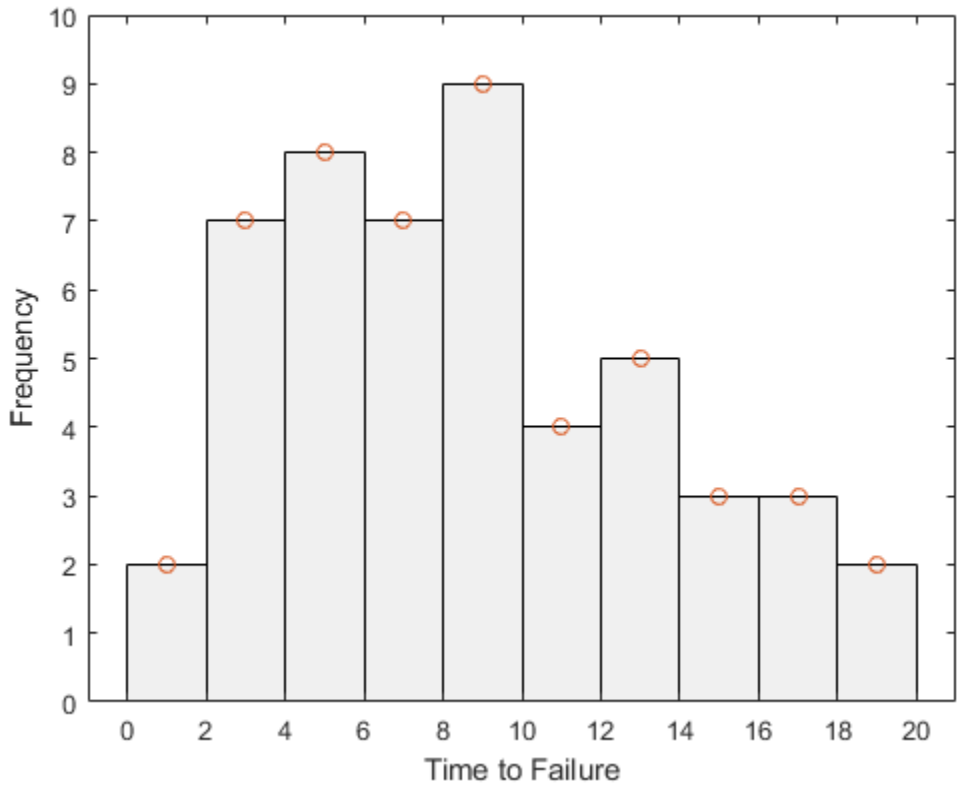
用直方图可视化数据。

```
binWidth = 2;
lastVal = ceil(max(life));
binEdges = 0:binWidth:lastVal+1;
h = histogram(life,binEdges);
xlabel('Time to Failure');
ylabel('Frequency');
ylim([0 10]);
```



由于使用寿命数据通常遵循 Weibull 分布，因此可以使用前一个曲线拟合示例中的 Weibull 曲线来拟合直方图。要尝试此方法，请将直方图转换为一个 (x,y) 点集，其中 x 是 bin 中心，y 是 bin 高度，然后对这些点进行曲线拟合。

```
counts = histcounts(life,binEdges);
binCtrs = binEdges(1:end-1) + binWidth/2;
h.FaceColor = [.9 .9 .9];
hold on
plot(binCtrs,counts,'o');
hold off
```



但是，对直方图进行曲线拟合容易出现问题，通常不建议这样做。

1. 该过程违背了最小二乘拟合的基本假设。bin 计数是非负值，这意味着测量误差不对称。而且，bin 计数在分布的尾部和中心具有不同的变异性。最后，bin 计数的总和是固定的，这意味着它们不是独立的测量值。
2. 如果您对条形高度进行 Weibull 曲线拟合，则必须对曲线进行约束，因为直方图是经过缩放的经验概率密度函数 (pdf)。
3. 对于连续数据，对直方图（而不是数据）进行曲线拟合会丢弃信息。
4. 直方图的条形高度依赖于 bin 边界和 bin 宽度的选择。

对于许多参数化分布，最大似然是更好的参数估计方法，因为它能避免这些问题。Weibull pdf 与 Weibull 曲线具有几乎相同的形式：

$$y = (b/a)(x/a)^{(b-1)}e^{-(x/a)^b}.$$

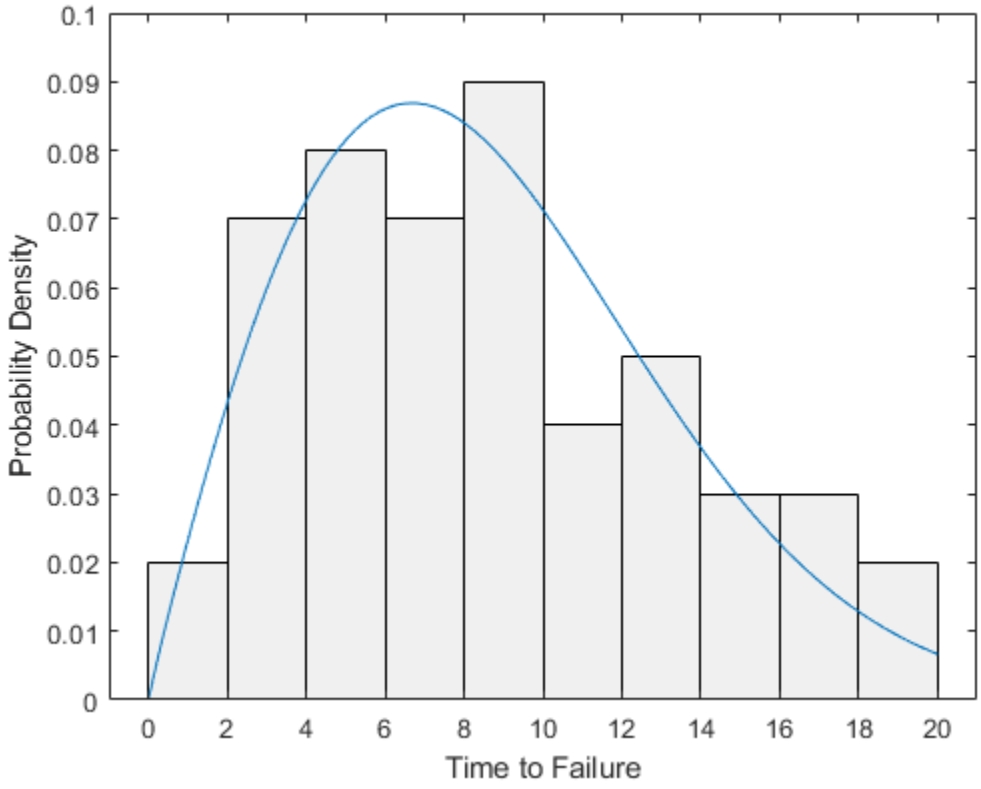
只不过 b/a 取代了尺度参数 c，因为该函数的积分必须为 1。要使用最大似然对数据进行 Weibull 分布拟合，请使用 **fitdist** 并将 **'Weibull'** 指定为分布名称。与最小二乘不同，最大似然会找到与缩放后的直方图最匹配的 Weibull pdf，而无需最小化 pdf 与条形高度之差的平方和。

```
pd = fitdist(life,'Weibull');
```

绘制经过缩放的数据直方图，并叠加绘制拟合的 pdf。

```
h = histogram(life,binEdges,'Normalization','pdf','FaceColor',[.9 .9 .9]);
xlabel('Time to Failure');
```

```
ylabel('Probability Density');
ylim([0 0.1]);
xgrid = linspace(0,20,100)';
pdfEst = pdf(pd,xgrid);
line(xgrid,pdfEst)
```



最佳做法是检查模型的拟合优度。

尽管通常不建议对直方图进行曲线拟合，但在某些情况下这样做是合适的。有关示例，请参阅“Fit Custom Distributions”。

分布拟合函数

- Statistics and Machine Learning Toolbox™ 包含用于对数据进行概率分布对象拟合的函数 **fitdist**。它还包括使用最大似然来拟合参数化分布的专用拟合函数（例如 **wblfit**）、用于拟合自定义分布（此时不使用专用拟合函数）的函数 **mle**，以及用于对数据进行非参数化分布模型拟合的函数 **ksdensity**。
- Statistics and Machine Learning Toolbox 还提供了 Distribution Fitter App，它能简化分布拟合中的许多任务，例如生成可视化和诊断图。
- 利用 Optimization Toolbox™ 中的函数，您可以拟合复杂的分布，包括具有参数约束的分布。
- MATLAB® 函数 **fminsearch** 提供最大似然分布拟合。

另请参阅

fitnlm | **fitglm** | **fitrgp** | **fitrsvm** | **polyfit** | **fminsearch** | **fitdist** | **mle** | **ksdensity** | **分布拟合器**

详细信息

- “Supported Distributions”

6

高斯过程

7

随机数生成

逻辑回归模型的贝叶斯分析

此示例说明如何使用 **slicesample** 对逻辑回归模型进行贝叶斯推断。

统计推断通常基于最大似然估计 (MLE)。MLE 选择能够使数据似然最大化的参数，是一种较为自然的方法。在 MLE 中，假定参数是未知但固定的数值，并在一定的置信度下进行计算。在贝叶斯统计中，使用概率来量化未知参数的不确定性，因而未知参数被视为随机变量。

贝叶斯推断

贝叶斯推断是结合有关模型或模型参数的先验知识来分析统计模型的过程。这种推断的根基是贝叶斯定理：

$$P(\text{parameters}|\text{data}) = \frac{P(\text{data}|\text{parameters}) \times P(\text{parameters})}{P(\text{data})} \propto \text{likelihood} \times \text{prior}$$

例如，假设我们有正态观测值

$$X|\theta \sim N(\theta, \sigma^2)$$

其中 sigma 是已知的，theta 的先验分布为

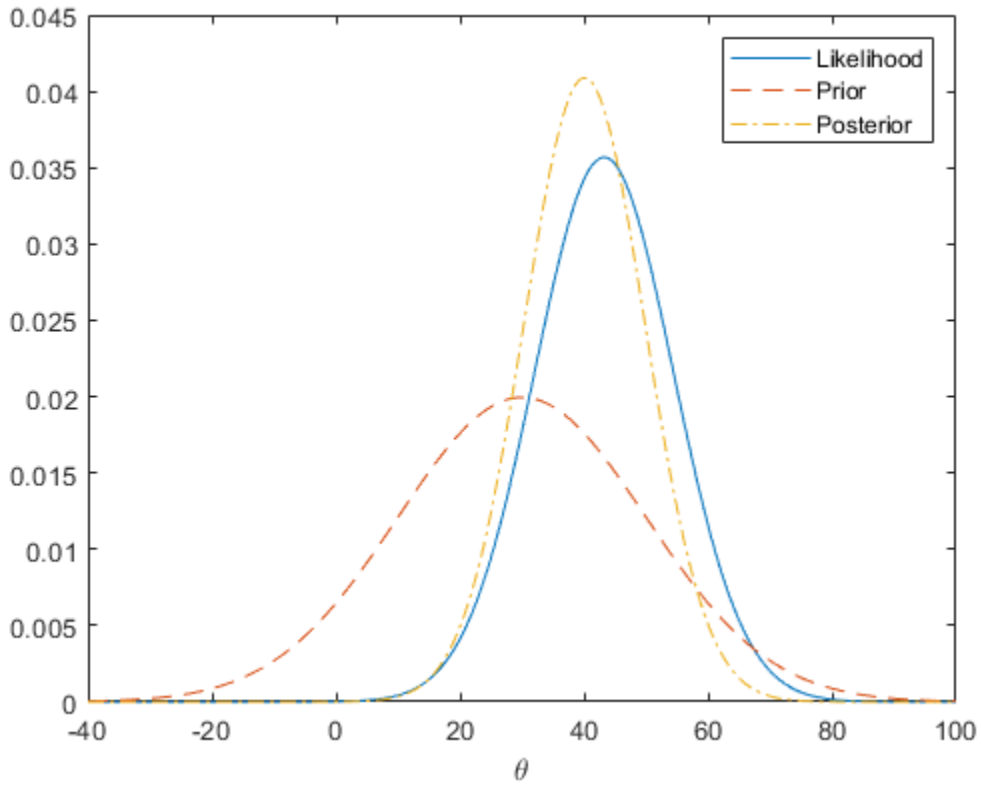
$$\theta \sim N(\mu, \tau^2)$$

在此公式中，mu 和 tau（有时也称为超参数）也是已知的。如果观察 **X** 的 **n** 个样本，我们可以获得 theta 的后验分布

$$\theta|X \sim N\left(\frac{\tau^2}{\sigma^2/n+\tau^2}\bar{X} + \frac{\sigma^2/n}{\sigma^2/n+\tau^2}\mu, \frac{(\sigma^2/n)\times\tau^2}{\sigma^2/n+\tau^2}\right)$$

下图显示 theta 的先验、似然和后验。

```
rng(0,'twister');

n = 20;
sigma = 50;
x = normrnd(10,sigma,n,1);
mu = 30;
tau = 20;
theta = linspace(-40, 100, 500);
y1 = normpdf(mean(x),theta,sigma/sqrt(n));
y2 = normpdf(theta,mu,tau);
postMean = tau^2*mean(x)/(tau^2+sigma^2/n) + sigma^2*mu/n/(tau^2+sigma^2/n);
postSD = sqrt(tau^2*sigma^2/n/(tau^2+sigma^2/n));
y3 = normpdf(theta, postMean,postSD);
plot(theta,y1,'-', theta,y2,'--', theta,y3,'-.')
legend('Likelihood','Prior','Posterior')
xlabel('\theta')
```



汽车试验数据

在一些简单的问题中，例如前面的正态均值推断示例，很容易计算出封闭形式的后验分布。但是，在涉及非共轭先验的一般问题中，后验分布很难或不可能通过分析来进行计算。我们将以逻辑回归作为示例。此示例包含一个试验，以帮助建模不同重量的汽车在里程测试中的未通过比例。数据包括被测汽车的重量、汽车数量以及失败次数等观测值。我们采用一组经过变换的重量，以减少回归参数估值的相关性。

```
% A set of car weights
weight = [2100 2300 2500 2700 2900 3100 3300 3500 3700 3900 4100 4300]';
weight = (weight-2800)/1000;    % recenter and rescale
% The number of cars tested at each weight
total = [48 42 31 34 31 21 23 23 21 16 17 21]';
% The number of cars that have poor mpg performances at each weight
poor = [1 2 0 3 8 8 14 17 19 15 17 21]';
```

逻辑回归模型

逻辑回归（广义线性模型的一种特例）适合这些数据，因为响应变量呈二项分布。逻辑回归模型可以写作：

$$P(\text{failure}) = \frac{e^{Xb}}{1+e^{Xb}}$$

其中 X 是设计矩阵，b 是包含模型参数的向量。在 MATLAB® 中，我们可以将此方程写作：

```
logitp = @(b,x) exp(b(1)+b(2).*x)./(1+exp(b(1)+b(2).*x));
```

如果您有一些先验知识或者已经具备某些非信息性先验，则可以指定模型参数的先验概率分布。例如，在此示例中，我们使用正态先验值表示截距 **b1** 和斜率 **b2**，即

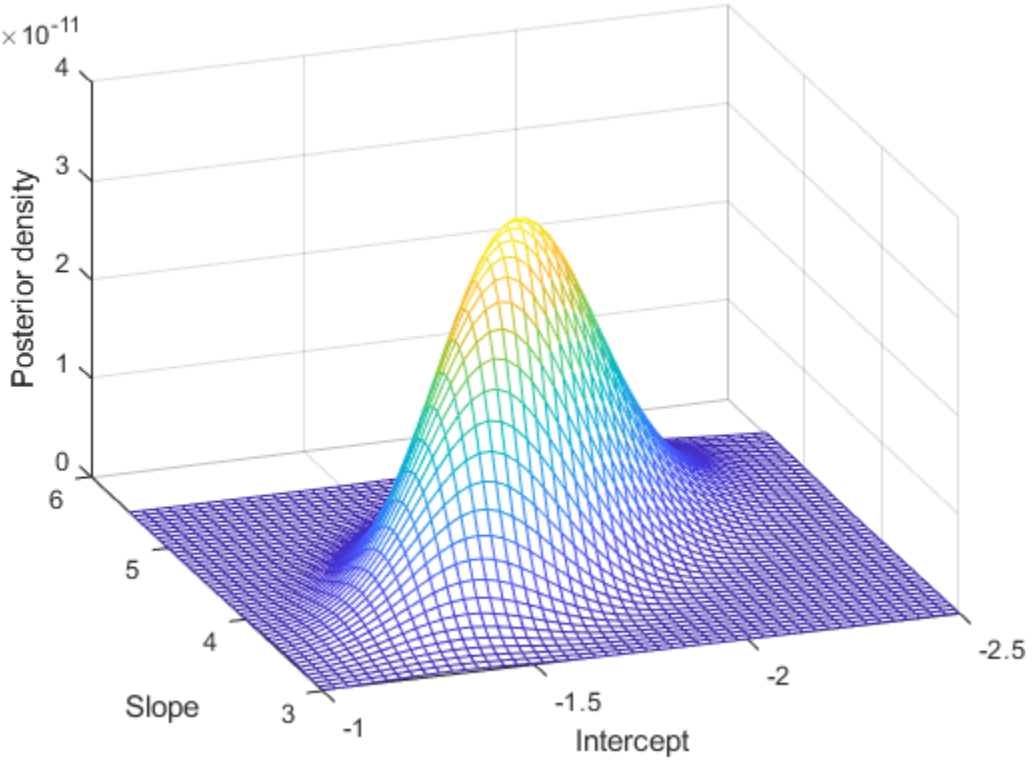
```
prior1 = @(b1) normpdf(b1,0,20);    % prior for intercept
prior2 = @(b2) normpdf(b2,0,20);    % prior for slope
```

根据贝叶斯定理，模型参数的联合后验分布与似然和先验的乘积成正比。

```
post = @(b) prod(binopdf(poor,total,logitp(b,weight))) ...  % likelihood
        * prior1(b(1)) * prior2(b(2));                      % priors
```

请注意，此模型中后验的归一化常数很难进行分析。但是，即使不知道归一化常数，如果您知道模型参数的大致范围，也可以可视化后验分布。

```
b1 = linspace(-2.5, -1, 50);
b2 = linspace(3, 5.5, 50);
simpost = zeros(50,50);
for i = 1:length(b1)
   for j = 1:length(b2)
      simpost(i,j) = post([b1(i), b2(j)]);
   end;
end;
mesh(b2,b1,simpost)
xlabel('Slope')
ylabel('Intercept')
zlabel('Posterior density')
view(-110,30)
```



此后验沿参数空间的对角线伸长，表明（在我们观察数据后）我们认为参数是相关的。这很有意思，因为在我们收集任何数据之前，我们假设它们是独立的。相关性来自我们的先验分布与似然函数的组合。

切片抽样

蒙特卡罗方法常用于在贝叶斯数据分析中汇总后验分布。其想法是，即使您不能通过分析的方式计算后验分布，也可以从分布中生成随机样本，并使用这些随机值来估计后验分布或推断的统计量，如后验均值、中位数、标准差等。切片抽样是一种算法，用于从具有任意密度函数的分布中进行抽样，已知项最多只有一个比例常数 - 而这正是从归一化常数未知的复杂后验分布中抽样所需要的。此算法不生成独立样本，而是生成马尔可夫序列，其平稳分布就是目标分布。因此，切片抽样器是一种马尔可夫链蒙特卡罗 (MCMC) 算法。但是，它与其他众所周知的 MCMC 算法不同，因为只需要指定缩放的后验，不需要建议分布或边缘分布。

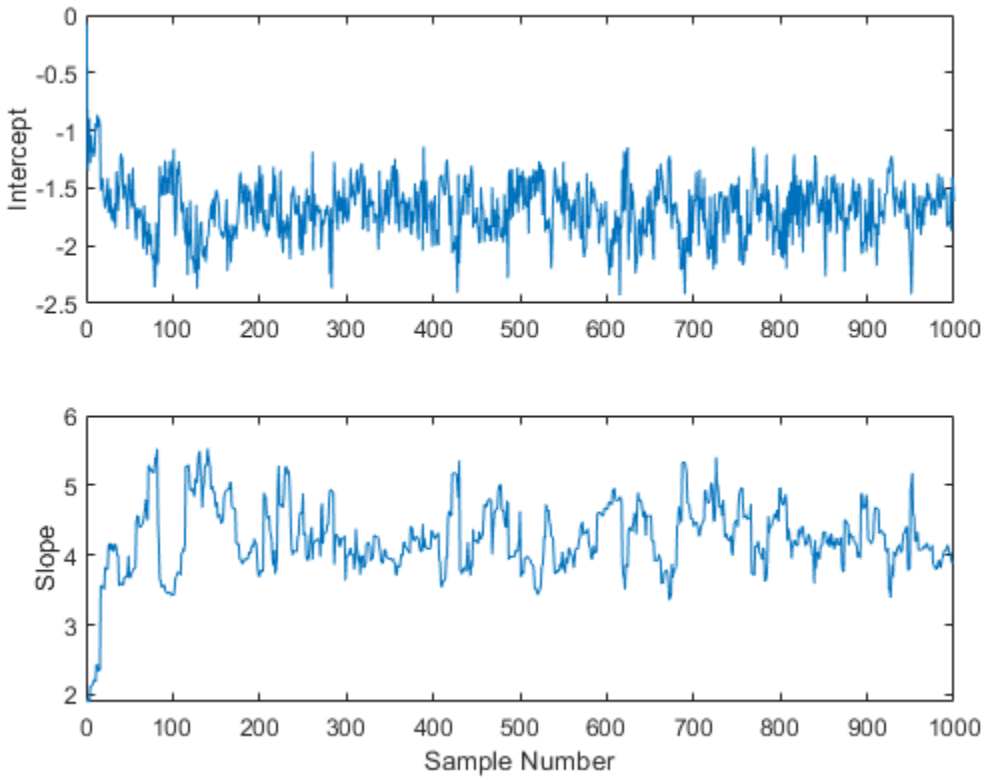
此示例说明如何使用切片抽样器作为里程测试逻辑回归模型的贝叶斯分析的一部分，包括从模型参数的后验分布生成随机样本、分析抽样器的输出，以及对模型参数进行推断。第一步是生成随机样本。

```
initial = [1 1];
nsamples = 1000;
trace = slicesample(initial,nsamples,'pdf',post,'width',[20 2]);
```

抽样器输出分析

从切片抽样器获取随机样本后，很重要的一点是研究诸如收敛和混合之类的问题，以确定将样本视为是来自目标后验分布的一组随机实现是否合理。观察边缘轨迹图是检查输出的最简单方法。

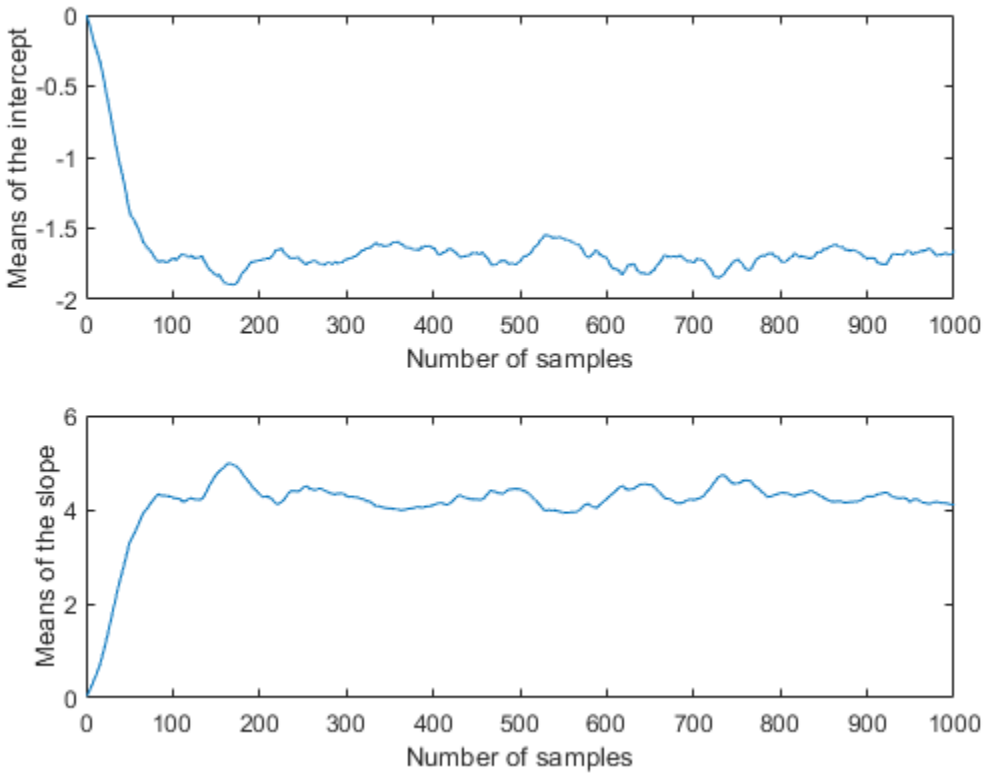
```
subplot(2,1,1)
plot(trace(:,1))
ylabel('Intercept');
subplot(2,1,2)
plot(trace(:,2))
ylabel('Slope');
xlabel('Sample Number');
```



从这些图中可以明显看出，在处理过程趋于平稳之前，参数起始值的影响会维持一段时间（大约 50 个样本）才会消失。

检查收敛以使用移动窗口计算统计量（例如样本的均值、中位数或标准差）也很有帮助。这样可以产生比原始样本轨迹更平滑的图，并且更容易识别和理解任何非平稳性。

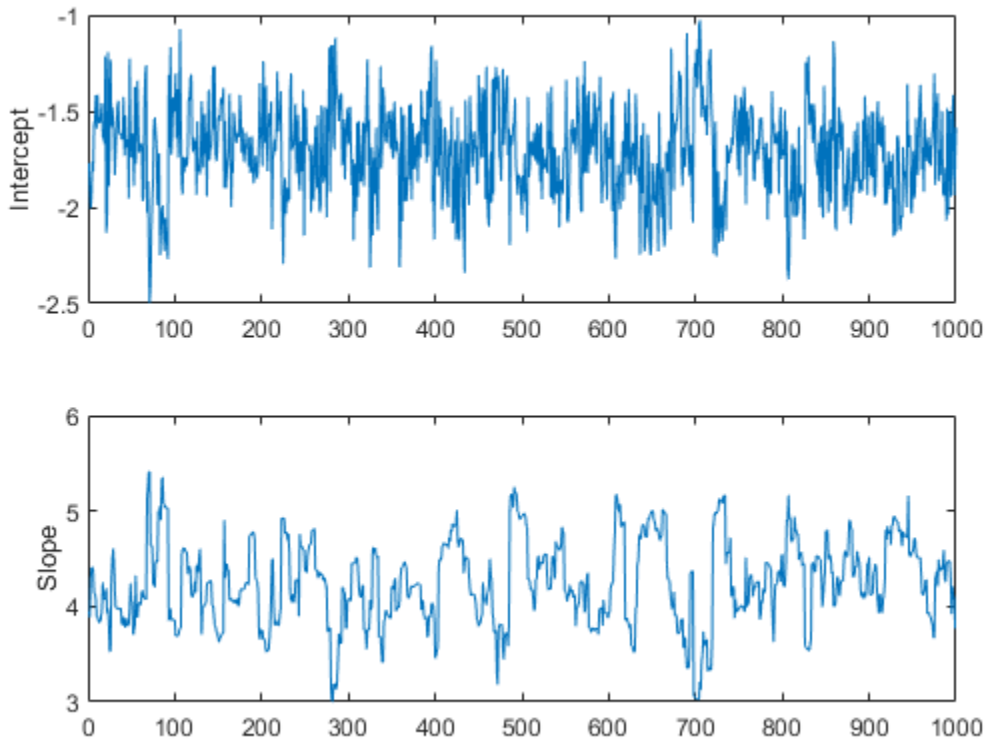
```
movavg = filter( (1/50)*ones(50,1), 1, trace);
subplot(2,1,1)
plot(movavg(:,1))
xlabel('Number of samples')
ylabel('Means of the intercept');
subplot(2,1,2)
plot(movavg(:,2))
xlabel('Number of samples')
ylabel('Means of the slope');
```



由于这些是基于包含 50 次迭代的窗口计算的移动平均值，因此前 50 个值无法与图中的其他值进行比较。然而，每个图的其他值似乎证实参数后验均值在 100 次左右迭代后收敛至平稳分布。同样显而易见的是，这两个参数彼此相关，与之前的后验密度图一致。

由于磨合期代表目标分布中不能合理视为随机实现的样本，因此不建议使用切片抽样器一开始输出的前 50 个左右的值。您可以简单地删除这些输出行，但也可以指定一个“预热”期。在已知合适的预热长度（可能来自先前的运行）时，这种方式很简便。

```
trace = slicesample(initial,nsamples,'pdf',post, ...
                    'width',[20 2],'burnin',50);
subplot(2,1,1)
plot(trace(:,1))
ylabel('Intercept');
subplot(2,1,2)
plot(trace(:,2))
ylabel('Slope');
```



这些跟踪图没有显示出任何不平稳，表明预热期已完成。

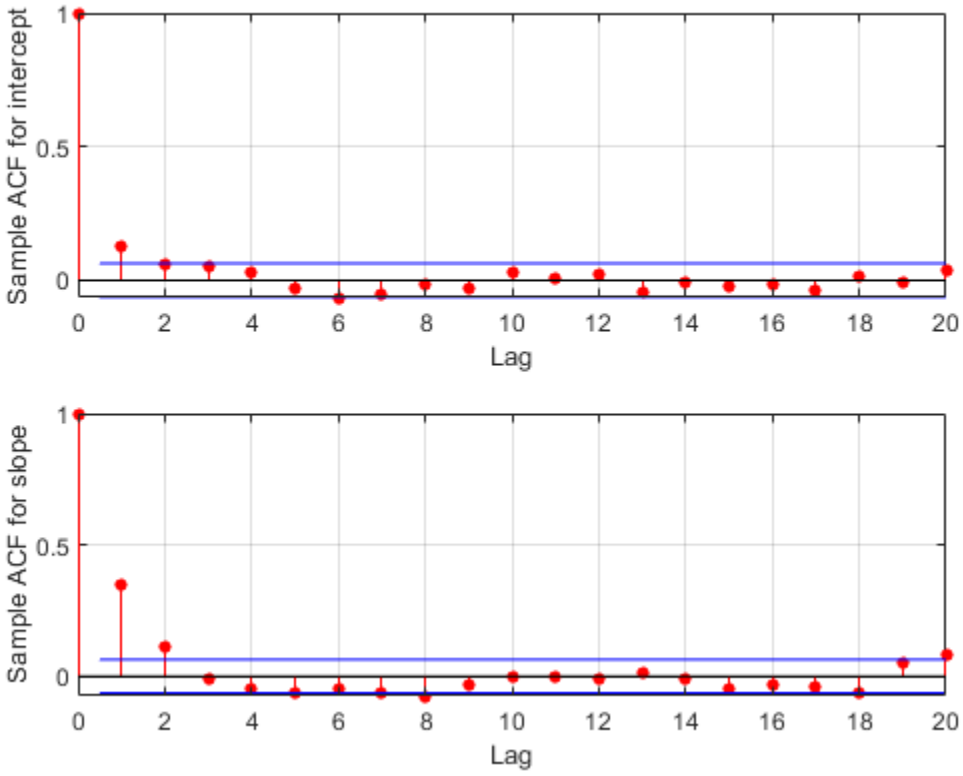
但是，还需要了解跟踪图的另一方面。虽然截距的轨迹看起来像高频噪声，但斜率的轨迹好像具有低频分量，表明相邻迭代的值之间存在自相关。虽然也可以从这个自相关样本计算均值，但我们通常会通过删除样本中的冗余数据这一简便的操作来降低存储要求。如果它同时消除了自相关，我们还可以将这些数据视为独立值样本。例如，您可以通过只保留第 10 个、第 20 个、第 30 个等值来稀释样本。

```
trace = slicesample(initial,nsamples,'pdf',post,'width',[20 2], ...
                                    'burnin',50,'thin',10);
```

要检查这种稀释的效果，可以根据轨迹估计样本自相关函数，并使用它们来检查样本是否快速混合。

```
F   = fft(detrend(trace,'constant'));
F   = F .* conj(F);
ACF = ifft(F);
ACF = ACF(1:21,:);                  % Retain lags up to 20.
ACF = real([ACF(1:21,1) ./ ACF(1,1) ...
        ACF(1:21,2) ./ ACF(1,2)]);      % Normalize.
bounds = sqrt(1/nsamples) * [2 ; -2];      % 95% CI for iid normal

labs = {'Sample ACF for intercept','Sample ACF for slope' };
for i = 1:2
   subplot(2,1,i)
   lineHandles = stem(0:20, ACF(:,i) , 'filled' , 'r-o');
   lineHandles.MarkerSize = 4;
   grid('on')
   xlabel('Lag')
```

```
    ylabel(labs{i})
    hold on
    plot([0.5 0.5 ; 20 20] , [bounds([1 1]) bounds([2 2])] , '-b');
    plot([0 20] , [0 0] , '-k');
    hold off
    a = axis;
    axis([a(1:3) 1]);
end
```

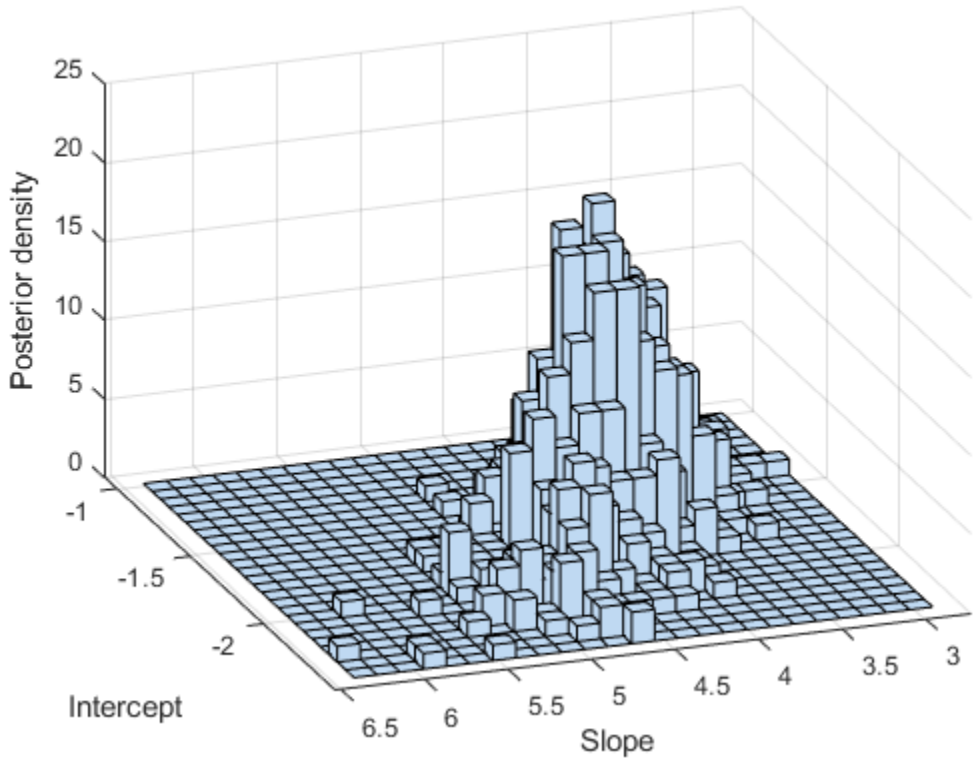


第一个滞后的自相关值对于截距参数很明显，对于斜率参数更是如此。我们可以使用更大的稀释参数重复抽样，以进一步降低相关性。但为了完成本示例的目的，我们将继续使用当前样本。

推断模型参数

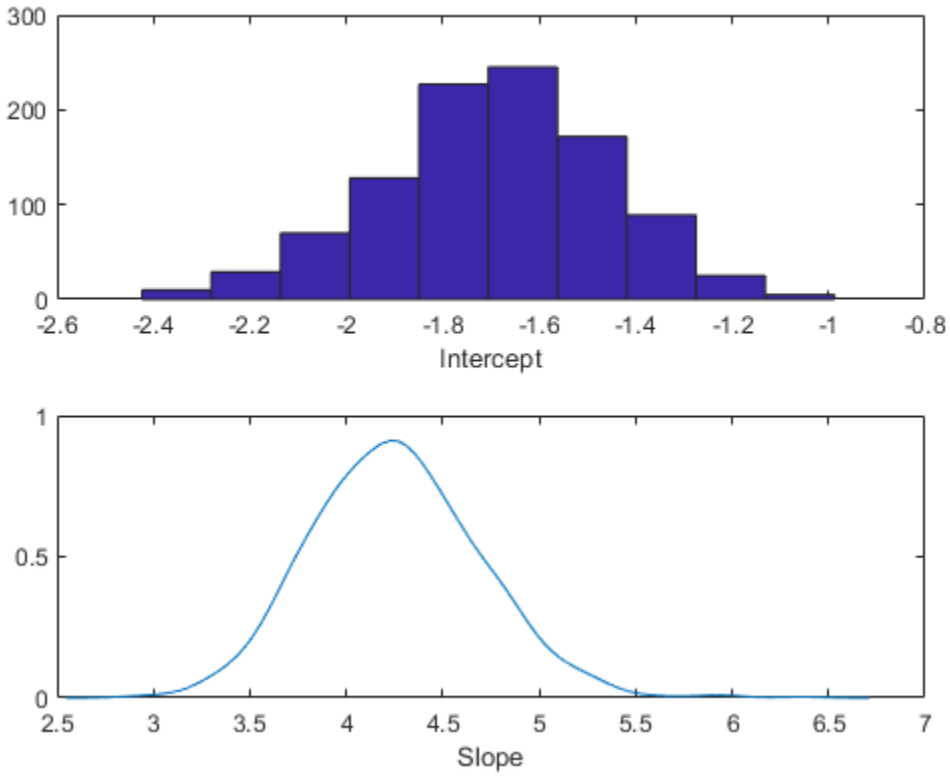
与预期相符，样本直方图模拟了后验密度图。

```
subplot(1,1,1)
hist3(trace,[25,25]);
xlabel('Intercept')
ylabel('Slope')
zlabel('Posterior density')
view(-110,30)
```



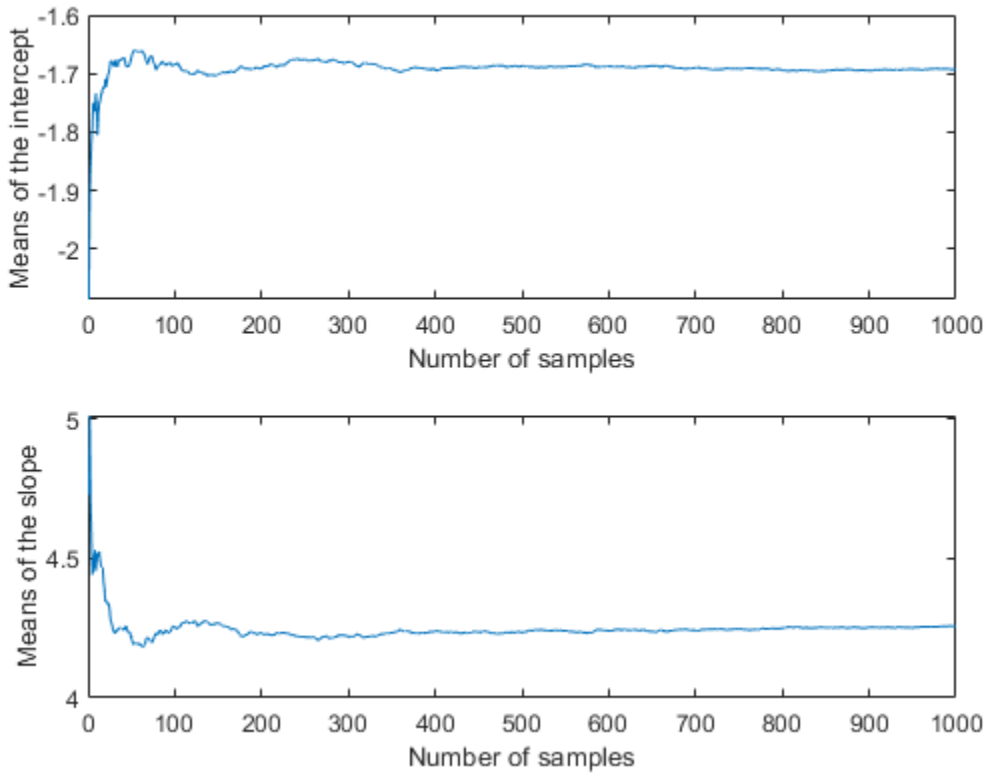
您可以使用直方图或核平滑密度估计值来总结后验样本的边缘分布属性。

```
subplot(2,1,1)
hist(trace(:,1))
xlabel('Intercept');
subplot(2,1,2)
ksdensity(trace(:,2))
xlabel('Slope');
```



您还可以计算描述性统计量，例如随机样本的后验均值或百分位数。为了确定样本大小是否足以实现所需的精度，将所需的轨迹统计量作为样本数的函数来进行监视会很有帮助。

```
csum = cumsum(trace);
subplot(2,1,1)
plot(csum(:,1)'./(1:nsamples))
xlabel('Number of samples')
ylabel('Means of the intercept');
subplot(2,1,2)
plot(csum(:,2)'./(1:nsamples))
xlabel('Number of samples')
ylabel('Means of the slope');
```



在这种情况下，样本大小 1000 似乎足以为后验均值估计值提供良好的精度。

```
bHat = mean(trace)
```

```
bHat =

   -1.6931    4.2569
```

总结

Statistics and Machine Learning Toolbox™ 提供了多种函数，让您能够轻松地指定似然和先验。您也可以将它们结合起来用于推断后验分布。使用 **slicesample** 函数，您可以通过马尔可夫链蒙特卡罗模拟在 MATLAB 中执行贝叶斯分析。甚至在使用标准随机数生成函数难以抽样的后验分布问题中也可以使用此函数。

8

假设检验

9

方差分析

贝叶斯优化

使用贝叶斯优化来优化 SVM 分类器拟合

此示例说明如何使用 **fitcsvm** 函数和 **OptimizeHyperparameters** 名称-值对组优化 SVM 分类。该分类基于高斯混合模型中点的位置来工作。有关该模型的描述，请参阅 The Elements of Statistical Learning，作者 Hastie、Tibshirani 和 Friedman (2009)，第 17 页。该模型从为“green”类生成 10 个基点开始，这些基点呈二维独立正态分布，均值为 (1,0) 且具有单位方差。它还为“red”类生成 10 个基点，这些基点呈二维独立正态分布，均值为 (0,1) 且具有单位方差。对于每个类（green 和 red），生成 100 个随机点，如下所示：

1. 随机均匀选择合适颜色的一个基点 m。
2. 生成一个呈二维正态分布的独立随机点，其均值为 m，方差为 I/5，其中 I 是 2×2 单位矩阵。在此示例中，使用方差 I/50 来更清楚地显示优化的优势。

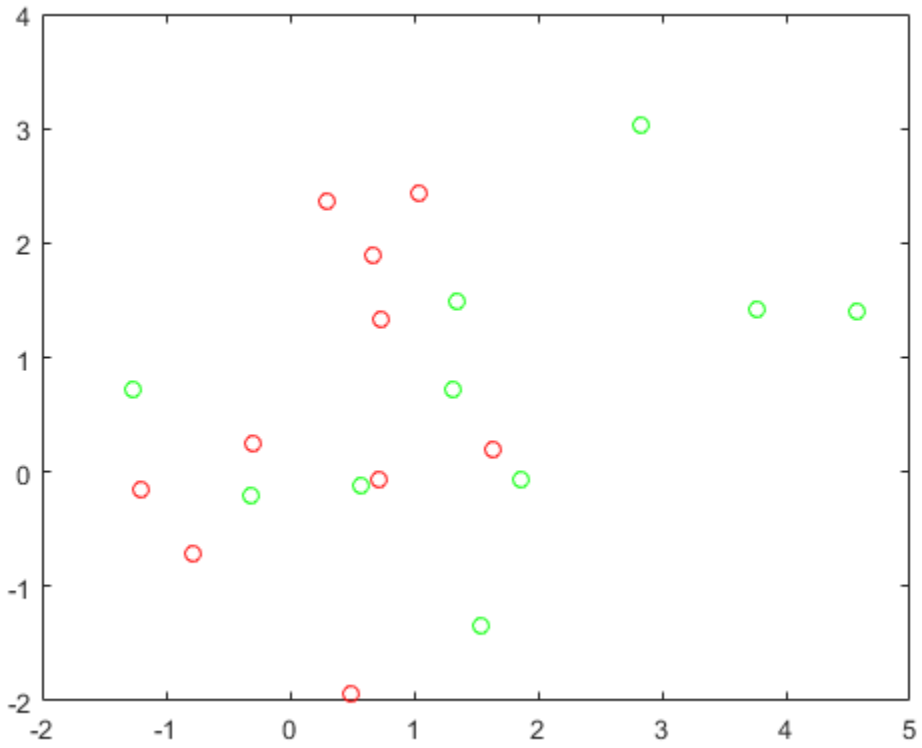
生成点和分类器

为每个类生成 10 个基点。

```
rng default % For reproducibility
grnpop = mvnrnd([1,0],eye(2),10);
redpop = mvnrnd([0,1],eye(2),10);
```

查看基点。

```
plot(grnpop(:,1),grnpop(:,2),'go')
hold on
plot(redpop(:,1),redpop(:,2),'ro')
hold off
```



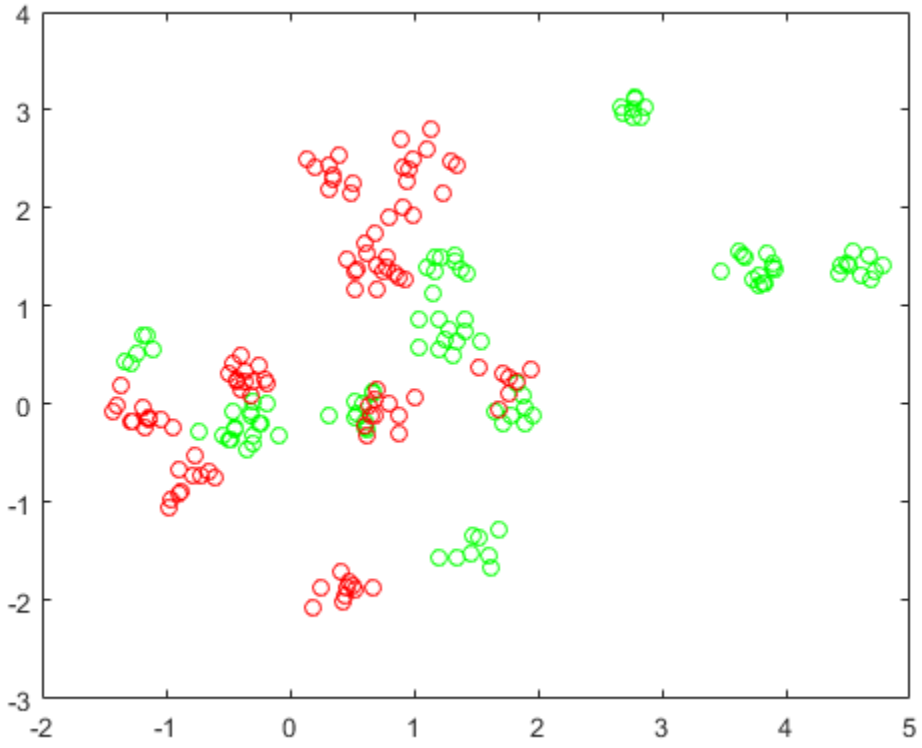
由于一些红色基点靠近绿色基点，因此很难仅基于位置对数据点进行分类。

生成每个类的 100 个数据点。

```
redpts = zeros(100,2);grnpts = redpts;
for i = 1:100
   grnpts(i,:) = mvnrnd(grnpop(randi(10),:),eye(2)*0.02);
   redpts(i,:) = mvnrnd(redpop(randi(10),:),eye(2)*0.02);
end
```

查看数据点。

```
figure
plot(grnpts(:,1),grnpts(:,2),'go')
hold on
plot(redpts(:,1),redpts(:,2),'ro')
hold off
```



为分类准备数据

将数据放入一个矩阵中，并创建向量 **grp**，该向量标记每个点的类。

```
cdata = [grnpts;redpts];
grp = ones(200,1);
% Green label 1, red label -1
grp(101:200) = -1;
```

准备交叉验证

为交叉验证设置一个分区。此步骤确定优化在每一步所使用的训练集和测试集。

```
c = cvpartition(200,'KFold',10);
```

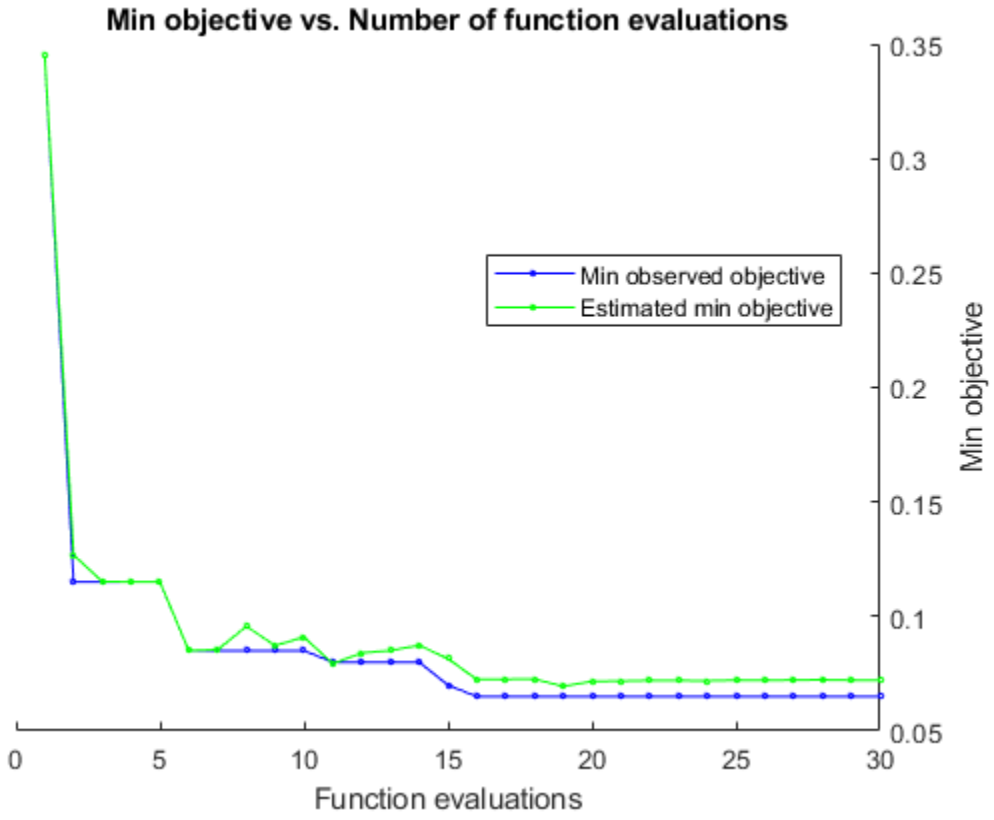
优化拟合

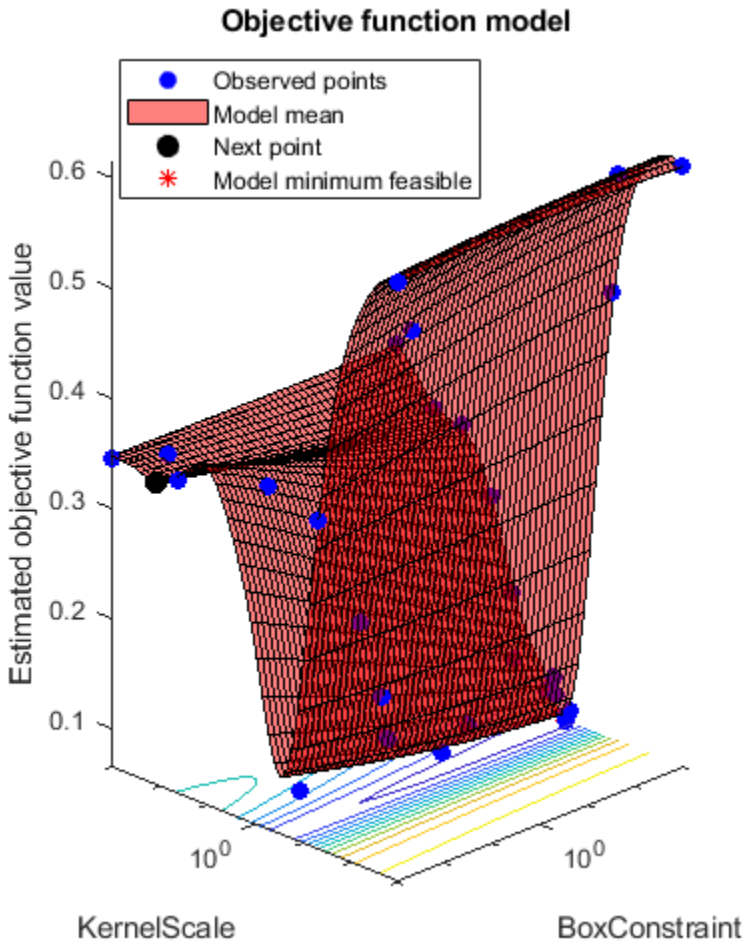
要找到好的拟合，即交叉验证损失低的拟合，请设置选项以使用贝叶斯优化。在所有优化中使用相同的交叉验证分区 `c`。

为了实现可再现性，请使用 `'expected-improvement-plus'` 采集函数。

```
opts = struct('Optimizer','bayesopt','ShowPlots',true,'CVPartition',c,...
   'AcquisitionFunctionName','expected-improvement-plus');
svmmod = fitcsvm(cdata,grp,'KernelFunction','rbf',...
   'OptimizeHyperparameters','auto','HyperparameterOptimizationOptions',opts)
```

```
|=====================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | BoxConstraint|  KernelScale |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|=====================================================================================================|
|    1 | Best   |       0.345 |     0.21295 |       0.345 |       0.345 |      0.00474 |       306.44 |
|    2 | Best   |       0.115 |     0.17149 |       0.115 |     0.12678 |       430.31 |       1.4864 |
|    3 | Accept |        0.52 |      0.1333 |       0.115 |      0.1152 |     0.028415 |     0.014369 |
|    4 | Accept |        0.61 |      0.2691 |       0.115 |     0.11504 |       133.94 |    0.0031427 |
|    5 | Accept |        0.34 |     0.35421 |       0.115 |     0.11504 |     0.010993 |       5.7742 |
|    6 | Best   |       0.085 |     0.11744 |       0.085 |    0.085039 |       885.63 |      0.68403 |
|    7 | Accept |       0.105 |     0.11036 |       0.085 |    0.085428 |       0.3057 |      0.58118 |
|    8 | Accept |        0.21 |     0.13858 |       0.085 |     0.09566 |      0.16044 |      0.91824 |
|    9 | Accept |       0.085 |       0.178 |       0.085 |     0.08725 |       972.19 |      0.46259 |
|   10 | Accept |         0.1 |      0.3127 |       0.085 |    0.090952 |       990.29 |        0.491 |
|   11 | Best   |        0.08 |     0.12274 |        0.08 |    0.079362 |       2.5195 |        0.291 |
|   12 | Accept |        0.09 |     0.11783 |        0.08 |     0.08402 |       14.338 |      0.44386 |
|   13 | Accept |         0.1 |     0.10892 |        0.08 |     0.08508 |    0.0022577 |      0.23803 |
|   14 | Accept |        0.11 |     0.10798 |        0.08 |    0.087378 |       0.2115 |      0.32109 |
|   15 | Best   |        0.07 |     0.17553 |        0.07 |    0.081507 |        910.2 |      0.25218 |
|   16 | Best   |       0.065 |     0.17637 |       0.065 |    0.072457 |       953.22 |      0.26253 |
|   17 | Accept |       0.075 |     0.26422 |       0.065 |    0.072554 |       998.74 |      0.23087 |
|   18 | Accept |       0.295 |     0.11691 |       0.065 |    0.072647 |       996.18 |       44.626 |
|   19 | Accept |        0.07 |     0.13689 |       0.065 |     0.06946 |       985.37 |      0.27389 |
|   20 | Accept |       0.165 |     0.11689 |       0.065 |    0.071622 |     0.065103 |      0.13679 |
|=====================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | BoxConstraint|  KernelScale |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|=====================================================================================================|
|   21 | Accept |       0.345 |     0.10901 |       0.065 |    0.071764 |        971.7 |       999.01 |
|   22 | Accept |        0.61 |     0.12691 |       0.065 |    0.071967 |    0.0010168 |    0.0010005 |
|   23 | Accept |       0.345 |     0.10781 |       0.065 |    0.071959 |    0.0010674 |       999.18 |
|   24 | Accept |        0.35 |     0.11365 |       0.065 |    0.071863 |    0.0010003 |       40.628 |
|   25 | Accept |        0.24 |      0.2432 |       0.065 |    0.072124 |       996.55 |       10.423 |
|   26 | Accept |        0.61 |      0.1947 |       0.065 |    0.072068 |       958.64 |    0.0010026 |
|   27 | Accept |        0.47 |      0.1147 |       0.065 |     0.07218 |       993.69 |     0.029723 |
|   28 | Accept |         0.3 |     0.10471 |       0.065 |    0.072291 |       993.15 |       170.01 |
|   29 | Accept |        0.16 |      0.2536 |       0.065 |    0.072104 |       992.81 |       3.8594 |
|   30 | Accept |       0.365 |     0.10891 |       0.065 |    0.072112 |    0.0010017 |     0.044287 |
```


Min objective vs. Number of function evaluations
Min observed objective
Estimated min objective
Min objective
Function evaluations
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
5
10
15
20
25
30



Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 50.7144 seconds
Total objective function evaluation time: 4.9196

Best observed feasible point:

BoxConstraint	KernelScale
953.22	0.26253

Observed objective function value = 0.065
Estimated objective function value = 0.073726
Function evaluation time = 0.17637

Best estimated feasible point (according to models):

BoxConstraint	KernelScale

```
    985.37        0.27389

Estimated objective function value = 0.072112
Estimated function evaluation time = 0.1773

svmmod =
  ClassificationSVM
                 ResponseName: 'Y'
        CategoricalPredictors: []
                  ClassNames: [-1 1]
              ScoreTransform: 'none'
             NumObservations: 200
  HyperparameterOptimizationResults: [1x1 BayesianOptimization]
                       Alpha: [77x1 double]
                        Bias: -0.2352
            KernelParameters: [1x1 struct]
             BoxConstraints: [200x1 double]
            ConvergenceInfo: [1x1 struct]
            IsSupportVector: [200x1 logical]
                    Solver: 'SMO'


  Properties, Methods
```

计算优化模型的损失。

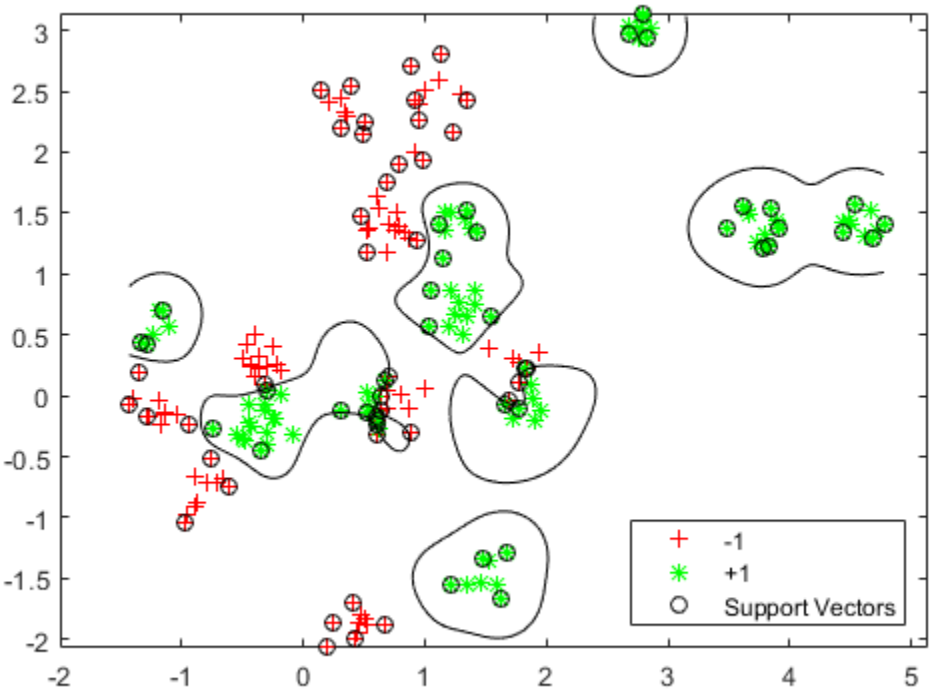
```
lossnew = kfoldLoss(fitcsvm(cdata,grp,'CVPartition',c,'KernelFunction','rbf',...
   'BoxConstraint',svmmod.HyperparameterOptimizationResults.XAtMinObjective.BoxConstraint,...
   'KernelScale',svmmod.HyperparameterOptimizationResults.XAtMinObjective.KernelScale))
```

```
lossnew = 0.0650
```

该损失与“观测的目标函数值”下优化输出中报告的损失相同。

可视化经过优化的分类器。

```
d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(cdata(:,1)):d:max(cdata(:,1)),...
   min(cdata(:,2)):d:max(cdata(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(svmmod,xGrid);
figure;
h = nan(3,1); % Preallocation
h(1:2) = gscatter(cdata(:,1),cdata(:,2),grp,'rg','+*');
hold on
h(3) = plot(cdata(svmmod.IsSupportVector,1),...
   cdata(svmmod.IsSupportVector,2),'ko');
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0 0],'k');
legend(h,{'-1','+1','Support Vectors'},'Location','Southeast');
axis equal
hold off
```



另请参阅

bayesopt | fitcsvm

相关示例

- “Optimize Cross-Validated Classifier Using bayesopt”

参数化回归分析

决定系数（R 方）

目的

决定系数（R 方）表示线性回归模型中由自变量 X 解释的响应变量 y 的变化比例。R 方越大，线性回归模型解释的变异越大。

定义

R 方是模型解释的平方总和的比例。拟合模型的一个属性 **Rsquared** 是包含两个字段的结构体：

- **Ordinary** - 普通（未经调整的）R 方

  $$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}.$$

- **Adjusted** - 根据系数的数量调整的 R 方

  $$R^2_{adj} = 1 - \left(\frac{n-1}{n-p}\right)\frac{SSE}{SST}.$$

  SSE 是误差平方和，SSR 是回归平方和，SST 是平方和的总和，n 是观测值的数量，而 p 是回归系数的数量。请注意，p 包含截距，因此，例如，对于线性拟合，p 为 2。由于在回归模型中随着预测变量的增加，R 方会增大，因此调整 R 方会根据模型中预测变量的数目进行调整。这样有助于比较具有不同预测变量个数的模型。

如何

在获得拟合模型（例如 **mdl**）后，您可以使用 **fitlm** 或 **stepwiselm**，通过圆点表示法对属性进行索引来获得标量 R 方值，例如，

```
mdl.Rsquared.Ordinary
mdl.Rsquared.Adjusted
```

您也可以使用 SSE、SSR 和 SST 属性名来获取相应值。

```
mdl.SSE
mdl.SSR
mdl.SST
```

显示决定系数

此示例说明如何显示 R 方（决定系数）和调整 R 方。加载样本数据并定义响应和自变量。

```
load hospital
y = hospital.BloodPressure(:,1);
X = double(hospital(:,2:5));
```

拟合线性回归模型。

```
mdl = fitlm(X,y)
```

```
mdl =
Linear regression model:
```

```
y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:
                   Estimate       SE       tStat       pValue
                   ________    ________    ________    _________

    (Intercept)       117.4      5.2451      22.383    1.1667e-39
    x1              0.88162      2.9473     0.29913       0.76549
    x2              0.08602     0.06731       1.278       0.20438
    x3            -0.016685    0.055714    -0.29947       0.76524
    x4                9.884      1.0406       9.498    1.9546e-15


Number of observations: 100, Error degrees of freedom: 95
Root Mean Squared Error: 4.81
R-squared: 0.508,  Adjusted R-Squared: 0.487
F-statistic vs. constant model: 24.5, p-value = 5.99e-14
```

R 方值和调整 R 方值分别为 0.508 和 0.487。模型解释了响应变量中大约 50% 的变异。

使用拟合的 **LinearModel** 对象的属性访问 R 方值和调整 R 方值。

```
mdl.Rsquared.Ordinary

ans = 0.5078

mdl.Rsquared.Adjusted

ans = 0.4871
```

调整 R 方值小于普通的 R 方值。

另请参阅

LinearModel | **fitlm** | **stepwiselm** | **anova**

相关示例

- “Examine Quality and Adjust Fitted Model”
- “Interpret Linear Regression Results”
- “Summary of Output and Diagnostic Statistics”

偏最小二乘回归和主成分回归

此示例说明如何应用偏最小二乘回归 (PLSR) 和主成分回归 (PCR)，并讨论这两种方法的有效性。当存在大量预测变量并且它们高度相关甚至共线时，PLSR 和 PCR 都可以作为建模响应变量的方法。这两种方法都将新的预测变量（称为成分）构建为原始预测变量的线性组合，但它们构建这些成分的方式不同。PCR 创建成分来解释在预测变量中观察到的变异性，而根本不考虑响应变量。而 PLSR 会考虑响应变量，因此常使模型能够拟合具有更少成分的响应变量。从实际应用上来说，这能否最终转化为更简约的模型要视情况而定。

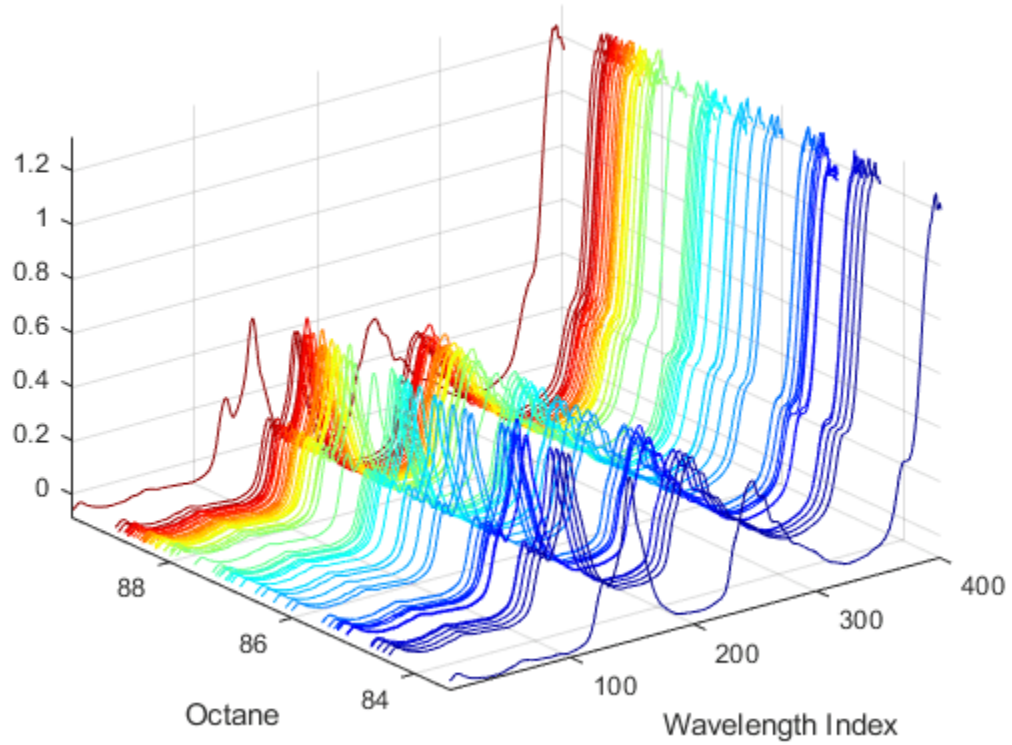
加载数据

加载包括 60 个汽油样本的 401 个波长的光频谱强度及其辛烷值的数据集。这些数据在以下文献中有详细描述：Kalivas, John H., "Two Data Sets of Near Infrared Spectra," Chemometrics and Intelligent Laboratory Systems, v.37 (1997) pp.255-259。

```
load spectra
whos NIR octane
```

```
  Name       Size            Bytes  Class    Attributes

  NIR       60x401         192480  double
  octane     60x1             480  double
```

```
[dummy,h] = sort(octane);
oldorder = get(gcf,'DefaultAxesColorOrder');
set(gcf,'DefaultAxesColorOrder',jet(60));
plot3(repmat(1:401,60,1)',repmat(octane(h),1,401)',NIR(h,:)');
set(gcf,'DefaultAxesColorOrder',oldorder);
xlabel('Wavelength Index'); ylabel('Octane'); axis('tight');
grid on
```



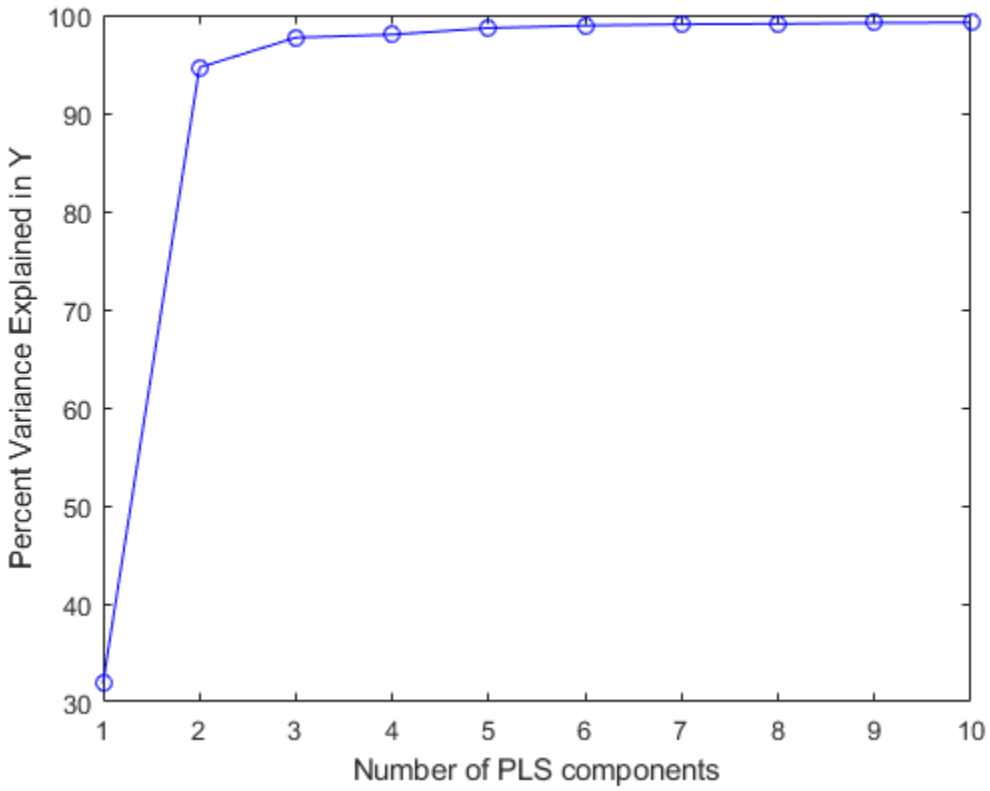
使用两个成分拟合数据

使用 **plsregress** 函数来拟合一个具有十个 PLS 成分和一个响应变量的 PLSR 模型。

```
X = NIR;
y = octane;
[n,p] = size(X);
[Xloadings,Yloadings,Xscores,Yscores,betaPLS10,PLSPctVar] = plsregress(...
    X,y,10);
```

十个成分可能超出充分拟合数据所需要的成分数量，但可以根据此拟合的诊断来选择具有更少成分的简单模型。例如，选择成分数量的一种快速方法是将响应变量中解释的方差百分比绘制为成分数量的函数。

```
plot(1:10,cumsum(100*PLSPctVar(2,:)),'-bo');
xlabel('Number of PLS components');
ylabel('Percent Variance Explained in Y');
```



在实际应用中，选择成分数量时可能需要更加谨慎。例如交叉验证就是一种广泛使用的方法，本示例稍后将进行说明。就目前来说，上图显示具有两个成分的 PLSR 即能解释观测的 y 中的大部分方差。计算双成分模型的拟合响应值。

```
[Xloadings,Yloadings,Xscores,Yscores,betaPLS] = plsregress(X,y,2);
yfitPLS = [ones(n,1) X]*betaPLS;
```

接下来，拟合具有两个主成分的 PCR 模型。第一步是使用 **pca** 函数对 **X** 执行主成分分析，并保留两个主成分。然后，PCR 就只是响应变量对这两个成分的线性回归。当不同变量的取值范围有很大差异时，通常应该先按照变量的标准差来归一化每个变量，但在本例中并没有这样做。

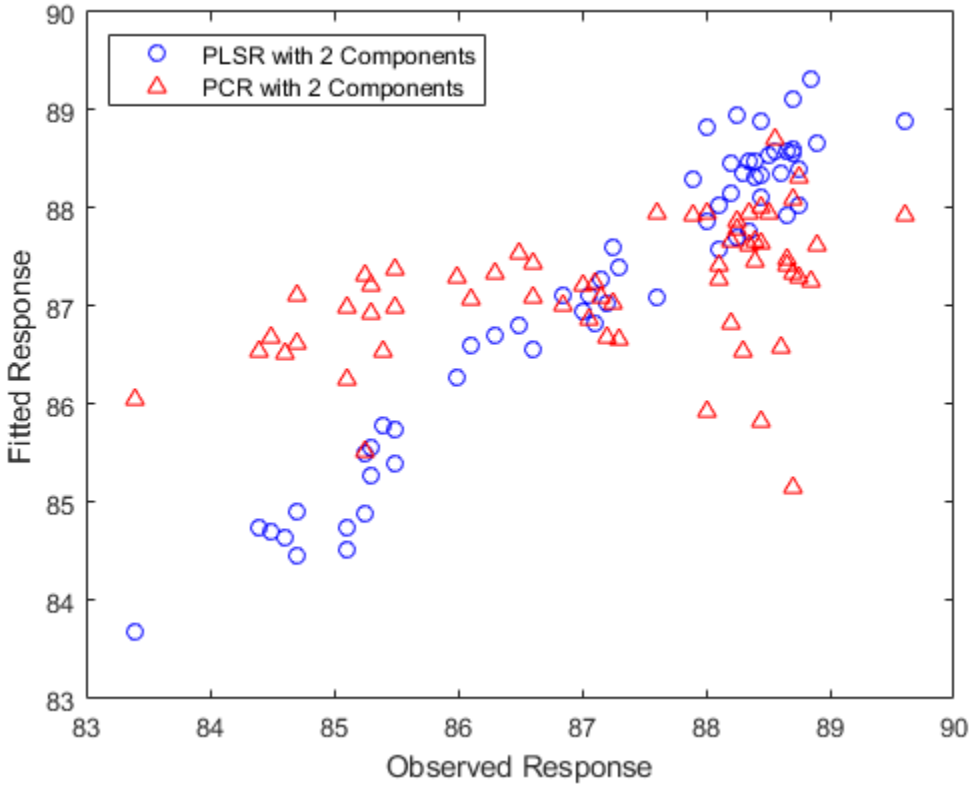
```
[PCALoadings,PCAScores,PCAVar] = pca(X,'Economy',false);
betaPCR = regress(y-mean(y), PCAScores(:,1:2));
```

为了更易于对比原始频谱数据解释 PCR 结果，对原始未中心化变量的回归系数进行转换。

```
betaPCR = PCALoadings(:,1:2)*betaPCR;
betaPCR = [mean(y) - mean(X)*betaPCR; betaPCR];
yfitPCR = [ones(n,1) X]*betaPCR;
```

绘制 PLSR 和 PCR 两种方法的拟合响应值对观察响应值的图。

```
plot(y,yfitPLS,'bo',y,yfitPCR,'r^');
xlabel('Observed Response');
ylabel('Fitted Response');
legend({'PLSR with 2 Components' 'PCR with 2 Components'}, ...
   'location','NW');
```



上图中的比较从某种意义上说并不合理 - 成分数量（两个）是通过观察具有两个成分的 PLSR 模型预测响应变量的效果来选择的，而对于 PCR 模型来说，并无充分理由将其成分数量限制为与 PLSR 一致。但是，在成分数量相同的情况下，PLSR 拟合 y 的效果更好。事实上，观察上图中拟合值的水平散点图可以看出，两个成分的 PCR 并不比使用常量模型更好。两种回归的 R 方值证实了这一点。

```
TSS = sum((y-mean(y)).^2);
RSS_PLS = sum((y-yfitPLS).^2);
rsquaredPLS = 1 - RSS_PLS/TSS
```

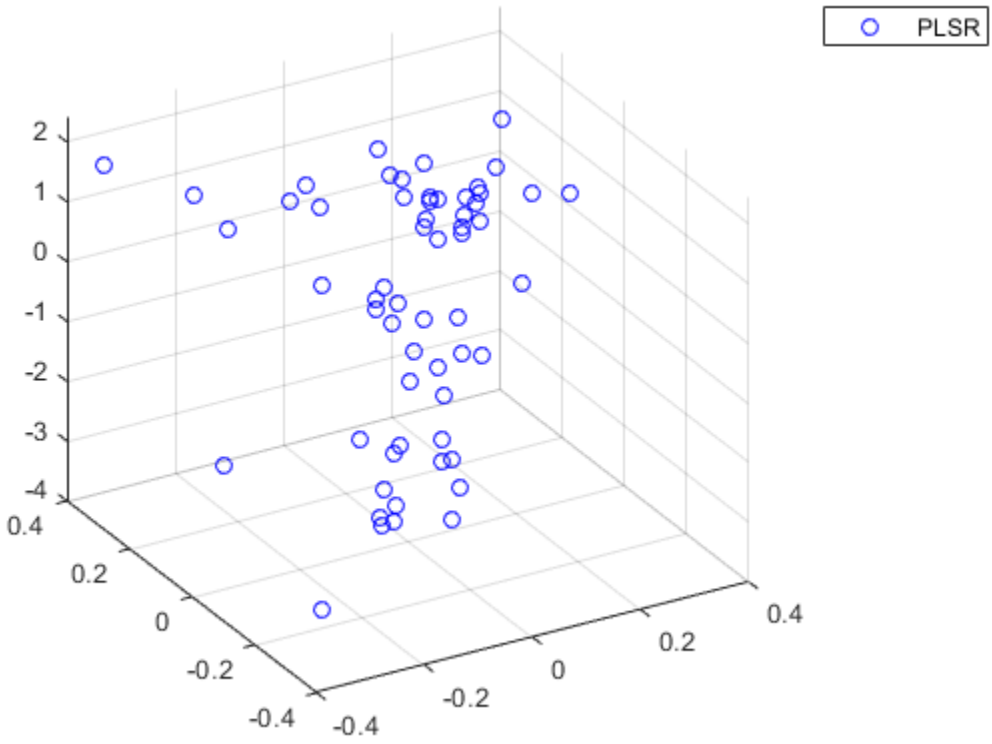
```
rsquaredPLS =

    0.9466
```

```
RSS_PCR = sum((y-yfitPCR).^2);
rsquaredPCR = 1 - RSS_PCR/TSS
```

```
rsquaredPCR =

    0.1962
```

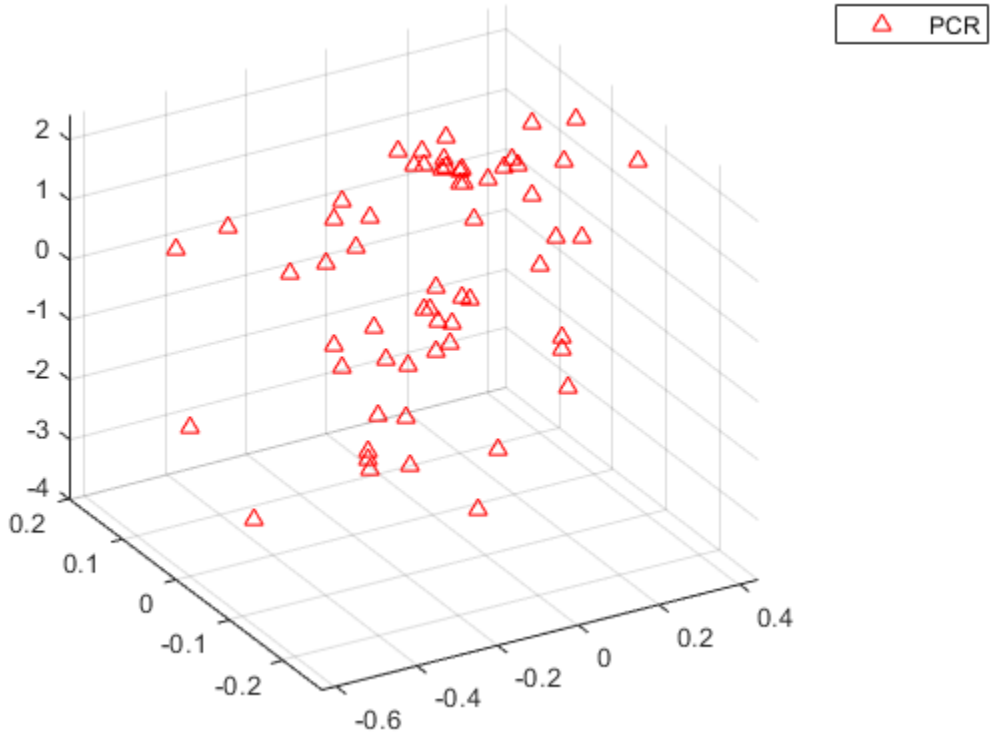
另一种比较两个模型的预测能力的方法是绘制两种情况下响应变量对两个预测变量的图。

```
plot3(Xscores(:,1),Xscores(:,2),y-mean(y),'bo');
legend('PLSR');
grid on; view(-30,30);
```



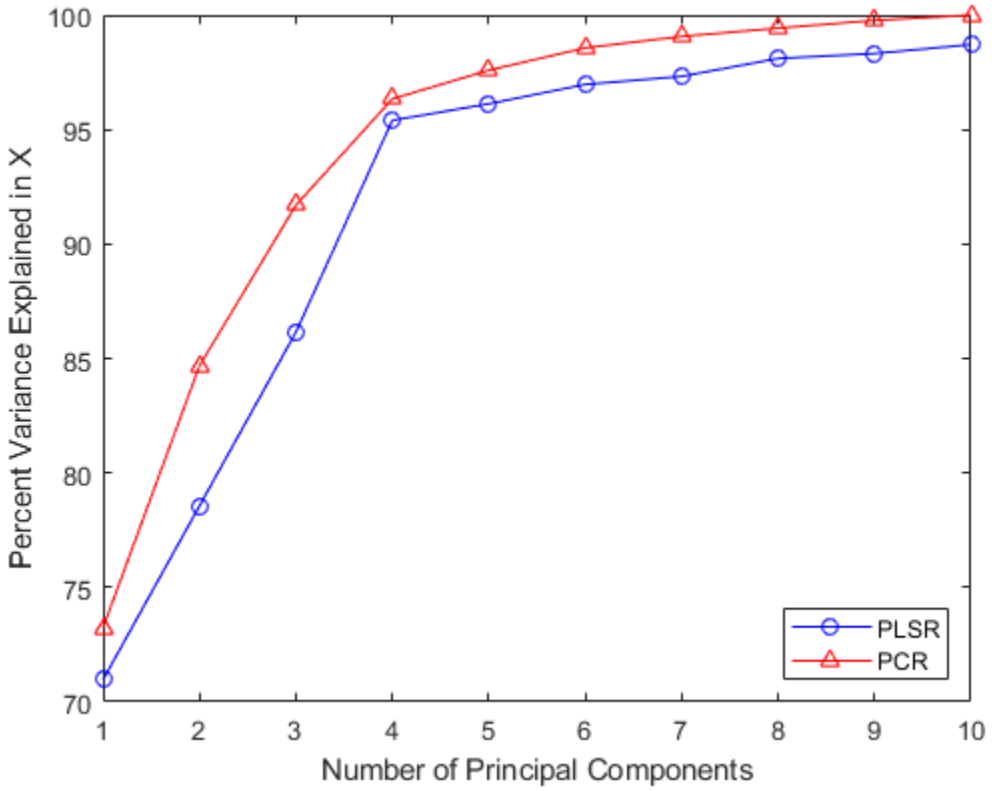
虽然可能需要交互旋转图形才能明显地看出分布情况，但上面的 PLSR 图还是显示出点几乎都散布在一个平面上。而下面的 PCR 图显示一片点云，几乎看不出线性关系。

```
plot3(PCAScores(:,1),PCAScores(:,2),y-mean(y),'r^');
legend('PCR');
grid on; view(-30,30);
```



请注意，虽然这两个 PLS 成分对 **y** 观测值而言预测效果较好，但下图显示，相比 PCR 所用的前两个主成分，这两个 PLS 成分解释的 **X** 观测值方差比例较少。

```
plot(1:10,100*cumsum(PLSPctVar(1,:)),'b-o',1:10, ...
   100*cumsum(PCAVar(1:10))/sum(PCAVar(1:10)),'r-^');
xlabel('Number of Principal Components');
ylabel('Percent Variance Explained in X');
legend({'PLSR' 'PCR'},'location','SE');
```

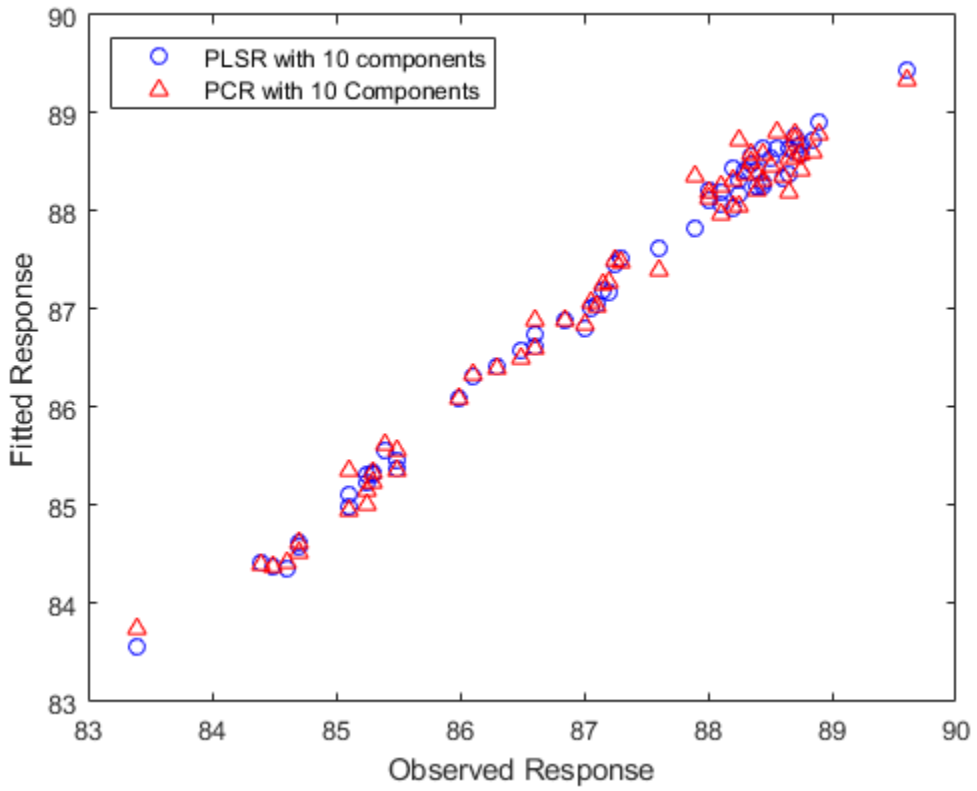


PCR 曲线更均匀的事实说明了为什么具有两个成分的 PCR 在拟合 **y** 方面比 PLSR 差。PCR 构造成分是为了最好地解释 **X**，因此前两个成分忽略了数据中对拟合观察到的 **y** 重要的信息。

使用更多成分拟合数据

在 PCR 中添加更多成分后，它必然可以更好地拟合原始数据 **y**，因为 **X** 中的大多数重要预测信息都会在某个点出现在主成分中。例如，下图显示，使用十个成分时，两种方法的残差差异远小于使用两个成分时。

```
yfitPLS10 = [ones(n,1) X]*betaPLS10;
betaPCR10 = regress(y-mean(y), PCAScores(:,1:10));
betaPCR10 = PCALoadings(:,1:10)*betaPCR10;
betaPCR10 = [mean(y) - mean(X)*betaPCR10; betaPCR10];
yfitPCR10 = [ones(n,1) X]*betaPCR10;
plot(y,yfitPLS10,'bo',y,yfitPCR10,'r^');
xlabel('Observed Response');
ylabel('Fitted Response');
legend({'PLSR with 10 components' 'PCR with 10 Components'}, ...
   'location','NW');
```



虽然 PLSR 拟合的准确度稍高一点，不过，两个模型都比较准确地拟合了 y。但是，十个成分对任一模型来说仍然是任意选择的数字。

通过交叉验证选择成分数量

在预测未来观测值对预测变量的响应时，选择适当数量的成分以最大程度减少预期误差通常很有用。简单地使用大量成分将能够很好地拟合当前观察到的数据，但这种策略往往会导致过拟合。太好地拟合当前数据会导致模型不能很好地推广到其他数据，并对预期误差给出过度乐观的估计。

交叉验证是统计学上更加合理的成分数量选择方法，不管是在 PLSR 中还是在 PCR 中。它在拟合模型和估计预测误差时使用不同的数据，从而避免过拟合数据。因此，预测误差的估计值不会乐观地偏小。

plsregress 有一个选项可以通过交叉验证来估计均方预测误差 (MSEP)，此处我们使用 10 折交叉验证。

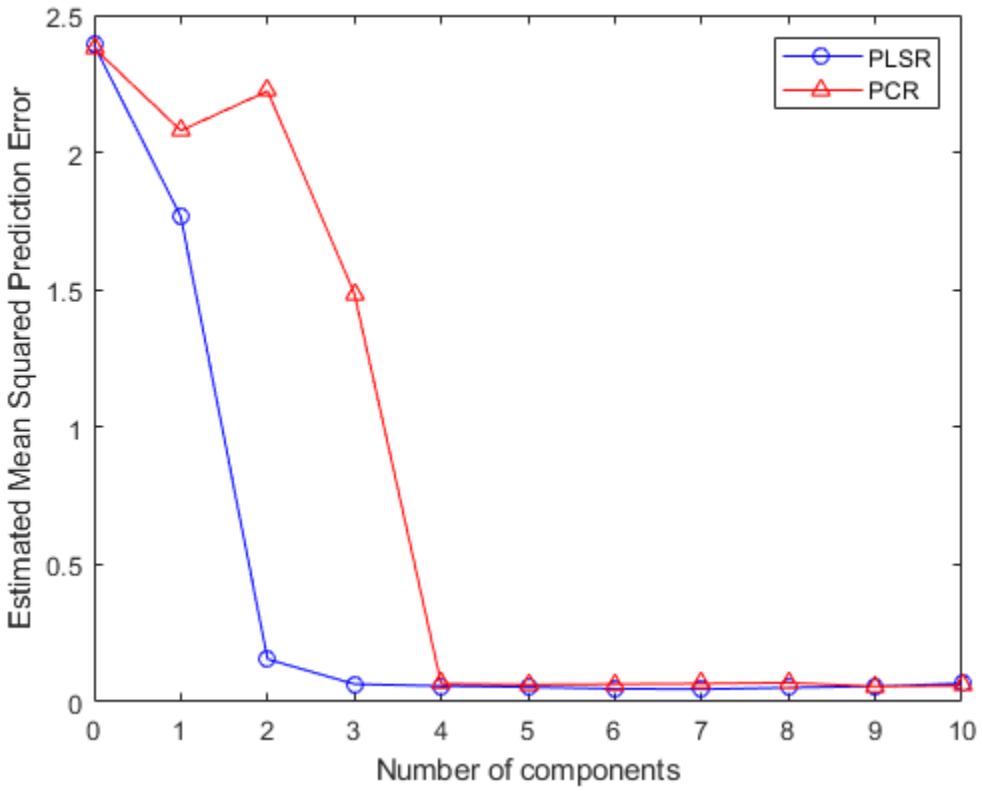
```
[Xl,Yl,Xs,Ys,beta,pctVar,PLSmsep] = plsregress(X,y,10,'CV',10);
```

对于 PCR，将 **crossval** 与一个简单的函数（计算 PCR 的平方误差之和）相结合，可以估计 MSEP，在此同样使用 10 折交叉验证。

```
PCRmsep = sum(crossval(@pcrsse,X,y,'KFold',10),1) / n;
```

PLSR 的 MSEP 曲线表明，两个或三个成分的效果已经足够好。另一方面，PCR 需要四个成分才能获得同样的预测准确度。

```
plot(0:10,PLSmsep(2,:),'b-o',0:10,PCRmsep,'r-^');
xlabel('Number of components');
ylabel('Estimated Mean Squared Prediction Error');
legend({'PLSR' 'PCR'},'location','NE');
```



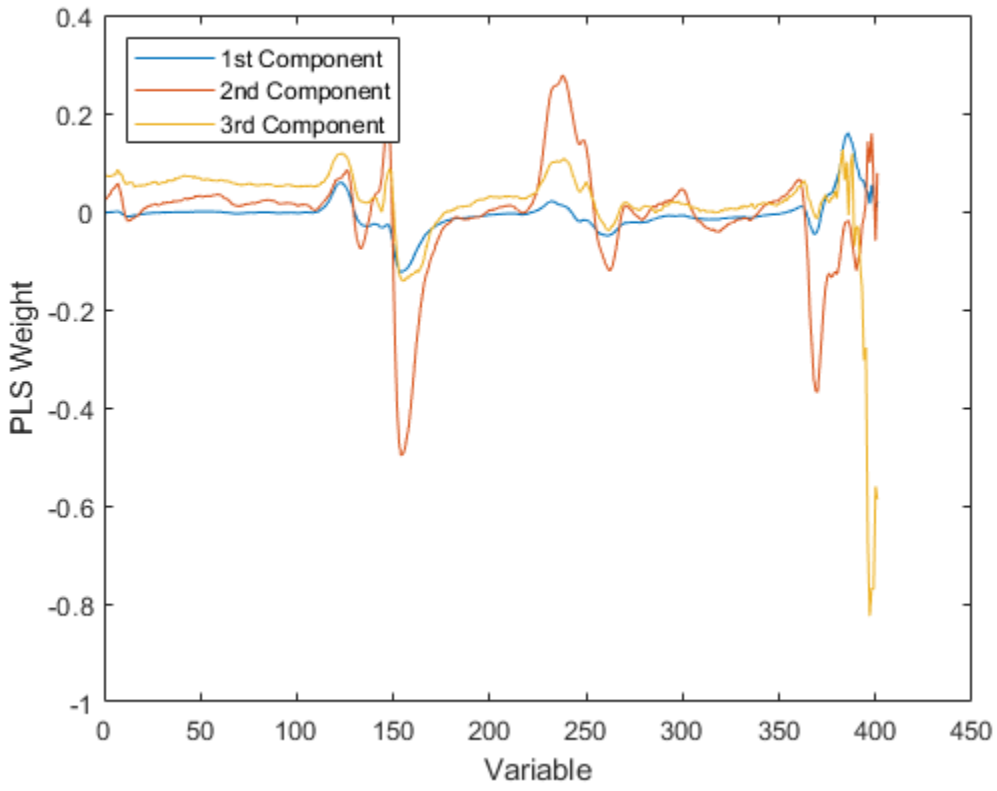
事实上，PCR 中的第二个成分会增加模型的预测误差，表明该成分中包含的预测变量的组合与 **y** 没有很强的相关性。同样，这是因为 PCR 的构造成分是为了解释 **X** 而非 **y** 的变异。

模型精简

因此，如果 PCR 需要四个成分才能获得与具有三个成分的 PLSR 相同的预测精度，是否说明 PLSR 模型更加精简呢？这取决于您考虑的是模型的哪个方面。

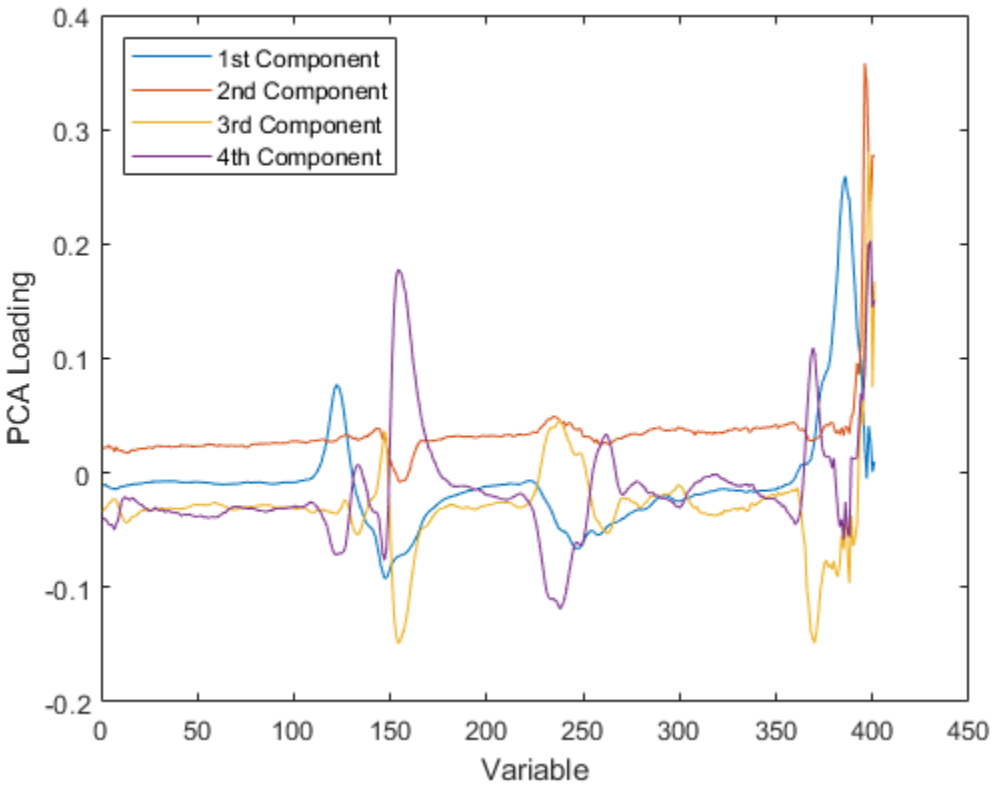
PLS 权重是定义 PLS 成分的原始变量的线性组合，即，它们描述 PLSR 中的每个成分依赖原始变量的程度和方向。

```
[Xl,Yl,Xs,Ys,beta,pctVar,mse,stats] = plsregress(X,y,3);
plot(1:401,stats.W,'-');
xlabel('Variable');
ylabel('PLS Weight');
legend({'1st Component' '2nd Component' '3rd Component'}, ...
   'location','NW');
```



同样，PCA 载重描述 PCR 中的每个成分依赖原始变量的程度。

```
plot(1:401,PCALoadings(:,1:4),'-');
xlabel('Variable');
ylabel('PCA Loading');
legend({'1st Component' '2nd Component' '3rd Component'  ...
   '4th Component'},'location','NW');
```



不管对于 PLSR 还是 PCR，都可以通过检查哪些变量的权重最大来为每个成分提供一个具有实际意义的解释。例如，利用这些频谱数据，也许能够从汽油中存在的化合物的角度解释强度波峰，然后观察特定成分的权重，从中选出少数几种化合物。从这个角度来说，成分越少越容易解释，而且因为 PLSR 通常需要更少的成分就能充分地预测响应，所以模型更精简。

另一方面，PLSR 和 PCR 为每个原始预测变量都生成一个回归系数和一个截距。从这个意义上讲，二者都不算精简，因为无论使用多少成分，两个模型都依赖于所有预测变量。再具体一点，对于这些数据，两个模型都需要 401 个频谱强度值才能进行预测。

然而，最终目的可能是将原始变量集缩减为更小但仍能准确预测响应的子集。例如，可以使用 PLS 权重或 PCA 载重只选择对每个成分贡献最大的那些变量。正如前文所示，PCR 模型拟合中的一些成分主要用于描述预测变量的变异，与响应的相关性不强的那些变量可能具有较大的权重。因此，PCR 可能会导致那些对预测不必要的变量被保留下来。

对于本例中使用的数据，PLSR 和 PCR 进行精确预测所需要的成分数量差别不是太大，而且 PLS 权重和 PCA 载重似乎选择了相同的变量。对于其他数据，情况可能并非如此。

广义线性模型

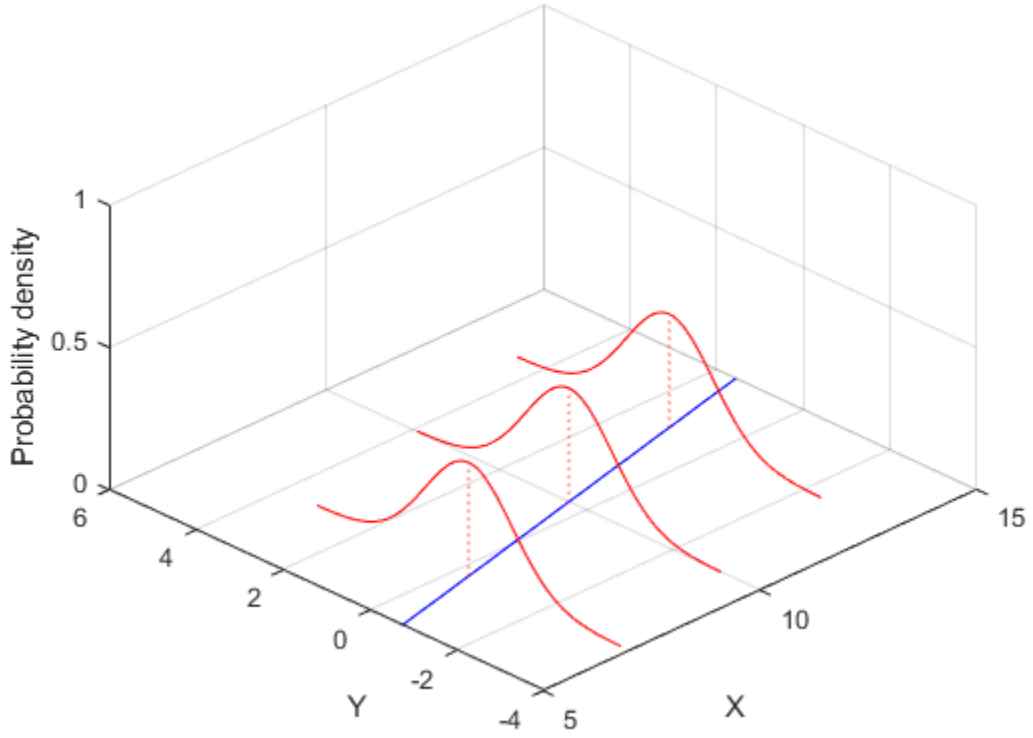
使用广义线性模型拟合数据

此示例说明如何使用 **glmfit** 和 **glmval** 来拟合和计算广义线性模型。普通线性回归可用于将直线或具有线性参数的函数与具有正态分布误差的数据相拟合。这是最常用的回归模型，但并非总是符合实际需要。广义线性模型通过两种方式对线性模型进行扩展。首先，通过引入联系函数，放宽了参数的线性假设。其次，可以对正态分布之外的误差分布进行建模。

广义线性模型

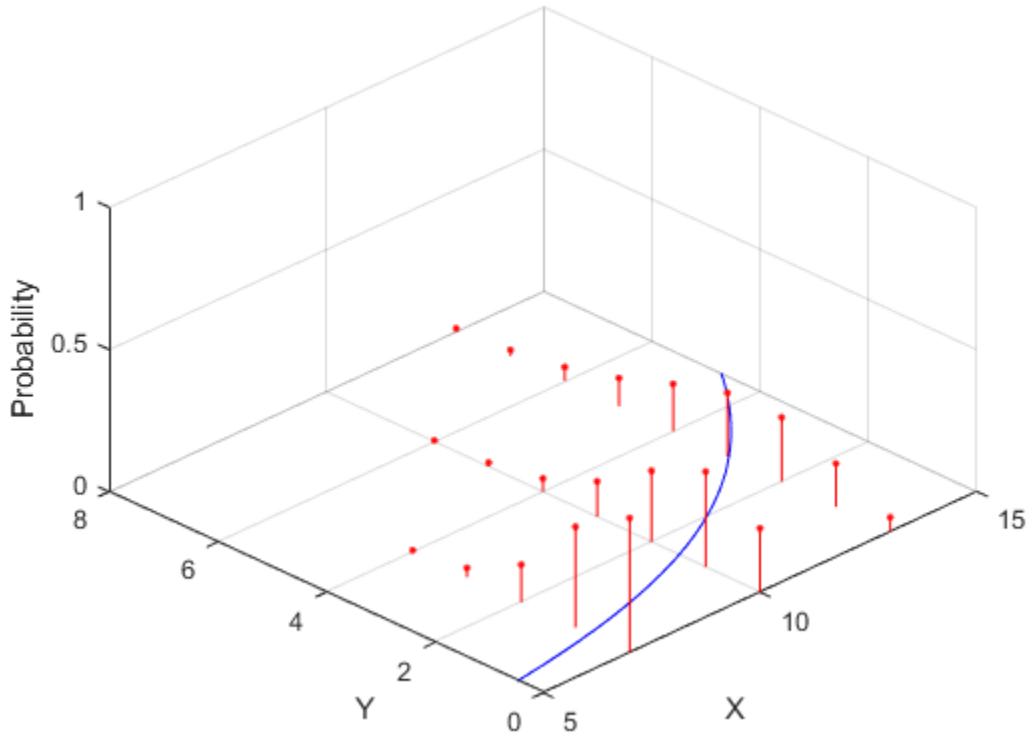
回归模型使用一个或多个预测变量（通常表示为 x1、x2 等）来定义一个响应变量（通常表示为 y）的分布。最常用的回归模型，即普通线性回归模型，将 y 建模为正态随机变量，其均值是预测变量的线性函数 b0 + b1*x1 + ...，其方差是常量。在只有一个预测变量 x 的最简单的情况下，该模型可以表示为由符合高斯分布的点组成的一条直线。

```
mu = @(x) -1.9+.23*x;
x = 5:.1:15;
yhat = mu(x);
dy = -3.5:.1:3.5; sz = size(dy); k = (length(dy)+1)/2;
x1 =  7*ones(sz); y1 = mu(x1)+dy; z1 = normpdf(y1,mu(x1),1);
x2 = 10*ones(sz); y2 = mu(x2)+dy; z2 = normpdf(y2,mu(x2),1);
x3 = 13*ones(sz); y3 = mu(x3)+dy; z3 = normpdf(y3,mu(x3),1);
plot3(x,yhat,zeros(size(x)),'b-', ...
    x1,y1,z1,'r-', x1([k k]),y1([k k]),[0 z1(k)],'r:', ...
    x2,y2,z2,'r-', x2([k k]),y2([k k]),[0 z2(k)],'r:', ...
    x3,y3,z3,'r-', x3([k k]),y3([k k]),[0 z3(k)],'r:');
zlim([0 1]);
xlabel('X'); ylabel('Y'); zlabel('Probability density');
grid on; view([-45 45]);
```



在广义线性模型中，响应变量的均值建模为对预测变量的线性函数的单调非线性变换 g(b0 + b1*x1 + ...)。变换 g 的逆称为“联系”函数。示例包括对数几率 (sigmoid) 联系和对数联系。此外，y 可能具有非正态分布，例如二项分布或泊松分布。例如，具有对数联系和一个预测变量 x 的泊松回归可以表示为由符合泊松分布的点组成的一条指数曲线。

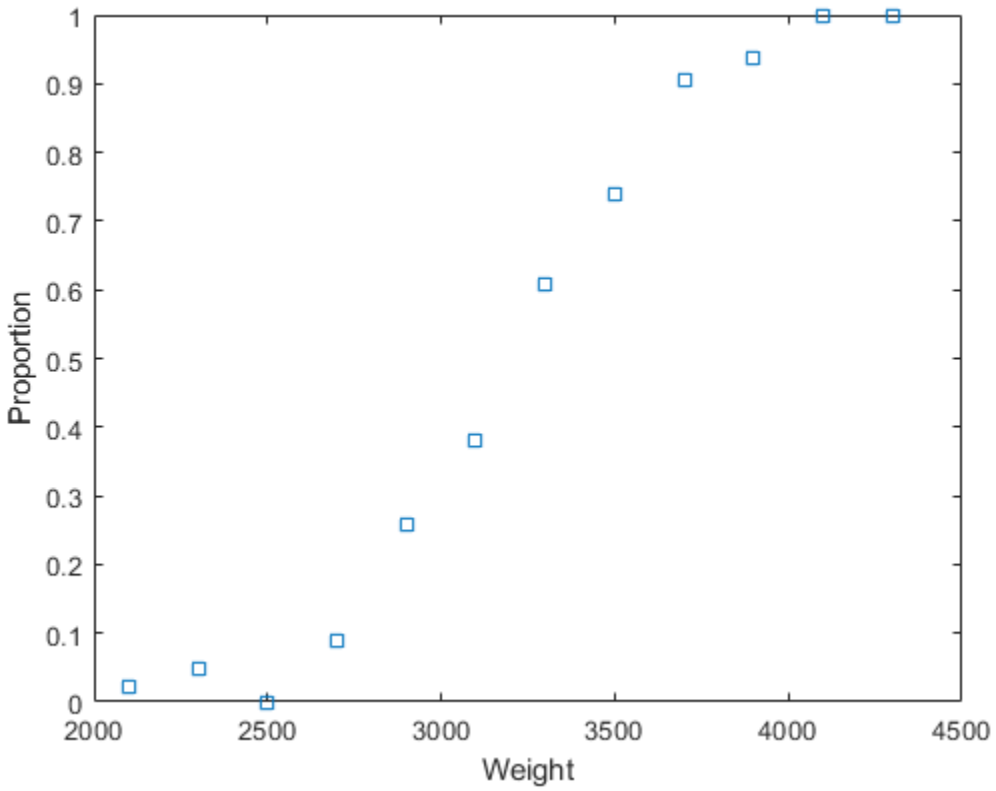
```
mu = @(x) exp(-1.9+.23*x);
x = 5:.1:15;
yhat = mu(x);
x1 =  7*ones(1,5);  y1 = 0:4; z1 = poisspdf(y1,mu(x1));
x2 = 10*ones(1,7); y2 = 0:6; z2 = poisspdf(y2,mu(x2));
x3 = 13*ones(1,9); y3 = 0:8; z3 = poisspdf(y3,mu(x3));
plot3(x,yhat,zeros(size(x)),'b-', ...
    [x1; x1],[y1; y1],[z1; zeros(size(y1))],'r-', x1,y1,z1,'r.', ...
    [x2; x2],[y2; y2],[z2; zeros(size(y2))],'r-', x2,y2,z2,'r.', ...
    [x3; x3],[y3; y3],[z3; zeros(size(y3))],'r-', x3,y3,z3,'r.');
zlim([0 1]);
xlabel('X'); ylabel('Y'); zlabel('Probability');
grid on; view([-45 45]);
```



拟合逻辑回归

此示例包含一个试验，以帮助建模不同重量的汽车在里程测试中的未通过比例。数据包括被测汽车的重量、汽车数量以及失败次数等观测值。

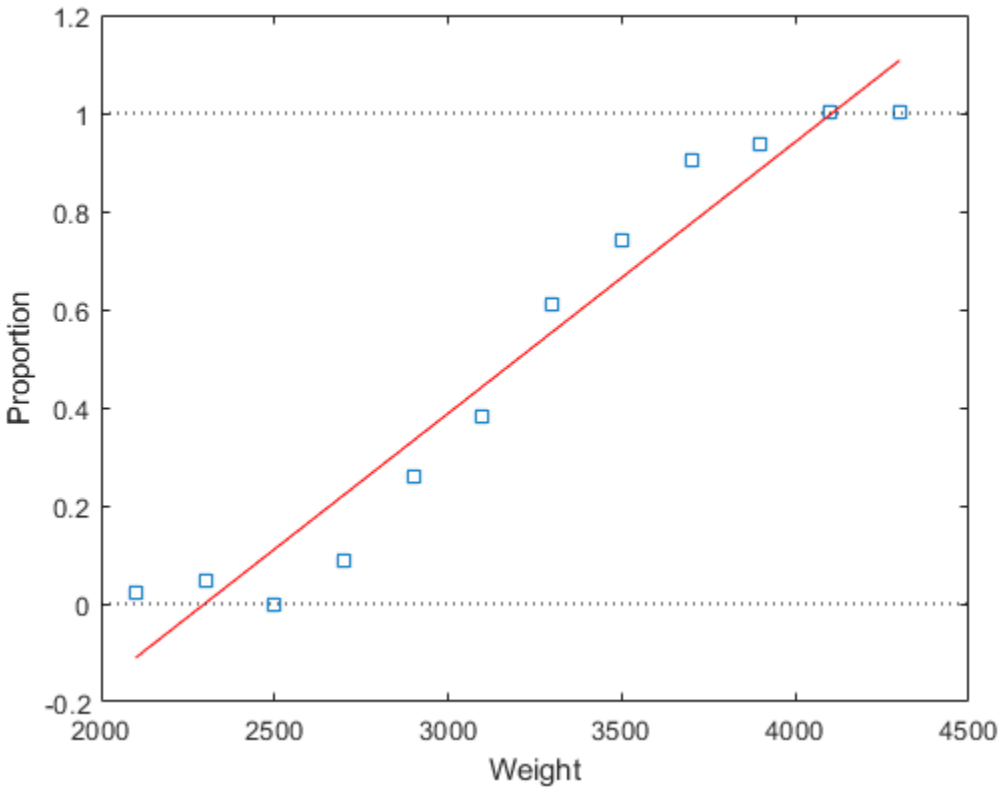
```
% A set of car weights
weight = [2100 2300 2500 2700 2900 3100 3300 3500 3700 3900 4100 4300]';
% The number of cars tested at each weight
tested = [48 42 31 34 31 21 23 23 21 16 17 21]';
% The number of cars failing the test at each weight
failed = [1 2 0 3 8 8 14 17 19 15 17 21]';
% The proportion of cars failing for each weight
proportion = failed ./ tested;

plot(weight,proportion,'s')
xlabel('Weight'); ylabel('Proportion');
```



此图是汽车失败比例值对汽车重量的函数关系图。可以合理地假设失败次数来自二项分布，其概率参数 P 随着重量而增加。但是，P 与重量的确切关系应该是怎样的呢？

我们可以尝试用一条直线来拟合这些数据。

```
linearCoef = polyfit(weight,proportion,1);
linearFit = polyval(linearCoef,weight);
plot(weight,proportion,'s', weight,linearFit,'r-', [2000 4500],[0 0],'k:', [2000 4500],[1 1],'k:')
xlabel('Weight'); ylabel('Proportion');
```



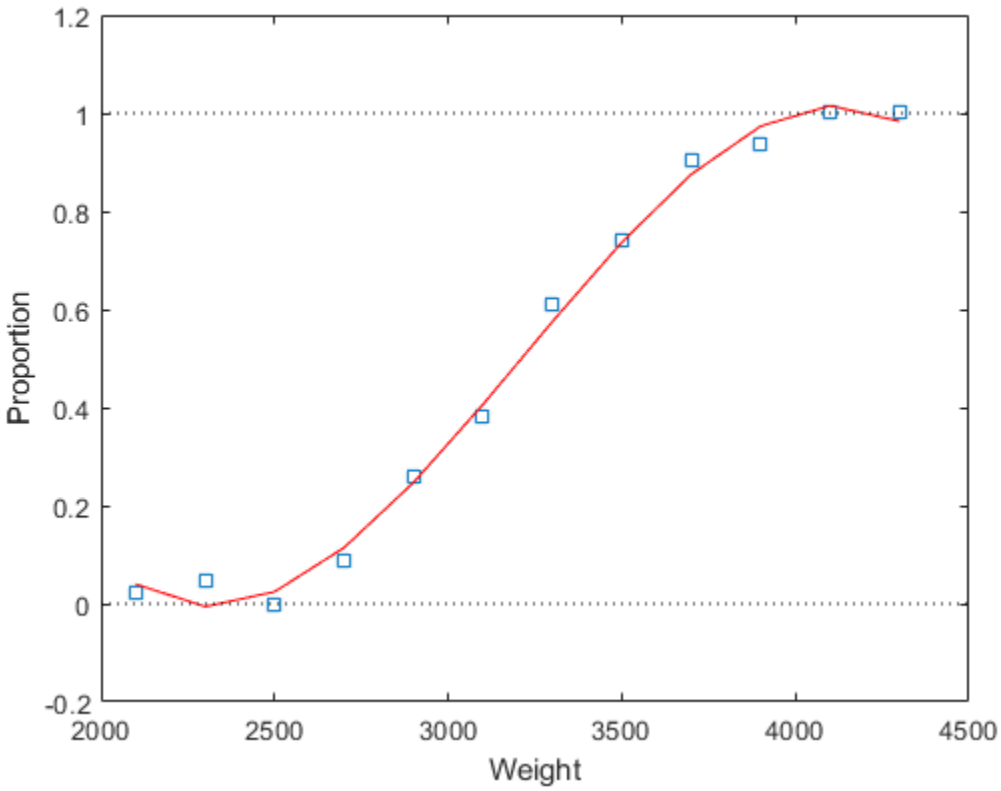
这种线性拟合有两个问题：

1) 线上存在小于 0 和大于 1 的预测比例值。

2) 因为这些比例值是有界的，因而不符合正态分布。这违反了拟合简单线性回归模型需满足的假设之一。

使用更高阶的多项式可能会有所帮助。

```
[cubicCoef,stats,ctr] = polyfit(weight,proportion,3);
cubicFit = polyval(cubicCoef,weight,[],ctr);
plot(weight,proportion,'s', weight,cubicFit,'r-', [2000 4500],[0 0],'k:', [2000 4500],[1 1],'k:')
xlabel('Weight'); ylabel('Proportion');
```

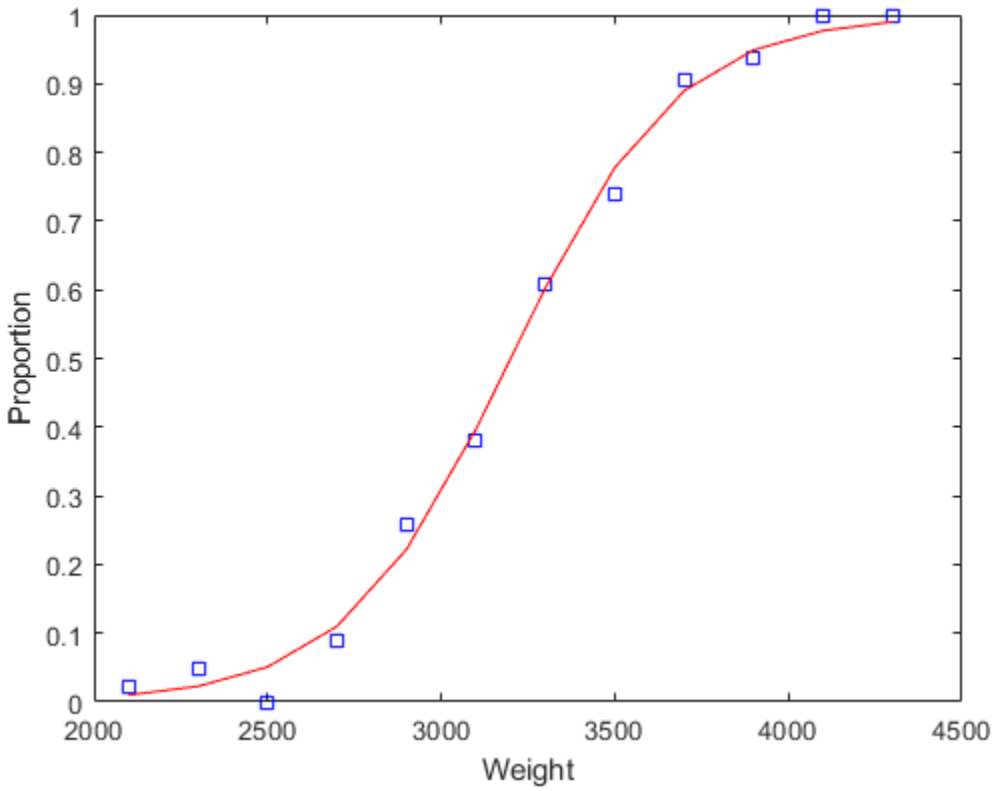


然而，此拟合仍然存在类似的问题。图中显示，当重量值超过 4000 时，拟合的失败比例值开始下降；但事实上，重量值进一步增大时，情况应与此相反。当然，这仍然违反正态分布的假设。

这种情况下，更好的方法是使用 **glmfit** 来拟合一个逻辑回归模型。逻辑回归是广义线性模型的特例，比线性回归更合适这些数据，原因有两个。首先，它使用适合二项分布的拟合方法。其次，逻辑联系将预测的比例值限制在 [0,1] 范围内。

对于逻辑回归，我们指定预测变量矩阵，再指定另一个一列包含失败次数、一列包含被测汽车数量的矩阵。我们还指定二项分布和对数几率联系。

```
[logitCoef,dev] = glmfit(weight,[failed tested],'binomial','logit');
logitFit = glmval(logitCoef,weight,'logit');
plot(weight,proportion,'bs', weight,logitFit,'r-');
xlabel('Weight'); ylabel('Proportion');
```



如图中所示，重量变小时，拟合比例值渐近为 0；重量变大时，拟合比例值渐近为 1。

模型诊断

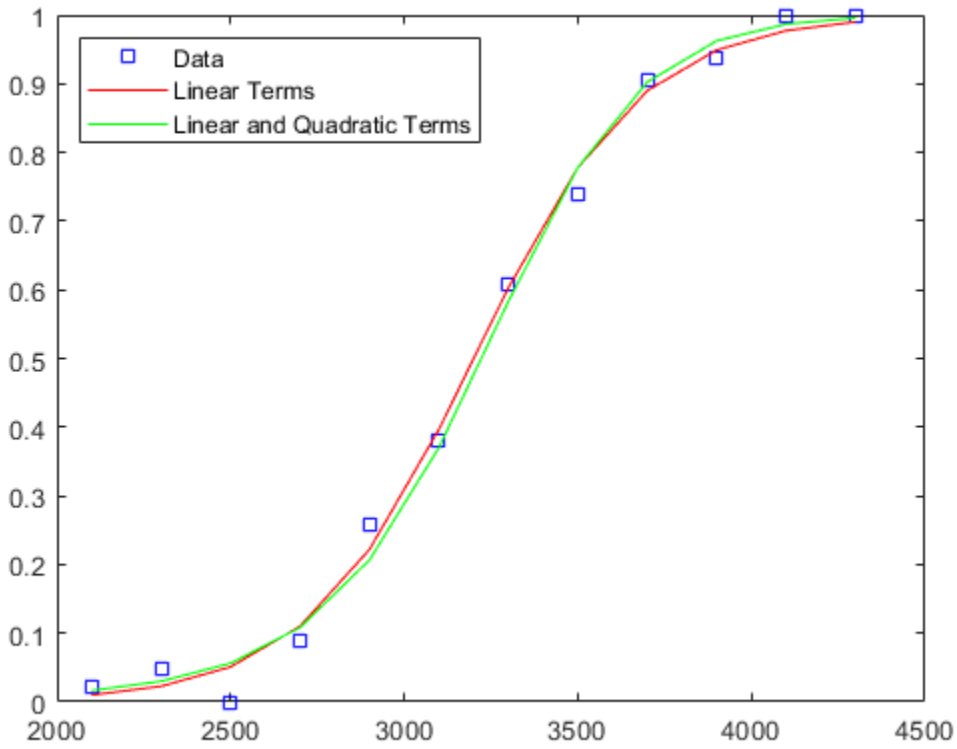
glmfit 函数提供几个输出，用于检查拟合和测试模型。例如，我们可以比较两个模型的偏差值，以确定平方项是否可以显著改善拟合。

```
[logitCoef2,dev2] = glmfit([weight weight.^2],[failed tested],'binomial','logit');
pval = 1 - chi2cdf(dev-dev2,1)
```

```
pval =

    0.4019
```

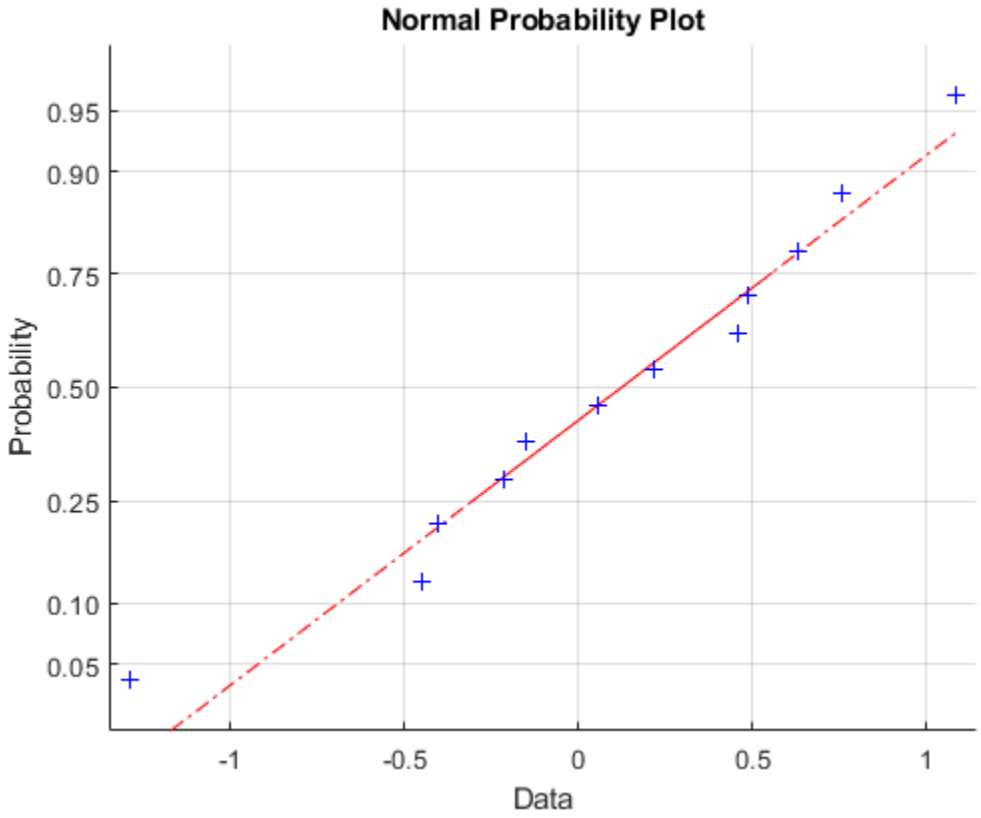
对于这些数据，较大的 p 值表明二次项没有显著改善拟合。两个拟合的图显示拟合几乎没有差异。

```
logitFit2 = glmval(logitCoef2,[weight weight.^2],'logit');
plot(weight,proportion,'bs', weight,logitFit,'r-', weight,logitFit2,'g-');
legend('Data','Linear Terms','Linear and Quadratic Terms','Location','northwest');
```



为了检查拟合的好坏，我们还可以查看 Pearson 残差的概率图。这些值已经归一化，因此当模型合理地拟合数据时，它们大致呈标准正态分布。（如果没有归一化，残差将具有不同的方差。）

```
[logitCoef,dev,stats] = glmfit(weight,[failed tested],'binomial','logit');
normplot(stats.residp);
```

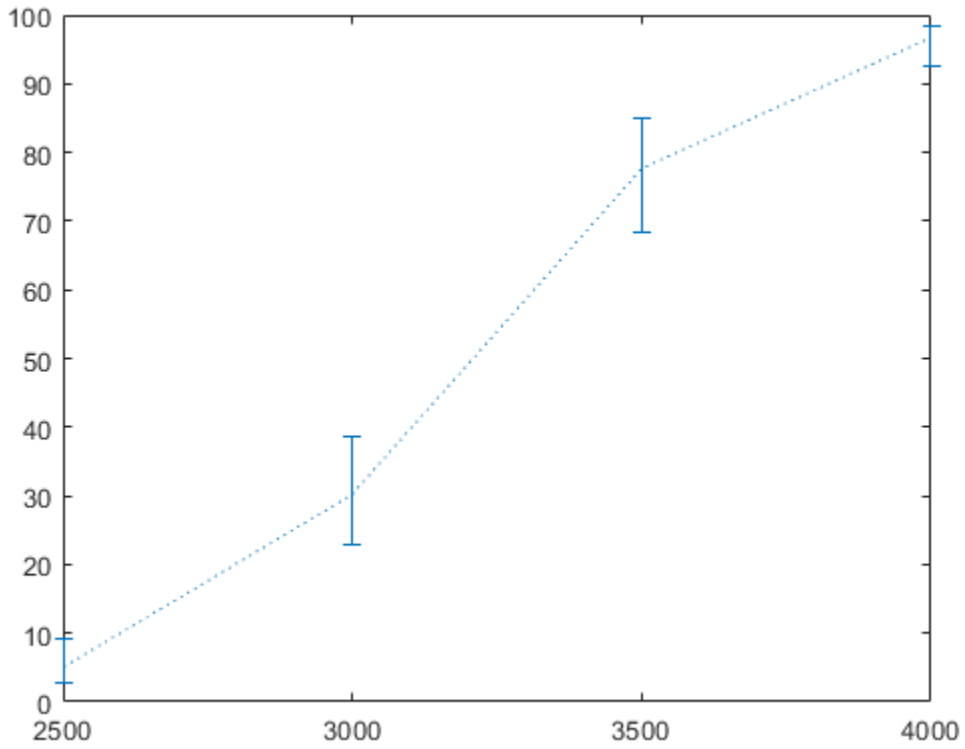


残差图显示非常符合正态分布。

计算模型预测

当我们对模型满意后，即可使用它来进行预测，包括计算置信边界。这里，我们分别对四种重量的汽车进行预测，看每 100 辆被测汽车中不能通过里程测试的汽车有多少辆。

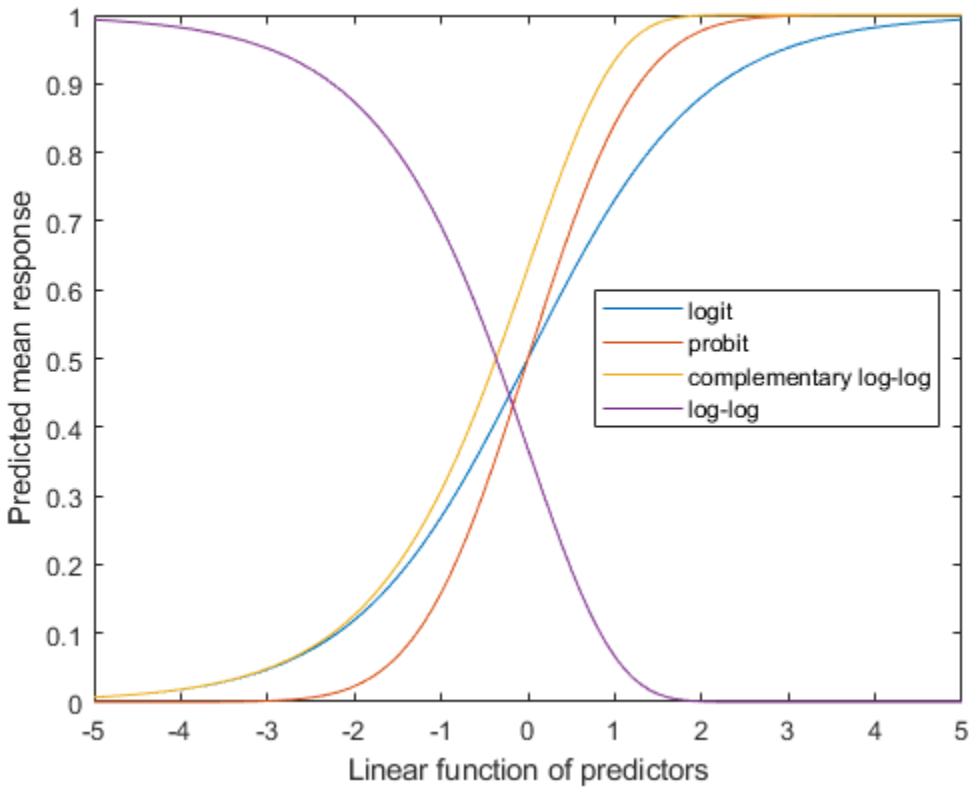
```
weightPred = 2500:500:4000;
[failedPred,dlo,dhi] = glmval(logitCoef,weightPred,'logit',stats,.95,100);
errorbar(weightPred,failedPred,dlo,dhi,':');
```



二项模型的联系函数

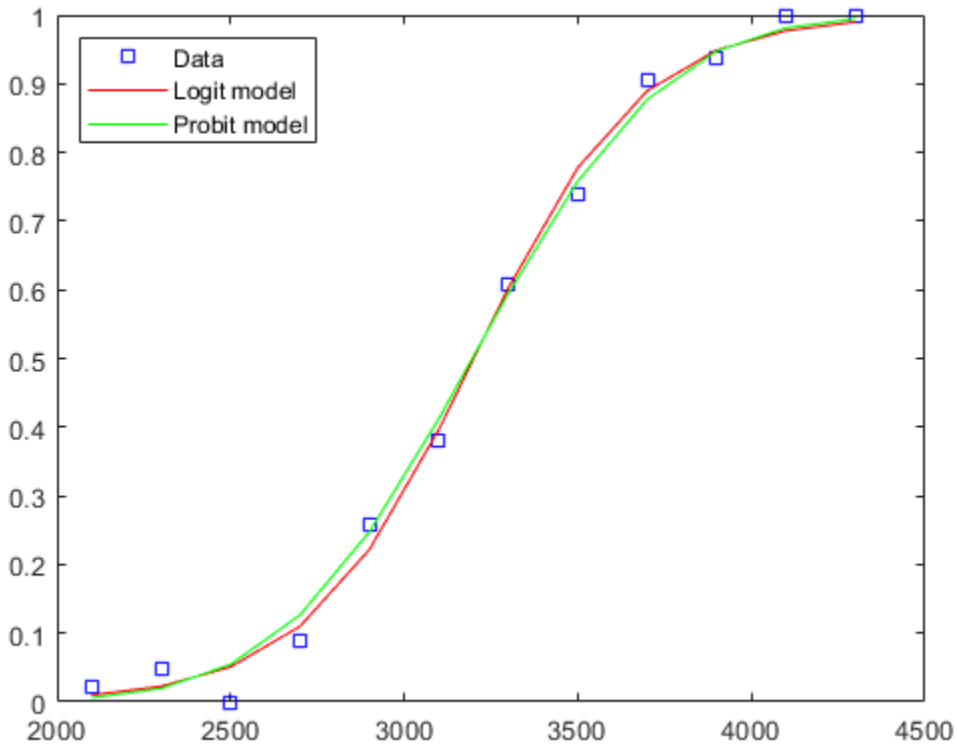
对于 **glmfit** 支持的五种分布，每一种都有一个典型（默认）的联系函数。对于二项分布，典型的联系为对数几率。但是，还有三个联系对二项模型敏感。所有四个联系的均值响应都在区间 [0, 1] 内。

```
eta = -5:.1:5;
plot(eta,1 ./ (1 + exp(-eta)),'-', eta,normcdf(eta), '-', ...
    eta,1 - exp(-exp(eta)),'-', eta,exp(-exp(eta)),'-');
xlabel('Linear function of predictors'); ylabel('Predicted mean response');
legend('logit','probit','complementary log-log','log-log','location','east');
```



例如，我们可以将具有概率比联系的拟合与具有对数几率联系的拟合进行比较。

```
probitCoef = glmfit(weight,[failed tested],'binomial','probit');
probitFit = glmval(probitCoef,weight,'probit');
plot(weight,proportion,'bs', weight,logitFit,'r-', weight,probitFit,'g-');
legend('Data','Logit model','Probit model','Location','northwest');
```



通常很难基于数据拟合效果判断四个联系函数中哪个较为适用，因此，我们往往基于理论来进行选择。

13

非线性回归

加权非线性回归

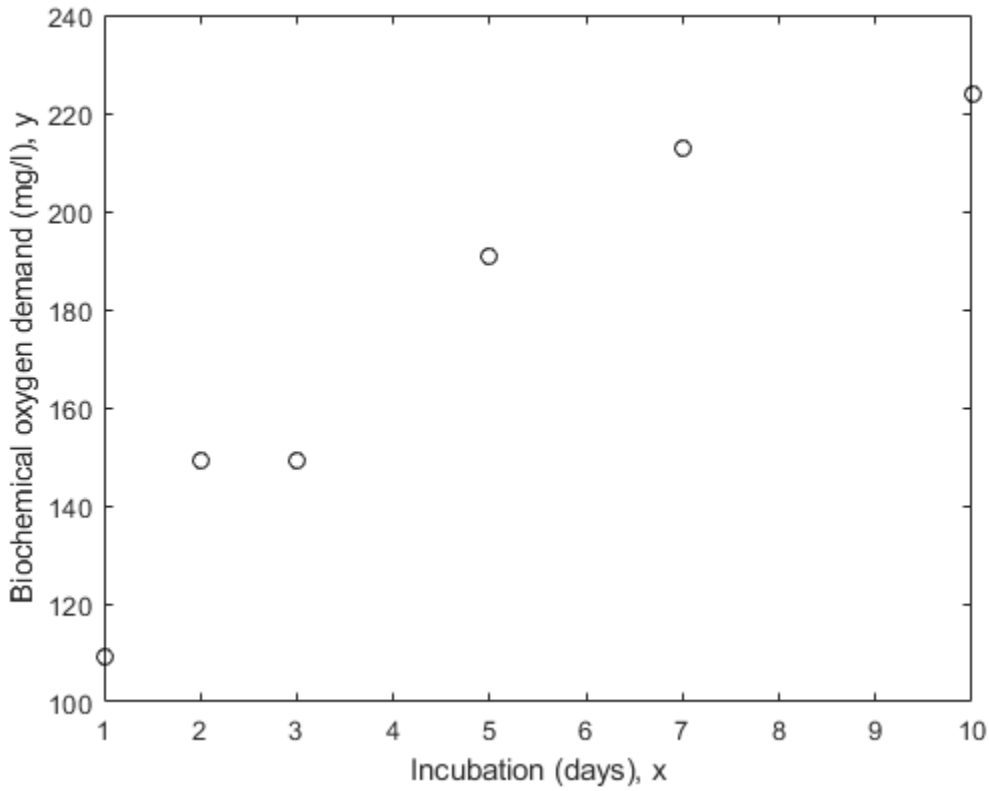
此示例说明如何将非线性回归模型与具有非常量误差方差的数据进行拟合。

当测量误差的方差全部相同时，适合使用常规非线性最小二乘算法。当这个假设不成立时，适合使用加权拟合。此示例说明如何将权重与 **fitnlm** 函数一起使用。

要拟合的数据和模型

我们将使用收集的数据来研究工业和生活垃圾造成的水污染。这些数据在以下文献中有详细描述：Box, G.P., W.G. Hunter, and J.S. Hunter, Statistics for Experimenters (Wiley, 1978, pp. 483-487)。响应变量是以 mg/l 为单位的生化需氧量，预测变量是以天为单位的潜伏期。

```
x = [1 2 3 5 7 10]';
y = [109 149 149 191 213 224]';

plot(x,y,'ko');
xlabel('Incubation (days), x'); ylabel('Biochemical oxygen demand (mg/l), y');
```



我们假设已知前两个观测值的精度低于其余观测值。比如说，它们可能是用不同仪器测量的。对数据进行加权的另一个常见原因是记录的每个观测值实际上是在相同的 x 值处提取的几个测量值的均值。在此处的数据中，假设前两个值各代表一个原始测量值，其余四个值中的每个值分别代表一个从 5 个原始测量值求得的均值。那么，根据每次观察的测量次数进行加权就是合适的。

```
w = [1 1 5 5 5 5]';
```

拟合这些数据的模型是一条缩放的指数曲线，当 x 变大时，曲线变为水平。

```
modelFun = @(b,x) b(1).*(1-exp(-b(2).*x));
```

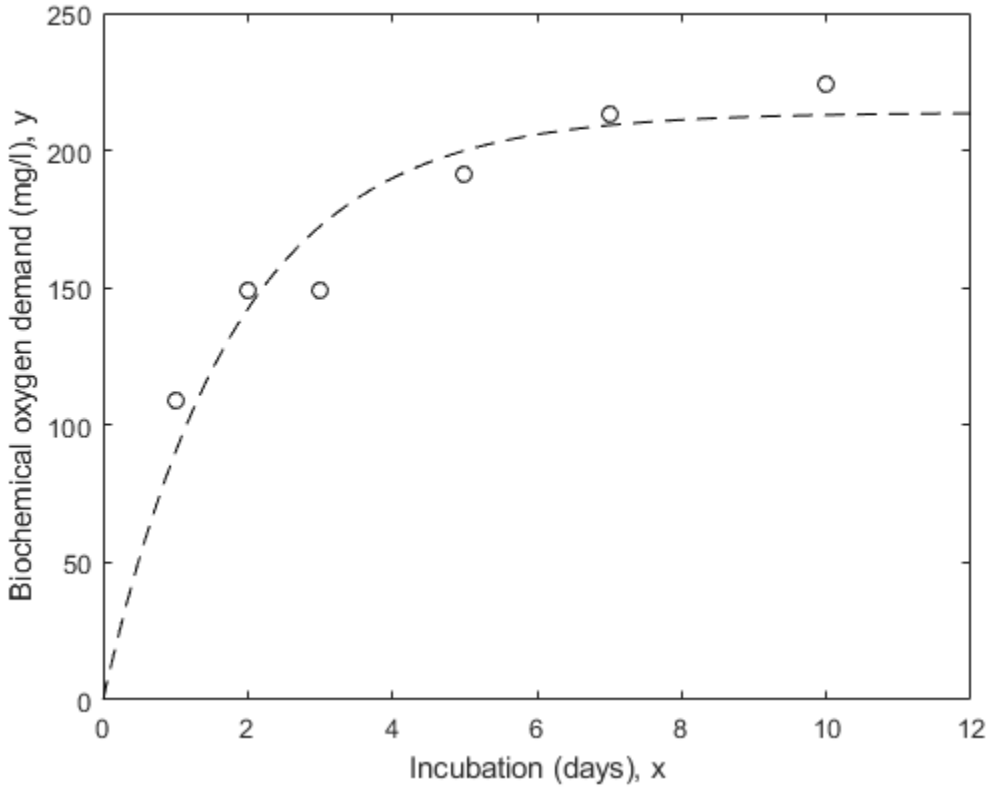
仅根据粗略的视觉拟合就可以看出，在 x = 15 附近，值大约为 240 的位置，沿这些点绘制的曲线似乎就会变平。因此，我们使用 240 作为 b1 的起始值，而且由于 e^(-.5*15) 比 1 小，所以我们使用 .5 作为 b2 的起始值。

```
start = [240; .5];
```

不加权拟合模型

忽略测量误差的危险在于拟合可能受不精确测量值的过度影响，因而可能无法很好地拟合已知精确的测量值。我们先进行不加权拟合，然后将数据与各点进行比较。

```
nlm = fitnlm(x,y,modelFun,start);
xx = linspace(0,12)';
line(xx,predict(nlm,xx),'linestyle','--','color','k')
```



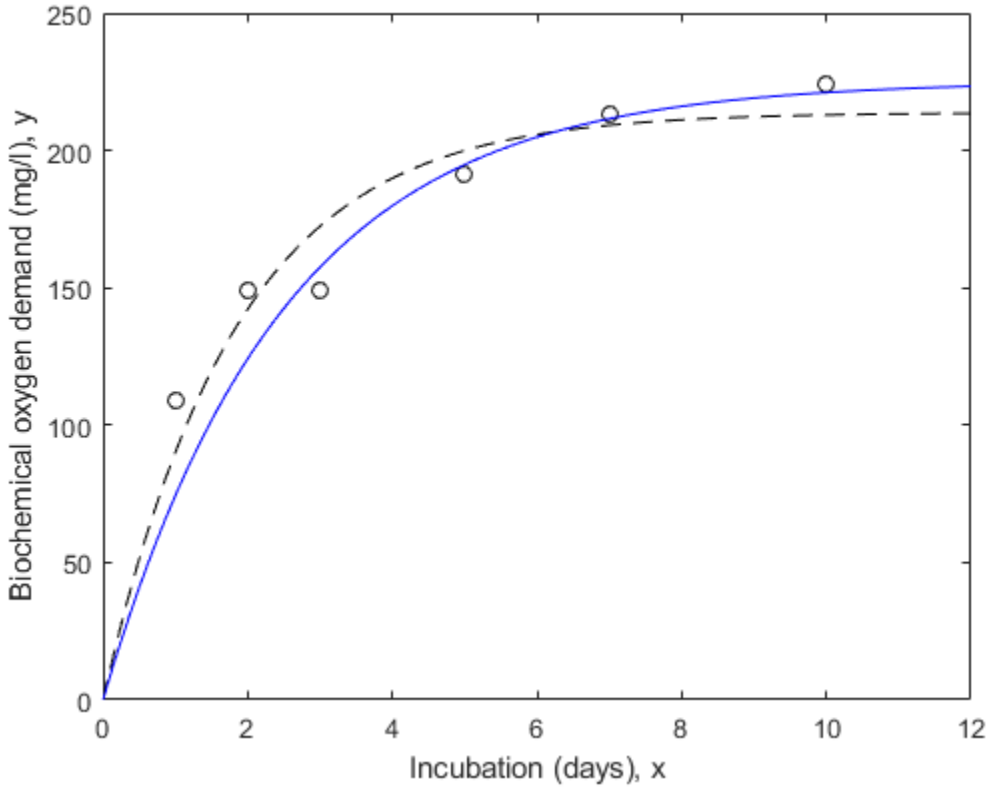
加权拟合模型

请注意，拟合曲线靠近前两个点，但似乎偏离了其他点的趋势。现在我们再尝试进行加权拟合。

```
wnlm = fitnlm(x,y,modelFun,start,'Weight',w)
line(xx,predict(wnlm,xx),'color','b')
```

```
wnlm =
```

```
Nonlinear regression model:
  y ~ b1*(1 - exp( - b2*x))

Estimated Coefficients:
       Estimate      SE      tStat      pValue
       ________    ________    _____    _________

   b1    225.17       10.7    21.045    3.0134e-05
   b2   0.40078    0.064296    6.2333     0.0033745


Number of observations: 6, Error degrees of freedom: 4
Root Mean Squared Error: 24
R-Squared: 0.908,  Adjusted R-Squared 0.885
F-statistic vs. zero model: 696, p-value = 8.2e-06
```



在这种情况下，估计的总体标准差描述了权重或测量精度为 1 的“标准”观测值的平均变化。

wnlm.RMSE

```
ans =

   24.0096
```

对于任何分析来说，估计模型拟合的精度都是重要的一部分。系数显示了参数的标准误差，但我们也可以计算参数的置信区间。

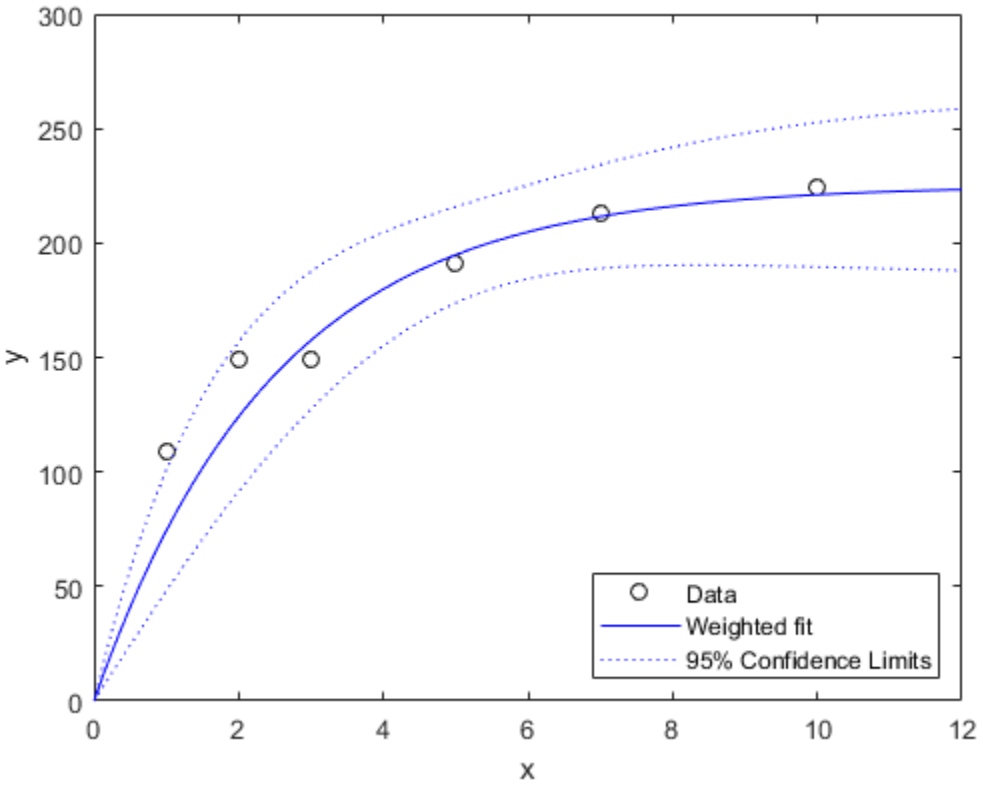
```
coefCI(wnlm)
```

```
ans =

  195.4650  254.8788
    0.2223    0.5793
```

估计响应曲线

接下来，我们将计算参数的拟合响应值和置信区间。默认情况下，这些宽度表示预测值的逐点置信边界，但我们将请求整个曲线的同时置信区间。

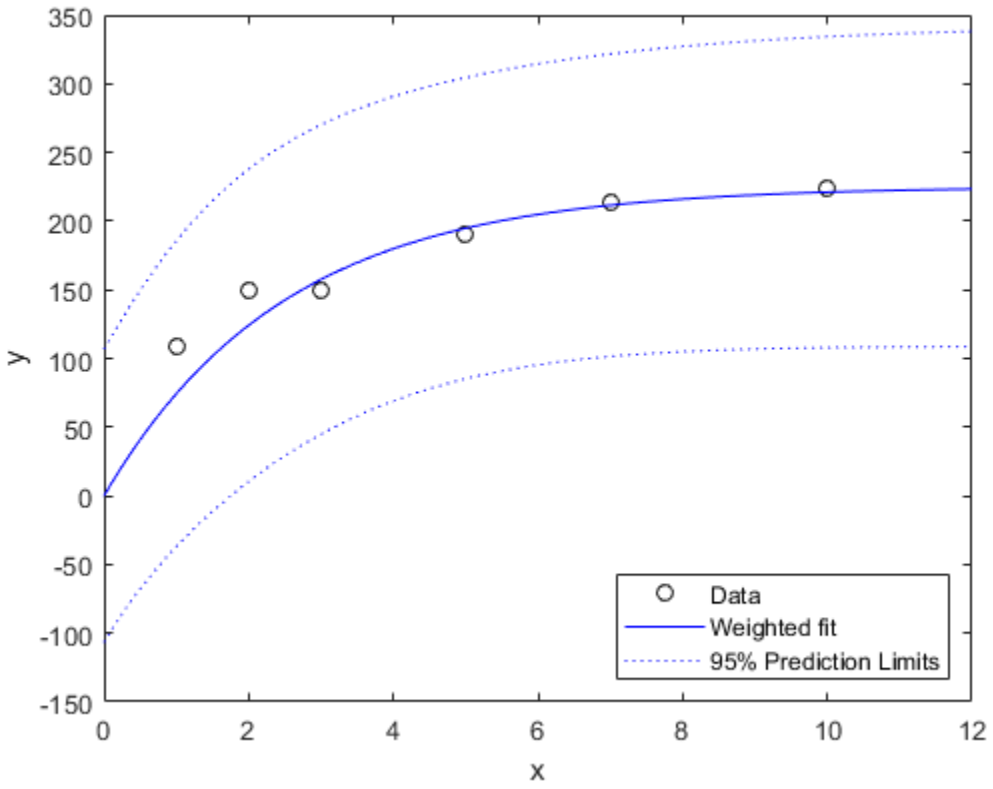
```
[ypred,ypredci] = predict(wnlm,xx,'Simultaneous',true);
plot(x,y,'ko', xx,ypred,'b-', xx,ypredci,'b:');
xlabel('x'); ylabel('y');
legend({'Data', 'Weighted fit', '95% Confidence Limits'},'location','SouthEast');
```



请注意，两个下加权点并没有像其他点一样很好地与曲线拟合。这符合您对加权拟合的预期。

您还可以估计以后在指定 x 值处所得观测值的预测区间。计算这些区间时，会假定权重或测量精度为 1。

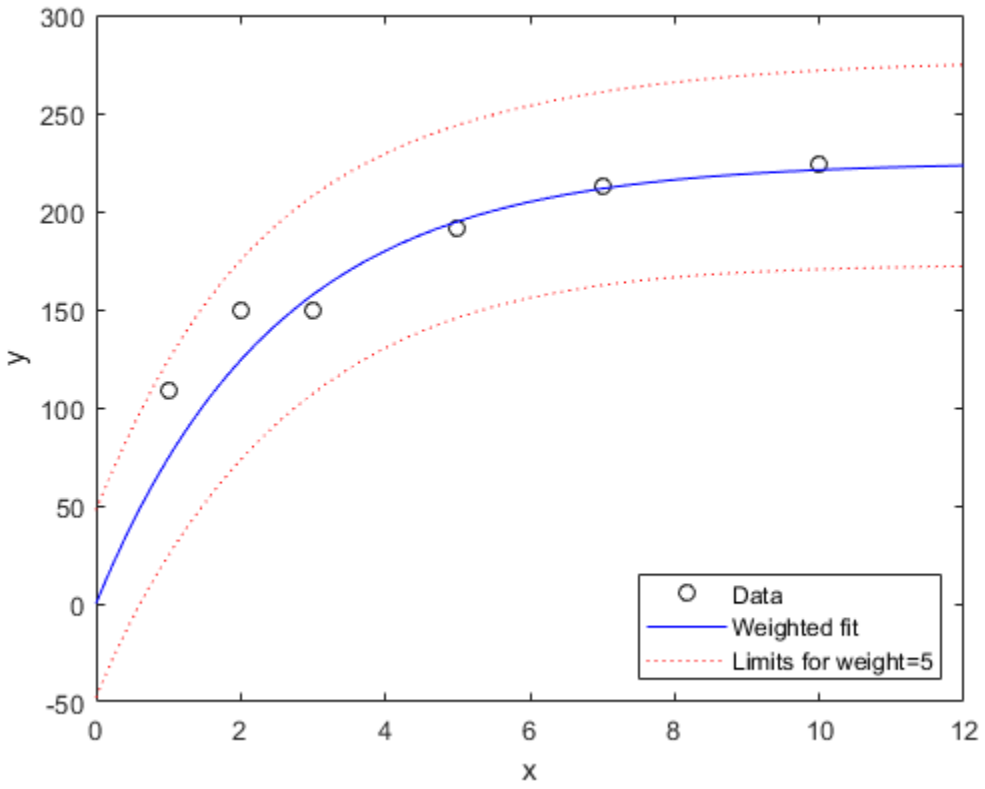
```
[ypred,ypredci] = predict(wnlm,xx,'Simultaneous',true,'Prediction','observation');
plot(x,y,'ko', xx,ypred,'b-', xx,ypredci,'b:');
xlabel('x'); ylabel('y');
legend({'Data', 'Weighted fit', '95% Prediction Limits'},'location','SouthEast');
```



权重的绝对尺度实际上不影响参数估计值。按任何常量重新调整权重会得到相同的估计值。但它们确实影响置信边界，因为边界代表权重为 1 的观测值。这里您可以看到，与置信界限相比，权重较大的点似乎太靠近拟合线。

虽然 **predict** 方法不允许我们更改权重，但我们可以进行一些后处理，并研究对于更精确的估计值，曲线的外观会如何。假设我们对一个新的观测值感兴趣，它基于五个测量值的均值，就像此图中后四个点一样。我们可以将区间宽度减小 sqrt(5) 分之一。

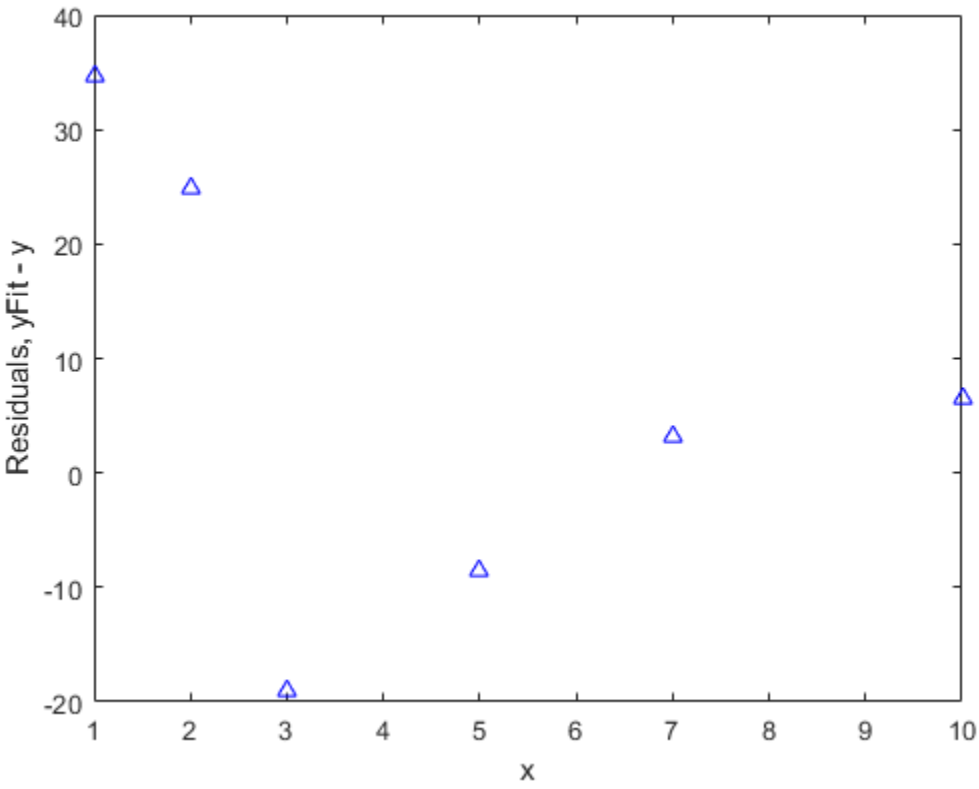
```
halfwidth = ypredci(:,2)-ypred;
newwidth = halfwidth/sqrt(5);
newci = [ypred-newwidth, ypred+newwidth];
plot(x,y,'ko', xx,ypred,'b-', xx,newci,'r:');
xlabel('x'); ylabel('y');
legend({'Data', 'Weighted fit', 'Limits for weight=5'},'location','SouthEast');
```



残差分析

除了绘制数据和拟合之外，我们还将根据预测变量绘制拟合残差，以诊断模型的任何问题。残差应该表现为独立同分布，但方差与权重的倒数成正比。我们可以将此方差归一化，使图更容易理解。

```
r = wnlm.Residuals.Raw;
plot(x,r.*sqrt(w),'b^');
xlabel('x'); ylabel('Residuals, yFit - y');
```



可以看出，此残差图存在一定的系统模式。请注意最后四个残差的线性趋势，这表明模型可能不会随着 x 增加而快速增加。而且，残差的模随 x 增大而减小，这表明测量误差可能取决于 x。这些特性值得进一步研究，但由于数据点太少，不足以确定这些明显模式的研究价值。

生存分析

多元方法

主成分分析 (PCA)

多元统计中的一个固有困难是可视化多变量数据的问题。函数 **plot** 显示两个变量之间的关系图。**plot3** 和 **surf** 命令显示不同三维视图。但是，当有三个以上的变量时，它们之间的关系就更难可视化。

幸运的是，在有许多变量的数据集中，变量组经常一起移动。其中一个原因是，可能有多个变量可用于衡量控制系统行为的同一个驱动原理。在许多系统中，这样的驱动力为数不多。但是，大量的检测可以让您测量几十个系统变量。当这种情况发生时，您可以利用信息的冗余。您可以通过用单个新变量替换一组变量来简化问题。

主成分分析以严格定量的方式来实现这种简化。该方法生成一组新变量，称为主成分。每个主成分是原始变量的线性组合。所有主成分相互正交，所以没有冗余信息。主成分作为一个整体构成了数据空间的一个正交基。

有无数种方法可以为几列数据构造正交基。主成分基有什么特别之处？

第一个主成分是空间中的单轴。当您将每个观测值投影到该轴上时，得到的值将形成一个新变量。此变量的方差是第一个轴的所有可能选择中的最大值。

第二个主成分是空间中的另一个轴，与第一个轴垂直。将观测值投影到此轴上会生成另一个新变量。此变量的方差是此第二个轴的所有可能选择中的最大值。

主成分的完整集合与原始变量的集合大小相同。但是，前几个主成分的方差之和通常会超过原始数据总方差的 80%。通过检查这几个新变量的图，研究人员通常会对生成原始数据的驱动因素有更深的理解。

您可以使用函数 **pca** 来查找主成分。要使用 **pca**，您需要有要分析的实际测量数据。但是，如果您缺少实际数据，但有数据的样本协方差或相关矩阵，您仍可以使用函数 **pcacov** 来执行主成分分析。有关其输入和输出的说明，请参阅 **pcacov** 的参考页。

另请参阅

pca | **pcacov** | **pcares** | **ppca**

相关示例

- “Analyze Quality of Life in U.S. Cities Using PCA”

选择用于高维数据分类的特征

此示例说明如何选择用于高维数据分类的特征。具体而言，示例说明如何执行序列特征选择，这是最常用的特征选择算法之一。示例还说明如何使用留出法和交叉验证来评估所选特征的分类性能。

减少特征（降低维度）数量在统计学习中很重要。对于许多具有大量特征和有限观测值的数据集（例如生物信息学数据）来说，往往有很多特征对实现理想的学习结果并无帮助，而有限的观测值又可能导致学习算法过拟合噪声。减少特征还可以节省存储空间和计算时间，并提高可解释性。

减少特征的方法主要有两种：特征选择和特征转换。特征选择算法从原始特征集中选择特征子集，特征转换方法将数据从原始高维特征空间转换到新的降维空间。

加载数据

血清蛋白质组图谱诊断可用于区分患者和非患者的观测值。图谱是使用表面增强激光解吸电离 (SELDI) 蛋白质质谱法产生的。这些特征是特定质荷比下的离子强度水平。

此示例用到了使用 WCX2 蛋白质阵列生成的高分辨率卵巢癌数据集。经过一些预处理步骤（类似于 Bioinformatics Toolbox™ 示例预处理原始质谱数据中所示的步骤），数据集现在包含两个变量 **obs** 和 **grp**。**obs** 变量包含 216 个观测值，具有 4000 个特征。**grp** 中的每个元素定义 **obs** 的对应行所属的组。

```
load ovariancancer;
whos
```

```
  Name      Size              Bytes  Class    Attributes

  grp     216x1              25056  cell
  obs     216x4000         3456000  single
```

将数据分成训练集和测试集

此示例中使用的一些函数调用 MATLAB® 内置随机数生成函数。要得到与此示例完全相同的结果，请执行以下命令，将随机数生成器设置为已知状态。否则，您的结果可能会有所不同。

```
rng(8000,'twister');
```

模型对训练数据的处理性能（再代入性能）并不能很好地反映模型对独立测试集的处理性能。再代入性能通常会过于乐观。要预测所选模型的性能，您需要使用另一个数据集（而不是用来构建模型的数据集）来评估模型性能。此处，我们使用 **cvpartition** 将数据分成大小为 160 的训练集和大小为 56 的测试集。训练集和测试集的组比例与 **grp** 中的组比例大致相同。我们使用训练数据选择特征，并使用测试数据判断所选特征的性能。这通常称为留出法验证。另一种简单但广泛使用的模型评估和选择方法是交叉验证，本示例稍后会进行说明。

```
holdoutCVP = cvpartition(grp,'holdout',56)
```

```
holdoutCVP =
Hold-out cross validation partition
   NumObservations: 216
      NumTestSets: 1
        TrainSize: 160
         TestSize: 56
```

```
dataTrain = obs(holdoutCVP.training,:);
grpTrain = grp(holdoutCVP.training);
```

使用全部特征对数据进行分类存在的问题

如果不先减少特征的数量，有些分类算法在处理本示例中使用的数据集时可能会失败，因为特征的数量远远大于观测值的数量。在本示例中，我们使用二次判别分析 (QDA) 作为分类算法。如果我们使用全部特征对数据应用 QDA，如下所示，我们将收到错误消息，因为每个组中没有足够的样本来估计协方差矩阵。

```
try
  yhat = classify(obs(test(holdoutCVP),:), dataTrain, grpTrain,'quadratic');
catch ME
  display(ME.message);
end
```

```
The covariance matrix of each group in TRAINING must be positive definite.
```

使用简单的筛选器方法选择特征

我们的目的是找出一小组能够实现良好分类性能的重要特征，从而减少数据的维度。特征选择算法可以大致分为两类：筛选器方法和封装器方法。筛选器方法依赖于数据的一般特性来评估和选择特征子集，而不涉及所选择的学习算法（此示例中为 QDA）。封装器方法利用所选学习算法的性能来评估每个候选特征子集。封装器方法会搜索更适合所选学习算法的特征，但如果学习算法需要很长时间运行，则封装器方法可能会比筛选器方法慢很多。有关“筛选器”和“封装器”的概念，请参阅以下文献：John G. Kohavi R. (1997) "Wrappers for feature subset selection", Artificial Intelligence, Vol.97, No.1-2, pp.272-324。本示例通过一个筛选器方法实例和一个封装器方法实例进行说明。

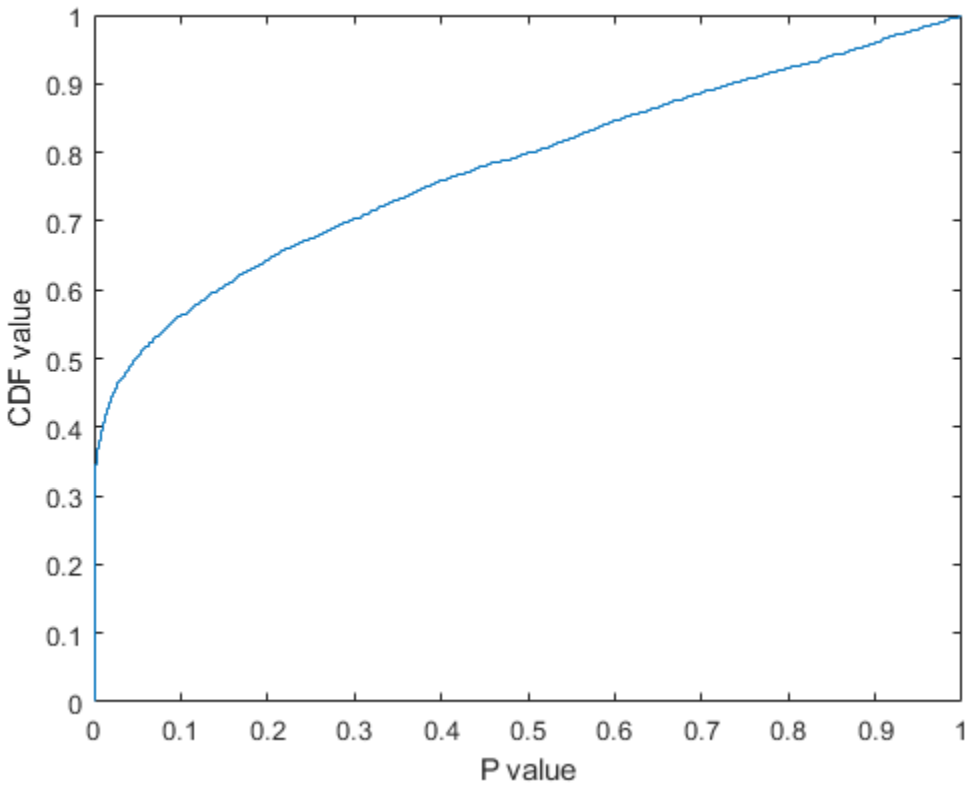
筛选器通常用作预处理步骤，因为它们简单快速。一种广泛用于生物信息学数据的筛选器方法是分别对每个特征应用单变量准则（假设特征之间没有交互作用）。

例如，我们可以对每个特征应用 t 检验，然后比较每个特征的 p 值（或 t 统计量的绝对值），以衡量该特征的分组效果如何。

```
dataTrainG1 = dataTrain(grp2idx(grpTrain)==1,:);
dataTrainG2 = dataTrain(grp2idx(grpTrain)==2,:);
[h,p,ci,stat] = ttest2(dataTrainG1,dataTrainG2,'Vartype','unequal');
```

为了大致了解每个特征区分两个组的效果，我们可以绘制 p 值的经验累积分布函数 (CDF)：

```
ecdf(p);
xlabel('P value');
ylabel('CDF value')
```



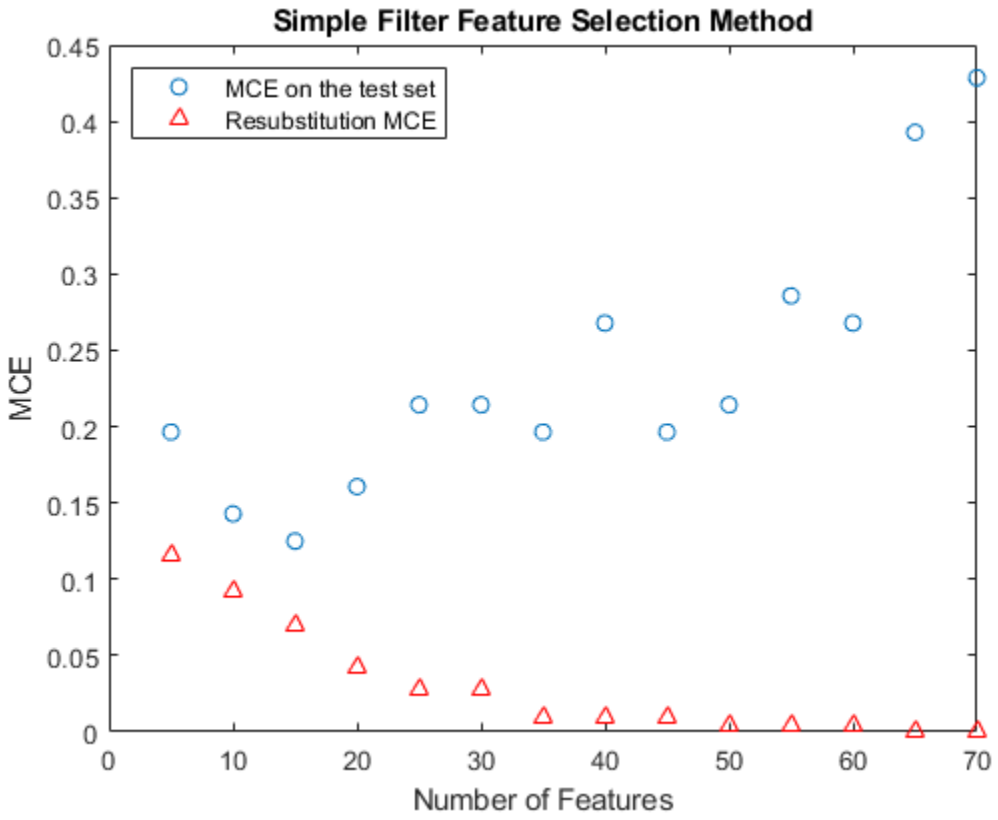
大约 35% 的特征的 p 值接近零，超过 50% 的特征的 p 值小于 0.05，这意味着 5000 个原始特征中有超过 2500 个特征具有强大的区分能力。您可以根据 p 值（或 t 统计量的绝对值）对这些特征进行排序，然后从排序的列表中选择一些特征。但是，除非您具备一定的专业知识，或者已经基于外部约束提前确定可以考虑的最大特征数，否则通常很难确定需要多少个特征。

要确定需要多少个特征，一种快速的方法是绘制测试集上的 MCE 对特征数的函数，MCE 是误分类误差，即误分类的观测值数除以观测值总数。由于训练集中只有 160 个观测值，因此应用 QDA 的最大特征数是有限的，否则，每个组中可能没有足够的样本来估计协方差矩阵。实际上，对于本示例中使用的数据，留出法分区和两个组的大小决定应用 QDA 的最大允许特征数约为 70。现在我们计算特征数介于 5 到 70 之间时的 MCE，并绘制 MCE 对特征数的函数。为了合理地估计所选模型的性能，应使用 160 个训练样本拟合 QDA 模型，并根据 56 个测试观测值（下图中的蓝色圆形标记）计算 MCE。为了说明再代入误差为什么不能很好地反映测试误差，我们同时用红色三角形标记显示再代入 MCE。

```
[~,featureIdxSortbyP] = sort(p,2); % sort the features
testMCE = zeros(1,14);
resubMCE = zeros(1,14);
nfs = 5:5:70;
classf = @(xtrain,ytrain,xtest,ytest) ...
        sum(~strcmp(ytest,classify(xtest,xtrain,ytrain,'quadratic')));
resubCVP = cvpartition(length(grp),'resubstitution')
```

```
resubCVP =
Resubstitution (no partition of data)
  NumObservations: 216
     NumTestSets: 1
       TrainSize: 216
        TestSize: 216
```

```
for i = 1:14
  fs = featureIdxSortbyP(1:nfs(i));
  testMCE(i) = crossval(classf,obs(:,fs),grp,'partition',holdoutCVP)...
     /holdoutCVP.TestSize;
  resubMCE(i) = crossval(classf,obs(:,fs),grp,'partition',resubCVP)/...
     resubCVP.TestSize;
end
 plot(nfs, testMCE,'o',nfs,resubMCE,'r^');
 xlabel('Number of Features');
 ylabel('MCE');
 legend({'MCE on the test set' 'Resubstitution MCE'},'location','NW');
 title('Simple Filter Feature Selection Method');
```



为方便起见，我们将 **classf** 定义为匿名函数。它将基于给定的训练集拟合 QDA，并返回基于给定测试集的误分类样本数。如果您开发自己的分类算法，则可能需要将其放在单独的文件中，如下所示：

```
%  function err = classf(xtrain,ytrain,xtest,ytest)
%       yfit = classify(xtest,xtrain,ytrain,'quadratic');
%        err = sum(~strcmp(ytest,yfit));
```

再代入 MCE 过于乐观。它随着所用特征数的增加而持续减少；当特征数超过 60 时，它会减少到零。但是，如果测试误差升高而再代入误差继续下降，则可能发生过拟合。当使用 15 个特征时，这种简单的筛选器特征选择方法获得了针对测试集的最小 MCE。该图显示，当使用的特征数等于或大于 20 时，将开始发生过拟合。针对测试集的最小 MCE 为 12.5%：

```
testMCE(3)
```

```
ans = 0.1250
```

下面是实现最小 MCE 的前 15 个特征：

```
featureIdxSortbyP(1:15)
```

```
ans = 1×15

    2814    2813    2721    2720    2452    2645    2644    2642    2650    2643    2731    2638    2
```

应用序列特征选择

上述特征选择算法不考虑特征之间的交互作用；此外，基于各个特征的排位从列表中选择的特征也可能包含冗余信息，因此并不需要全部特征。例如，选择的第一个特征（2814 列）和第二个特征（2813 列）之间的线性相关系数几乎为 0.95。

```
corr(dataTrain(:,featureIdxSortbyP(1)),dataTrain(:,featureIdxSortbyP(2)))
```

```
ans = single
   0.9447
```

这种简单的特征选择过程速度很快，因而通常用作预处理步骤。更高级的特征选择算法则用于提高性能。序列特征选择是使用最广泛的方法之一。它通过按顺序添加特征（前向搜索）或删除特征（后向搜索）来选择特征子集，直到满足某些停止条件为止。

在此示例中，我们以封装器方法使用前向序列特征选择来查找重要特征。具体而言，由于分类的目的通常是为了使 MCE 最小化，因此特征选择过程使用每个候选特征子集的学习算法 QDA 的 MCE，将其作为该子集的性能指标来执行序列搜索。训练集用于选择特征并拟合 QDA 模型，测试集用于评估最终选择的特征的性能。在特征选择过程中，为了评估和比较每个候选特征子集的性能，我们对训练集应用分层 10 折交叉验证。稍后将说明为什么要对训练集应用交叉验证。

首先，我们为训练集生成一个分层 10 折分区：

```
tenfoldCVP = cvpartition(grpTrain,'kfold',10)
```

```
tenfoldCVP =
K-fold cross validation partition
  NumObservations: 160
     NumTestSets: 10
       TrainSize: 144  144  144  144  144  144  144  144  144  144
        TestSize: 16  16  16  16  16  16  16  16  16  16
```

然后，我们将上一节中的筛选作为预处理步骤，使用其结果来选择特征。例如，我们在此选择 150 个特征：

```
fs1 = featureIdxSortbyP(1:150);
```

对这 150 个特征应用前向序列特征选择。函数 **sequentialfs** 提供一种简单的方法（默认选项）来决定需要多少个特征。它在找到交叉验证 MCE 的第一个局部最小值时停止。

```
fsLocal = sequentialfs(classf,dataTrain(:,fs1),grpTrain,'cv',tenfoldCVP);
```

所选特征如下：

```
fs1(fsLocal)
```

```
ans = 1×3

    2337     864    3288
```

为了评估具有这三个特征的所选模型的性能，我们基于 56 个测试样本计算 MCE。

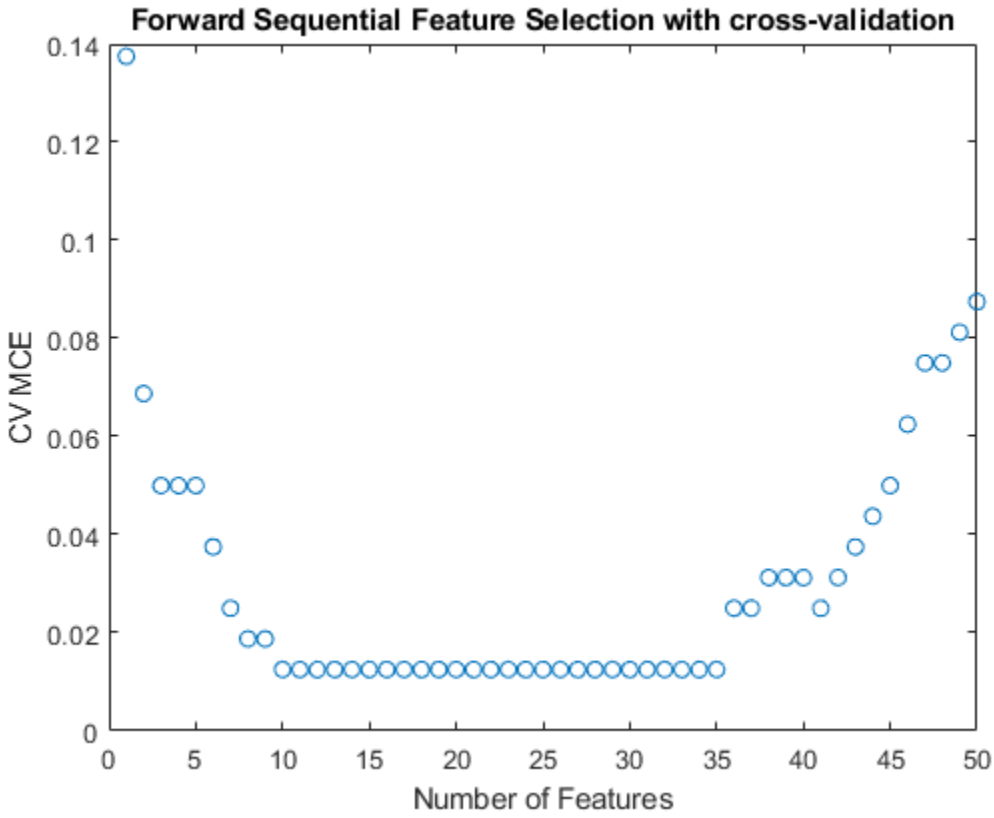
```
testMCELocal = crossval(classf,obs(:,fs1(fsLocal)),grp,'partition',...
   holdoutCVP)/holdoutCVP.TestSize
```

```
testMCELocal = 0.0714
```

因为只选择了三个特征，此 MCE 仅比使用简单筛选器特征选择方法的最小 MCE 的一半大一点。

算法可能过早停止。有时，通过在合理的特征数量范围内寻找交叉验证 MCE 的最小值，可以实现更小的 MCE。例如，我们绘制交叉验证 MCE 对特征数（最多 50 个）的函数。

```
[fsCVfor50,historyCV] = sequentialfs(classf,dataTrain(:,fs1),grpTrain,...
   'cv',tenfoldCVP,'Nf',50);
plot(historyCV.Crit,'o');
xlabel('Number of Features');
ylabel('CV MCE');
title('Forward Sequential Feature Selection with cross-validation');
```



当使用 10 个特征时，交叉验证 MCE 达到最小值，并且这条曲线在 10 到 35 个特征范围内保持平直。而且，当使用的特征超过 35 个时，曲线上升，意味着发生过拟合。

通常最好使用较少的特征，因此我们选择 10 个特征：

```
fsCVfor10 = fs1(historyCV.In(10,:))
```

```
fsCVfor10 = 1×10
```

```
    2814    2721    2720    2452    2650    2731    2337    2658     864    3288
```

要按照序列前向过程中选择特征的顺序显示这 10 个特征，我们在 **historyCV** 输出中找到它们首次变为 true 的行：

```
[orderlist,ignore] = find( [historyCV.In(1,:); diff(historyCV.In(1:10,:) )]' );
fs1(orderlist)
```

```
ans = 1×10

    2337     864    3288    2721    2814    2658    2452    2731    2650    2720
```

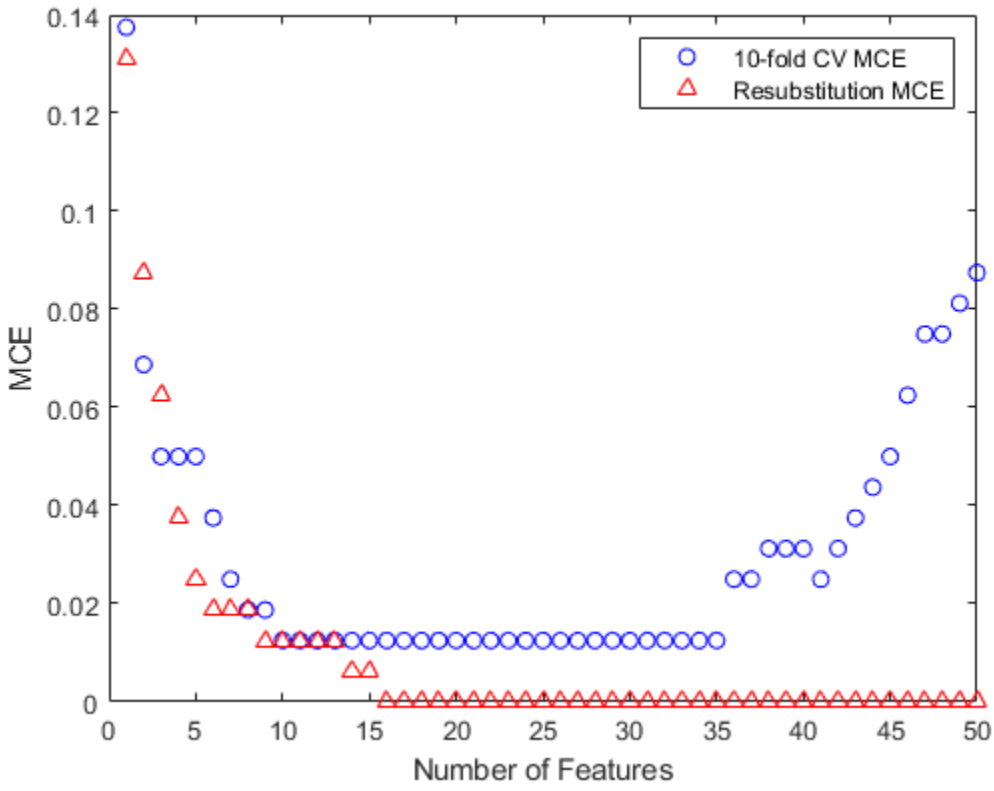
为了评估这 10 个特征，我们基于测试集计算这些特征的 QDA 的 MCE。我们获得到目前为止最小的 MCE 值：

```
testMCECVfor10 = crossval(classf,obs(:,fsCVfor10),grp,'partition',...
   holdoutCVP)/holdoutCVP.TestSize
```

```
testMCECVfor10 = 0.0357
```

我们不妨绘制训练集上的再代入 MCE 值（即，在特征选择过程中不执行交叉验证）对特征数的函数：

```
[fsResubfor50,historyResub] = sequentialfs(classf,dataTrain(:,fs1),...
    grpTrain,'cv','resubstitution','Nf',50);
plot(1:50, historyCV.Crit,'bo',1:50, historyResub.Crit,'r^');
xlabel('Number of Features');
ylabel('MCE');
legend({'10-fold CV MCE' 'Resubstitution MCE'},'location','NE');
```



这里的再代入 MCE 值同样过于乐观。其中大多数都小于交叉验证 MCE 值，且使用 16 个特征时，再代入 MCE 值开始变为零。我们可以计算这 16 个特征针对测试集的 MCE 值，以查看它们的真实性能：

```
fsResubfor16 = fs1(historyResub.In(16,:));
testMCEResubfor16 = crossval(classf,obs(:,fsResubfor16),grp,'partition',...
    holdoutCVP)/holdoutCVP.TestSize
```

```
testMCEResubfor16 = 0.0714
```

这 16 个特征（在特征选择过程中通过再代入选出）针对测试集的性能 **testMCEResubfor16** 大约是 10 个特征（在特征选择过程中通过 10 折交叉验证选出）针对测试集的性能 **testMCECVfor10** 的两倍。这再次表明，在评估和选择特征时，再代入误差往往并不能很好地反映性能。不管是在最终评估期间，还是在特征选择过程中，我们都要避免使用再代入误差。

另请参阅

sequentialfs

详细信息

- “Introduction to Feature Selection”
- “Sequential Feature Selection”

聚类分析

聚类分析

此示例说明如何使用 Statistics and Machine Learning Toolbox™ 中的聚类分析来检查观测值或对象的相似性和相异性。数据通常可以自然划分到观测值组或簇中，同一簇中对象的特性是相似的，不同簇中对象的特性是相异的。

K 均值和层次聚类

Statistics and Machine Learning Toolbox 中的一些函数可执行 K 均值聚类和层次聚类。

K 均值聚类是一种分区方法，它将数据中的观测值视为具有位置和相互间距离的对象。它将对象划分为 K 个互斥簇，使每个簇中的对象尽可能彼此靠近，并尽可能远离其他簇中的对象。每个簇的特性由其质心或中心点决定。当然，聚类中使用的距离通常不代表空间距离。

层次聚类是通过创建聚类树，同时在多个距离尺度内调查数据分组的一种方法。与 K-均值法不同，树并不是一组簇的简单组合，而是一个多级层次结构，较低级别的簇在相邻的更高级别合并成新的簇。使用这种方法，您可以选择最适合您的应用场景的聚类尺度或级别。

此示例中使用的一些函数调用 MATLAB® 内置随机数生成函数。要得到与此示例完全相同的结果，请执行以下命令，将随机数生成器设置为已知状态。如果您不设置状态，结果可能会略有不同，例如，簇可能会以不同的顺序编号。另外，有可能产生次优聚类解（本示例也讨论了次优解，以及避免次优解的方法）。

```
rng(6,'twister')
```

Fisher 鸢尾花数据

20 世纪 20 年代，植物学家收集了 150 个鸢尾花标本（三个品种各取 50 个标本）的萼片长度、萼片宽度、花瓣长度和花瓣宽度的测量值。这些测量值被称为 Fisher 鸢尾花数据集。

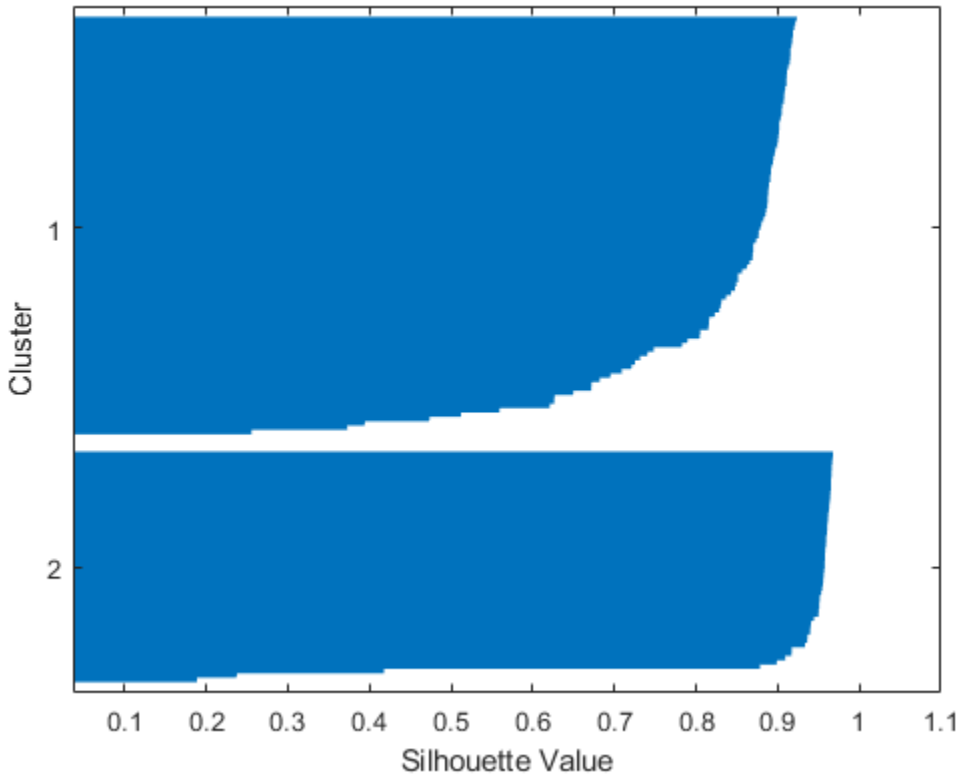
在此数据集中，每个观测值都来自一个已知的品种，因此已经有明显的方法对数据进行分组。现在，我们将忽略品种信息，仅使用原始测量值对数据进行聚类。完成后，我们可以将得到的簇与实际的品种进行比较，以了解这三种鸢尾花是否具有不同的特性。

使用 K 均值聚类方法对 Fisher 鸢尾花数据进行聚类

函数 **kmeans** 使用迭代算法执行 K 均值聚类，该算法将对象分配给簇，使每个对象到其所在簇质心的距离之和最小。对 Fisher 鸢尾花数据应用该函数时，它将根据鸢尾花标本萼片和花瓣的测量值找到它们的自然分组。使用 K 均值聚类，您必须指定要创建的簇数。

首先，加载数据并调用 **kmeans**，将所需的簇数设置为 2，并使用平方欧几里德距离。要了解生成的簇区分数据的效果，可以绘制轮廓图。轮廓图显示一个簇中的每个点与相邻簇中的点的接近程度。

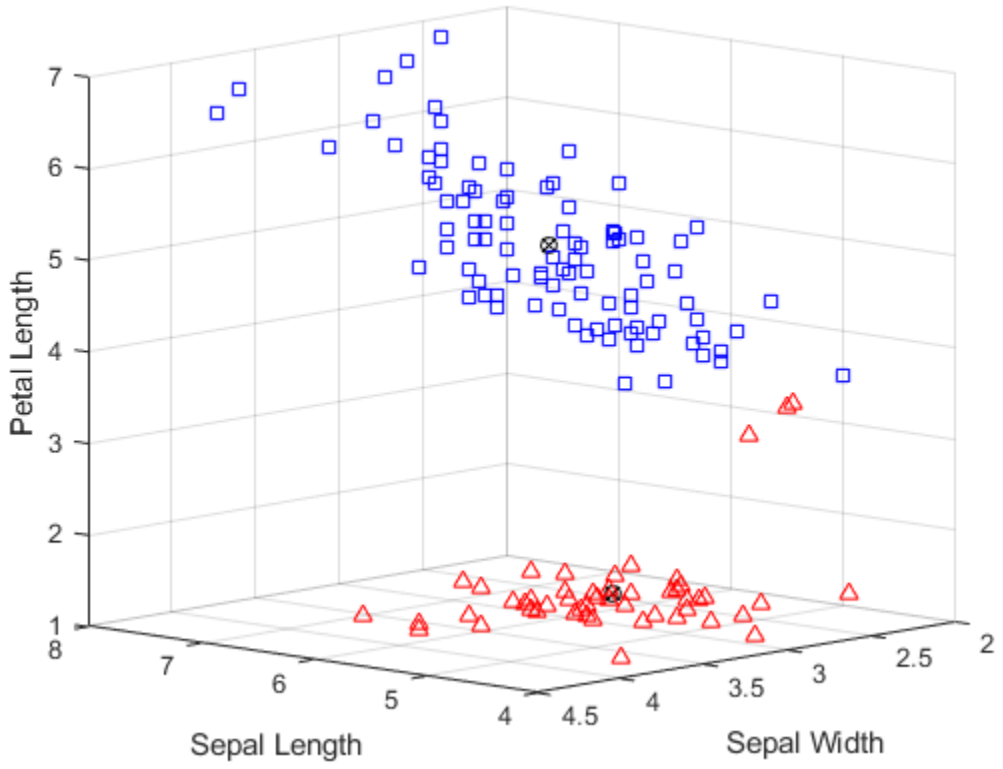
```
load fisheriris
[cidx2,cmeans2] = kmeans(meas,2,'dist','sqeuclidean');
[silh2,h] = silhouette(meas,cidx2,'sqeuclidean');
```



从轮廓图中可以看出，两个簇中的大多数点都具有较大的轮廓值（大于 0.8），表明这些点可以与相邻簇很好地区分。但是，每个簇中还有一些点的轮廓值较低，表示它们靠近其他簇中的点。

事实证明，这些数据中的第四个测量值（花瓣宽度）与第三个测量值（花瓣长度）高度相关，因此前三个测量值的三维图可以很好地表示数据，而无需求助于四个维度。如果绘制数据，则使用不同的符号表示 **kmeans** 创建的每个簇，就可以识别轮廓值较小的点，也就是与其他簇中的点相距较近的点。

```
ptsymb = {'bs','r^','md','go','c+'};
for i = 1:2
   clust = find(cidx2==i);
   plot3(meas(clust,1),meas(clust,2),meas(clust,3),ptsymb{i});
   hold on
end
plot3(cmeans2(:,1),cmeans2(:,2),cmeans2(:,3),'ko');
plot3(cmeans2(:,1),cmeans2(:,2),cmeans2(:,3),'kx');
hold off
xlabel('Sepal Length');
ylabel('Sepal Width');
zlabel('Petal Length');
view(-137,10);
grid on
```



每个簇的质心用带圆圈的 X 表示。下方簇中用三角形表示的点中，有三个非常靠近上方簇中用正方形表示的点。由于上方簇相当分散，这三个点更靠近下方簇的质心而不是上方簇的质心，虽然这些点与所在簇中的大部分点存在明显距离。由于 K 均值聚类只考虑距离而不是密度，因此可能会出现这种结果。

您可以增加簇数以查看 **kmeans** 是否可以在数据中找到进一步的分组结构。这次，使用可选的 **'Display'** 名称-值对组参数显示聚类算法中有关每次迭代的信息。

```
[cidx3,cmeans3] = kmeans(meas,3,'Display','iter');
```

```
  iter    phase      num         sum
    1       1       150       146.424
    2       1         5       144.333
    3       1         4       143.924
    4       1         3        143.61
    5       1         1       143.542
    6       1         2       143.414
    7       1         2       143.023
    8       1         2       142.823
    9       1         1       142.786
   10       1         1       142.754
Best total sum of distances = 142.754
```

在每次迭代时，**kmeans** 算法（请参阅 “算法” （第 33-131 页））在簇之间重新分配点，以减小点到质心距离的总和，然后重新计算新簇分配的簇质心。请注意，距离总和与重新分配次数随着每次迭代而减小，直到算法达到最小值。**kmeans** 中使用的算法由两个阶段组成。在此示例中，算法的第二阶段未进行任何重新分配，表明第一阶段在进行几次迭代后就达到了最小值。

默认情况下，**kmeans** 使用随机选择的一组初始质心位置开始聚类过程。**kmeans** 算法会收敛于作为局部最小值的解；也就是说，**kmeans** 可对数据进行分区，使得将任一点移到其他簇都会增加距离总和。但是，同许多其他类型的数值最小化问题一样，**kmeans** 达到的解有时取决于起点。因此，该数据可能存在其他具有更小距离总和的解（局部最小值）。您可以使用可选的 **'Replicates'** 名称-值对组参数来检验不同的解。指定多次重复时，**kmeans** 会在每次重复时从不同的随机选择质心重新开始聚类过程。**kmeans** 随后返回所有重复中距离总和最小的解。

```
[cidx3,cmeans3,sumd3] = kmeans(meas,3,'replicates',5,'display','final');
```

```
Replicate 1, 9 iterations, total sum of distances = 78.8557.
Replicate 2, 10 iterations, total sum of distances = 78.8557.
Replicate 3, 8 iterations, total sum of distances = 78.8557.
Replicate 4, 8 iterations, total sum of distances = 78.8557.
Replicate 5, 1 iterations, total sum of distances = 78.8514.
Best total sum of distances = 78.8514
```

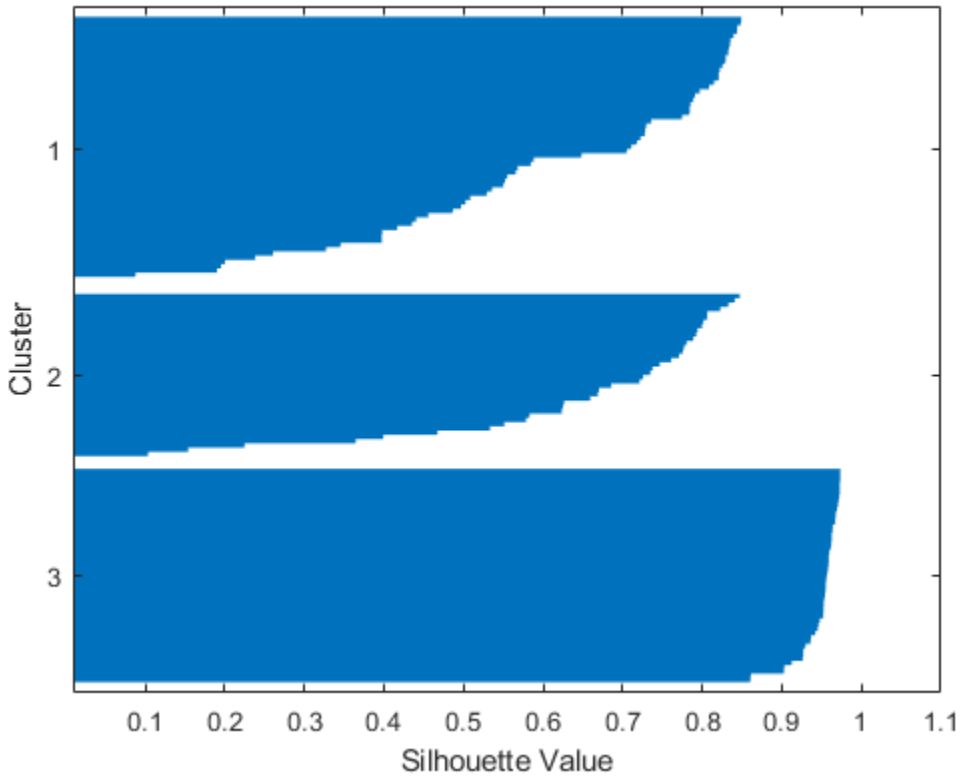
输出表明，即使对于这样相对简单的问题，也存在非全局最小值。这五次重复分别从一组不同的初始质心开始。根据起始位置，**kmeans** 达到两个不同的解之一。但是，**kmeans** 最终返回的解是所有重复中距离总和最小的解。第三个输出参数包含每个簇中对于该最佳解的距离总和。

```
sum(sumd3)
```

```
ans =

   78.8514
```

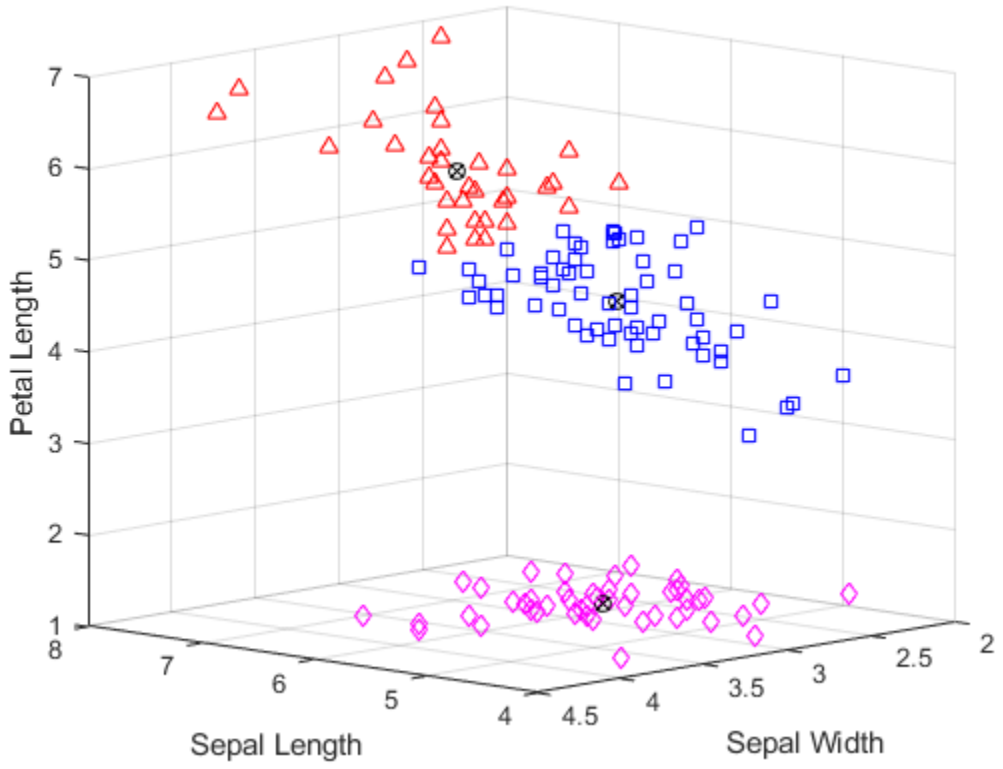
该三簇解的轮廓图显示，有一个簇区分效果很好，但另外两个簇的区别不是很明显。

```
[silh3,h] = silhouette(meas,cidx3,'sqeuclidean');
```



同样，您可以绘制原始数据以查看 **kmeans** 如何将点分配给簇。

```
for i = 1:3
   clust = find(cidx3==i);
   plot3(meas(clust,1),meas(clust,2),meas(clust,3),ptsymb{i});
   hold on
end
plot3(cmeans3(:,1),cmeans3(:,2),cmeans3(:,3),'ko');
plot3(cmeans3(:,1),cmeans3(:,2),cmeans3(:,3),'kx');
hold off
xlabel('Sepal Length');
ylabel('Sepal Width');
zlabel('Petal Length');
view(-137,10);
grid on
```



您可以看到，**kmeans** 已将双簇解中的上方簇一分为二，而且这两个簇非常接近。跟之前的双簇解相比，这个三簇解是否更为有用取决于您在聚类后要对数据进行的操作。**silhouette** 的第一个输出参数包含每个点的轮廓值，您可以使用这些值对两个解进行定量比较。双簇解的平均轮廓值更大，表明就簇的区分效果而言，这是更好的解。

```
[mean(silh2) mean(silh3)]
```

```
ans =

   0.8504    0.7357
```

您还可以使用不同距离对这些数据进行聚类。余弦距离可能适用于这些数据，因为它会忽略测量值的绝对大小，只考虑相对大小。因此，对于两朵大小不同但花瓣和萼片的形状相似的花，使用平方欧几里德距离可能不接近，但使用余弦距离则会接近。

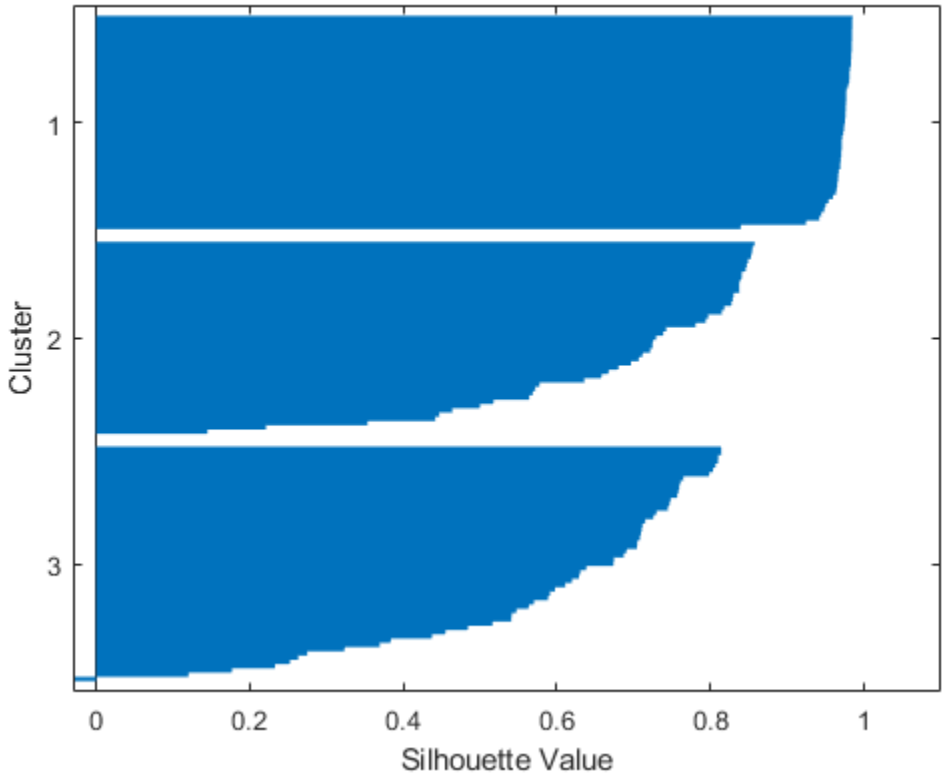
```
[cidxCos,cmeansCos] = kmeans(meas,3,'dist','cos');
```

从轮廓图中可以看出，这些簇的区分效果好像只比使用平方欧几里德距离时稍好一点。

```
[silhCos,h] = silhouette(meas,cidxCos,'cos');
[mean(silh2) mean(silh3) mean(silhCos)]
```

```
ans =
```

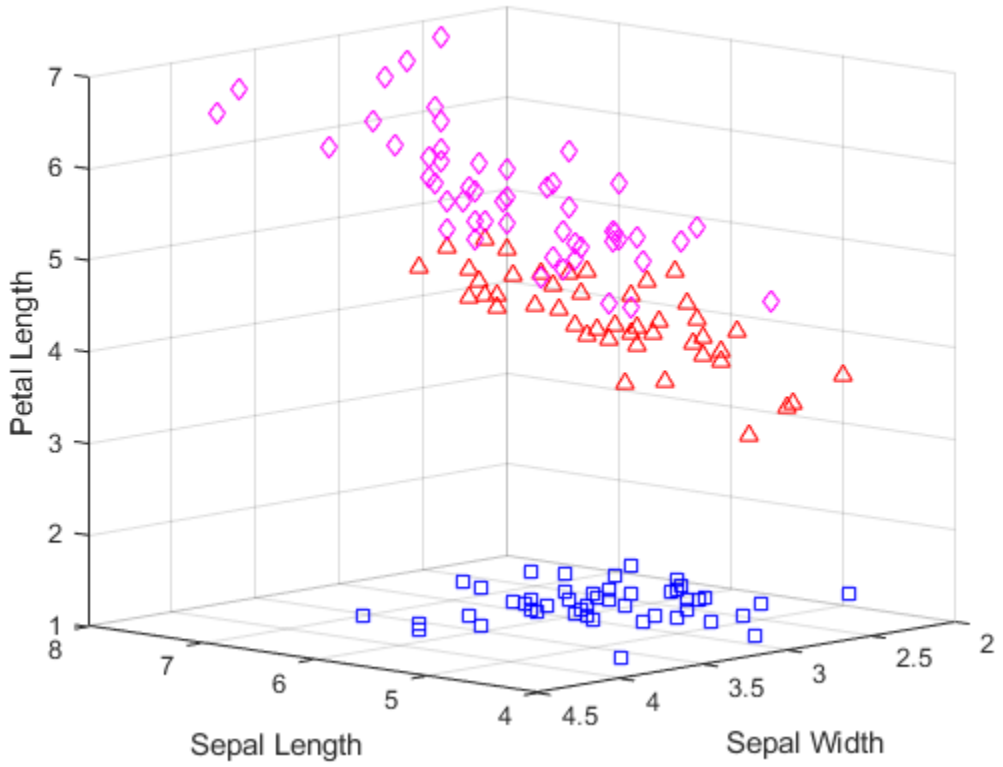
```
    0.8504    0.7357    0.7491
```



请注意，簇的顺序与之前的轮廓图不一样。这是因为 **kmeans** 会随机选择初始簇分配。

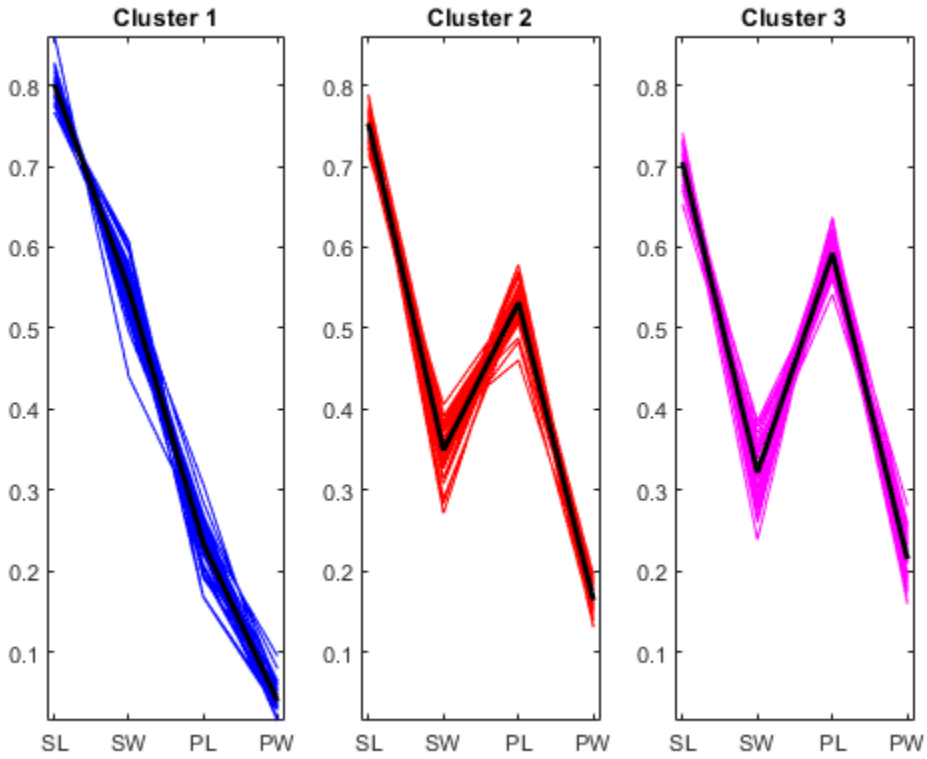
通过绘制原始数据，您可以看到使用两种不同距离创建的簇在形状上的差异。两个解较为相似，但使用余弦距离时，上方的两个簇向原点拉伸。

```
for i = 1:3
   clust = find(cidxCos==i);
   plot3(meas(clust,1),meas(clust,2),meas(clust,3),ptsymb{i});
   hold on
end
hold off
xlabel('Sepal Length');
ylabel('Sepal Width');
zlabel('Petal Length');
view(-137,10);
grid on
```



此图不包括簇质心，因为使用余弦距离时，质心是以原始数据空间原点为起点的射线。但是，您可以绘制归一化数据点的平行坐标图，以直观地了解簇质心的差异。

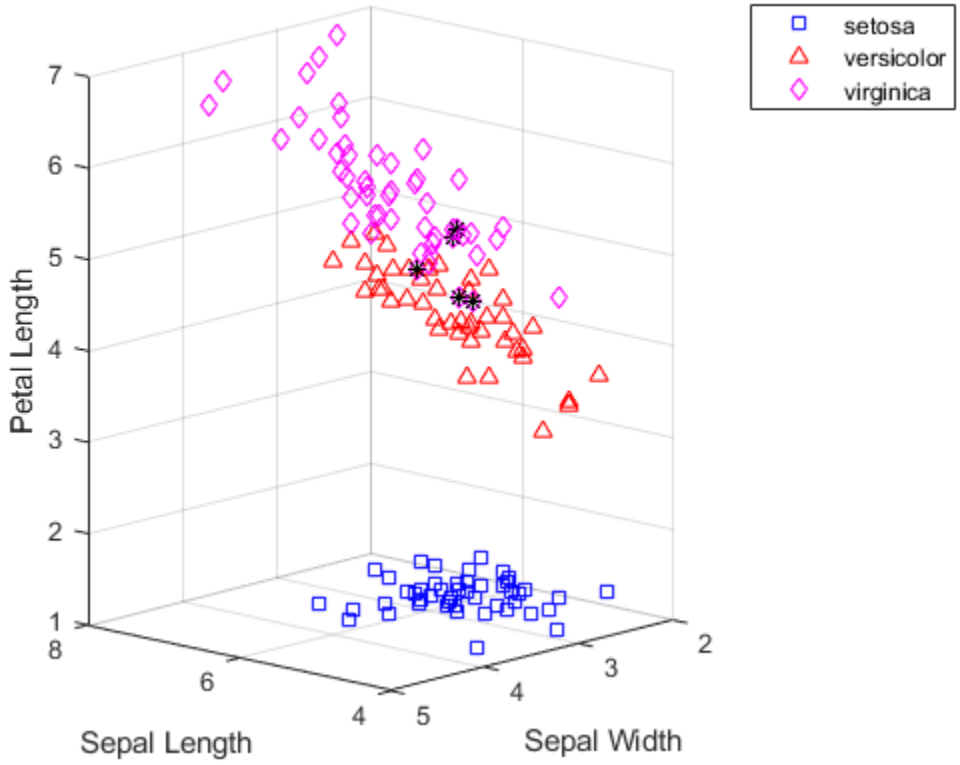
```
lnsymb = {'b-','r-','m-'};
names = {'SL','SW','PL','PW'};
meas0 = meas ./ repmat(sqrt(sum(meas.^2,2)),1,4);
ymin = min(min(meas0));
ymax = max(max(meas0));
for i = 1:3
   subplot(1,3,i);
   plot(meas0(cidxCos==i,:)',lnsymb{i});
   hold on;
   plot(cmeansCos(i,:)','k-','LineWidth',2);
   hold off;
   title(sprintf('Cluster %d',i));
   xlim([.9, 4.1]);
   ylim([ymin, ymax]);
   h_gca = gca;
   h_gca.XTick = 1:4;
   h_gca.XTickLabel = names;
end
```



从图中可以清楚地看出，三个簇所包含的标本在花瓣和萼片的平均相对大小方面明显不同。第一个簇的花瓣严格小于萼片。后面两个簇的花瓣和萼片的大小重叠，但第三个簇的重叠超过第二个簇。您还可以看到，第二个和第三个簇包含一些非常相似的标本。

因为我们知道数据中每个观测值所属的品种，所以可以将 **kmeans** 发现的簇与实际品种进行比较，以了解这三个品种是否具有明显不同的物理特性。实际上，如下图所示，使用余弦距离创建的簇与实际品种分组的不同之处只有五朵花。这五个点（用星形标记绘制）都靠近上方两个簇的边缘交界处。

```
subplot(1,1,1);
for i = 1:3
   clust = find(cidxCos==i);
   plot3(meas(clust,1),meas(clust,2),meas(clust,3),ptsymb{i});
   hold on
end
xlabel('Sepal Length');
ylabel('Sepal Width');
zlabel('Petal Length');
view(-137,10);
grid on
sidx = grp2idx(species);
miss = find(cidxCos ~= sidx);
plot3(meas(miss,1),meas(miss,2),meas(miss,3),'k*');
legend({'setosa','versicolor','virginica'});
hold off
```



使用层次聚类方法对 Fisher 鸢尾花数据进行聚类

K 均值聚类只产生一个鸢尾花数据分区，但您可能还想按照不同的分组尺度来研究数据。要实现这一点，您可以通过层次聚类创建层次聚类树。

首先，使用鸢尾花数据中观测值之间的距离创建一个聚类树。先使用欧几里德距离。

```
eucD = pdist(meas,'euclidean');
clustTreeEuc = linkage(eucD,'average');
```

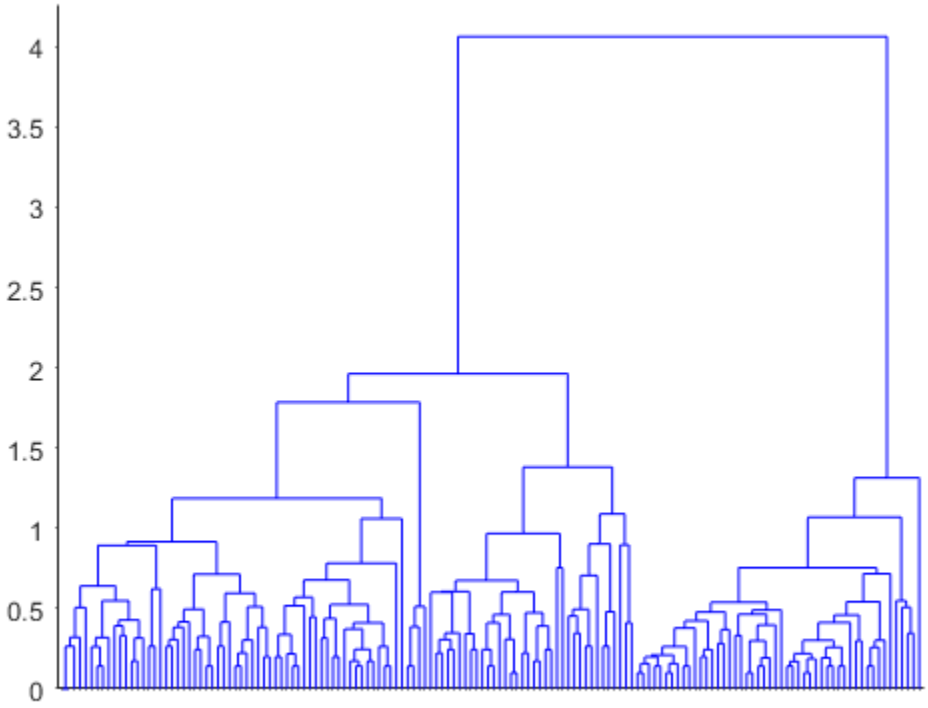
共性相关是验证聚类树与原始距离是否一致的一种方法。较大的值表明树对距离的拟合良好，即观测值之间的两两连接与它们实际上的两两距离是相关的。此树对距离的拟合看起来相当好。

```
cophenet(clustTreeEuc,eucD)
```

```
ans =

    0.8770
```

要可视化聚类的层次结构，可以绘制树状图。

```
[h,nodes] = dendrogram(clustTreeEuc,0);
h_gca = gca;
h_gca.TickDir = 'out';
h_gca.TickLength = [.002 0];
h_gca.XTickLabel = [];
```



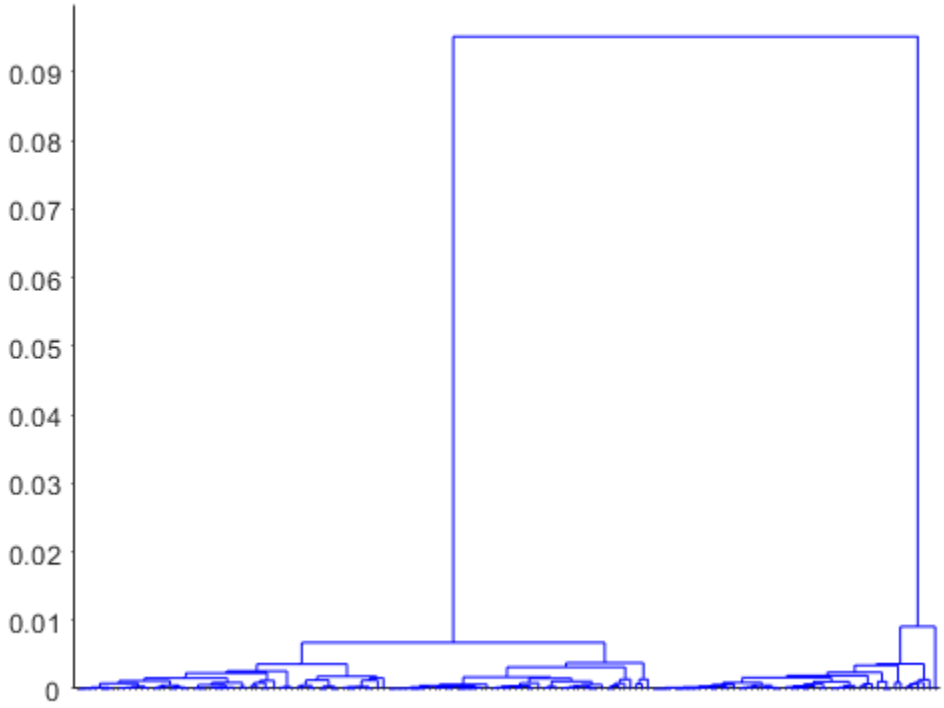
树中的根节点远远高于其余节点，证实了您在 K 均值聚类中所看到的：两个很大的、区别明显的观测值组。在每个组中可以看到，随着距离尺度越来越小，较低级别的组逐渐出现。有许多级别不同、大小不同、区别度不同的组。

根据 K 均值聚类的结果，我们也不妨使用余弦来测度距离。由此生成的层次结构树很不一样，让我们能够以一种完全不同的方式观察鸢尾花数据的组结构。

```
cosD = pdist(meas,'cosine');
clustTreeCos = linkage(cosD,'average');
cophenet(clustTreeCos,cosD)
```

```
ans =

    0.9360
```

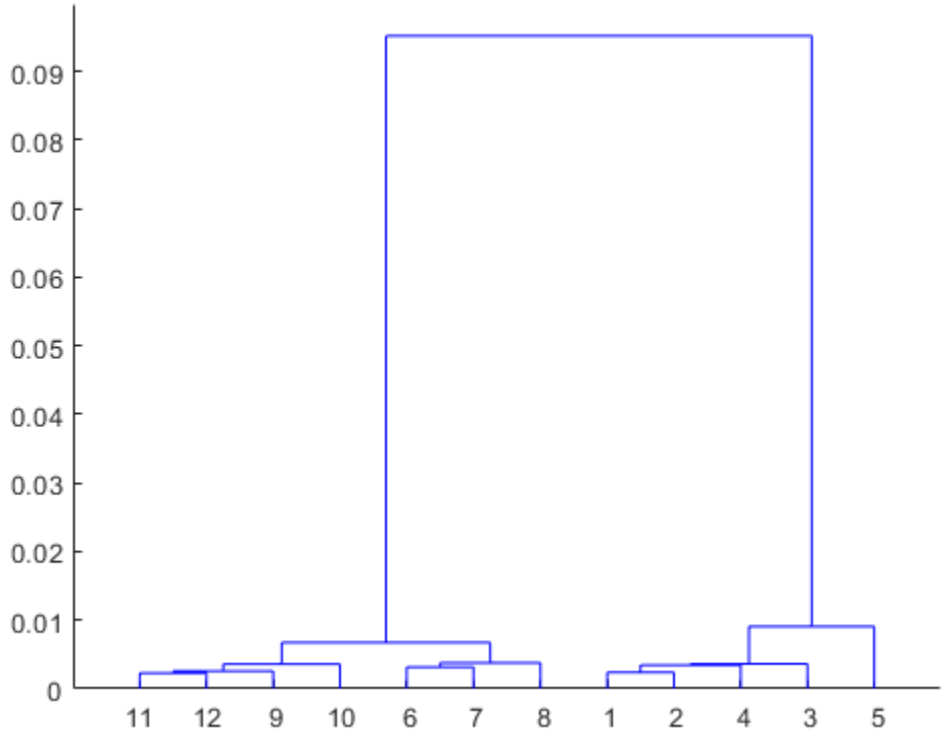
```
[h,nodes] = dendrogram(clustTreeCos,0);
h_gca = gca;
h_gca.TickDir = 'out';
h_gca.TickLength = [.002 0];
h_gca.XTickLabel = [];
```



此树的最高级别将鸢尾花标本分成两个区别很大的组。树状图显示，使用余弦距离，相对于组间差异的组内差异远小于使用欧几里德距离的情况。这正是您期望的数据结果，因为对于从原点出发的相同“方向”上的对象，余弦距离将其两两距离计为零。

包含 150 个观测值的图较为杂乱，但您可以制作简化的树状图，不显示树中靠近底部的级别。

```
[h,nodes] = dendrogram(clustTreeCos,12);
```



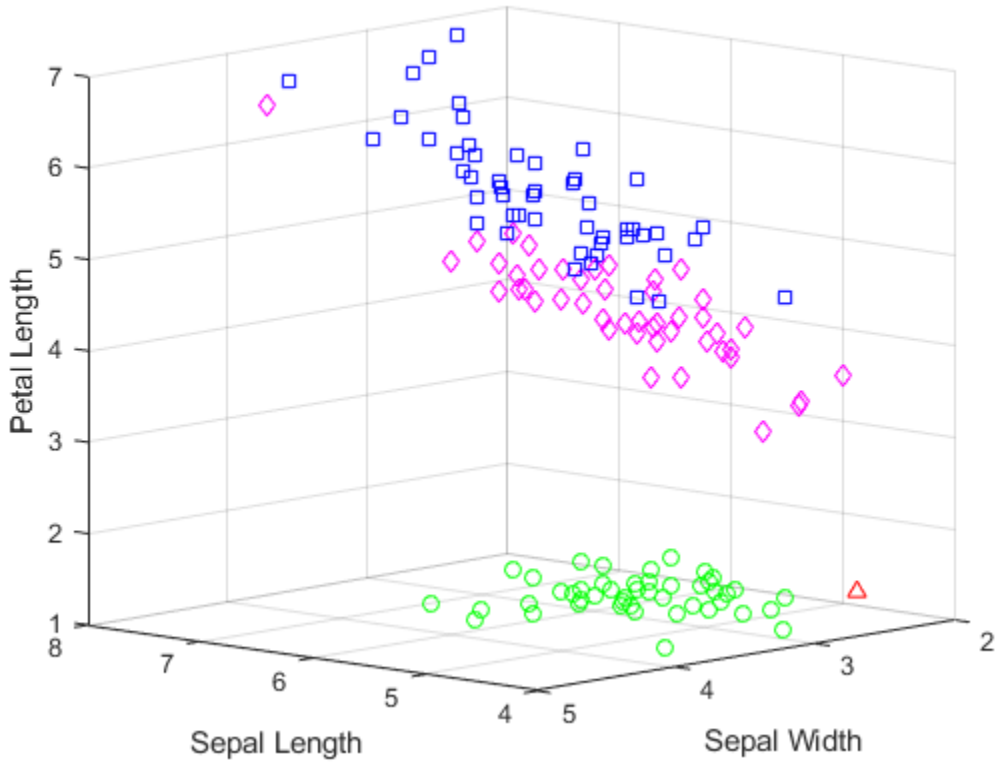
此树中的三个最高节点划分出三个大小相同的组，以及一个与任何其他组都不相近的标本（标记为叶节点 5）。

```
[sum(ismember(nodes,[11 12 9 10])) sum(ismember(nodes,[6 7 8])) ...
          sum(ismember(nodes,[1 2 4 3])) sum(nodes==5)]
```

```
ans =

   54   46   49    1
```

在很多情况下，树状图结果或许已能满足分类要求。但是，您可以更进一步，使用 **cluster** 函数来裁剪树并将观测值显式划分为特定的聚类，就像使用 K 均值一样。根据使用余弦距离的层次结构创建聚类，指定连接高度以在三个最高节点下方裁剪树，创建四个聚类，然后绘制经过聚类的原始数据。

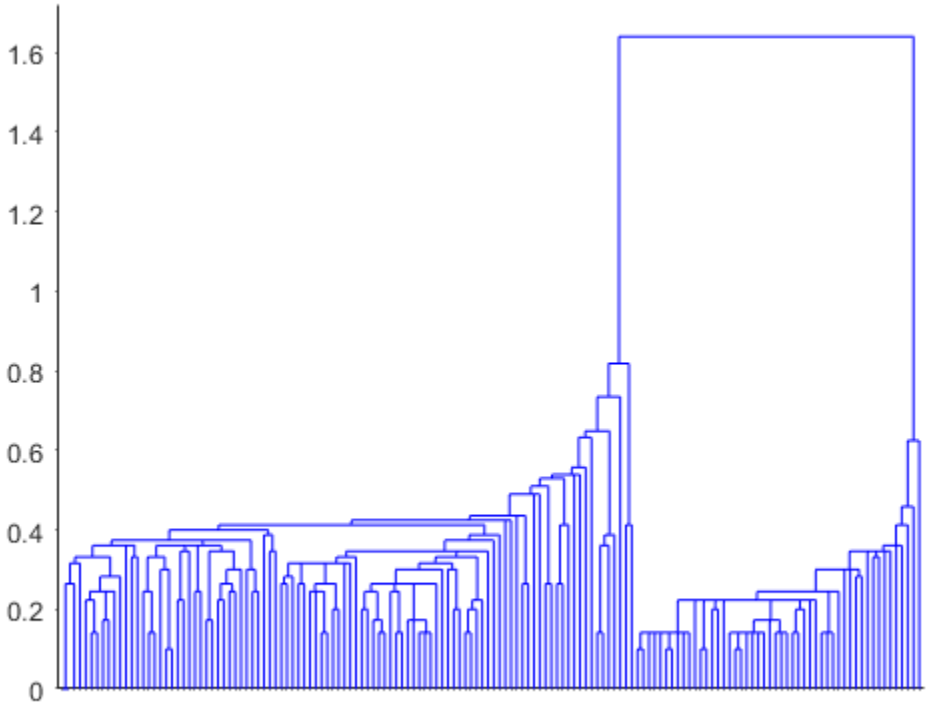
```
hidx = cluster(clustTreeCos,'criterion','distance','cutoff',.006);
for i = 1:5
   clust = find(hidx==i);
   plot3(meas(clust,1),meas(clust,2),meas(clust,3),ptsymb{i});
   hold on
end
hold off
xlabel('Sepal Length');
ylabel('Sepal Width');
zlabel('Petal Length');
view(-137,10);
grid on
```



此图显示，就其性质而言，使用余弦距离的层次聚类结果与三簇 K 均值聚类的结果类似。但是，创建层次聚类树可以一次性可视化所有结果，而在 K 均值聚类中需要使用不同的 K 值进行大量试验。

层次聚类还允许您使用不同的连接进行试验。例如，相比基于平均距离连接聚类，基于单连接对鸢尾花数据进行聚类时，往往将距离更大的对象连接在一起，从而给出对数据结构的另一种解释。

```
clustTreeSng = linkage(eucD,'single');
[h,nodes] = dendrogram(clustTreeSng,0);
h_gca = gca;
h_gca.TickDir = 'out';
h_gca.TickLength = [.002 0];
h_gca.XTickLabel = [];
```


1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0

参数化分类

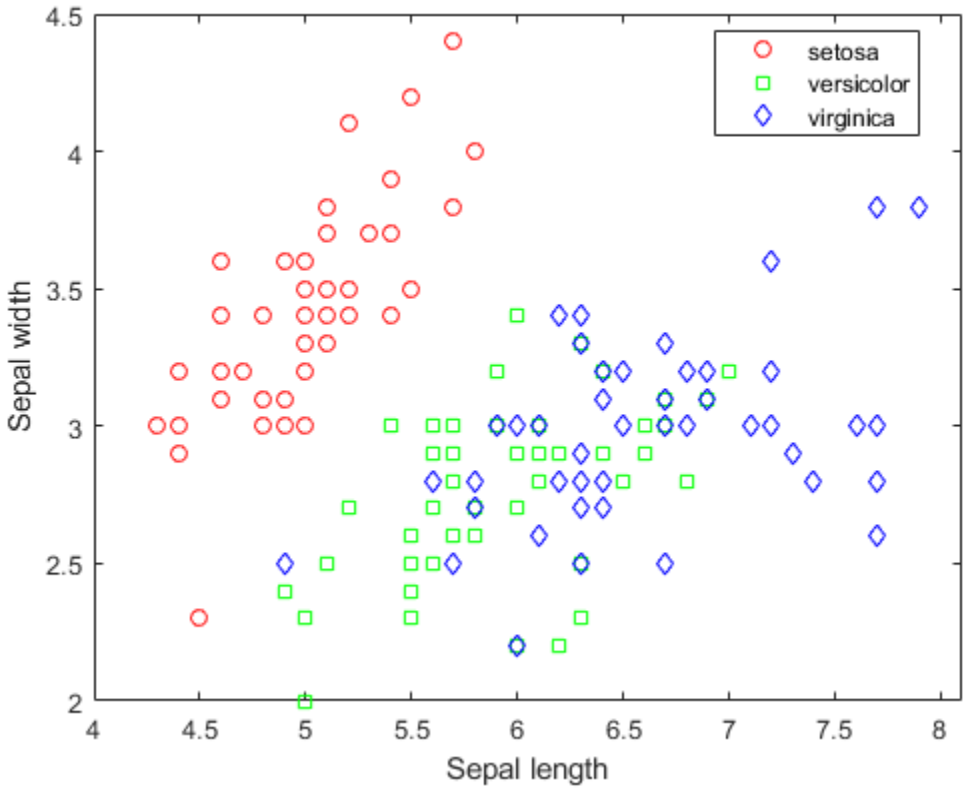
分类

此示例说明如何使用判别分析、朴素贝叶斯分类器和决策树进行分类。假设您有一个数据集，其中包含由不同变量（称为预测变量）的测量值组成的观测值，以及这些观测值的已知类标签。如果得到新观测值的预测变量值，您能判断这些观测值可能属于哪些类吗？这就是分类问题。

Fisher 鸢尾花数据

Fisher 鸢尾花数据包括 150 个鸢尾花标本的萼片长度、萼片宽度、花瓣长度和花瓣宽度的测量值。三个品种各有 50 个标本。加载数据，查看萼片测量值在不同品种间有何差异。您可以使用包含萼片测量值的两列。

```
load fisheriris
f = figure;
gscatter(meas(:,1), meas(:,2), species,'rgb','osd');
xlabel('Sepal length');
ylabel('Sepal width');
N = size(meas,1);
```



假设您测量了一朵鸢尾花的萼片和花瓣，并且需要根据这些测量值确定它所属的品种。解决此问题的一种方法称为判别分析。

线性判别分析和二次判别分析

fitcdiscr 函数可以使用不同类型的判别分析进行分类。首先使用默认的线性判别分析 (LDA) 对数据进行分类。

```
lda = fitcdiscr(meas(:,1:2),species);
ldaClass = resubPredict(lda);
```

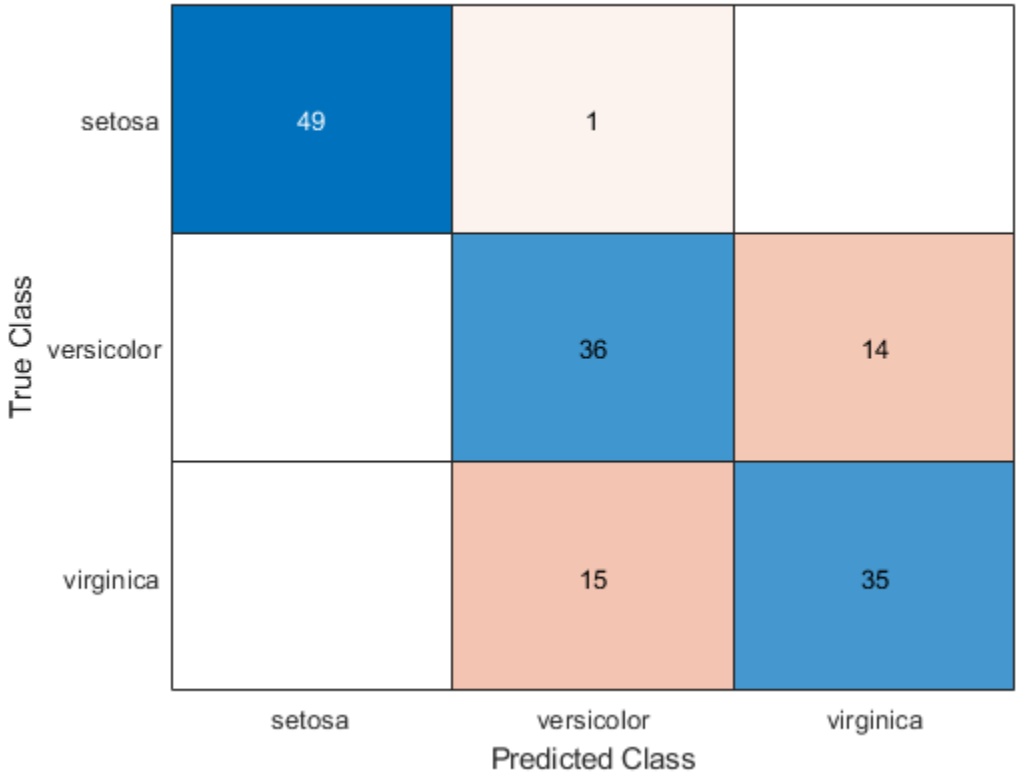
带有已知类标签的观测值通常称为训练数据。现在计算再代入误差，即针对训练集的误分类误差（误分类的观测值所占的比例）。

```
ldaResubErr = resubLoss(lda)
```

```
ldaResubErr =

    0.2000
```

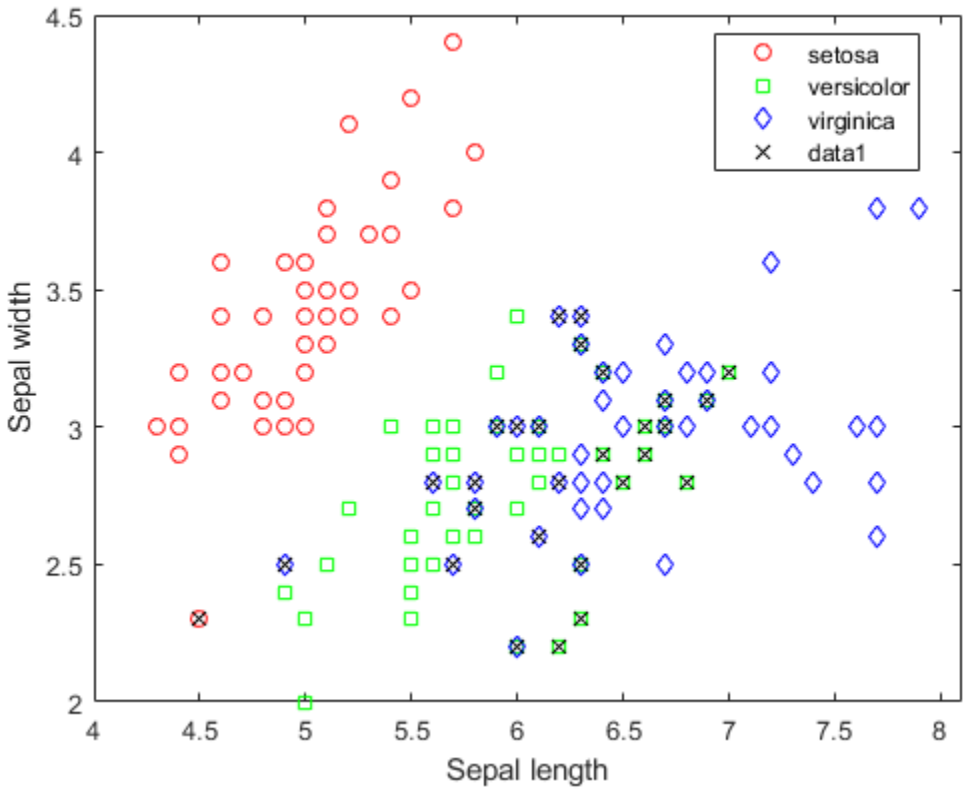
您还可以计算基于训练集的混淆矩阵。混淆矩阵包含有关已知类标签和预测类标签的信息。通常来说，混淆矩阵中的元素 (i,j) 是已知类标签为 i、预测类标签为 j 的样本的数量。对角元素表示正确分类的观测值。

```
figure
ldaResubCM = confusionchart(species,ldaClass);
```



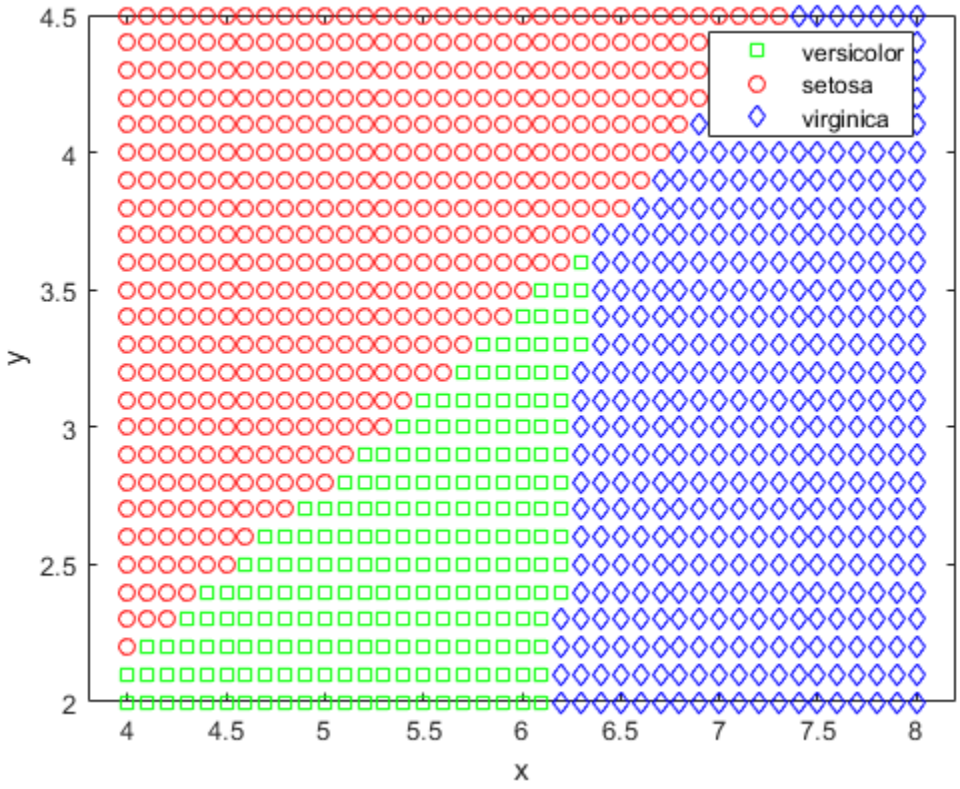
在 150 个训练观测值中，有 20% 的（即 30 个）观测值被线性判别函数错误分类。您可以将错误分类的点画上 X 来查看是哪些观测值。

```
figure(f)
bad = ~strcmp(ldaClass,species);
hold on;
plot(meas(bad,1), meas(bad,2), 'kx');
hold off;
```



该函数将平面分成几个由直线分隔的区域，并为不同的品种分配了不同的区域。要可视化这些区域，一种方法是创建 (x,y) 值网格，并将分类函数应用于该网格。

```
[x,y] = meshgrid(4:.1:8,2:.1:4.5);
x = x(:);
y = y(:);
j = classify([x y],meas(:,1:2),species);
gscatter(x,y,j,'grb','sod')
```



对于某些数据集，直线不能很好地分隔各个类的区域。这种情况下，不适合使用线性判别分析。取而代之，您可以尝试对数据进行二次判别分析 (QDA)。

计算二次判别分析的再代入误差。

```
qda = fitcdiscr(meas(:,1:2),species,'DiscrimType','quadratic');
qdaResubErr = resubLoss(qda)
```

```
qdaResubErr =

    0.2000
```

您已经计算出再代入误差。通常人们更关注测试误差（也称为泛化误差），即针对独立集合预计会得出的预测误差。事实上，再代入误差可能会低估测试误差。

在此示例中，您并没有另一个带标签的数据集，但您可以通过交叉验证来模拟一个这样的数据集。分层 10 折交叉验证是估计分类算法的测试误差的常用选择。它将训练集随机分为 10 个不相交的子集。每个子集的大小大致相同，类比例也与训练集中的类比例大致相同。取出一个子集，使用其他九个子集训练分类模型，然后使用训练过的模型对刚才取出的子集进行分类。您可以轮流取出十个子集中的每个子集并重复此操作。

由于交叉验证随机划分数据，因此结果取决于初始随机种子。要重现与此示例完全相同的结果，请执行以下命令：

```
rng(0,'twister');
```

首先使用 **cvpartition** 生成 10 个不相交的分层子集。

```
cp = cvpartition(species,'KFold',10)
```

```
cp =

K-fold cross validation partition
  NumObservations: 150
     NumTestSets: 10
       TrainSize: 135  135  135  135  135  135  135  135  135  135
        TestSize: 15  15  15  15  15  15  15  15  15  15
```

crossval 和 **kfoldLoss** 方法可以使用给定的数据分区 **cp** 来估计 LDA 和 QDA 的误分类误差。

使用 10 折分层交叉验证估计 LDA 的真实测试误差。

```
cvlda = crossval(lda,'CVPartition',cp);
ldaCVErr = kfoldLoss(cvlda)
```

```
ldaCVErr =

    0.2000
```

LDA 交叉验证误差的值与此数据的 LDA 再代入误差相同。

使用 10 折分层交叉验证估计 QDA 的真实测试误差。

```
cvqda = crossval(qda,'CVPartition',cp);
qdaCVErr = kfoldLoss(cvqda)
```

```
qdaCVErr =

    0.2200
```

QDA 的交叉验证误差略大于 LDA。它表明简单模型的性能可能不逊于甚至超过复杂模型。

朴素贝叶斯分类器

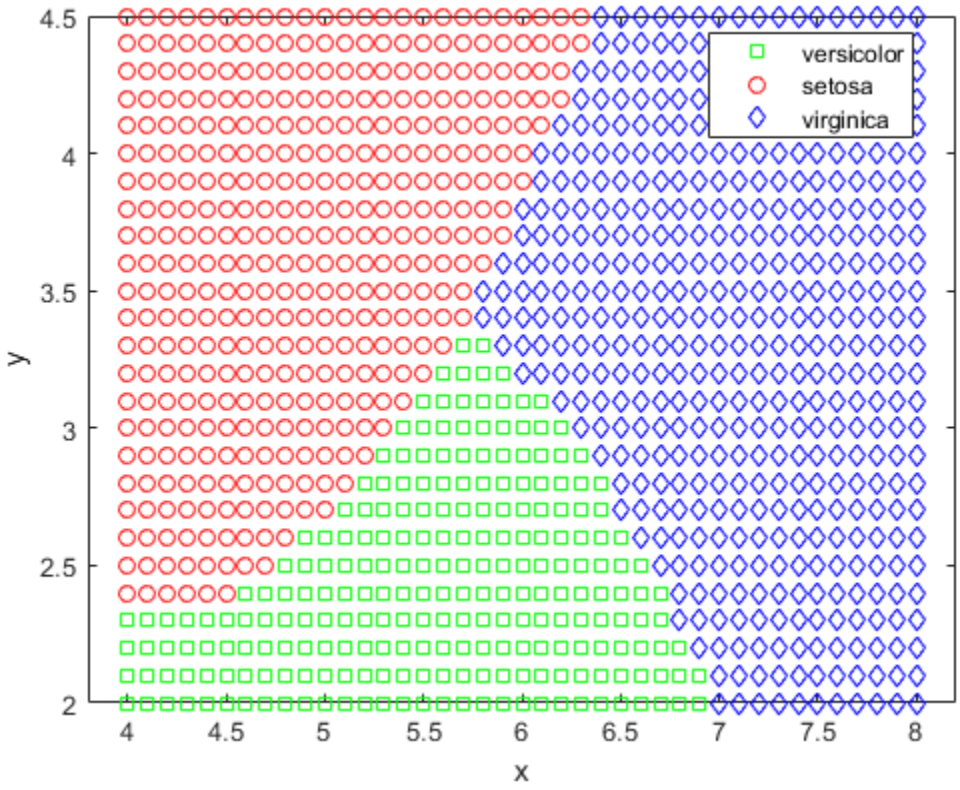
fitcdiscr 函数还有另外两种类型，'DiagLinear' 和 'DiagQuadratic'。它们类似于 'linear' 和 'quadratic'，但具有对角协方差矩阵估计值。这些对角选择是朴素贝叶斯分类器的具体例子，因为它们假定变量在类标签给定的情况下是条件独立的。朴素贝叶斯分类器是最常用的分类器之一。虽然假定变量之间类条件独立通常并不正确，但已经在实践中发现朴素贝叶斯分类器可以很好地处理许多数据集。

fitcnb 函数可用于创建更通用类型的朴素贝叶斯分类器。

首先使用高斯分布对每个类中的每个变量进行建模。您可以计算再代入误差和交叉验证误差。

```
nbGau = fitcnb(meas(:,1:2), species);
nbGauResubErr = resubLoss(nbGau)
nbGauCV = crossval(nbGau, 'CVPartition',cp);
nbGauCVErr = kfoldLoss(nbGauCV)

labels = predict(nbGau, [x y]);
gscatter(x,y,labels,'grb','sod')
```

```
nbGauResubErr =

    0.2200

nbGauCVErr =

    0.2200
```

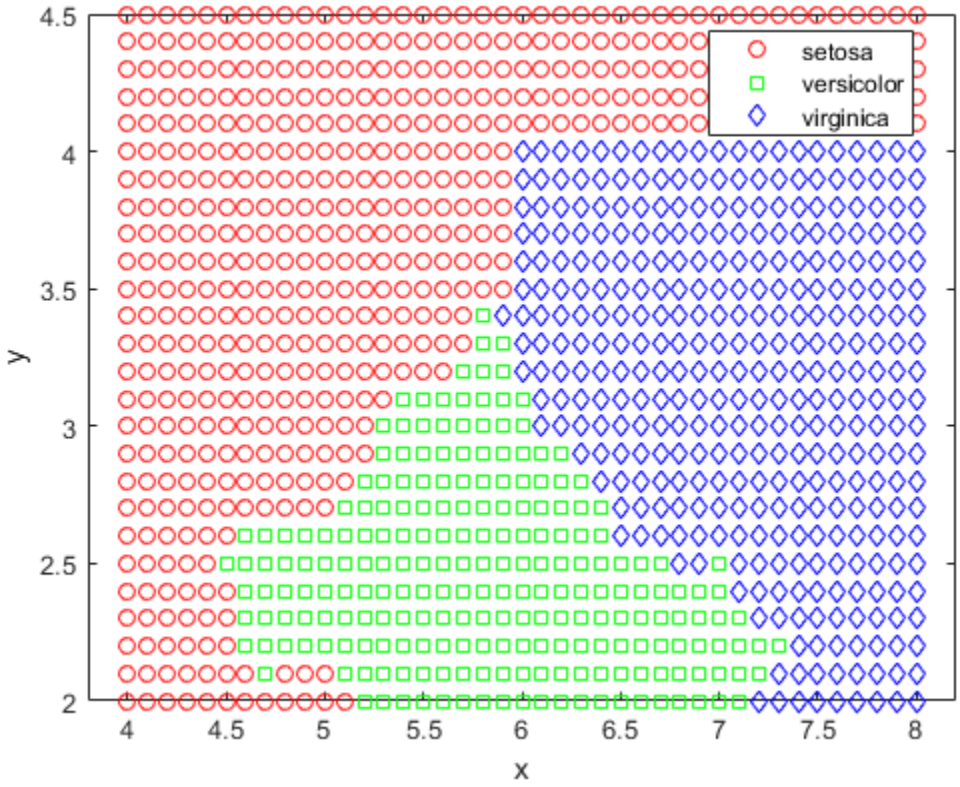


目前为止，您都假设每个类的变量都具有多元正态分布。通常这是合理的假设，但有时您可能不愿意这么假设，或者您可能很清楚地了解它是无效的。现在尝试使用核密度估计对每个类中的每个变量进行建模，这是一种更灵活的非参数化方法。此处我们将核设置为 **box**。

```
nbKD = fitcnb(meas(:,1:2), species, 'DistributionNames','kernel', 'Kernel','box');
nbKDResubErr = resubLoss(nbKD)
nbKDCV = crossval(nbKD, 'CVPartition',cp);
nbKDCVErr = kfoldLoss(nbKDCV)

labels = predict(nbKD, [x y]);
gscatter(x,y,labels,'rgb','osd')
```

```
nbKDResubErr =

    0.2067
```

```
nbKDCVErr =

    0.2133
```



对于此数据集，相比使用高斯分布的朴素贝叶斯分类器，使用核密度估计的朴素贝叶斯分类器得到的再代入误差和交叉验证误差较小。

决策树

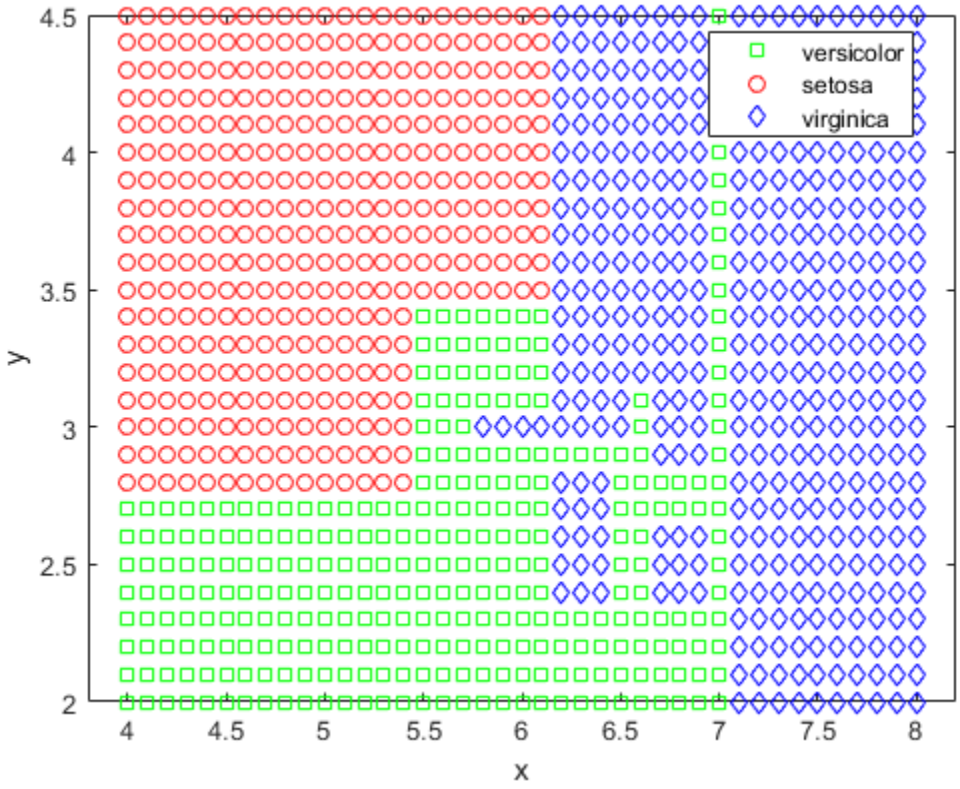
另一种分类算法基于决策树。决策树是一组简单的规则，例如，“如果萼片长度小于 5.45，则将样本分类为山鸢尾”。决策树也是非参数化的，因为它们不需要对每个类中的变量分布进行任何假设。

使用 **fitctree** 函数可创建决策树。为鸢尾花数据创建决策树，查看它对鸢尾花品种的分类效果。

```
t = fitctree(meas(:,1:2), species,'PredictorNames',{'SL' 'SW' });
```

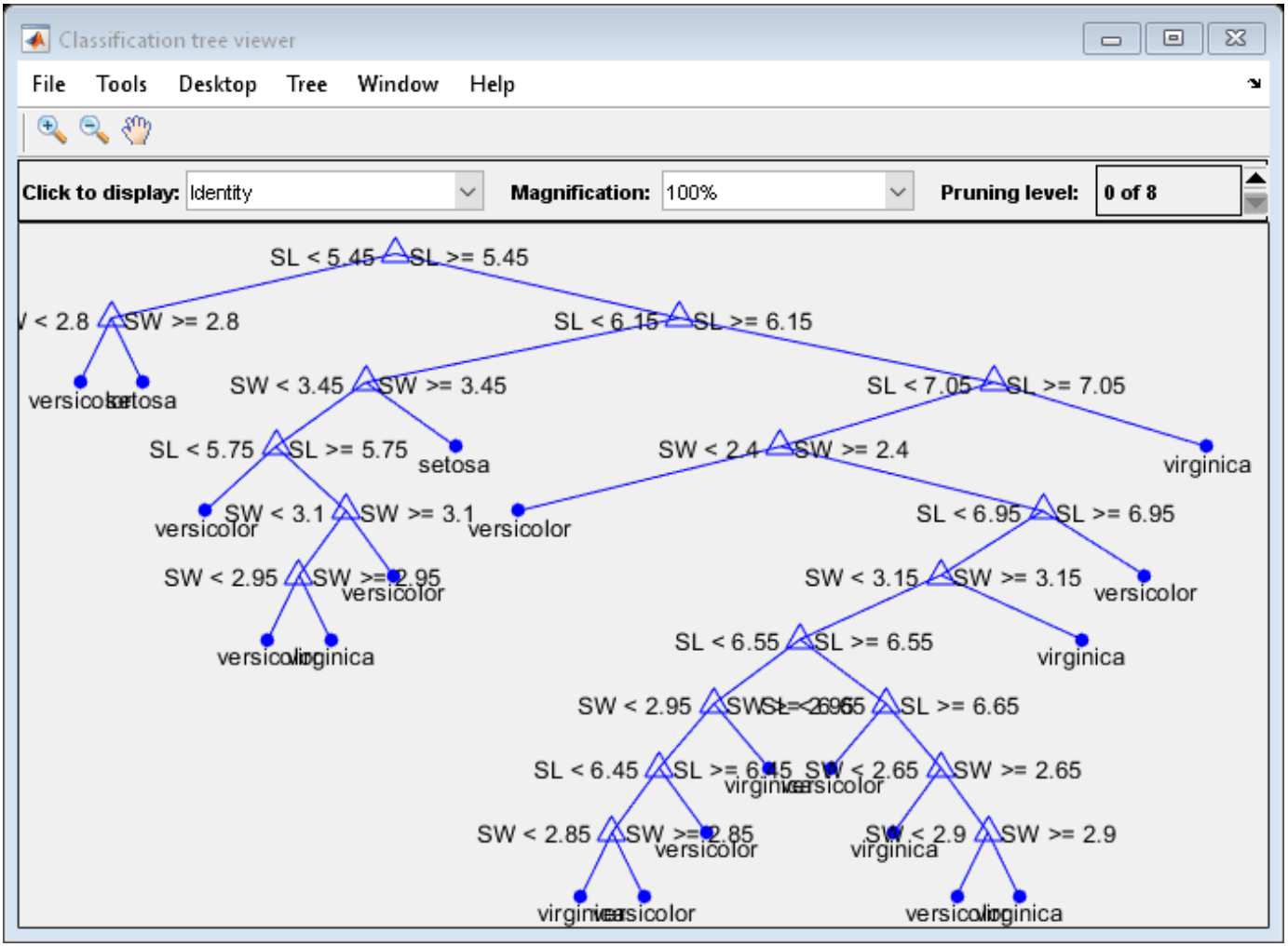
观察决策树方法如何划分平面很有意思。使用与之前一样的方法，可视化分配给每个品种的区域。

```
[grpname,node] = predict(t,[x y]);
gscatter(x,y,grpname,'grb','sod')
```



可视化决策树的另一种方法是绘制决策规则和类分配图。

```
view(t,'Mode','graph');
```



这个看起来有些杂乱的树使用一系列形如 "SL < 5.45" 的规则将每个样本划分到 19 个终端节点之一。要确定观测值的品种分配，请从顶部节点开始应用规则。如果该点满足该规则，则沿左侧路线前进，如果不满足，则沿右侧路线前进。最后您将到达一个终端节点，将观测值分配给三个品种之一。

计算决策树的再代入误差和交叉验证误差。

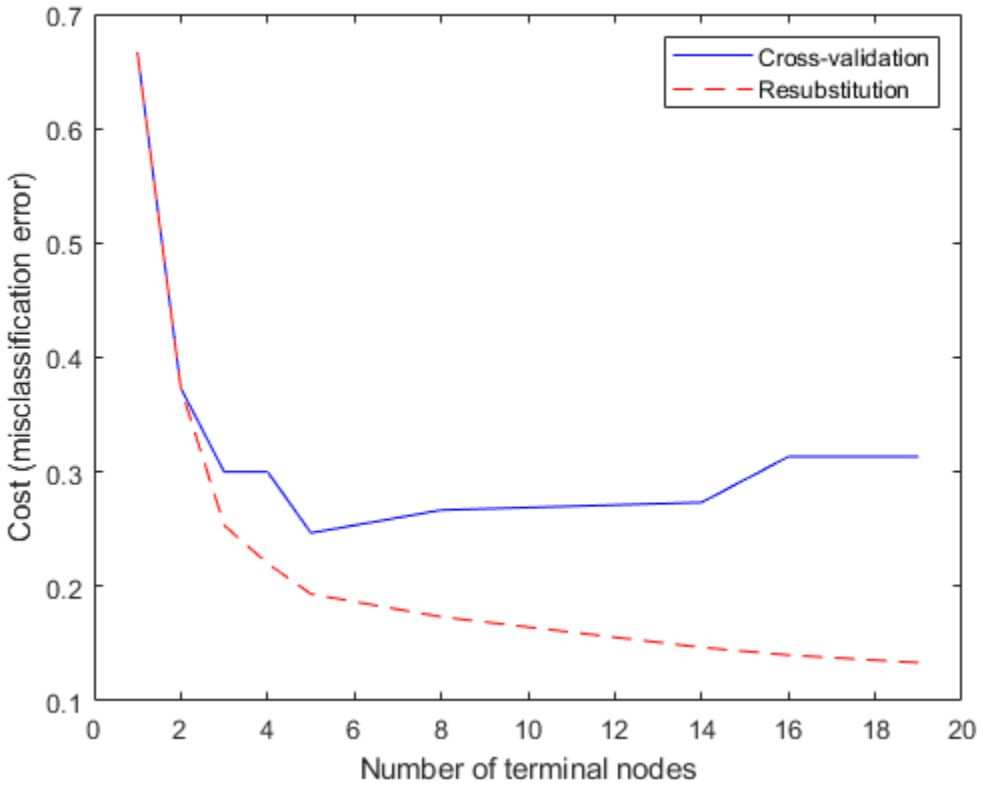
```
dtResubErr = resubLoss(t)

cvt = crossval(t,'CVPartition',cp);
dtCVErr = kfoldLoss(cvt)
```

```
dtResubErr =

    0.1333


dtCVErr =

    0.3000
```

对于决策树算法，交叉验证误差估计值明显大于再代入误差。这表明生成的树对训练集过拟合。也就是说，此树可以很好地对原始训练集进行分类，但树的结构仅对这个特定的训练集敏感，因此对新数据的分类效果可能会变差。通常我们可以找到一个更为简单的树，它在处理新数据时要比复杂的树效果好。

尝试对树进行剪枝。首先计算原始树的各种子集的再代入误差。然后计算这些子树的交叉验证误差。图中显示再代入误差过于乐观。它随着树大小的增加而不断降低，但在某一点之后，随着树大小的增加，交叉验证误差率也随之增加。

```
resubcost = resubLoss(t,'Subtrees','all');
[cost,secost,ntermnodes,bestlevel] = cvloss(t,'Subtrees','all');
plot(ntermnodes,cost,'b-', ntermnodes,resubcost,'r--')
figure(gcf);
xlabel('Number of terminal nodes');
ylabel('Cost (misclassification error)')
legend('Cross-validation','Resubstitution')
```

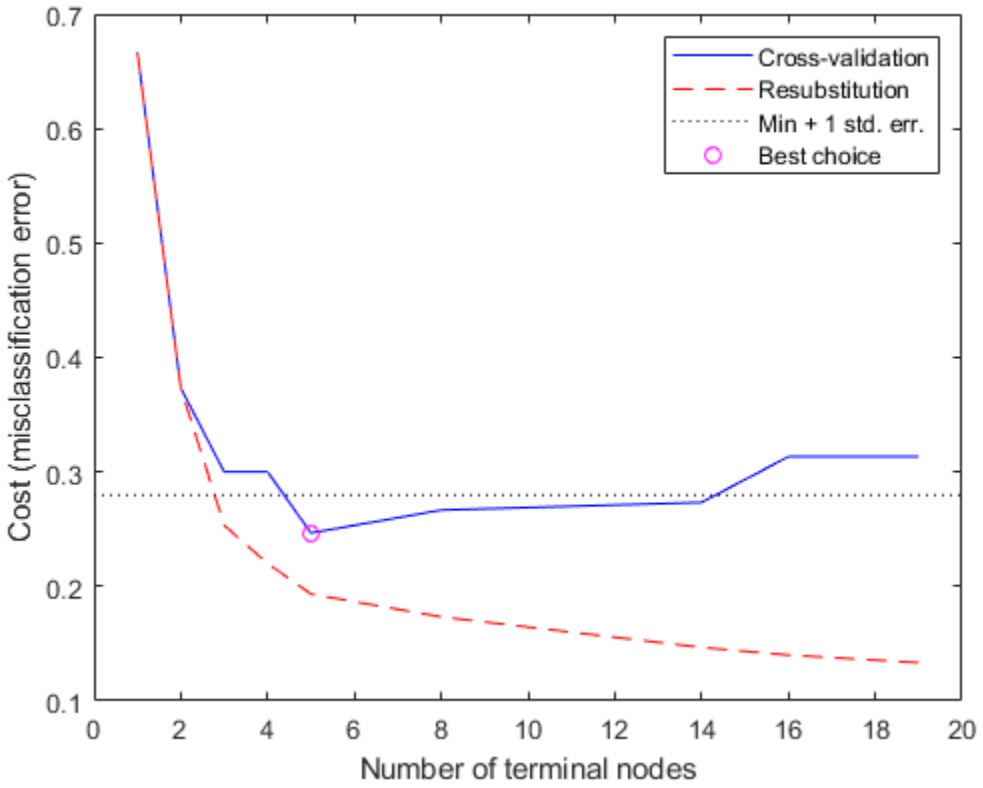


您应该选择哪个树？一个简单的原则就是选择交叉验证误差最小的树。如果简单的树和复杂的树都能提供大致满意的结果，您可能更愿意使用简单的树。对于此示例，我们选择与最小值的距离在一个标准误差范围内的最简单的树。这是 **ClassificationTree** 的 **cvloss** 方法采用的默认规则。

通过计算截止值（等于最小成本加上一个标准误差），您可以在图上显示这一点。由 **cvloss** 方法计算的“最佳”级别是此截止值下的最小树。（注意，**bestlevel**=0 对应于未修剪的树，因此您必须加上 1 才能将其用作 **cvloss** 的向量输出的索引。）

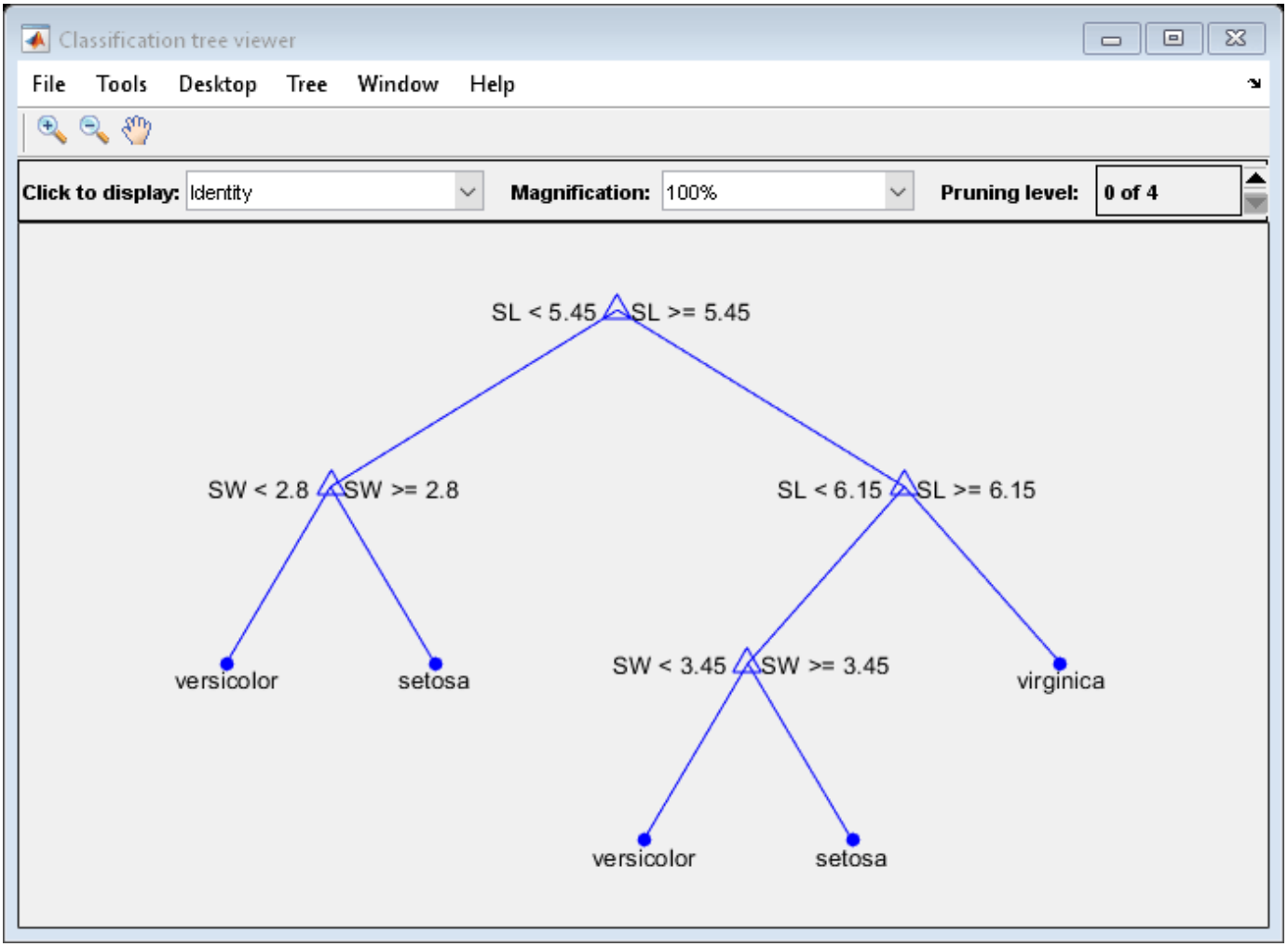
```
[mincost,minloc] = min(cost);
cutoff = mincost + secost(minloc);
hold on
```

```
plot([0 20], [cutoff cutoff], 'k:')
plot(ntermnodes(bestlevel+1), cost(bestlevel+1), 'mo')
legend('Cross-validation','Resubstitution','Min + 1 std. err.','Best choice')
hold off
```



最后，您可以查看修剪后的树并计算估计的误分类误差。

```
pt = prune(t,'Level',bestlevel);
view(pt,'Mode','graph')
```




```
cost(bestlevel+1)
```

```
ans =

    0.2467
```

结论

此示例说明如何使用 Statistics and Machine Learning Toolbox™ 函数在 MATLAB® 中执行分类。

此示例并非 Fisher 鸢尾花数据的理想分析模型，事实上，使用花瓣测量值代替萼片测量值或者将二者相结合可以实现更好的分类。此外，此示例也不是要比较不同分类算法的优缺点。我们希望它对您分析其他数据集和比较不同算法能有所启发。还有一些工具箱函数可以实现其他分类算法。例如，您可以使用 **TreeBagger** 执行自助汇聚以集成决策树，如示例“Bootstrap Aggregation (Bagging) of Classification Trees Using TreeBagger”中所述。

非参数化有监督学习

用于二类分类的支持向量机

本节内容
“了解支持向量机” （第 18-2 页）
“使用支持向量机” （第 18-6 页）
“用高斯核训练 SVM 分类器” （第 18-7 页）
“使用自定义核训练 SVM 分类器” （第 18-10 页）
“使用贝叶斯优化来优化 SVM 分类器拟合” （第 18-14 页）
“绘制 SVM 分类模型的后验概率区域” （第 18-21 页）
“使用线性支持向量机分析图像” （第 18-23 页）

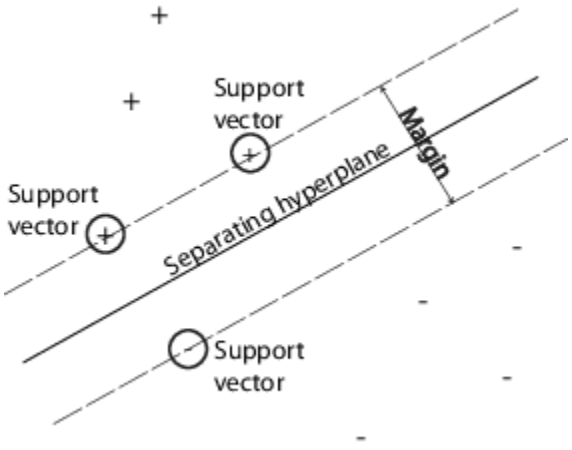
了解支持向量机

- “可分离数据” （第 18-2 页）
- “不可分离的数据” （第 18-3 页）
- “核的非线性变换” （第 18-5 页）

可分离数据

当数据正好有两个类时，可以使用支持向量机 (SVM)。SVM 通过找到将一个类的所有数据点与另一个类的所有数据点分离的最佳超平面对数据进行分类。SVM 的最佳超平面是指使两个类之间的边距最大的超平面。边距是指平行于超平面的内部不含数据点的平板的最大宽度。

支持向量是最接近分离超平面的数据点；这些点在平板的边界上。下图说明了这些定义，其中 + 表示类型 1 的数据点，而 - 表示类型 -1 的数据点。



数学表达式：原问题

此讨论基于 Hastie、Tibshirani 和 Friedman [1] 以及 Christianini 和 Shawe-Taylor [2]的研究。

训练数据是一组点（向量）x_j 及其类别 y_j。对于某个维度 d，$x_j \in R^d$ 且 $y_j = \pm 1$。超平面的方程是

$$f(x) = x'\beta + b = 0$$

其中 $\beta \in R^d$ 和 b 是实数。

以下问题定义最佳分离超平面（即决策边界）。计算使 ||β|| 最小的 β 和 b，使得所有数据点 (x_j,y_j) 满足

$$y_j f(x_j) \geq 1.$$

支持向量是边界上的 x_j，满足 $y_j f(x_j) = 1.$

为了数学上的方便，此问题通常表示为等效的最小化 $\|\beta\|$ 的问题。这是二次规划问题。最优解 $(\widehat{\beta}, \widehat{b})$ 能够对向量 z 进行如下分类：

$$\text{class}(z) = \text{sign}\left(z'\widehat{\beta} + \widehat{b}\right) = \text{sign}\left(\widehat{f}(z)\right).$$

$\widehat{f}(z)$ 是分类分数，表示 z 与决策边界的距离。

数学表达式：对偶问题

求解对偶二次规划问题在计算上更简单。要获得对偶问题，将正的 Lagrange 乘数 α_j 乘以每个约束，然后从目标函数中减去它们：

$$L_P = \frac{1}{2}\beta'\beta - \sum_j \alpha_j\left(y_j\left(x_j'\beta + b\right) - 1\right),$$

您要从中计算 L_P 在 β 和 b 上的平稳点。将 L_P 的梯度设置为 0，您将获得

$$\begin{aligned}\beta &= \sum_j \alpha_j y_j x_j \\ 0 &= \sum_j \alpha_j y_j .\end{aligned} \tag{18-1}$$

代入 L_P，您将获得对偶问题 L_D：

$$L_D = \sum_j \alpha_j - \frac{1}{2}\sum_j\sum_k \alpha_j\alpha_k y_j y_k x_j' x_k,$$

您需要求 $\alpha_j \geq 0$ 上的最大值。一般情况下，许多 α_j 在最大值时为 0。对偶问题解中非零的 α_j 定义超平面，如 “公式 18-1” 所示，这可以得出 β 成为 $\alpha_j y_j x_j$ 的总和。对应于非零 α_j 的数据点 x_j 是支持向量。

L_D 关于非零 α_j 的导数在最佳情况下为 0。这可得出

$$y_j f(x_j) - 1 = 0.$$

具体来说，通过取非零 α_j 作为任何 j 的值，可得出 b 在解处的值。

此对偶问题是标准的二次规划问题。Optimization Toolbox™ **quadprog** 求解器可用于求解这类问题。

不可分离的数据

您的数据可能不存在一个分离超平面。在这种情况下，SVM 可以使用软边距，获取一个可分离许多数据点，但并非所有数据点的超平面。

软边距有两个标准表达式。两者都包括添加松弛变量 ξ_j 和罚分参数 C。

- L^1-范数问题是：

$$\min_{\beta, b, \xi}\left(\frac{1}{2}\beta'\beta + C\sum_j \xi_j\right)$$

满足

$$y_j f(x_j) \geq 1 - \xi_j$$
$$\xi_j \geq 0.$$

L^1-范数表示使用 ξ_j（而不是其平方）作为松弛变量。**fitcsvm** 的三个求解器选项 **SMO**、**ISDA** 和 **L1QP** 可求解 L^1-范数最小化问题。

- L^2-范数问题是：

$$\min_{\beta, b, \xi}\left(\frac{1}{2}\beta'\beta + C\sum_j \xi_j^2\right)$$

受限于同样的约束。

在这些公式中，您可以看到，增加 C 会增加松弛变量 ξ_j 的权重，这意味着优化尝试在类之间进行更严格的分离。等效地，将 C 朝 0 方向减少会降低误分类的重要性。

数学表达式：对偶问题

为了便于计算，这里使用软边距的 L^1 对偶问题表达式。使用 Lagrange 乘数 μ_j，可得到 L^1-范数最小化问题的函数：

$$L_P = \frac{1}{2}\beta'\beta + C\sum_j \xi_j - \sum_j \alpha_j\left(y_i f(x_j) - (1 - \xi_j)\right) - \sum_j \mu_j \xi_j,$$

您要从中计算 L_P 在 β、b 和正 ξ_j 上的平稳点。将 L_P 的梯度设置为 0，您将获得

$$\beta = \sum_j \alpha_j y_j x_j$$
$$\sum_j \alpha_j y_j = 0$$
$$\alpha_j = C - \mu_j$$
$$\alpha_j, \mu_j, \xi_j \geq 0.$$

通过这些方程可直接获取对偶问题表达式：

$$\max_\alpha \sum_j \alpha_j - \frac{1}{2}\sum_j \sum_k \alpha_j \alpha_k y_j y_k x_j' x_k$$

它受限于以下约束

$$\sum_j y_j \alpha_j = 0$$
$$0 \leq \alpha_j \leq C.$$

最后一组不等式 $0 \leq \alpha_j \leq C$ 说明了为什么 C 有时称为框约束。C 将 Lagrange 乘数 α_j 的允许值保持在一个“框”中，即有界区域内。

b 的梯度方程给出了对应于支持向量的非零 α_j 集合的解 b。

您可以用类似的方式编写和求解 L^2-范数问题的对偶问题。有关详细信息，请参阅 Christianini and Shawe-Taylor [2], Chapter 6。

fitcsvm 实现

以上两个对偶软边距问题均为二次规划问题。在内部，**fitcsvm** 提供了几种不同算法来求解这些问题。

- 对于一类或二类分类，如果您没有设置数据中预期离群值的占比（请参阅 **OutlierFraction**），则默认求解器是序列最小优化 (SMO)。SMO 通过一系列两点最小化来求解 1-范数最小化问题。在优化过程中，SMO 遵守线性约束 $\sum_i \alpha_i y_i = 0$,，并在模型中显式包含偏置项。SMO 相对较快。有关 SMO 的详细信息，请参阅 [3]。
- 对于二类分类，如果您设置了数据中预期离群值的占比，则默认求解器是迭代单点数据算法。与 SMO 相同，ISDA 用于求解 1-范数问题。与 SMO 不同的是，ISDA 通过一系列单点最小化来进行最小化求解，不遵守线性约束，也不在模型中显式包含偏置项。有关 ISDA 的详细信息，请参阅 [4]。
- 对于一类或二类分类，如果您有 Optimization Toolbox 许可证，您可以选择使用 **quadprog** 来求解 1-范数问题。**quadprog** 会占用大量内存，但能以高精度求解二次规划。有关详细信息，请参阅“二次规划定义” (Optimization Toolbox)。

核的非线性变换

一些二类分类问题没有简单的超平面作为有用的分离标准。对于这些问题，有一种变通的数学方法可保留 SVM 分离超平面的几乎所有简易性。

这种方法利用了再生核理论的下列成果：

- 存在一类具有以下属性的函数类 $G(x_1,x_2)$。存在一个线性空间 S 和将 x 映射到 S 函数 φ，满足

  $G(x_1,x_2) = <\varphi(x_1),\varphi(x_2)>$。

  点积发生在空间 S 中。
- 该类函数包括：
  - 多项式：对于某些正整数 p，

    $G(x_1,x_2) = (1 + x_1'x_2)^p$。
  - 径向基函数（高斯）：

    $G(x_1,x_2) = \exp(-\|x_1-x_2\|^2)$。
  - 多层感知机或 sigmoid（神经网络）：对于正数 p_1 和负数 p_2，

    $G(x_1,x_2) = \tanh(p_1x_1'x_2 + p_2)$。

    注意

    - 并非每组 p_1 和 p_2 都能产生有效的再生核。
    - **fitcsvm** 不支持 sigmoid 核。在这种情况下，您可以定义 sigmoid 核，并通过使用 **'KernelFunction'** 名称-值对组参数来指定它。有关详细信息，请参阅 “使用自定义核训练 SVM 分类器” （第 18-10 页）。

使用核的数学方法依赖于超平面的计算方法。所有超平面分类的计算只使用点积。因此，非线性核可以使用相同的计算和解算法，并获得非线性分类器。得到的分类器是某个空间 S 中的超曲面，但不需要标识或检查空间 S。

使用支持向量机

与任何有监督学习模型一样，您首先训练支持向量机，然后交叉验证分类器。使用经过训练的机器对新数据进行分类（预测）。此外，为了获得令人满意的预测准确度，可以使用各种 SVM 核函数，并且必须调整核函数的参数。

训练 SVM 分类器

使用 **fitcsvm** 训练并（可选）交叉验证 SVM 分类器。最常见的语法是：

```
SVMModel = fitcsvm(X,Y,'KernelFunction','rbf',...
   'Standardize',true,'ClassNames',{'negClass','posClass'});
```

输入是：

- **X** - 预测变量数据的矩阵，其中每行是一个观测值，每列是一个预测变量。
- **Y** - 类标签数组，其中每行对应于 **X** 中对应行的值。**Y** 可以是分类数组、字符数组或字符串数组、逻辑值或数值向量或字符向量元胞数组。
- **KernelFunction** - 二类学习的默认值为 **'linear'**，它通过超平面分离数据。值 **'gaussian'**（或 **'rbf'**）是一类学习的默认值，它指定使用高斯（或径向基函数）核。成功训练 SVM 分类器的重要步骤是选择合适的核函数。
- **Standardize** - 指示软件在训练分类器之前是否应标准化预测变量的标志。
- **ClassNames** - 区分负类和正类，或指定要在数据中包括哪些类。负类是第一个元素（或字符数组的行），例如 **'negClass'**，正类是第二个元素（或字符数组的行），例如 **'posClass'**。**ClassNames** 必须与 **Y** 具有相同的数据类型。指定类名是很好的做法，尤其是在比较不同分类器的性能时。

生成的训练模型 (**SVMModel**) 包含来自 SVM 算法的优化参数，使您能够对新数据进行分类。

有关可用于控制训练的更多名称-值对组，请参阅 **fitcsvm** 参考页。

用 SVM 分类器对新数据进行分类

使用 **predict** 对新数据进行分类。使用经过训练的 SVM 分类器 (**SVMModel**) 对新数据进行分类的语法如下：

```
[label,score] = predict(SVMModel,newX);
```

生成的向量 **label** 表示 **X** 中每行的分类。**score** 是软分数的 n×2 矩阵。每行对应于 **X** 中的一行，即新观测值。第一列包含分类为负类的观测值的分数，第二列包含分类为正类的观测值的分数。

要估计后验概率而不是分数，请首先将经过训练的 SVM 分类器 (**SVMModel**) 传递给 **fitPosterior**，该方法对分数进行分数-后验概率转换函数拟合。语法是：

```
ScoreSVMModel = fitPosterior(SVMModel,X,Y);
```

分类器 **ScoreSVMModel** 的属性 **ScoreTransform** 包含最佳转换函数。将 **ScoreSVMModel** 传递给 **predict**。输出参数 **score** 不返回分数，而是包含分类为负类（**score** 的列 1）或正类（**score** 的列 2）的观测值的后验概率。

调整 SVM 分类器

使用 **fitcsvm** 的 **'OptimizeHyperparameters'** 名称-值对组参数来求得使交叉验证损失最小的参数值。可使用的参数包括 **'BoxConstraint'**、**'KernelFunction'**、**'KernelScale'**、**'PolynomialOrder'** 和 **'Standardize'**。有关示例，请参阅“使用贝叶斯优化来优化 SVM 分类器拟合” （第 18-14 页）。您也可以使用 **bayesopt** 函数，如“Optimize Cross-Validated Classifier Using bayesopt”中所示。**bayesopt** 函数允许更加灵活地自定义优化。您可以使用 **bayesopt** 函数优化任何参数，包括使用 **fitcsvm** 函数时不能优化的参数。

您还可以尝试根据以下方案手动调整分类器的参数：

1. 将数据传递给 **fitcsvm**，并设置名称-值对组参数 **'KernelScale','auto'**。假设经过训练的 SVM 模型称为 **SVMModel**。软件使用启发式过程来选择核尺度。启发式过程使用二次抽样。因此，为了重现结果，在训练分类器之前，请使用 **rng** 设置随机数种子。
2. 通过将分类器传递给 **crossval** 以交叉验证分类器。默认情况下，软件进行 10 折交叉验证。
3. 将经过交叉验证的 SVM 模型传递给 **kfoldLoss** 以估计和保留分类误差。
4. 重新训练 SVM 分类器，但调整 **'KernelScale'** 和 **'BoxConstraint'** 名称-值对组参数。

   - **BoxConstraint** - 一种策略是尝试框约束参数的等比数列。例如，取依次增长 10 倍的 11 个值，从 **1e-5** 到 **1e5**。增加 **BoxConstraint** 可能会减少支持向量的数量，但也会增加训练时间。
   - **KernelScale** - 一种策略是尝试基于原始核尺度缩放的 RBF sigma 参数的等比数列。为此，请执行以下操作：

     a. 使用圆点表示法检索原始核尺度，例如 **ks**：**ks = SVMModel.KernelParameters.Scale**。

     b. 将原始核尺度的缩放值用作新的核尺度。例如，将 **ks** 乘以逐项递增 10 倍的 11 个值，从 **1e-5** 到 **1e5**。

选择产生最低分类误差的模型。您可能需要进一步细化参数以获得更高的准确度。从初始参数开始，执行另一个交叉验证步骤，这次使用因子 1.2。

用高斯核训练 SVM 分类器

此示例说明如何使用高斯核函数生成非线性分类器。首先，在二维单位圆盘内生成一个由点组成的类，在半径为 1 到半径为 2 的环形空间内生成另一个由点组成的类。然后，使用高斯径向基函数核基于数据生成一个分类器。默认的线性分类器显然不适合此问题，因为模型具有圆对称特性。将框约束参数设置为 **Inf** 以进行严格分类，这意味着没有误分类的训练点。其他核函数可能无法使用这一严格的框约束，因为它们可能无法提供严格的分类。即使 rbf 分类器可以将类分离，结果也可能会过度训练。

生成在单位圆盘上均匀分布的 100 个点。为此，可先计算均匀随机变量的平方根以得到半径 r，并在 (0, 2π) 中均匀生成角度 t，然后将点置于 (r cos(t), r sin(t)) 位置上。

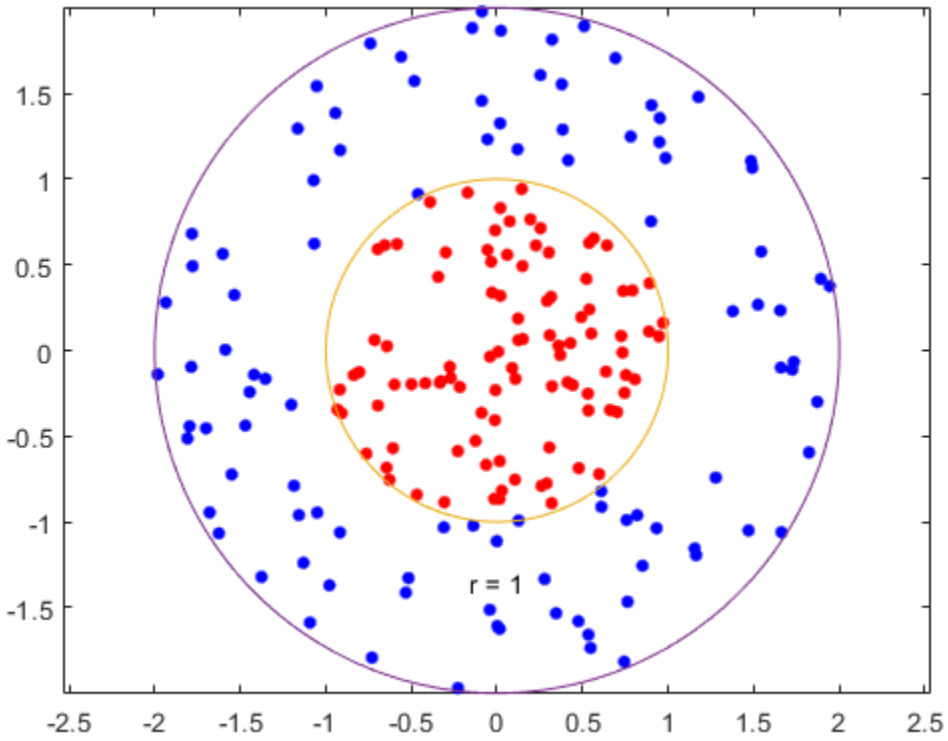
```
rng(1); % For reproducibility
r = sqrt(rand(100,1)); % Radius
t = 2*pi*rand(100,1);  % Angle
data1 = [r.*cos(t), r.*sin(t)]; % Points
```

生成在环形空间中均匀分布的 100 个点。半径同样与平方根成正比，这次采用从 1 到 4 均匀分布值的平方根。

```
r2 = sqrt(3*rand(100,1)+1); % Radius
t2 = 2*pi*rand(100,1);      % Angle
data2 = [r2.*cos(t2), r2.*sin(t2)]; % points
```

绘制各点，并绘制半径为 1 和 2 的圆进行比较。

```
figure;
plot(data1(:,1),data1(:,2),'r.','MarkerSize',15)
hold on
plot(data2(:,1),data2(:,2),'b.','MarkerSize',15)
ezpolar(@(x)1);ezpolar(@(x)2);
axis equal
hold off
```



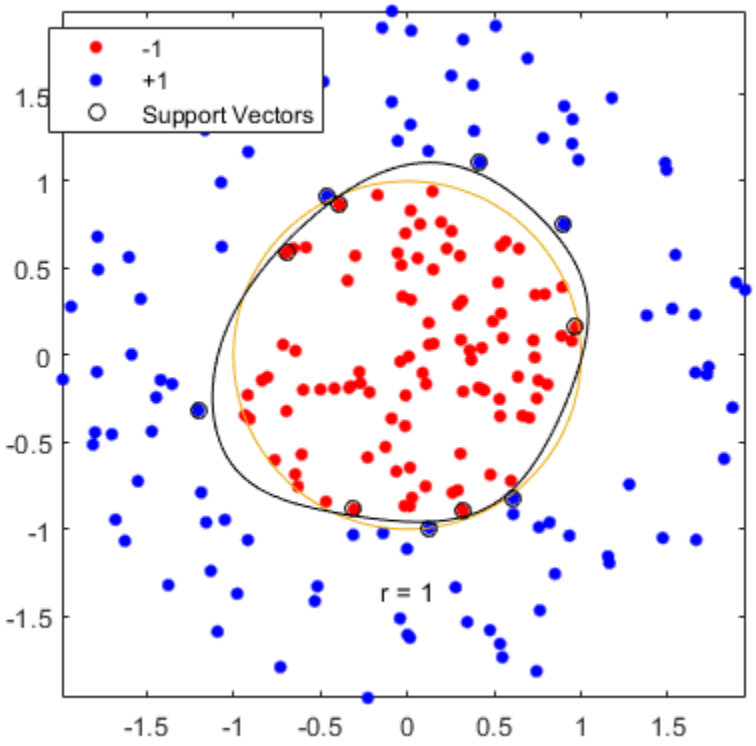
将数据放在一个矩阵中，并建立一个分类向量。

```
data3 = [data1;data2];
theclass = ones(200,1);
theclass(1:100) = -1;
```

将 **KernelFunction** 设置为 **'rbf'**，**BoxConstraint** 设置为 **Inf** 以训练 SVM 分类器。绘制决策边界并标记支持向量。

```
%Train the SVM Classifier
cl = fitcsvm(data3,theclass,'KernelFunction','rbf',...
    'BoxConstraint',Inf,'ClassNames',[-1,1]);

% Predict scores over the grid
d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(data3(:,1)):d:max(data3(:,1)),...
    min(data3(:,2)):d:max(data3(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
```

```
[~,scores] = predict(cl,xGrid);

% Plot the data and the decision boundary
figure;
h(1:2) = gscatter(data3(:,1),data3(:,2),theclass,'rb','.');
hold on
ezpolar(@(x)1);
h(3) = plot(data3(cl.IsSupportVector,1),data3(cl.IsSupportVector,2),'ko');
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0 0],'k');
legend(h,{'-1','+1','Support Vectors'});
axis equal
hold off
```

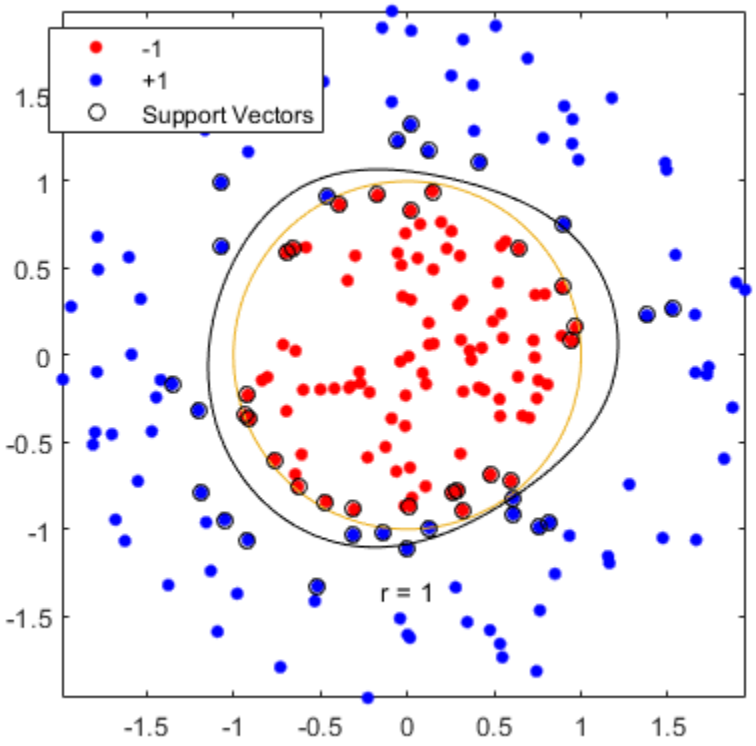


fitcsvm 生成一个接近半径为 1 的圆的分类器。差异是随机训练数据造成的。

使用默认参数进行训练会形成更接近圆形的分类边界，但会对一些训练数据进行误分类。此外，**BoxConstraint** 的默认值为 **1**，因此支持向量更多。

```
cl2 = fitcsvm(data3,theclass,'KernelFunction','rbf');
[~,scores2] = predict(cl2,xGrid);

figure;
h(1:2) = gscatter(data3(:,1),data3(:,2),theclass,'rb','.');
hold on
ezpolar(@(x)1);
h(3) = plot(data3(cl2.IsSupportVector,1),data3(cl2.IsSupportVector,2),'ko');
contour(x1Grid,x2Grid,reshape(scores2(:,2),size(x1Grid)),[0 0],'k');
legend(h,{'-1','+1','Support Vectors'});
```

```
axis equal
hold off
```



使用自定义核训练 SVM 分类器

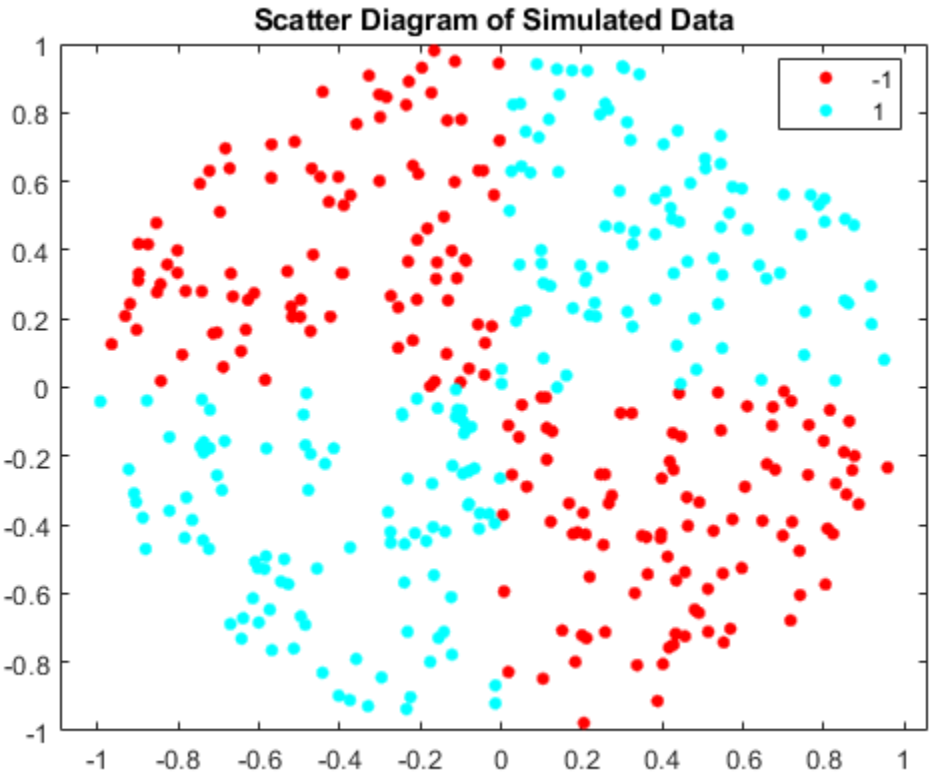
此示例说明如何使用自定义核函数（例如 sigmoid 核）训练 SVM 分类器，并调整自定义核函数参数。

在单位圆内生成一组随机点。将第一和第三象限中的点标记为属于正类，将第二和第四象限中的点标记为属于负类。

```
rng(1);  % For reproducibility
n = 100; % Number of points per quadrant

r1 = sqrt(rand(2*n,1));                  % Random radii
t1 = [pi/2*rand(n,1); (pi/2*rand(n,1)+pi)]; % Random angles for Q1 and Q3
X1 = [r1.*cos(t1) r1.*sin(t1)];          % Polar-to-Cartesian conversion

r2 = sqrt(rand(2*n,1));
t2 = [pi/2*rand(n,1)+pi/2; (pi/2*rand(n,1)-pi/2)]; % Random angles for Q2 and Q4
X2 = [r2.*cos(t2) r2.*sin(t2)];

X = [X1; X2];       % Predictors
Y = ones(4*n,1);
Y(2*n + 1:end) = -1; % Labels
```

绘制数据图。

```
figure;
gscatter(X(:,1),X(:,2),Y);
title('Scatter Diagram of Simulated Data')
```



编写一个函数，该函数接受特征空间中的两个矩阵作为输入，并使用 sigmoid 核将它们转换为 Gram 矩阵。

```
function G = mysigmoid(U,V)
% Sigmoid kernel function with slope gamma and intercept c
gamma = 1;
c = -1;
G = tanh(gamma*U*V' + c);
end
```

将此代码保存为您的 MATLAB® 路径上名为 **mysigmoid** 的文件。

使用 sigmoid 核函数训练 SVM 分类器。标准化数据是一种良好的做法。

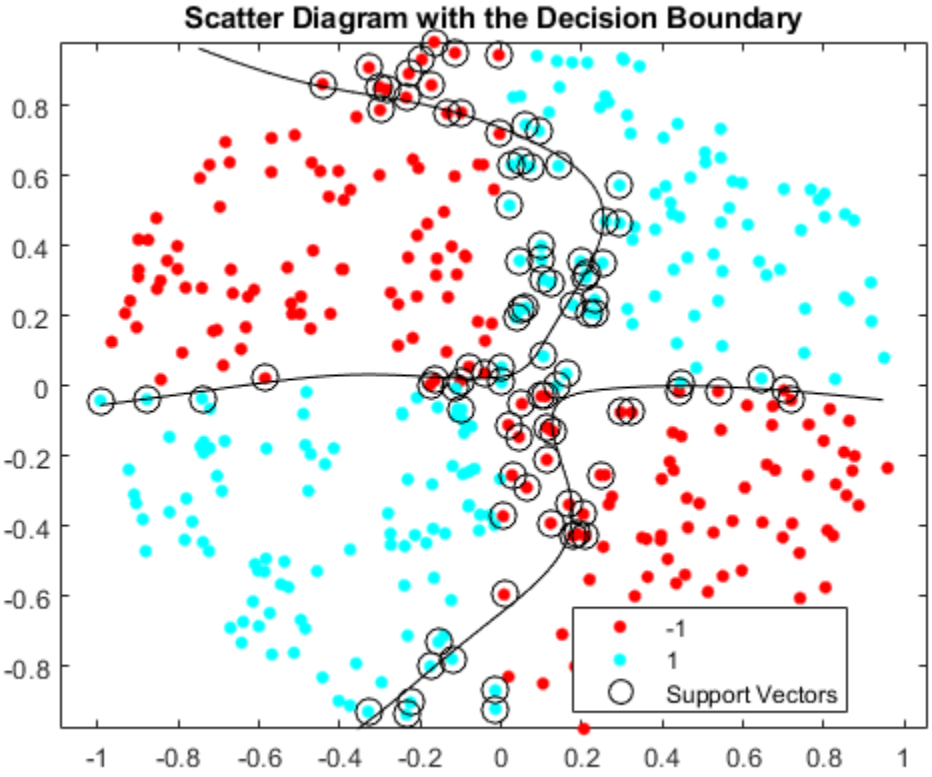
```
Mdl1 = fitcsvm(X,Y,'KernelFunction','mysigmoid','Standardize',true);
```

Mdl1 是 **ClassificationSVM** 分类器，其中包含估计的参数。

绘制数据，并确定支持向量和决策边界。

```
   % Compute the scores over a grid
d = 0.02; % Step size of the grid
```

```
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
   min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];       % The grid
[~,scores1] = predict(Mdl1,xGrid); % The scores

figure;
h(1:2) = gscatter(X(:,1),X(:,2),Y);
hold on
h(3) = plot(X(Mdl1.IsSupportVector,1),...
   X(Mdl1.IsSupportVector,2),'ko','MarkerSize',10);
   % Support vectors
contour(x1Grid,x2Grid,reshape(scores1(:,2),size(x1Grid)),[0 0],'k');
   % Decision boundary
title('Scatter Diagram with the Decision Boundary')
legend({'-1','1','Support Vectors'},'Location','Best');
hold off
```



您可以调整核参数，尝试改进决策边界的形状。这也可能降低样本内的误分类率，但您应首先确定样本外的误分类率。

使用 10 折交叉验证确定样本外的误分类率。

```
CVMdl1 = crossval(Mdl1);
misclass1 = kfoldLoss(CVMdl1);
misclass1
```

```
misclass1 =
```

```
    0.1350
```

样本外的误分类率为 13.5%。

编写另一个 sigmoid 函数，但设置 **gamma = 0.5;**。

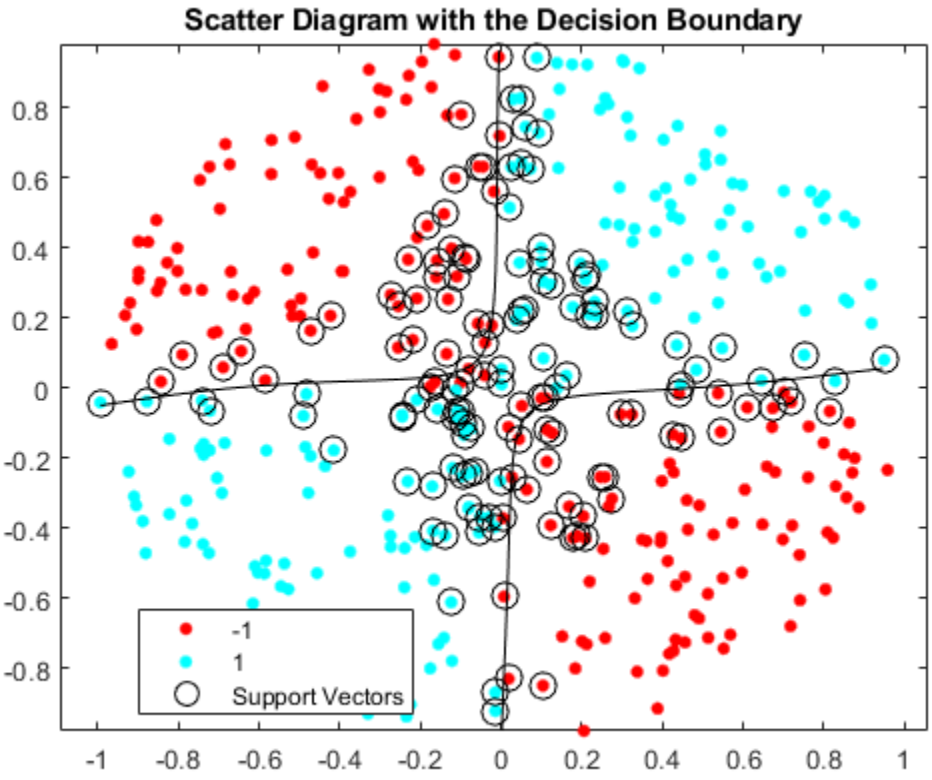
```
function G = mysigmoid2(U,V)
% Sigmoid kernel function with slope gamma and intercept c
gamma = 0.5;
c = -1;
G = tanh(gamma*U*V' + c);
end
```

将此代码保存为您的 MATLAB® 路径上名为 **mysigmoid2** 的文件。

使用调整后的 sigmoid 核训练另一个 SVM 分类器。绘制数据和决策区域，并确定样本外的误分类率。

```
Mdl2 = fitcsvm(X,Y,'KernelFunction','mysigmoid2','Standardize',true);
[~,scores2] = predict(Mdl2,xGrid);

figure;
h(1:2) = gscatter(X(:,1),X(:,2),Y);
hold on
h(3) = plot(X(Mdl2.IsSupportVector,1),...
   X(Mdl2.IsSupportVector,2),'ko','MarkerSize',10);
title('Scatter Diagram with the Decision Boundary')
contour(x1Grid,x2Grid,reshape(scores2(:,2),size(x1Grid)),[0 0],'k');
legend({'-1','1','Support Vectors'},'Location','Best');
hold off

CVMdl2 = crossval(Mdl2);
misclass2 = kfoldLoss(CVMdl2);
misclass2
```

```
misclass2 =

    0.0450
```



在 sigmoid 斜率调整后，新决策边界似乎提供更好的样本内拟合，交叉验证率收缩 66% 以上。

使用贝叶斯优化来优化 SVM 分类器拟合

此示例说明如何使用 **fitcsvm** 函数和 **OptimizeHyperparameters** 名称-值对组优化 SVM 分类。该分类基于高斯混合模型中点的位置来工作。有关该模型的描述，请参阅 The Elements of Statistical Learning，作者 Hastie、Tibshirani 和 Friedman (2009)，第 17 页。该模型从为“green”类生成 10 个基点开始，这些基点呈二维独立正态分布，均值为 (1,0) 且具有单位方差。它还为“red”类生成 10 个基点，这些基点呈二维独立正态分布，均值为 (0,1) 且具有单位方差。对于每个类（green 和 red），生成 100 个随机点，如下所示：

1 随机均匀选择合适颜色的一个基点 m。

2 生成一个呈二维正态分布的独立随机点，其均值为 m，方差为 I/5，其中 I 是 2×2 单位矩阵。在此示例中，使用方差 I/50 来更清楚地显示优化的优势。

生成点和分类器

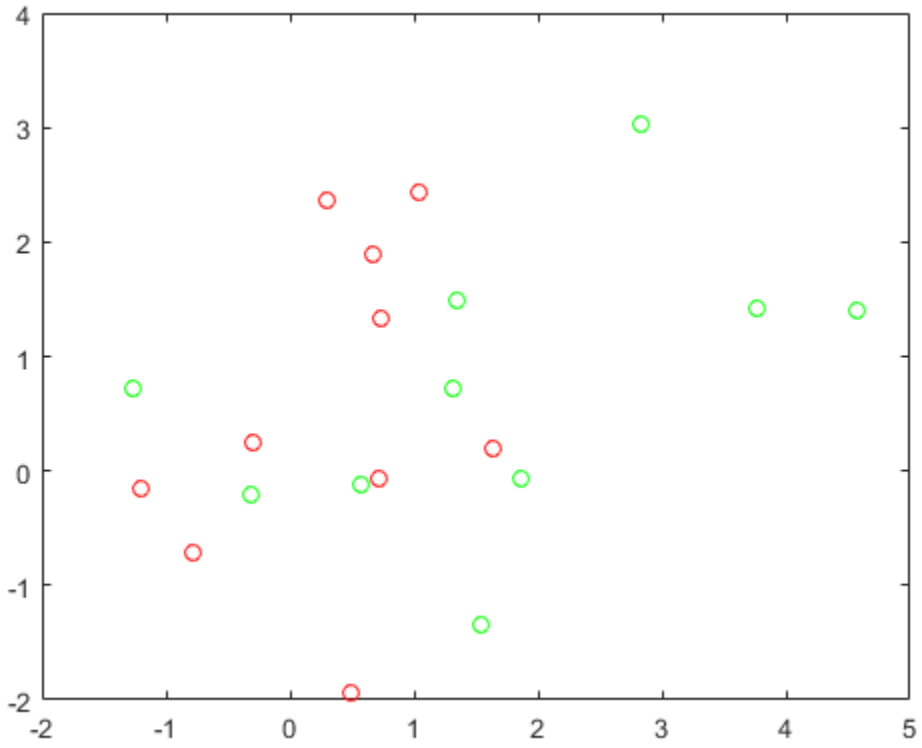
为每个类生成 10 个基点。

```
rng default % For reproducibility
grnpop = mvnrnd([1,0],eye(2),10);
redpop = mvnrnd([0,1],eye(2),10);
```

查看基点。

```
plot(grnpop(:,1),grnpop(:,2),'go')
hold on
```

```
plot(redpop(:,1),redpop(:,2),'ro')
hold off
```



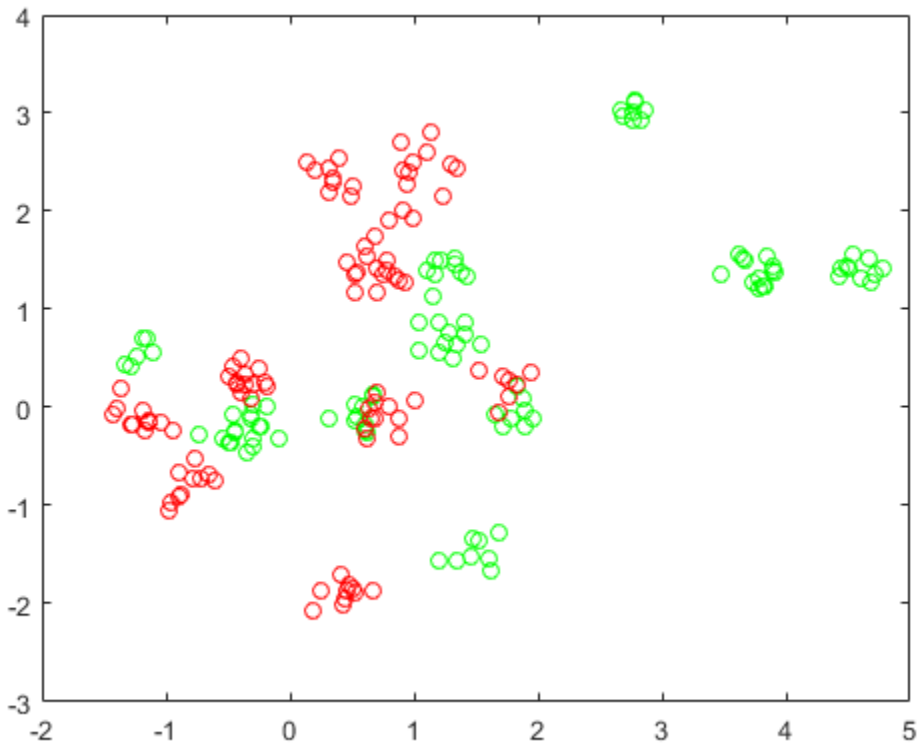
由于一些红色基点靠近绿色基点，因此很难仅基于位置对数据点进行分类。

生成每个类的 100 个数据点。

```
redpts = zeros(100,2);grnpts = redpts;
for i = 1:100
   grnpts(i,:) = mvnrnd(grnpop(randi(10),:),eye(2)*0.02);
   redpts(i,:) = mvnrnd(redpop(randi(10),:),eye(2)*0.02);
end
```

查看数据点。

```
figure
plot(grnpts(:,1),grnpts(:,2),'go')
hold on
plot(redpts(:,1),redpts(:,2),'ro')
hold off
```



为分类准备数据

将数据放入一个矩阵中，并创建向量 **grp**，该向量标记每个点的类。

```
cdata = [grnpts;redpts];
grp = ones(200,1);
% Green label 1, red label -1
grp(101:200) = -1;
```

准备交叉验证

为交叉验证设置一个分区。此步骤确定优化在每一步所使用的训练集和测试集。

```
c = cvpartition(200,'KFold',10);
```

优化拟合

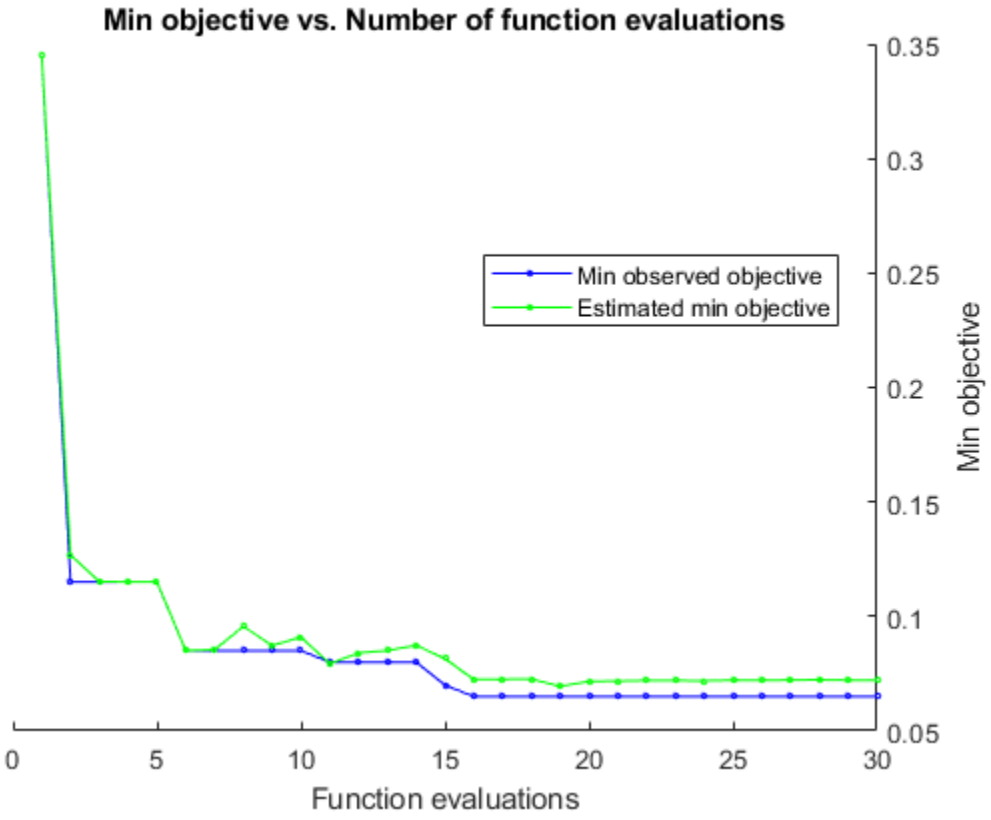
要找到好的拟合，即交叉验证损失低的拟合，请设置选项以使用贝叶斯优化。在所有优化中使用相同的交叉验证分区 **c**。

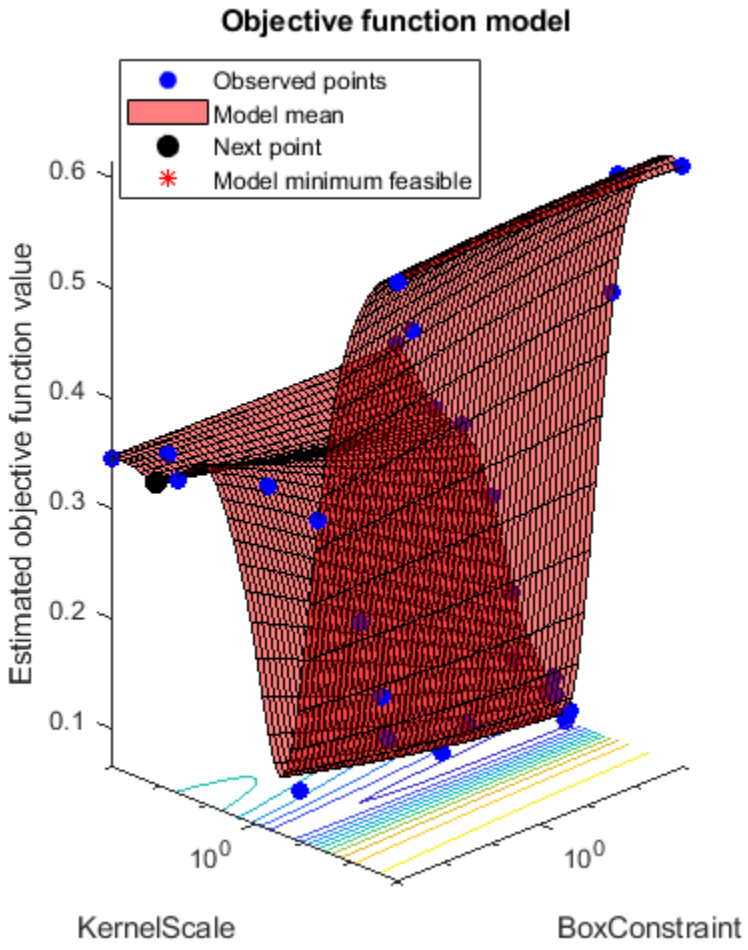
为了实现可再现性，请使用 **'expected-improvement-plus'** 采集函数。

```
opts = struct('Optimizer','bayesopt','ShowPlots',true,'CVPartition',c,...
   'AcquisitionFunctionName','expected-improvement-plus');
svmmod = fitcsvm(cdata,grp,'KernelFunction','rbf',...
   'OptimizeHyperparameters','auto','HyperparameterOptimizationOptions',opts)
```

```
|=====================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | BoxConstraint|  KernelScale |
```

```
|     | result |          | runtime     | (observed)  | (estim.)    |             |             |
|=====================================================================================================|
|   1 | Best   |    0.345 |    0.21295 |     0.345 |     0.345 |    0.00474 |     306.44 |
|   2 | Best   |    0.115 |    0.17149 |     0.115 |    0.12678 |     430.31 |     1.4864 |
|   3 | Accept |     0.52 |     0.1333 |     0.115 |     0.1152 |   0.028415 |   0.014369 |
|   4 | Accept |     0.61 |     0.2691 |     0.115 |    0.11504 |     133.94 |  0.0031427 |
|   5 | Accept |     0.34 |    0.35421 |     0.115 |    0.11504 |   0.010993 |     5.7742 |
|   6 | Best   |    0.085 |    0.11744 |     0.085 |   0.085039 |     885.63 |    0.68403 |
|   7 | Accept |    0.105 |    0.11036 |     0.085 |   0.085428 |     0.3057 |    0.58118 |
|   8 | Accept |     0.21 |    0.13858 |     0.085 |    0.09566 |    0.16044 |    0.91824 |
|   9 | Accept |    0.085 |     0.178 |     0.085 |    0.08725 |     972.19 |    0.46259 |
|  10 | Accept |      0.1 |     0.3127 |     0.085 |   0.090952 |     990.29 |     0.491 |
|  11 | Best   |     0.08 |    0.12274 |      0.08 |   0.079362 |     2.5195 |     0.291 |
|  12 | Accept |     0.09 |    0.11783 |      0.08 |    0.08402 |     14.338 |    0.44386 |
|  13 | Accept |      0.1 |    0.10892 |      0.08 |    0.08508 |  0.0022577 |    0.23803 |
|  14 | Accept |     0.11 |    0.10798 |      0.08 |   0.087378 |     0.2115 |    0.32109 |
|  15 | Best   |     0.07 |    0.17553 |      0.07 |   0.081507 |      910.2 |    0.25218 |
|  16 | Best   |    0.065 |    0.17637 |     0.065 |   0.072457 |     953.22 |    0.26253 |
|  17 | Accept |    0.075 |    0.26422 |     0.065 |   0.072554 |     998.74 |    0.23087 |
|  18 | Accept |    0.295 |    0.11691 |     0.065 |   0.072647 |     996.18 |     44.626 |
|  19 | Accept |     0.07 |    0.13689 |     0.065 |    0.06946 |     985.37 |    0.27389 |
|  20 | Accept |    0.165 |    0.11689 |     0.065 |   0.071622 |   0.065103 |    0.13679 |
|=====================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | BoxConstraint|  KernelScale |
|     | result |          | runtime     | (observed)  | (estim.)    |             |             |
|=====================================================================================================|
|  21 | Accept |    0.345 |    0.10901 |     0.065 |   0.071764 |      971.7 |     999.01 |
|  22 | Accept |     0.61 |    0.12691 |     0.065 |   0.071967 |  0.0010168 |  0.0010005 |
|  23 | Accept |    0.345 |    0.10781 |     0.065 |   0.071959 |  0.0010674 |     999.18 |
|  24 | Accept |     0.35 |    0.11365 |     0.065 |   0.071863 |  0.0010003 |     40.628 |
|  25 | Accept |     0.24 |     0.2432 |     0.065 |   0.072124 |     996.55 |     10.423 |
|  26 | Accept |     0.61 |     0.1947 |     0.065 |   0.072068 |     958.64 |  0.0010026 |
|  27 | Accept |     0.47 |     0.1147 |     0.065 |    0.07218 |     993.69 |   0.029723 |
|  28 | Accept |      0.3 |    0.10471 |     0.065 |   0.072291 |     993.15 |     170.01 |
|  29 | Accept |     0.16 |     0.2536 |     0.065 |   0.072104 |     992.81 |     3.8594 |
|  30 | Accept |    0.365 |    0.10891 |     0.065 |   0.072112 |  0.0010017 |   0.044287 |
```


Min objective vs. Number of function evaluations
Min observed objective
Estimated min objective
Min objective
Function evaluations
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
5
10
15
20
25
30




```
Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 50.7144 seconds
Total objective function evaluation time: 4.9196

Best observed feasible point:
    BoxConstraint    KernelScale
    _____________    ___________

       953.22          0.26253

Observed objective function value = 0.065
Estimated objective function value = 0.073726
Function evaluation time = 0.17637

Best estimated feasible point (according to models):
    BoxConstraint    KernelScale
    _____________    ___________
```

```
    985.37        0.27389

Estimated objective function value = 0.072112
Estimated function evaluation time = 0.1773

svmmod =
  ClassificationSVM
                 ResponseName: 'Y'
        CategoricalPredictors: []
                  ClassNames: [-1 1]
              ScoreTransform: 'none'
             NumObservations: 200
  HyperparameterOptimizationResults: [1x1 BayesianOptimization]
                       Alpha: [77x1 double]
                        Bias: -0.2352
            KernelParameters: [1x1 struct]
             BoxConstraints: [200x1 double]
            ConvergenceInfo: [1x1 struct]
            IsSupportVector: [200x1 logical]
                    Solver: 'SMO'


  Properties, Methods
```

计算优化模型的损失。

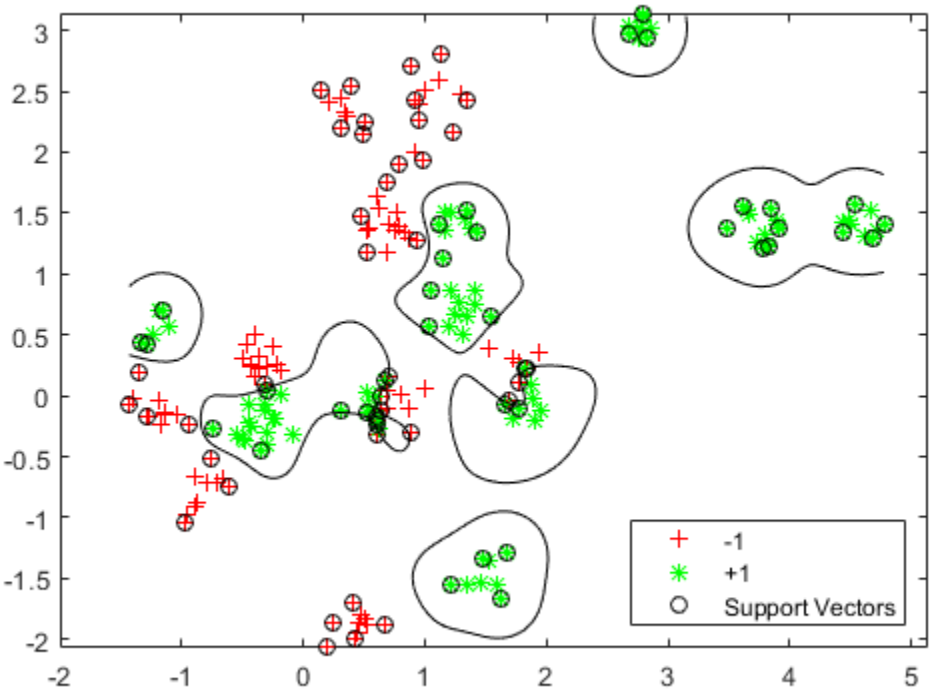
```
lossnew = kfoldLoss(fitcsvm(cdata,grp,'CVPartition',c,'KernelFunction','rbf',...
   'BoxConstraint',svmmod.HyperparameterOptimizationResults.XAtMinObjective.BoxConstraint,...
   'KernelScale',svmmod.HyperparameterOptimizationResults.XAtMinObjective.KernelScale))
```

```
lossnew = 0.0650
```

该损失与“观测的目标函数值”下优化输出中报告的损失相同。

可视化经过优化的分类器。

```
d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(cdata(:,1)):d:max(cdata(:,1)),...
   min(cdata(:,2)):d:max(cdata(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(svmmod,xGrid);
figure;
h = nan(3,1); % Preallocation
h(1:2) = gscatter(cdata(:,1),cdata(:,2),grp,'rg','+*');
hold on
h(3) = plot(cdata(svmmod.IsSupportVector,1),...
   cdata(svmmod.IsSupportVector,2),'ko');
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0 0],'k');
legend(h,{'-1','+1','Support Vectors'},'Location','Southeast');
axis equal
hold off
```



绘制 SVM 分类模型的后验概率区域

此示例说明如何预测 SVM 模型在观测网格上的后验概率，然后绘制该网格上的后验概率。绘制后验概率会显现出决策边界。

加载 Fisher 鸢尾花数据集。使用花瓣长度和宽度训练分类器，并从数据中删除海滨锦葵物种。

```
load fisheriris
classKeep = ~strcmp(species,'virginica');
X = meas(classKeep,3:4);
y = species(classKeep);
```

使用数据训练 SVM 分类器。指定类的顺序是很好的做法。

```
SVMModel = fitcsvm(X,y,'ClassNames',{'setosa','versicolor'});
```

估计最佳分数转换函数。

```
rng(1); % For reproducibility
[SVMModel,ScoreParameters] = fitPosterior(SVMModel);
```

```
Warning: Classes are perfectly separated. The optimal score-to-posterior transformation is a step function.
```

```
ScoreParameters
```

```
ScoreParameters = struct with fields:
                  Type: 'step'
```

```
          LowerBound: -0.8431
          UpperBound: 0.6897
PositiveClassProbability: 0.5000
```

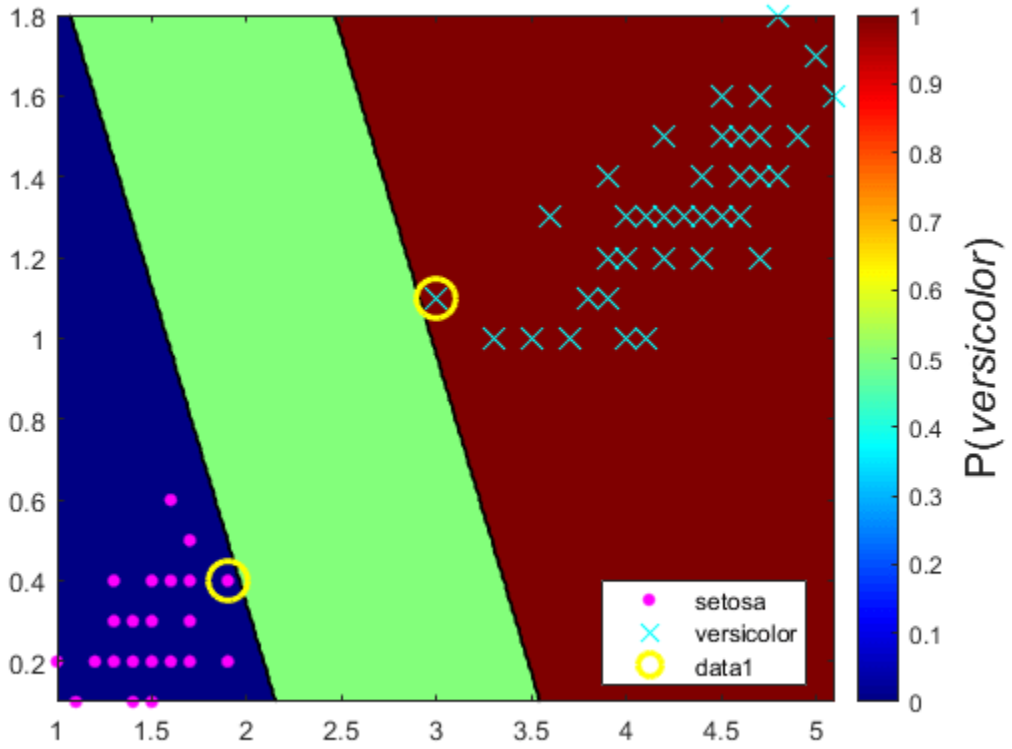
最佳分数转换函数是阶跃函数，因为类是可分离的。**ScoreParameters** 的字段 **LowerBound** 和 **UpperBound** 指示与类分离超平面（边距）内观测值对应的分数区间的下端点和上端点。没有训练观测值落在边距内。如果区间中有一个新分数，则软件会为对应的观测值分配一个正类后验概率，即 **ScoreParameters** 的 **PositiveClassProbability** 字段中的值。

在观测到的预测变量空间中定义一个数值网格。预测该网格中每个实例的后验概率。

```
xMax = max(X);
xMin = min(X);
d = 0.01;
[x1Grid,x2Grid] = meshgrid(xMin(1):d:xMax(1),xMin(2):d:xMax(2));

[~,PosteriorRegion] = predict(SVMModel,[x1Grid(:),x2Grid(:)]);
```

绘制正类后验概率区域和训练数据。

```
figure;
contourf(x1Grid,x2Grid,...
    reshape(PosteriorRegion(:,2),size(x1Grid,1),size(x1Grid,2)));
h = colorbar;
h.Label.String = 'P({\it{versicolor}})';
h.YLabel.FontSize = 16;
caxis([0 1]);
colormap jet;

hold on
gscatter(X(:,1),X(:,2),y,'mc','.x',[15,10]);
sv = X(SVMModel.IsSupportVector,:);
plot(sv(:,1),sv(:,2),'yo','MarkerSize',15,'LineWidth',2);
axis tight
hold off
```



在二类学习中，如果类是可分离的，则有三个区域：一个是包含正类后验概率为 0 的观测值的区域，另一个是包含正类后验概率为 1 的观测值的区域，还有一个是包含具有正类先验概率的观测值的区域。

使用线性支持向量机分析图像

此示例说明如何通过训练由线性 SVM 二类学习器组成的纠错输出编码 (ECOC) 模型来确定形状占据图像的哪个象限。此示例还说明存储支持向量、其标签和估计的 α 系数的 ECOC 模型的磁盘空间消耗。

创建数据集

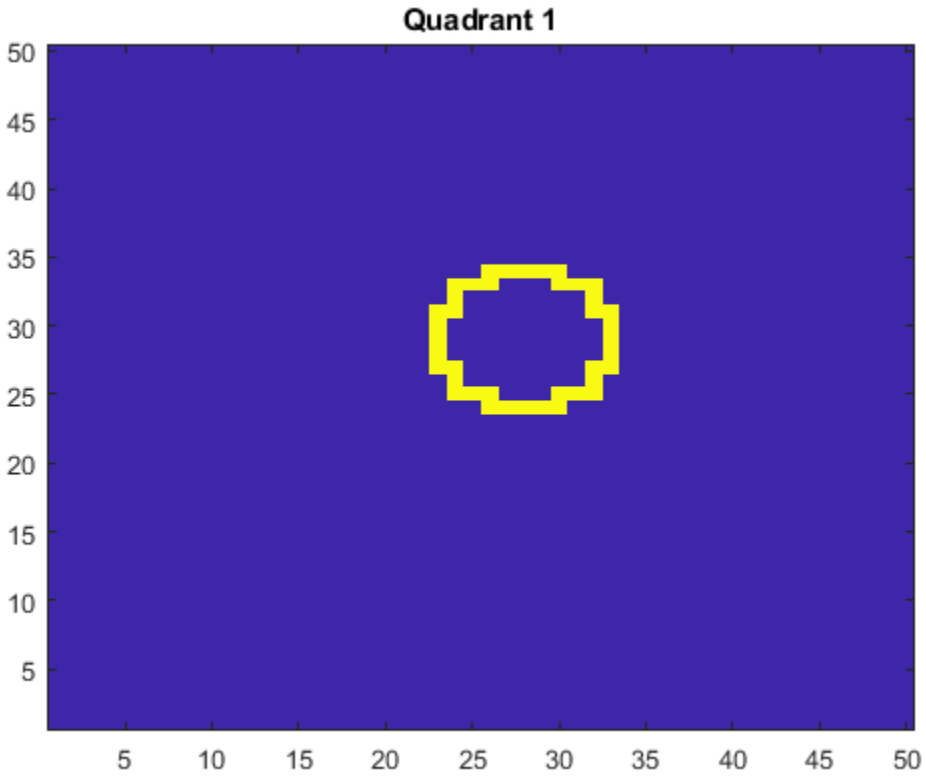
在一个 50×50 图像中随机放置一个半径为 5 的圆。生成 5000 个图像。为每个图像创建一个标签，指示圆占据的象限。象限 1 在右上角，象限 2 在左上角，象限 3 在左下角，象限 4 在右下角。预测变量是每个像素的强度。

```
d = 50;  % Height and width of the images in pixels
n = 5e4; % Sample size

X = zeros(n,d^2); % Predictor matrix preallocation
Y = zeros(n,1);   % Label preallocation
theta = 0:(1/d):(2*pi);
r = 5;          % Circle radius
rng(1);         % For reproducibility

for j = 1:n
   figmat = zeros(d);                % Empty image
```

```
    c = datasample((r + 1):(d - r - 1),2);   % Random circle center
    x = r*cos(theta) + c(1);                 % Make the circle
    y = r*sin(theta) + c(2);
    idx = sub2ind([d d],round(y),round(x));  % Convert to linear indexing
    figmat(idx) = 1;                         % Draw the circle
    X(j,:) = figmat(:);                      % Store the data
    Y(j) = (c(2) >= floor(d/2)) + 2*(c(2) < floor(d/2)) + ...
        (c(1) < floor(d/2)) + ...
        2*((c(1) >= floor(d/2)) & (c(2) < floor(d/2))); % Determine the quadrant
end
```

绘制观测值。

```
figure
imagesc(figmat)
h = gca;
h.YDir = 'normal';
title(sprintf('Quadrant %d',Y(end)))
```



训练 ECOC 模型

使用 25% 的留出样本，并指定训练和留出样本索引。

```
p = 0.25;
CVP = cvpartition(Y,'Holdout',p); % Cross-validation data partition
isIdx = training(CVP);            % Training sample indices
oosIdx = test(CVP);               % Test sample indices
```

创建一个 SVM 模板，它指定存储二类学习器的支持向量。将它和训练数据传递给 **fitcecoc** 以训练模型。确定训练样本分类误差。

```
t = templateSVM('SaveSupportVectors',true);
MdlSV = fitcecoc(X(isIdx,:),Y(isIdx),'Learners',t);
isLoss = resubLoss(MdlSV)
```

```
isLoss = 0
```

MdlSV 是经过训练的 **ClassificationECOC** 多类模型。它存储每个二类学习器的训练数据和支持向量。对于大型数据集，如图像分析中的数据集，该模型会消耗大量内存。

确定 ECOC 模型消耗的磁盘空间量。

```
infoMdlSV = whos('MdlSV');
mbMdlSV = infoMdlSV.bytes/1.049e6
```

```
mbMdlSV = 763.6150
```

该模型消耗 763.6 MB。

提高模型效率

您可以评估样本外的性能。您还可以使用不包含支持向量、其相关参数和训练数据的压缩模型来评估模型是否过拟合。

从经过训练的 ECOC 模型中丢弃支持向量和相关参数。然后，使用 **compact** 从生成的模型中丢弃训练数据。

```
Mdl = discardSupportVectors(MdlSV);
CMdl = compact(Mdl);
info = whos('Mdl','CMdl');
[bytesCMdl,bytesMdl] = info.bytes;
memReduction = 1 - [bytesMdl bytesCMdl]/infoMdlSV.bytes
```

```
memReduction = 1×2

    0.0626    0.9996
```

在本例中，丢弃支持向量可将内存消耗减少大约 6%。压缩和丢弃支持向量会将大小减小大约 99.96%。

管理支持向量的另一种方法是通过指定更大的框约束（如 100）来减少训练过程中的支持向量数量。虽然使用更少支持向量的 SVM 模型更可取并且消耗更少的内存，但增加框约束的值往往会增加训练时间。

从工作区中删除 **MdlSV** 和 **Mdl**。

```
clear Mdl MdlSV
```

评估留出样本性能

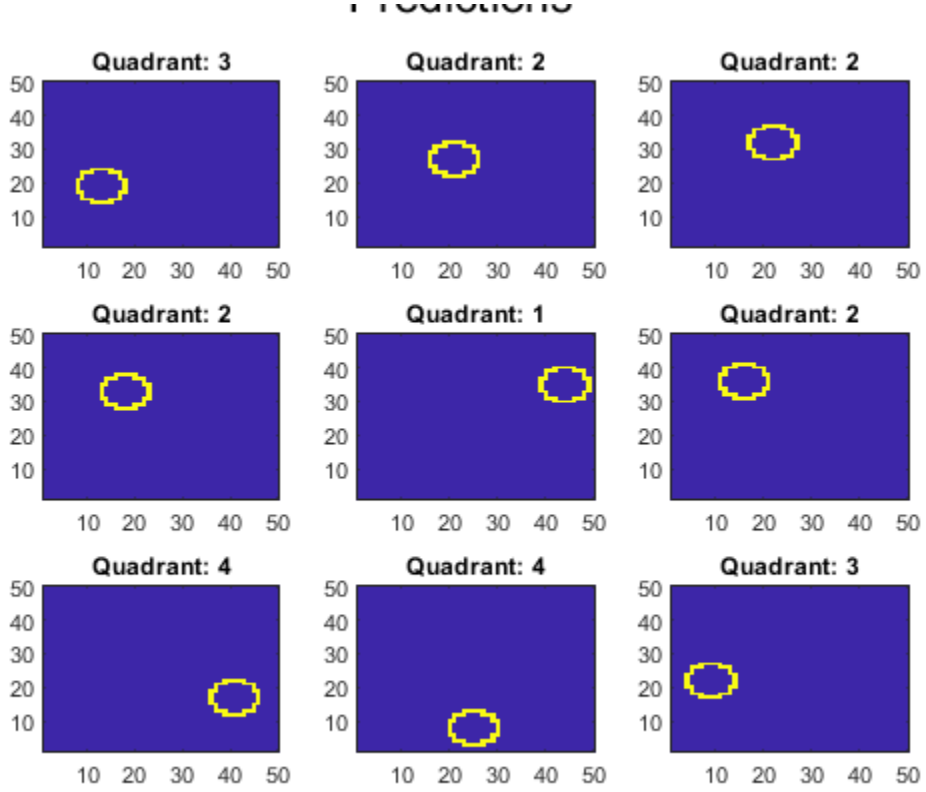
计算留出样本的分类误差。绘制留出样本预测的样本。

```
oosLoss = loss(CMdl,X(oosIdx,:),Y(oosIdx))
```

```
oosLoss = 0
```

```
yHat = predict(CMdl,X(oosIdx,:));
nVec = 1:size(X,1);
```

```
oosIdx = nVec(oosIdx);

figure;
for j = 1:9
   subplot(3,3,j)
   imagesc(reshape(X(oosIdx(j),:),[d d]))
   h = gca;
   h.YDir = 'normal';
   title(sprintf('Quadrant: %d',yHat(j)))
end
text(-1.33*d,4.5*d + 1,'Predictions','FontSize',17)
```



该模型不会对任何留出样本观测值进行误分类。

另请参阅

fitcsvm | **bayesopt** | **kfoldLoss**

详细信息

- “Train Support Vector Machines Using Classification Learner App”
- “Optimize Cross-Validated Classifier Using bayesopt”

参考

[1] Hastie, T., R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*, second edition. New York: Springer, 2008.

[2] Christianini, N., and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.

[3] Fan, R.-E., P.-H. Chen, and C.-J. Lin. "Working set selection using second order information for training support vector machines." *Journal of Machine Learning Research*, Vol 6, 2005, pp. 1889–1918.

[4] Kecman V., T. -M. Huang, and M. Vogt. "Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance." In *Support Vector Machines: Theory and Applications*. Edited by Lipo Wang, 255–274. Berlin: Springer-Verlag, 2005.

参考书目

[1] Agresti, A. *Categorical Data Analysis*, 2nd Ed. Hoboken, NJ: John Wiley & Sons, Inc., 2002.

[2] Allwein, E., R. Schapire, and Y. Singer. "Reducing multiclass to binary: A unifying approach for margin classifiers." *Journal of Machine Learning Research*. Vol. 1, 2000, pp. 113–141.

[3] Alpaydin, E. "Combined 5 x 2 CV F Test for Comparing Supervised Classification Learning Algorithms." *Neural Computation*, Vol. 11, No. 8, 1999, pp. 1885–1992.

[4] Blackard, J. A. and D. J. Dean. "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables". *Computers and Electronics in Agriculture* Vol. 24, Issue 3, 1999, pp. 131–151.

[5] Bottou, L., and Chih-Jen Lin. "Support Vector Machine Solvers." *Large Scale Kernel Machines* (L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, eds.). Cambridge, MA: MIT Press, 2007.

[6] Bouckaert. R. "Choosing Between Two Learning Algorithms Based on Calibrated Tests." *International Conference on Machine Learning*, pp. 51–58, 2003.

[7] Bouckaert, R. and E. Frank. "Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms." *In Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference*, 2004, pp. 3–12.

[8] Breiman, L. "Bagging Predictors." *Machine Learning* 26, 1996, pp. 123–140.

[9] Breiman, L. "Random Forests." *Machine Learning* 45, 2001, pp. 5–32.

[10] Breiman, L. **https://www.stat.berkeley.edu/~breiman/RandomForests/**

[11] Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Boca Raton, FL: Chapman & Hall, 1984.

[12] Christianini, N., and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.

[13] Dietterich, T. "Approximate statistical tests for comparing supervised classification learning algorithms." *Neural Computation*, Vol. 10, No. 7, 1998, pp. 1895–1923.

[14] Dietterich, T., and G. Bakiri. "Solving Multiclass Learning Problems Via Error-Correcting Output Codes." *Journal of Artificial Intelligence Research*. Vol. 2, 1995, pp. 263–286.

[15] Escalera, S., O. Pujol, and P. Radeva. "On the decoding process in ternary error-correcting output codes." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 32, Issue 7, 2010, pp. 120–134.

[16] Escalera, S., O. Pujol, and P. Radeva. "Separability of ternary codes for sparse designs of error-correcting output codes." *Pattern Recogn*. Vol. 30, Issue 3, 2009, pp. 285–297.

[17] Fan, R.-E., P.-H. Chen, and C.-J. Lin. "Working set selection using second order information for training support vector machines." *Journal of Machine Learning Research*, Vol 6, 2005, pp. 1889–1918.

[18] Fagerlan, M.W., S Lydersen, P. Laake. "The McNemar Test for Binary Matched-Pairs Data: Mid-p and Asymptotic Are Better Than Exact Conditional." *BMC Medical Research Methodology*. Vol. 13, 2013, pp. 1–8.

[19] Freund, Y. "A more robust boosting algorithm." arXiv:0905.2138v1, 2009.

[20] Freund, Y. and R. E. Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." *J. of Computer and System Sciences*, Vol. 55, 1997, pp. 119–139.

[21] Friedman, J. "Greedy function approximation: A gradient boosting machine." *Annals of Statistics,* Vol. 29, No. 5, 2001, pp. 1189–1232.

[22] Friedman, J., T. Hastie, and R. Tibshirani. "Additive logistic regression: A statistical view of boosting." *Annals of Statistics*, Vol. 28, No. 2, 2000, pp. 337–407.

[23] Hastie, T., and R. Tibshirani. "Classification by Pairwise Coupling." *Annals of Statistics*. Vol. 26, Issue 2, 1998, pp. 451–471.

[24] Hastie, T., R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*, second edition. New York: Springer, 2008.

[25] Ho, C. H. and C. J. Lin. "Large-Scale Linear Support Vector Regression." *Journal of Machine Learning Research*, Vol. 13, 2012, pp. 3323–3348.

[26] Ho, T. K. "The random subspace method for constructing decision forests." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 20, No. 8, 1998, pp. 832–844.

[27] Hsieh, C. J., K. W. Chang, C. J. Lin, S. S. Keerthi, and S. Sundararajan. "A Dual Coordinate Descent Method for Large-Scale Linear SVM." *Proceedings of the 25th International Conference on Machine Learning, ICML ' 08*, 2001, pp. 408–415.

[28] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. *A Practical Guide to Support Vector Classification*. Available at **https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf**.

[29] Hu, Q., X. Che, L. Zhang, and D. Yu. "Feature Evaluation and Selection Based on Neighborhood Soft Margin." *Neurocomputing*. Vol. 73, 2010, pp. 2114–2124.

[30] Kecman V., T. -M. Huang, and M. Vogt. "Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance." In *Support Vector Machines: Theory and Applications*. Edited by Lipo Wang, 255–274. Berlin: Springer-Verlag, 2005.

[31] Kohavi, R. "Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid." *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[32] Lancaster, H.O. "Significance Tests in Discrete Distributions." *JASA*, Vol. 56, Number 294, 1961, pp. 223–234.

[33] Langford, J., L. Li, and T. Zhang. "Sparse Online Learning Via Truncated Gradient." *J. Mach. Learn. Res.*, Vol. 10, 2009, pp. 777–801.

[34] Loh, W.Y. "Regression Trees with Unbiased Variable Selection and Interaction Detection." *Statistica Sinica*, Vol. 12, 2002, pp. 361–386.

[35] Loh, W.Y. and Y.S. Shih. "Split Selection Methods for Classification Trees." *Statistica Sinica*, Vol. 7, 1997, pp. 815–840.

[36] McNemar, Q. "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages." *Psychometrika*, Vol. 12, Number 2, 1947, pp. 153–157.

[37] Meinshausen, N. "Quantile Regression Forests." *Journal of Machine Learning Research*, Vol. 7, 2006, pp. 983–999.

[38] Mosteller, F. "Some Statistical Problems in Measuring the Subjective Response to Drugs." *Biometrics*, Vol. 8, Number 3, 1952, pp. 220–226.

[39] Nocedal, J. and S. J. Wright. *Numerical Optimization*, 2nd ed., New York: Springer, 2006.

[40] Schapire, R. E. et al. "Boosting the margin: A new explanation for the effectiveness of voting methods." *Annals of Statistics,* Vol. 26, No. 5, 1998, pp. 1651–1686.

[41] Schapire, R., and Y. Singer. "Improved boosting algorithms using confidence-rated predictions." *Machine Learning,* Vol. 37, No. 3, 1999, pp. 297–336.

[42] Shalev-Shwartz, S., Y. Singer, and N. Srebro. "Pegasos: Primal Estimated Sub-Gradient Solver for SVM." *Proceedings of the 24th International Conference on Machine Learning, ICML ′ 07*, 2007, pp. 807–814.

[43] Seiffert, C., T. Khoshgoftaar, J. Hulse, and A. Napolitano. "RUSBoost: Improving classification performance when training data is skewed." *19th International Conference on Pattern Recognition,* 2008, pp. 1–4.

[44] Warmuth, M., J. Liao, and G. Ratsch. "Totally corrective boosting algorithms that maximize the margin." *Proc. 23rd Int′ l. Conf. on Machine Learning, ACM,* New York, 2006, pp. 1001–1008.

[45] Wu, T. F., C. J. Lin, and R. Weng. "Probability Estimates for Multi-Class Classification by Pairwise Coupling." *Journal of Machine Learning Research*. Vol. 5, 2004, pp. 975–1005.

[46] Wright, S. J., R. D. Nowak, and M. A. T. Figueiredo. "Sparse Reconstruction by Separable Approximation." *Trans. Sig. Proc.*, Vol. 57, No 7, 2009, pp. 2479–2493.

[47] Xiao, Lin. "Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization." *J. Mach. Learn. Res.*, Vol. 11, 2010, pp. 2543–2596.

[48] Xu, Wei. "Towards Optimal One Pass Large Scale Learning with Averaged Stochastic Gradient Descent." *CoRR*, abs/1107.2490, 2011.

[49] Zadrozny, B. "Reducing Multiclass to Binary by Coupling Probability Estimates." *NIPS 2001: Proceedings of Advances in Neural Information Processing Systems 14*, 2001, pp. 1041–1048.

[50] Zadrozny, B., J. Langford, and N. Abe. "Cost-Sensitive Learning by Cost-Proportionate Example Weighting." *Third IEEE International Conference on Data Mining*, 435–442. 2003.

[51] Zhou, Z.-H. and X.-Y. Liu. "On Multi-Class Cost-Sensitive Learning." *Computational Intelligence.* Vol. 26, Issue 3, 2010, pp. 232–257 CiteSeerX.

决策树

判别分析

朴素贝叶斯

高维数据的分类和回归

分类学习器

回归学习器

支持向量机

增量学习

马尔可夫模型

试验设计

统计过程控制

tall 数组

并行统计

代码生成

函数

boxplot

用箱线图可视化汇总统计量

语法

boxplot(x)
boxplot(x,g)

boxplot(ax, ___)

boxplot(___ ,Name,Value)

说明

boxplot(x) 创建 **x** 中数据的箱线图。如果 **x** 是向量，**boxplot** 绘制一个箱子。如果 **x** 是矩阵，**boxplot** 为 **x** 的每列绘制一个箱子。

在每个箱子上，中心标记表示中位数，箱子的底边和顶边分别表示第 25 个和 75 个百分位数。须线会延伸到不是离群值的最远端数据点，离群值会以 '+' 符号单独绘制。

boxplot(x,g) 使用 **g** 中包含的一个或多个分组变量创建箱线图。**boxplot** 为具有相同的一个或多个 **g** 值的各组 **x** 值创建一个单独的箱子

boxplot(ax, ___) 使用坐标区图形对象 **ax** 指定的坐标区和任何上述语法创建箱线图。

boxplot(___ ,Name,Value) 使用由一个或多个 **Name,Value** 对组参数指定的附加选项创建箱线图。例如，您可以指定箱子样式或顺序。

示例

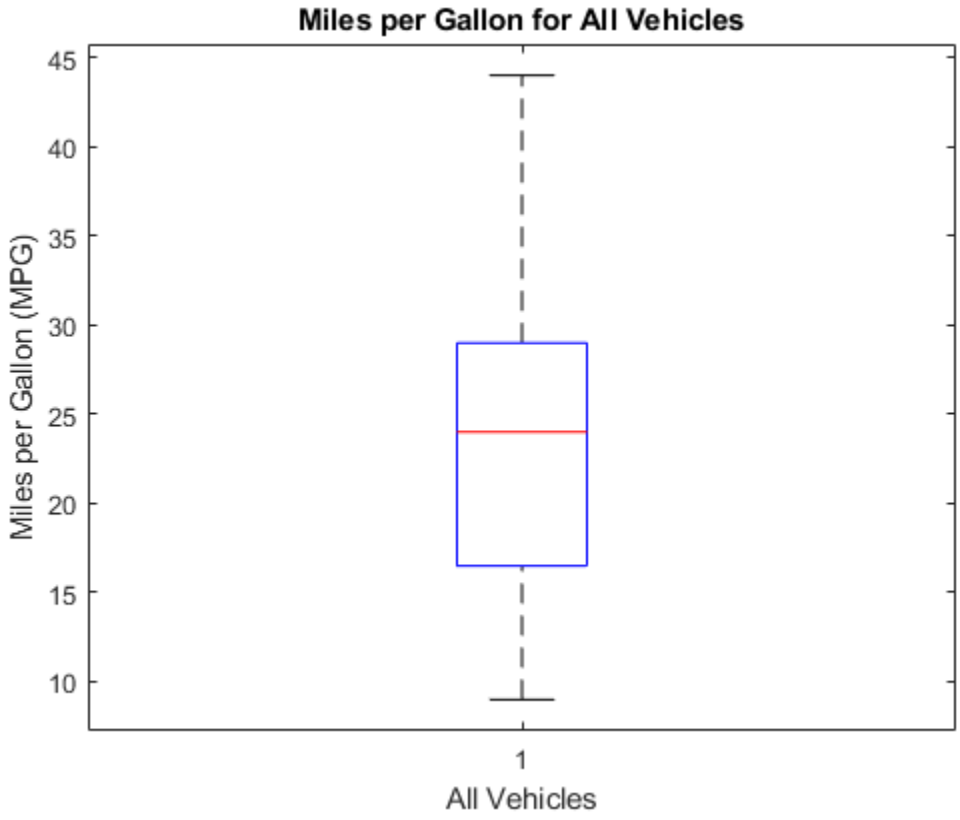
创建箱线图

加载样本数据。

```
load carsmall
```

创建一个表示每加仑英里数 (MPG) 测量值的箱线图。添加标题并为坐标区加标签。

```
boxplot(MPG)
xlabel('All Vehicles')
ylabel('Miles per Gallon (MPG)')
title('Miles per Gallon for All Vehicles')
```



箱线图显示，样本数据中所有车辆的每加仑英里数的中位数约为 24 英里。最小值约为 9，最大值约为 44。

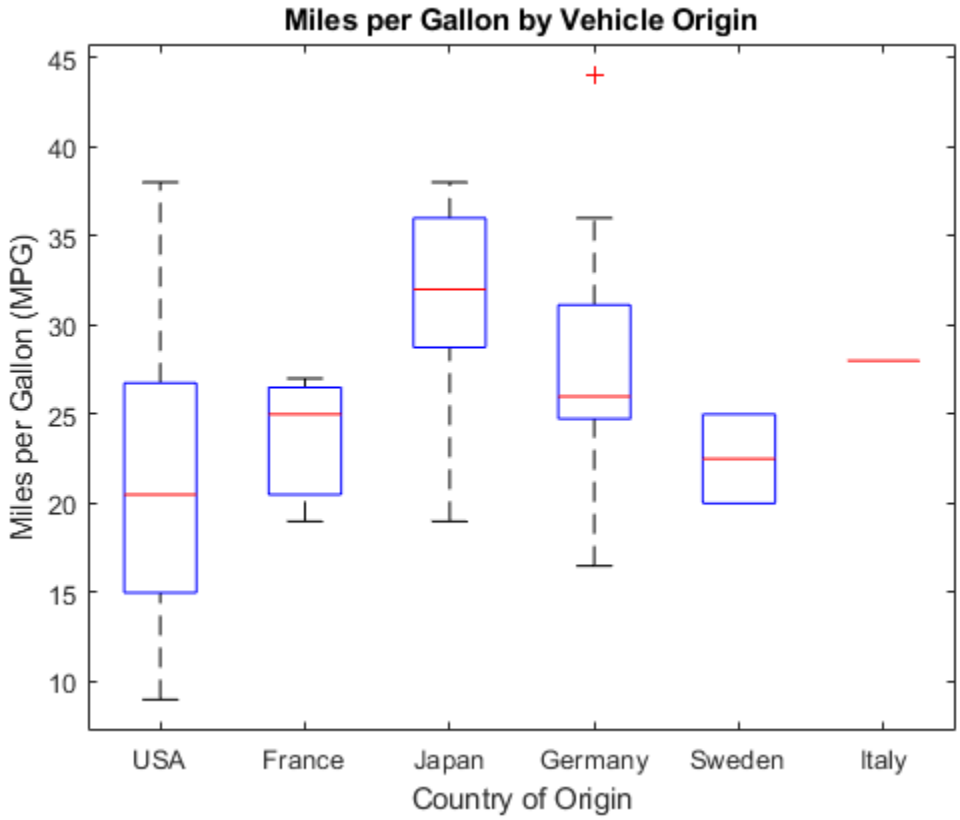
为分组数据创建箱线图

加载样本数据。

```
load carsmall
```

根据样本数据创建每加仑英里数 (MPG) 测量值的箱线图，按车辆的原产国 (Origin) 分组。添加标题并为坐标区加标签。

```
boxplot(MPG,Origin)
title('Miles per Gallon by Vehicle Origin')
xlabel('Country of Origin')
ylabel('Miles per Gallon (MPG)')
```



每个箱子直观地表示来自指定国家/地区的汽车的 MPG 数据。意大利的“箱子”显示为一条线，因为样本数据只包含该组的一个观测值。

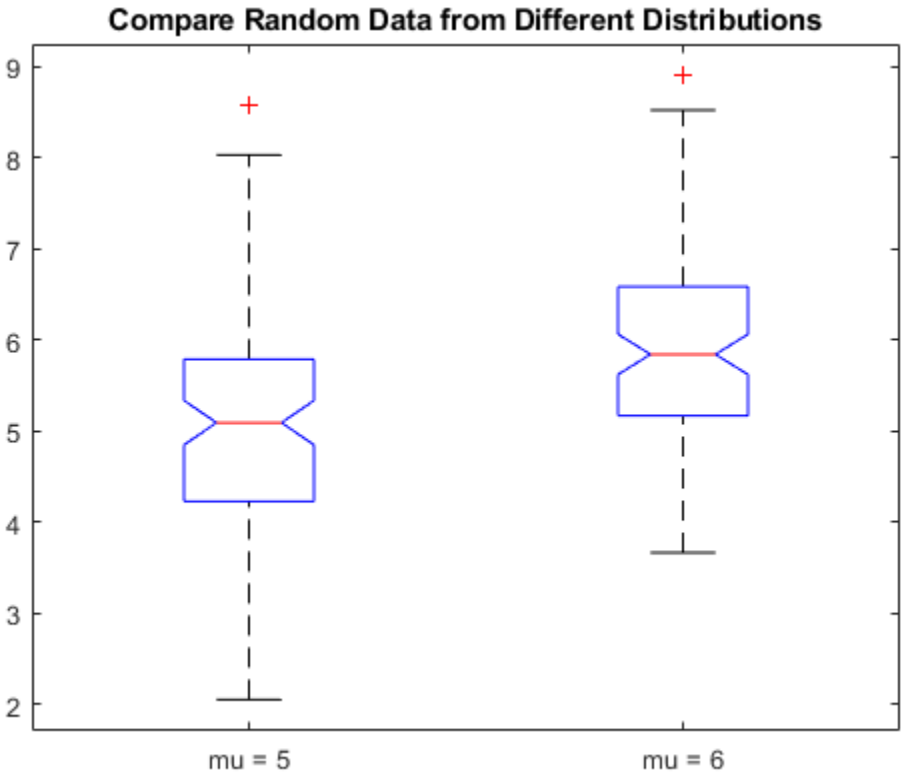
创建带缺口的箱线图

生成两组样本数据。第一个样本 **x1** 包含从 **mu = 5** 和 **sigma = 1** 的正态分布生成的随机数。第二个样本 **x2** 包含从 **mu = 6** 和 **sigma = 1** 的正态分布生成的随机数。

```
rng default  % For reproducibility
x1 = normrnd(5,1,100,1);
x2 = normrnd(6,1,100,1);
```

创建 **x1** 和 **x2** 的带缺口的箱线图。用对应的 **mu** 值对每个箱子加标签。

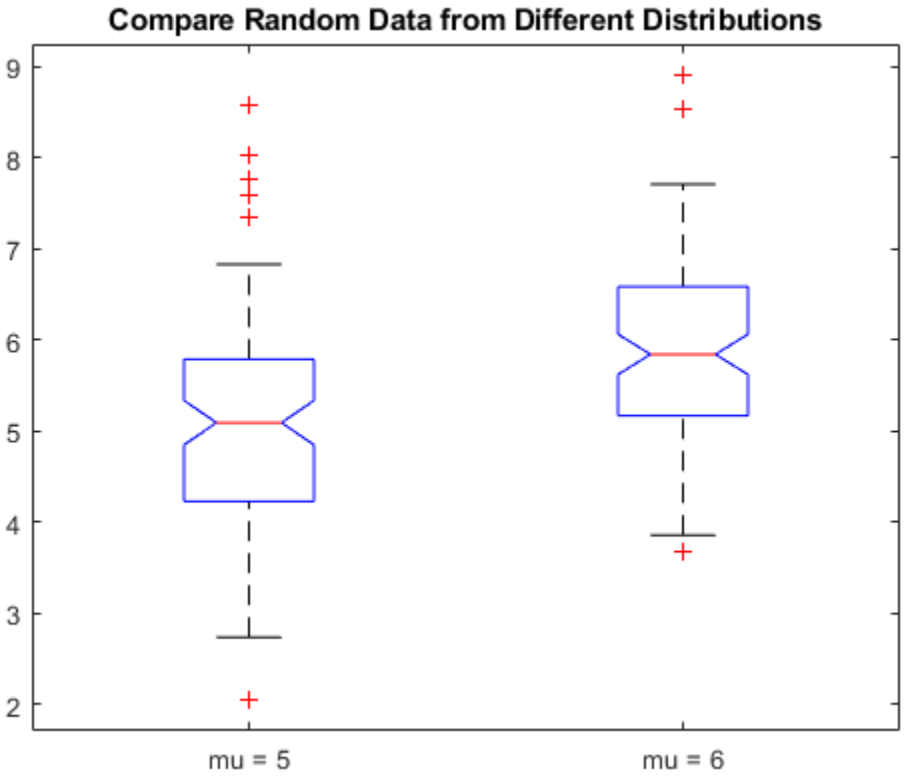
```
figure
boxplot([x1,x2],'Notch','on','Labels',{'mu = 5','mu = 6'})
title('Compare Random Data from Different Distributions')
```



箱线图显示两个组的中位数之间的差值约为 1。由于箱线图中的缺口不重叠，可以有 95% 的置信度认为真实的中位数不同。

下图显示了相同数据的箱线图，其中最大须线长度指定为四分位差的 1.0 倍。须线以外的数据点使用 + 显示。

```
figure
boxplot([x1,x2],'Notch','on','Labels',{'mu = 5','mu = 6'},'Whisker',1)
title('Compare Random Data from Different Distributions')
```



须线越短，**boxplot** 将越多的数据点显示为离群值。

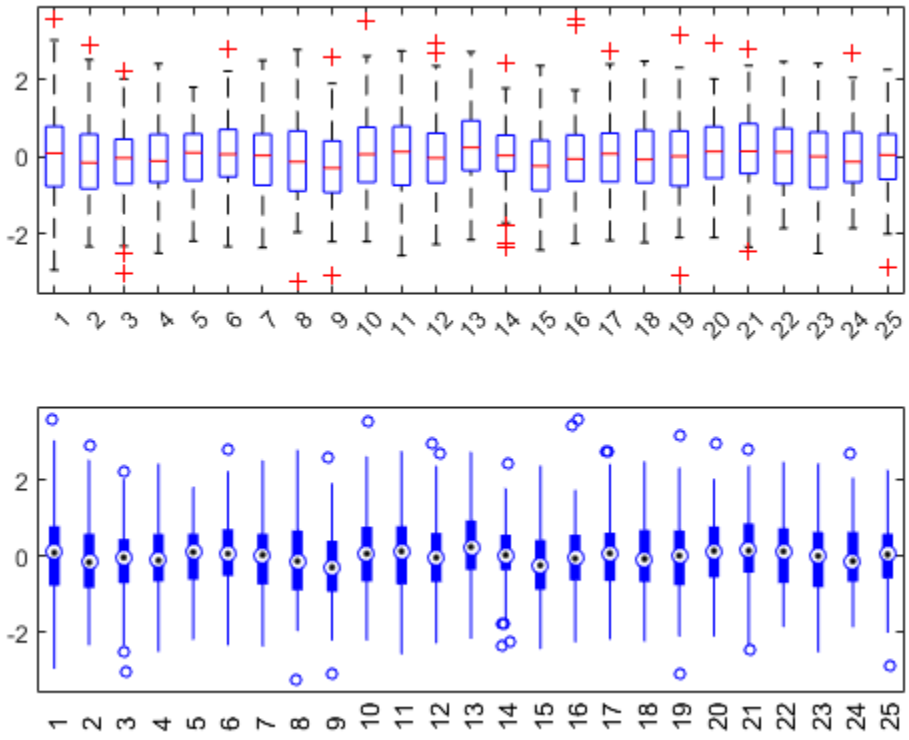
创建紧凑箱线图

创建从标准正态分布生成的一个 100×25 随机数矩阵，用作样本数据。

```
rng default  % For reproducibility
x = randn(100,25);
```

在同一图上为 **x** 中的数据创建两个箱线图。顶部图使用默认格式，底部图使用紧凑格式。

```
figure

subplot(2,1,1)
boxplot(x)

subplot(2,1,2)
boxplot(x,'PlotStyle','compact')
```



每个图显示相同的数据，但紧凑格式可以提高具有多个箱子的图的可读性。

可变长度向量的箱线图

通过使用分组变量为不同长度的数据向量创建箱线图。

随机生成三个不同长度的列向量：长度分别为 **5**、**10** 和 **15**。将数据合并为一个长度为 **30** 的列向量。

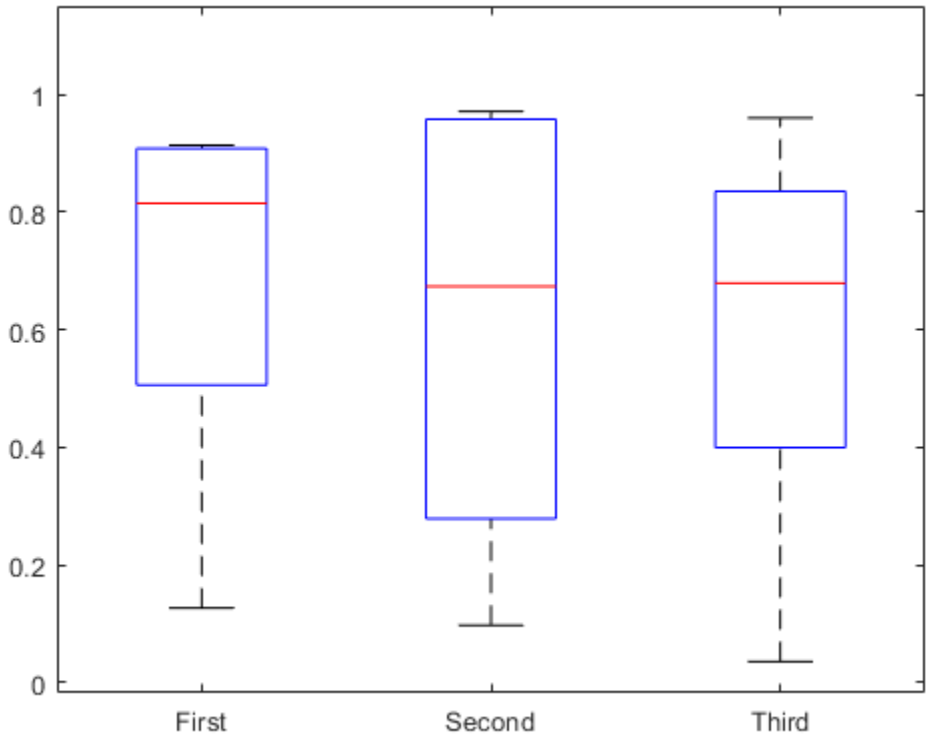
```
rng('default')  % For reproducibility
x1 = rand(5,1);
x2 = rand(10,1);
x3 = rand(15,1);
x = [x1; x2; x3];
```

创建一个分组变量，将同一值赋给 **x** 中来自于同一向量的行。例如，**g** 的前五行具有相同的值 **First**，因为 **x** 的前五行都来自同一个向量 **x1**。

```
g1 = repmat({'First'},5,1);
g2 = repmat({'Second'},10,1);
g3 = repmat({'Third'},15,1);
g = [g1; g2; g3];
```

创建箱线图。

```
boxplot(x,g)
```



输入参数

x - 输入数据
数值向量 | 数值矩阵

输入数据，指定为数值向量或数值矩阵。如果 **x** 是向量，**boxplot** 绘制一个箱子。如果 **x** 是矩阵，**boxplot** 为 **x** 的每列绘制一个箱子。

在每个箱子上，中心标记表示中位数，箱子的底边和顶边分别表示第 25 个和 75 个百分位数。须线会延伸到不是离群值的最远端数据点，离群值会以 '+' 符号单独绘制。

数据类型： **single** | **double**

g - 分组变量
数值向量 | 字符数组 | 字符串数组 | 元胞数组 | 分类数组

分组变量，指定为数值向量、字符数组、字符串数组、元胞数组或分类数组。您可以通过使用上述变量类型的元胞数组或矩阵在 **g** 中指定多个分组变量。如果指定多个分组变量，它们必须具有相同的长度。

如果 **x** 是向量，则对于 **x** 的每个元素，分组变量都包含一个对应的行。如果 **x** 是矩阵，则对于 **x** 的每个列，分组变量都包含一个对应的行。分组变量中包含缺失值 (**NaN**)、空字符向量、空值或 **<missing>** 字符串或 **<undefined>** 值的组将被忽略，并且不会计入其他参数考虑的组数中。

默认情况下，**boxplot** 按在数据中最初出现的顺序对字符和字符串分组变量进行排序，按水平顺序对分类分组变量进行排序，并按数值顺序对数值分组变量进行排序。要控制组的顺序，请执行以下操作之一：

- 在 **g** 中使用分类变量，并指定其水平的顺序。
- 使用 **'GroupOrder'** 名称-值对组参数。
- 对您的数据进行预先排序。

数据类型： **single** | **double** | **char** | **string** | **cell** | **categorical**

ax - 用于绘图的坐标区
坐标区图形对象

用于绘图的坐标区，指定为坐标区图形对象。如果未指定 **ax**，则 **boxplot** 使用当前轴创建绘图。有关创建坐标区图形对象的详细信息，请参阅 **axes** 和 Axes。

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例： **'Notch','on','Labels',{'mu = 5','mu = 6'}** 创建带缺口的箱线图，并从左到右对两个箱子 **mu = 5** 和 **mu = 6** 加标签

箱子外观

BoxStyle - 箱子样式
'outline' | **'filled'**

箱子样式，指定为以逗号分隔的对组，其中包含 **'BoxStyle'** 和下列项之一。

名称	值
'outline'	使用带虚须线的空心箱绘制箱子。如果 **'PlotStyle'** 是 **'traditional'**，这是默认值。
'filled'	使用带实须线的窄实心箱绘制箱子。如果 **'PlotStyle'** 是 **'compact'**，这是默认值。

示例： **'BoxStyle','filled'**

Colors - 箱子颜色
RGB 三元组 | 颜色名称的字符向量或字符串标量

箱子颜色，指定为以逗号分隔的对组，其中包含 **'Colors'** 和 RGB 三元组、字符向量或字符串标量。RGB 三元组是包含三个元素的行向量，其元素分别指定颜色中红、绿、蓝分量的强度。每个强度必须在 [0,1] 范围内。

下表列出了可用的颜色字符及其等效的 RGB 三元组值。

长名称	短名称	RGB 三元组
黄色	**'y'**	**[1 1 0]**
品红	**'m'**	**[1 0 1]**
青色	**'c'**	**[0 1 1]**
红色	**'r'**	**[1 0 0]**
绿色	**'g'**	**[0 1 0]**

长名称	短名称	RGB 三元组
蓝色	'b'	[0 0 1]
白色	'w'	[1 1 1]
黑色	'k'	[0 0 0]

您可以将多种颜色指定为字符向量或颜色名称的字符串标量（例如，'rgbm'）或包含 RGB 值的三列矩阵。序列会根据需要复制或截断，例如，'rb' 会交替显示红色和蓝色的箱子。

如果未指定名称-值对组 'ColorGroup'，则 boxplot 对所有箱子使用相同的颜色方案。如果您指定 'ColorGroup'，则默认为修改后的 hsv colormap。

示例： 'Colors','rgbm'

MedianStyle - 中位数样式
'line' | 'target'

中位数样式，指定为以逗号分隔的对组，其中包含 'MedianStyle' 和下列项之一。

名称	值
'line'	画一条线来表示每个箱子中的中位数。当 'PlotStyle' 是 'traditional' 时，这是默认值。
'target'	绘制带黑心的圆来表示每个箱子的中位数。当 'PlotStyle' 是 'compact' 时，这是默认值。

示例： 'MedianStyle','target'

Notch - 比较区间的标记
'off'（默认）| 'on' | 'marker'

比较区间的标记，指定为以逗号分隔的对组，其中包含 'Notch' 和下列项之一。

名称	值
'off'	在箱子显示中省略比较区间。
'on'	如果 'PlotStyle' 是 'traditional'，则使用缺口绘制比较区间。如果 'PlotStyle' 是 'compact'，则使用三角形标记绘制比较区间。
'marker'	使用三角形标记绘制比较区间。

如果两个中位数的区间不重叠，则这两个中位数在 5% 显著性水平上有显著差异。boxplot 使用缺口的极值或三角形标记的中心表示区间端点。缺口极值对应于 $q_2 - 1.57(q_3 - q_1)/\text{sqrt}(n)$ 和 $q_2 + 1.57(q_3 - q_1)/\text{sqrt}(n)$，其中 q_2 是中位数（第 50 百分位数），q_1 和 q_3 分别是第 25 个和第 75 个百分位数，n 是没有任何 NaN 值的观测值的数目。如果样本量很小，缺口可能会超出箱子的末端。

示例： 'Notch','on'

OutlierSize - 离群值的标记大小
正数值

离群值的标记大小，指定为由 'OutlierSize' 和正数值组成的以逗号分隔的对组。指定的值表示以磅为单位的标记大小。

如果 **'PlotStyle'** 是 **'traditional'**，则 **OutlierSize** 的默认值为 6。如果 **'PlotStyle'** 是 **'compact'**，则 **OutlierSize** 的默认值为 4。

示例： **'OutlierSize',8**

数据类型： **single** | **double**

PlotStyle - 绘图样式
'traditional' （默认） | **'compact'**

绘图样式，指定为以逗号分隔的对组，其中包含 **'PlotStyle'** 和下列项之一。

名称	值
'traditional'	使用传统箱子样式绘制箱子。
'compact'	使用较小的箱子样式绘制箱子，该箱子样式适用于具有许多组的绘图。此样式会更改其他一些参数的默认值。

示例： **'PlotStyle','compact'**

Symbol - 离群值的符号和颜色
线条设定

离群值的符号和颜色，指定为以逗号分隔的对组，其中包含 **'Symbol'** 和线条设定。有关可用的线条设定，请参阅 **plot** 中的 **LineSpec** 参数。

如果 **'PlotStyle'** 是 **'traditional'**，则默认值为 **'r+'**，它使用红色 **'+'** 符号绘制每个离群值。

如果 **'PlotStyle'** 是 **'compact'**，则默认值为 **'o'**，它使用与对应箱子颜色相同的 **'o'** 符号绘制每个离群值。

如果省略符号，则离群值不可见。如果省略颜色，则离群值将以与箱子相同的颜色出现。

示例： **'kx'**

Widths - 箱子宽度
数值标量 | 数值向量

箱子宽度，指定为以逗号分隔的对组，其中包含 **'Widths'** 和数值标量或数值向量。如果箱子的数量不等于指定的宽度值的数量，则根据需要复制或截断值列表。

此名称-值对组参数不会更改箱子之间的间距。因此，如果您为 **'Widths'** 指定较大的值，这些箱子可能会重叠。

当 **'Positions'** 名称-值对组参数取默认值时，默认箱子宽度等于箱子间最小间距的一半，即 0.5。

示例： **'Widths',0.3**

数据类型： **single** | **double**

组外观

ColorGroup - 决定箱子颜色是否变化的分组变量
[] （默认） | 数值向量 | 字符数组 | 字符串数组 | 元胞数组 | 分类数组

决定箱子颜色是否变化的分组变量，指定为以逗号分隔的对组，其中包含 **'ColorGroup'** 和分组变量。分组变量是数值向量、字符数组、字符串数组、元胞数组或分类数组。当指定的分组变量值有变化时，箱子的颜色也会变化。默认值 **[]** 表示箱子颜色不会因组而变化。

数据类型： **single** | **double** | **char** | **string** | **cell** | **categorical**

FactorDirection - 图上因子的显示顺序
'data' （默认） | **'list'** | **'auto'**

图上因子的显示顺序，指定为以逗号分隔的对组，其中包含 **'FactorDirection'** 和下列项之一。

名称	值
'data'	首个因子值显示在绘图原点旁边。
'list'	因子在 x 轴上从左到右显示，在 y 轴上从上到下显示。
'auto'	如果分组变量是数值，则 **boxplot** 使用 **'data'**。如果分组变量是字符数组、字符串数组、元胞数组或分类数组，则 **boxplot** 使用 **'list'**。

FullFactors - 绘制所有组因子
'off' （默认） | **'on'**

绘制所有组因子，指定为以逗号分隔的对组，其中包含 **'FullFactors'** 和 **'off'** 或 **'on'**。如果指定 **'off'**，则 **boxplot** 为每个唯一的分组变量行绘制一个箱子。如果指定 **'on'**，则 **boxplot** 为分组变量值的每个可能组合绘制一个箱子，包括数据中未出现的组合。

示例： **'FullFactors','on'**

FactorGap - 不同分组因子之间的距离
[] | 正数值 | 正数值向量 | **'auto'**

不同分组因子之间的距离，指定为以逗号分隔的对组，其中包含 **'FactorGap'** 和正数值、正数值向量或 **'auto'**。如果指定向量，则向量长度必须小于或等于分组变量的数目。

'FactorGap' 表示一个分组变量的不同因子之间的间隙，以绘图宽度的百分比表示。例如，如果您指定 **[3,1]**，则第一个分组变量具有不同值的组之间的间隙是绘图宽度的 3%；第一个分组变量的值相同而第二个分组变量具有不同值的组之间的间隙是绘图宽度的 1%。

如果您指定 **'auto'**，则 **boxplot** 会自动选择间隙距离。值 **[]** 表示不同因子之间的间隙大小没有变化。

如果 **'PlotStyle'** 是 **'traditional'**，则 **FactorGap** 的默认值为 **[]**。如果 **'PlotStyle'** 是 **'compact'**，则默认值为 **'auto'**。

示例： **'FactorGap',[3,1]**

数据类型： **single** | **double** | **char** | **string**

FactorSeparator - 分组因子之间的分隔
[] | 正整数 | 正整数向量 | **'auto'**

分组因子之间的分隔，指定为以逗号分隔的对组，其中包含 **'FactorSeparator'** 和正整数或正整数向量或 **'auto'**。如果指定向量，则向量的长度应小于或等于分组变量的数目。整数值必须在 [1,G] 范围内，其中 G 是分组变量的数目。

'FactorSeparator' 指定哪些因子的值应该由网格线分隔。例如，当第一个或第二个分组变量更改值时，**[1,2]** 会添加分隔线。

如果 **'PlotStyle'** 是 **'traditional'**，则 **FactorSeparator** 的默认值为 **[]**。如果 **'PlotStyle'** 是 **'compact'**，则默认值为 **'auto'**。

示例： **'FactorSeparator',[1,2]**

数据类型： **single** | **double** | **char** | **string**

GroupOrder - 绘制组的顺序
[] （默认） | 字符串数组 | 元胞数组

绘制组的顺序，指定为以逗号分隔的对组，其中包含 **'GroupOrder'** 和包含分组变量名称的字符串数组或元胞数组。如果有多个分组变量，请用逗号分隔值。您也可以使用分类数组作为分组变量来控制箱子的顺序。默认值 **[]** 不会对这些箱子重新排序。

数据类型： **string** | **cell**

数据范围和最大距离

DataLim - 极值数据范围
[-Inf,Inf] （默认） | 二元素数值向量

极值数据范围，指定为以逗号分隔的对组，分别包含 **'DataLim'** 和包含下限和上限的二元素数值向量。**'ExtremeMode'** 使用为 **'DataLim'** 指定的值来确定哪些数据点为极值。

数据类型： **single** | **double**

ExtremeMode - 极值数据的处理方法
'clip' （默认） | **'compress'**

极值数据的处理方法，指定为以逗号分隔的对组，其中包含 **'ExtremeMode'** 和下列项之一。

名称	值
'clip'	如果有任何数据值超出 **'DataLim'** 指定的范围，则 **boxplot** 会在图中的 **DataLim** 处显示这些值。
'compress'	如果有任何数据值超出 **'DataLim'** 指定的范围，则 **boxplot** 会在 **DataLim** 以外的紧邻区域均匀显示这些值，并保留点的相对顺序。

如果有任何数据点超出 **'DataLim'** 指定的范围，则范围用虚线标记。如果有任何数据点被压缩，则会使用两条灰线标记压缩区域。**–Inf** 或 **Inf** 处的值可以被裁剪或压缩，但 **NaN** 值不会出现在绘图上。箱子缺口基于样本规模绘制，如果中位数在指定范围内，箱子缺口可能会延伸出界外。如果中位数超出指定范围，则不会绘制箱子缺口。

示例： **'ExtremeMode','compress'**

Jitter - 最大离群值位移距离
数值

最大离群值位移距离，指定为由 **'Jitter'** 和数值组成的以逗号分隔的对组。**Jitter** 沿因子轴将重复离群值移动随机均匀量的最大距离，以使它们清楚可见。如果您指定 **'Jitter'** 等于 1，则抖动区域为最近相邻组之间的整个区域。

如果 **'PlotStyle'** 是 **'traditional'**，则 **Jitter** 的默认值为 0。如果 **'PlotStyle'** 是 **'compact'**，则默认值为 0.5。

示例： **'Jitter',1**

数据类型： **single** | **double**

Whisker - 用于计算最大须线长度的乘数
1.5 （默认） | 正数值

用于计算最大须线长度的乘数，以逗号分隔的对组形式指定，其中包含 **'Whisker'** 和正数值。最大须线长度是 **Whisker** 和四分位差的乘积。

boxplot 将大于 $q_3 + w \times (q_3 - q_1)$ 或小于 $q_1 - w \times (q_3 - q_1)$ 的点绘制为离群值，其中 w 是乘数 **Whisker**，而 q_1 和 q_3 分别是样本数据的第 25 个和第 75 个百分位数。

如果数据呈正态分布，**'Whisker'** 的默认值大约对应于 +/–2.7σ 和 99.3% 的覆盖率。绘制的须线会延伸到邻近值，该值是非离群值的最远端数据值。

将 **'Whisker'** 指定为 0，表示不带须线，q_1 和 q_3 之外的所有点都将视为离群值。

示例： **'Whisker',0**

数据类型： **single** | **double**

绘图外观

Labels - 箱子标签
字符数组 | 字符串数组 | 元胞数组 | 数值向量 | 数值矩阵

箱子标签，指定为以逗号分隔的对组，其中包含 **'Labels'** 和包含箱子标签名称的字符数组、字符串数组、元胞数组或数值向量。为每个 **x** 值指定一个标签，或为每个组指定一个标签。要指定多个标签变量，请使用包含任何接受的数据类型的数值矩阵或元胞数组。

要从绘图中删除标签，请使用以下命令：**set(gca,'XTickLabel',{' '})**。

数据类型： **char** | **string** | **cell** | **single** | **double**

LabelOrientation - 标签方向
'inline' | **'horizontal'**

标签方向，指定为以逗号分隔的对组，其中包含 **'LabelOrientation'** 和以下项之一。

名称	值
'inline'	将箱子标签旋转至垂直状态。当 **'PlotStyle'** 是 **'compact'** 时，这是默认值。
'horizontal'	保持箱子标签处于水平状态。当 **'PlotStyle'** 是 **'traditional'** 时，这是默认值。

如果标签位于 y 轴上，则两种设置都保持标签处于水平状态。

示例： **'LabelOrientation','inline'**

LabelVerbosity - 要在绘图上显示的标签
'all' | **'minor'** | **'majorminor'**

要在绘图上显示的标签，指定为以逗号分隔的对组，其中包含 **'LabelVerbosity'** 和以下项之一。

名称	值
'all'	为分组变量的每个值显示一个标签。当 **'PlotStyle'** 是 **'traditional'** 时，这是默认值。
'minor'	对于任何分组变量，仅当箱子 **j** 的对应值不同于其前一个箱子 **(j – 1)** 的对应值时，才显示其对应值。
'majorminor'	对于任何分组变量 **g(:,i)**，仅当箱子 **j** 的对应值不同于箱子 **(j – 1)** 对应的 **g(:,i)** 值，或 **g(:,1),…,g(:,i-1)** 中至少有一个分组变量满足上述条件时，才显示 j 的对应值。当 **'PlotStyle'** 是 **'compact'** 时，这是默认值。

示例： **'LabelVerbosity','minor'**

Orientation - 绘图方向
'vertical' （默认） | **'horizontal'**

绘图方向，指定为以逗号分隔的对组，其中包含 **'Orientation'** 和以下项之一。

名称	值
'vertical'	在 y 轴上绘制 **x**。
'horizontal'	在 x 轴上绘制 **x**。

示例： **'horizontal'**

Positions - 箱子位置
数值向量

箱子位置，指定为以逗号分隔的对组，其中包含 **'Positions'** 和一个数值向量，对于每个组或 **x** 值，该数值向量都包含一个对应的条目。默认值为 1:NumGroups，其中 NumGroups 是组数。

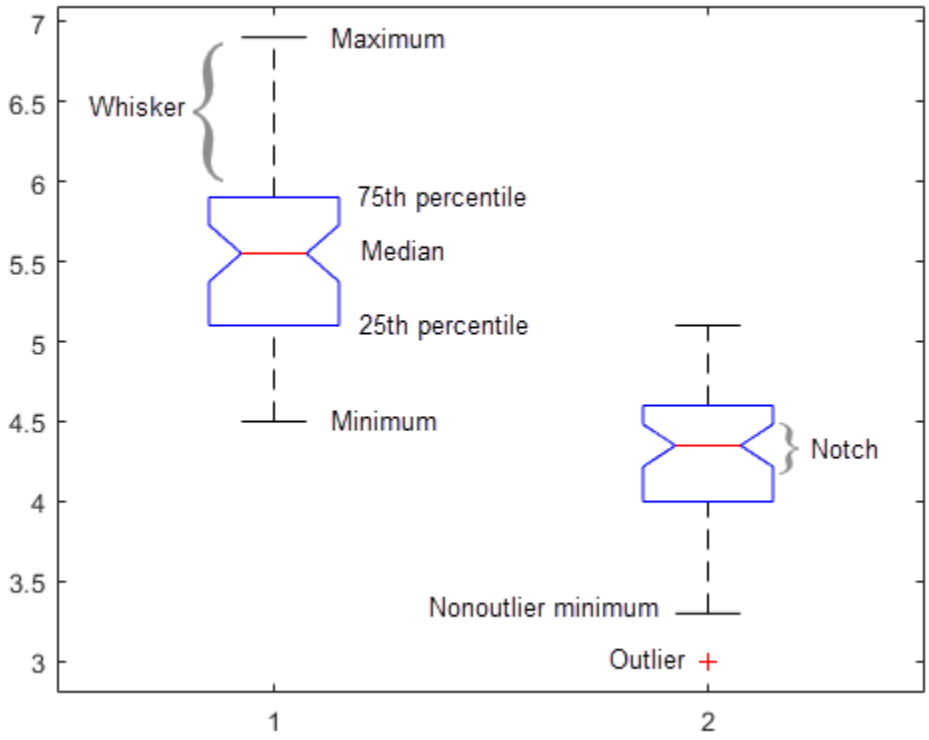
数据类型： **single** | **double**

详细信息

箱线图

箱线图提供样本数据的汇总统计量的可视化，并包含以下特性：

- 每个箱子的底部和顶部分别表示样本的第 25 个和第 75 个百分位数。每个箱子的底部和顶部之间的距离表示四分位差。
- 每个箱子中间的红线表示样本中位数。如果中位数不在箱子的中心，则绘图显示样本偏度。
- 须线是自每个箱子的顶部向上延伸和底部向下延伸的线条。须线从四分位差的端点延伸到须线长度内最远的观测值（相邻值）。
- 超出须线长度的观测值标记为离群值。默认情况下，离群值是距离箱子底部或顶部超过 1.5 倍四分位差的值。不过，您可以通过使用额外的输入参数来调整此值。离群值显示为红色 + 号。
- 缺口显示样本间中位数的变异性。计算缺口的宽度，使得缺口不重叠的框在 5% 显著性水平上具有不同中位数。显著性水平基于正态分布假设，但对于其他分布，中位数比较也可合理地认为是稳健的。比较箱线图中位数就像目测假设检验，类似于用于均值的 t 检验。



提示

- **boxplot** 创建数据的可视化表示，但不返回数值。要计算样本数据的相关汇总统计量，请使用以下函数：
  - **min** - 找到样本数据中的最小值。
  - **max** - 找到样本数据中的最大值。
  - **median** - 找到样本数据中的中位数值。
  - **quantile** - 找到样本数据中的分位数值。例如，要计算 **x** 的第 25 个和第 75 个百分位数，请指定 **quantile(x,[0.25 0.75])**。有关如何计算百分位数的详细信息，请参阅 “Algorithms” 。
  - **iqr** - 找到样本数据中的四分位差。
  - **grpstats** - 计算样本数据的分组汇总统计量。
- 您可以在图窗窗口中使用数据游标查看数据值和组名称。游标显示受 **datalim** 参数影响的任何点的原始值。您可以使用 **gname** 函数对离群值所属的组加标签。
- 要修改箱线图组件的图形属性，请使用 **findobj** 和 **Tag** 属性来查找组件的句柄。箱线图组件的 **Tag** 值取决于参数设置，如下表所示。

参数设置	标记值
所有设置	• **'Box'** • **'Outliers'**

参数设置	标记值
当 **'PlotStyle'** 是 **'traditional'** 时	• **'Median'** • **'Upper Whisker'** • **'Lower Whisker'** • **'Upper Adjacent Value'** • **'Lower Adjacent Value'**
当 **'PlotStyle'** 是 **'compact'** 时	• **'Whisker'** • **'MedianOuter'** • **'MedianInner'**
当 **'Notch'** 是 **'marker'** 时	• **'NotchLo'** • **'NotchHi'**

替代功能

您还可以使用 **boxchart** 函数创建 **BoxChart** 对象。虽然 **boxchart** 并未涵盖 **boxplot** 的所有功能，但它有一些优势。与 **boxplot** 不同，**boxchart** 函数：

- 支持沿组轴的分类标尺
- 提供图例选项
- 适合与 **hold on** 命令结合使用
- 具有改进的外观设计，帮助您更轻松地查看缺口

要控制对象的外观和行为，请更改 BoxChart Properties。

参考

[1] McGill, R., J. W. Tukey, and W. A. Larsen. “Variations of Boxplots.” *The American Statistician*. Vol. 32, No. 1, 1978, pp. 12–16.

[2] Velleman, P.F., and D.C. Hoaglin. *Applications, Basics, and Computing of Exploratory Data Analysis*. Pacific Grove, CA: Duxbury Press, 1981.

[3] Nelson, L. S. “Evaluating Overlapping Confidence Intervals.” *Journal of Quality Technology*. Vol. 21, 1989, pp. 140–141.

[4] Langford, E. “Quartiles in Elementary Statistics”, *Journal of Statistics Education*. Vol. 14, No. 3, 2006.

另请参阅

anova1 | **kruskalwallis** | **multcompare** | **min** | **max** | **median** | **quantile** | **grpstats**

主题

“Compare Grouped Data Using Box Plots”

“Grouping Variables”

在 R2006a 之前推出

caseread

从文件中读取个案名称

语法

```
names = caseread(filename)
names = caseread
```

说明

names = caseread(filename) 读取 **filename** 的内容，并返回字符数组 **names**。**caseread** 函数将文件的每行都视为单一个案名称。将 **filename** 指定为当前文件夹中的文件名或文件的完整路径名称。

filename 可以使用下列文件扩展名之一：

- **.txt**、**.dat** 或 **.csv**（适用于带分隔符的文本文件）
- **.xls**、**.xlsm** 或 **.xlsx**（适用于 Excel® 电子表格文件）

names = caseread 打开“选择要打开的文件”对话框，以便您可以交互选择要读取的文件。

示例

写入和读取个案名称

创建表示月份的个案名称字符数组。

```
months = char('January','February', ...
   'March','April','May');
```

将这些名称写入名为 **months.dat** 的文件。使用 **type** 函数查看文件内容。

```
casewrite(months,'months.dat')
type months.dat
```

```
January
February
March
April
May
```

读取 **months.dat** 文件中的名称。

```
names = caseread('months.dat')
```

```
names = 5x8 char array
   'January '
   'February'
   'March   '
   'April   '
   'May     '
```

输入参数

filename - 要读取的文件的名称
字符向量 | 字符串标量

要读取的文件的名称，指定为字符向量或字符串标量。

根据文件的位置，**filename** 为以下形式之一。

文件的位置	形式
当前文件夹，或 MATLAB® 路径中的文件夹	在 **filename** 中指定文件的名称。 **示例：'myTextFile.csv'**
非当前文件夹，或非 MATLAB 路径中的文件夹	请在 **filename** 中指定完整或相对路径名称。 **示例：'C:\myFolder\myTextFile.csv'**

示例： **'months.dat'**

数据类型： **char | string**

替代功能

对于字符数组，一般使用 **casewrite** 和 **caseread**，而对于元胞数组，请考虑使用 **writecell** 和 **readcell**。例如：

```
months = {'January';'February';'March';'April';'May'};
writecell(months,'months.dat')
names = readcell('months.dat')
```

```
names =

  5×1 cell array

    {'January' }
    {'February'}
    {'March'   }
    {'April'   }
    {'May'     }
```

另请参阅

casewrite | gname | readcell | writecell | readtable | writetable

在 R2006a 之前推出

cdf

包： prob

累积分布函数

语法

```
y = cdf('name',x,A)
y = cdf('name',x,A,B)
y = cdf('name',x,A,B,C)
y = cdf('name',x,A,B,C,D)

y = cdf(pd,x)

y = cdf( ___ ,'upper')
```

说明

y = cdf('name',x,A) 基于 **x** 中的值计算并返回由 **'name'** 和分布参数 **A** 指定的单参数分布族的累积分布函数 (cdf) 值。

y = cdf('name',x,A,B) 基于 **x** 中的值计算并返回由 **'name'** 以及分布参数 **A** 和 **B** 指定的双参数分布族的 cdf。

y = cdf('name',x,A,B,C) 基于 **x** 中的值计算并返回由 **'name'** 以及分布参数 **A**、**B** 和 **C** 指定的三参数分布族的 cdf。

y = cdf('name',x,A,B,C,D) 基于 **x** 中的值计算并返回由 **'name'** 以及分布参数 **A**、**B**、**C** 和 **D** 指定的四参数分布族的 cdf。

y = cdf(pd,x) 基于 **x** 中的值计算并返回概率分布对象 **pd** 的 cdf。

y = cdf(___ ,'upper') 使用可更精确计算极值上尾概率的算法返回 cdf 的补函数。**'upper'** 可以跟在上述语法中的任何输入参数之后。

示例

计算正态分布 cdf

创建均值 μ 等于 0、标准差 σ 等于 1 的标准正态分布对象。

```
mu = 0;
sigma = 1;
pd = makedist('Normal','mu',mu,'sigma',sigma);
```

定义输入向量 x 以包含用于计算 cdf 的值。

```
x = [-2,-1,0,1,2];
```

基于 x 中的值计算标准正态分布的 cdf 值。

```
y = cdf(pd,x)
```

```
y = 1×5

   0.0228    0.1587    0.5000    0.8413    0.9772
```

y 中的每个值对应于输入向量 x 中的一个值。例如，在值 x 等于 1 时，对应的 cdf 值 y 等于 0.8413。

您也可以不创建概率分布对象而直接计算同样的 cdf 值。使用 **cdf** 函数，再使用同样的 μ 和 σ 参数值指定一个标准正态分布。

```
y2 = cdf('Normal',x,mu,sigma)
```

```
y2 = 1×5

   0.0228    0.1587    0.5000    0.8413    0.9772
```

cdf 值与使用概率分布对象计算的值相同。

计算泊松分布 cdf

创建一个泊松分布对象，使用的速率参数 λ 等于 2。

```
lambda = 2;
pd = makedist('Poisson','lambda',lambda);
```

定义输入向量 x 以包含用于计算 cdf 的值。

```
x = [0,1,2,3,4];
```

基于 x 中的值计算泊松分布的 cdf 值。

```
y = cdf(pd,x)
```

```
y = 1×5

   0.1353    0.4060    0.6767    0.8571    0.9473
```

y 中的每个值对应于输入向量 x 中的一个值。例如，在值 x 等于 3 时，对应的 cdf 值 y 等于 0.8571。

您也可以不创建概率分布对象而直接计算同样的 cdf 值。使用 **cdf** 函数，并使用相同的速率参数值指定泊松分布，λ。

```
y2 = cdf('Poisson',x,lambda)
```

```
y2 = 1×5

   0.1353    0.4060    0.6767    0.8571    0.9473
```

cdf 值与使用概率分布对象计算的值相同。

绘制标准正态分布 cdf

创建一个标准正态分布对象。

```
pd = makedist('Normal')
```

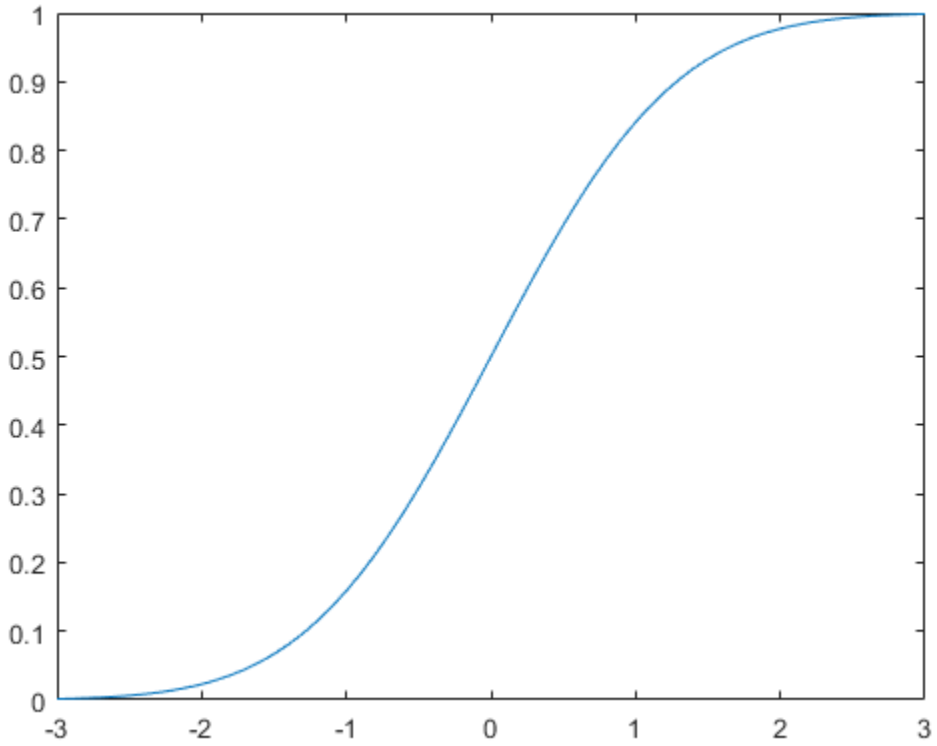
```
pd =
  NormalDistribution

  Normal distribution
     mu = 0
    sigma = 1
```

指定 x 值并计算 cdf。

```
x = -3:.1:3;
p = cdf(pd,x);
```

绘制标准正态分布的 cdf。

```
plot(x,p)
```



绘制 gamma 分布 cdf

创建三个 gamma 分布对象。第一个使用默认参数值。第二个指定 a = 1 和 b = 2。第三个指定 a = 2 和 b = 1。

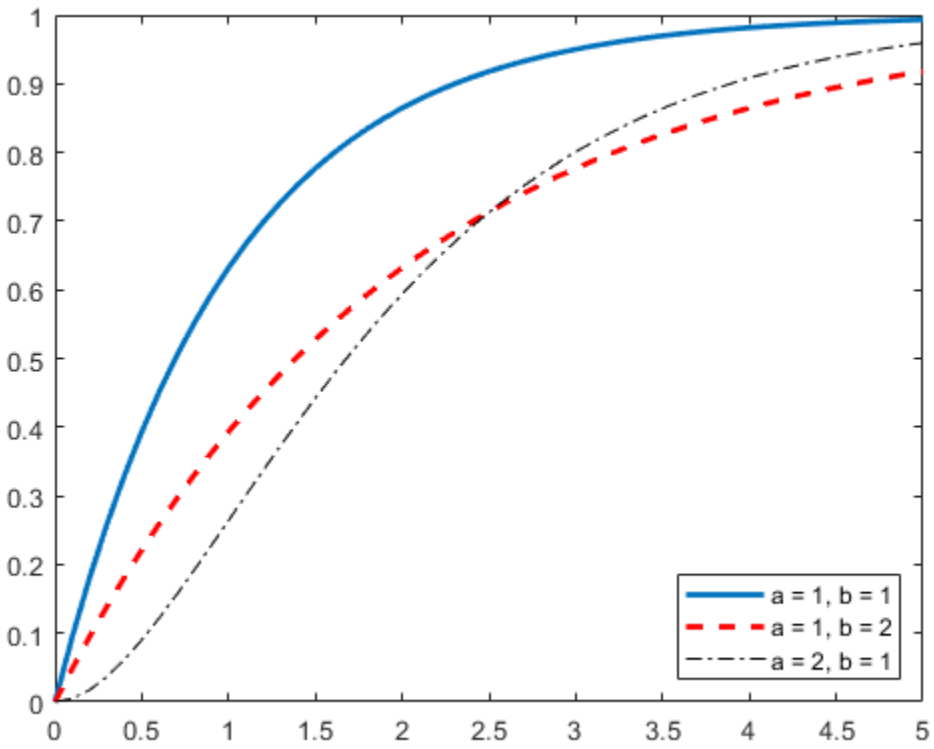
```
pd_gamma = makedist('Gamma')

pd_gamma =
  GammaDistribution

  Gamma distribution
    a = 1
    b = 1


pd_12 = makedist('Gamma','a',1,'b',2)

pd_12 =
  GammaDistribution

  Gamma distribution
    a = 1
    b = 2


pd_21 = makedist('Gamma','a',2,'b',1)

pd_21 =
  GammaDistribution

  Gamma distribution
    a = 2
    b = 1
```

指定 **x** 值，并计算每个分布的 cdf。

```
x = 0:.1:5;
cdf_gamma = cdf(pd_gamma,x);
cdf_12 = cdf(pd_12,x);
cdf_21 = cdf(pd_21,x);
```

创建一个绘图，该绘图用于可视化在为形状参数 **a** 和 **b** 指定不同值时 gamma 分布的 cdf 如何变化。

```
figure;
J = plot(x,cdf_gamma);
hold on;
K = plot(x,cdf_12,'r--');
L = plot(x,cdf_21,'k-.');
set(J,'LineWidth',2);
set(K,'LineWidth',2);
legend([J K L],'a = 1, b = 1','a = 1, b = 2','a = 2, b = 1','Location','southeast');
hold off;
```



对 t 分布进行帕累托尾拟合，并计算 cdf

对累积概率为 0.1 和 0.9 的 *t* 分布进行帕累托尾拟合。

```
t = trnd(3,100,1);
obj = paretotails(t,0.1,0.9);
[p,q] = boundary(obj)
```

```
p = 2×1

    0.1000
    0.9000


q = 2×1

   -1.8487
    2.0766
```

基于 **q** 中的值计算 cdf。

```
cdf(obj,q)
```

```
ans = 2×1

    0.1000
```

```
0.9000
```

输入参数

'name' - 概率分布名称
概率分布名称的字符向量或字符串标量

概率分布名称，指定为下表中的概率分布名称之一。

'name'	分布	输入参数 A	输入参数 B	输入参数 C	输入参数 D
'Beta'	“Beta Distribution”	a 第一个形状参数	b 第二个形状参数	—	—
'Binomial'	“Binomial Distribution”	n 试验次数	p 每次试验成功的概率	—	—
'BirnbaumSaunders'	“Birnbaum-Saunders Distribution”	β 尺度参数	γ 形状参数	—	—
'Burr'	“Burr Type XII Distribution”	α 尺度参数	c 第一个形状参数	k 第二个形状参数	—
'Chisquare'	“Chi-Square Distribution”	ν 自由度	—	—	—
'Exponential'	“Exponential Distribution”	μ 均值	—	—	—
'Extreme Value'	“Extreme Value Distribution”	μ 位置参数	σ 尺度参数	—	—
'F'	“F Distribution”	ν_1 分子自由度	ν_2 分母自由度	—	—
'Gamma'	“Gamma Distribution”	a 形状参数	b 尺度参数	—	—
'Generalized Extreme Value'	“Generalized Extreme Value Distribution”	k 形状参数	σ 尺度参数	μ 位置参数	—
'Generalized Pareto'	“Generalized Pareto Distribution”	k 尾部指数（形状）参数	σ 尺度参数	μ 阈值（位置）参数	—
'Geometric'	“Geometric Distribution”	p 概率参数	—	—	—
'HalfNormal'	“Half-Normal Distribution”	μ 位置参数	σ 尺度参数	—	—
'Hypergeometric'	“Hypergeometric Distribution”	m 总体的大小	k 总体中具有所需特征的项数	n 抽取的样本数量	—
'InverseGaussian'	“Inverse Gaussian Distribution”	μ 尺度参数	λ 形状参数	—	—
'Logistic'	“Logistic Distribution”	μ 均值	σ 尺度参数	—	—
'LogLogistic'	“Loglogistic Distribution”	μ 对数值的均值	σ 对数值的尺度参数	—	—
'Lognormal'	“Lognormal Distribution”	μ 对数值的均值	σ 对数值的标准差	—	—
'Nakagami'	“Nakagami Distribution”	μ 形状参数	ω 尺度参数	—	—
'Negative Binomial'	“Negative Binomial Distribution”	r 成功次数	p 单个试验的成功概率	—	—

'name'	分布	输入参数 A	输入参数 B	输入参数 C	输入参数 D
'Noncentral F'	“Noncentral F Distribution”	ν_1 分子自由度	ν_2 分母自由度	δ 非中心参数	—
'Noncentral t'	“Noncentral t Distribution”	ν 自由度	δ 非中心参数	—	—
'Noncentral Chi-square'	“Noncentral Chi-Square Distribution”	ν 自由度	δ 非中心参数	—	—
'Normal'	“正态分布” （第 B-2 页）	μ 均值	σ 标准差	—	—
'Poisson'	“泊松分布” （第 B-13 页）	λ 均值	—	—	—
'Rayleigh'	“Rayleigh Distribution”	b 尺度参数	—	—	—
'Rician'	“Rician Distribution”	s 非中心参数	σ 尺度参数	—	—
'Stable'	“Stable Distribution”	α 第一个形状参数	β 第二个形状参数	γ 尺度参数	δ 位置参数
'T'	“Student's t Distribution”	ν 自由度	—	—	—
'tLocationScale'	“t Location-Scale Distribution”	μ 位置参数	σ 尺度参数	ν 形状参数	—
'Uniform'	“Uniform Distribution (Continuous)”	a 下部端点（最小值）	b 上部端点（最大值）	—	—
'Discrete Uniform'	“Uniform Distribution (Discrete)”	n 最大可观测值	—	—	—
'Weibull'	“Weibull Distribution”	a 尺度参数	b 形状参数	—	—

示例： 'Normal'

x - 用于计算 cdf 的值
标量值 | 标量值组成的数组

用于计算 cdf 的值，指定为标量值或标量值组成的数组。

如果输入参数 x、A、B、C 和 D 中的一个或多个是数组，则数组大小必须相同。在这种情况下，cdf 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 'name' 了解每个分布的 A、B、C 和 D 的定义。

示例： [0.1,0.25,0.5,0.75,0.9]

数据类型： single | double

A - 第一概率分布参数
标量值 | 标量值组成的数组

第一概率分布参数，指定为标量值或标量值组成的数组。

如果输入参数 x、A、B、C 和 D 中的一个或多个是数组，则数组大小必须相同。在这种情况下，cdf 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 'name' 了解每个分布的 A、B、C 和 D 的定义。

数据类型： single | double

B - 第二概率分布参数
标量值 | 标量值组成的数组

第二概率分布参数，指定为标量值或标量值组成的数组。

如果输入参数 **x**、**A**、**B**、**C** 和 **D** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**cdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 **'name'** 了解每个分布的 **A**、**B**、**C** 和 **D** 的定义。

数据类型： **single** | **double**

C - 第三概率分布参数
标量值 | 标量值组成的数组

第三概率分布参数，指定为标量值或标量值组成的数组。

如果输入参数 **x**、**A**、**B**、**C** 和 **D** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**cdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 **'name'** 了解每个分布的 **A**、**B**、**C** 和 **D** 的定义。

数据类型： **single** | **double**

D - 第四概率分布参数
标量值 | 标量值组成的数组

第四概率分布参数，指定为标量值或标量值组成的数组。

如果输入参数 **x**、**A**、**B**、**C** 和 **D** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**cdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 **'name'** 了解每个分布的 **A**、**B**、**C** 和 **D** 的定义。

数据类型： **single** | **double**

pd - 概率分布
概率分布对象

概率分布，指定为使用下表中的函数或 App 创建的概率分布对象。

函数或 App	说明
makedist	使用指定的参数值创建一个概率分布对象。
fitdist	对样本数据进行概率分布对象拟合。
分布拟合器	使用交互式分布拟合器对样本数据进行概率分布拟合，并将拟合的对象导出到工作区。
paretotails	创建一个分段分布对象，它在尾部具有广义帕累托分布。

输出参数

y - cdf 值
标量值 | 标量值组成的数组

cdf 值，以标量值或标量值组成的数组形式返回。在经过任何必要的标量扩展后，**y** 的大小与 **x** 相同。**y** 中的每个元素均为由分布参数（**A**、**B**、**C** 和 **D**）中的对应元素或概率分布对象 (**pd**) 指定的分布的 cdf 值，其值在 **x** 中的对应元素处进行计算。

替代功能

- **cdf** 是泛型函数，它按名称 **'name'** 或概率分布对象 **pd** 接受分布。使用分布特有的函数更快，例如正态分布特有的 **normcdf**，二项分布特有的 **binocdf**。有关特定于分布的函数的列表，请参阅“Supported Distributions”。
- 使用 **Probability Distribution Function** App 为概率分布创建累积分布函数 (cdf) 或概率密度函数 (pdf) 的交互图。

扩展功能

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

- 输入参数 **'name'** 必须为编译时常量。例如，要使用正态分布，请将 **coder.Constant('Normal')** 包含在 **codegen** 的 **-args** 值中。
- 输入参数 **pd** 可以是 beta、指数、极值、对数正态、正态和 Weibull 分布的拟合概率分布对象。通过对来自 **fitdist** 函数的样本数据进行概率分布拟合，创建 **pd**。有关示例，请参阅“Code Generation for Probability Distribution Objects”。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “General Code Generation Workflow”。

另请参阅
pdf | **ecdf** | **icdf** | **mle** | **random** | **makedist** | **fitdist** | **分布拟合器** | **paretotails**

主题
“Working with Probability Distributions”
“Supported Distributions”

在 R2006a 之前推出

corr

线性或秩相关性

语法

```
rho = corr(X)
rho = corr(X,Y)
[rho,pval] = corr(X,Y)
[rho,pval] = corr( ___ ,Name,Value)
```

说明

rho = corr(X) 返回输入矩阵 **X** 中各列之间的两两线性相关系数矩阵。

rho = corr(X,Y) 返回输入矩阵 **X** 和 **Y** 中各列之间的两两相关系数矩阵。

[rho,pval] = corr(X,Y) 还返回 **pval**，它是一个 p 值矩阵，用于基于非零相关性备择假设来检验无相关性假设。

除了上述语法中的输入参数之外，**[rho,pval] = corr(___ ,Name,Value)** 还使用一个或多个名称-值对组参数指定选项。例如，**'Type','Kendall'** 指定计算 Kendall tau 相关系数。

示例

计算两个矩阵之间的相关性

计算两个矩阵之间的相关性，并将其与两个列向量之间的相关性进行比较。

生成样本数据。

```
rng('default')
X = randn(30,4);
Y = randn(30,4);
```

在矩阵 **X** 的第二列和矩阵 **Y** 的第四列之间引入相关性。

```
Y(:,4) = Y(:,4)+X(:,2);
```

计算 **X** 和 **Y** 的列之间的相关性。

```
[rho,pval] = corr(X,Y)
```

```
rho = 4×4

   -0.1686   -0.0363    0.2278    0.3245
    0.3022    0.0332   -0.0866    0.7653
   -0.3632   -0.0987   -0.0200   -0.3693
   -0.1365   -0.1804    0.0853    0.0279


pval = 4×4
```

```
    0.3731    0.8489    0.2260    0.0802
    0.1045    0.8619    0.6491    0.0000
    0.0485    0.6039    0.9166    0.0446
    0.4721    0.3400    0.6539    0.8837
```

与预期相符，**X** 的第二列和 **Y** 的第四列之间的相关系数 **rho(2,4)** 最高，表示这两列之间存在高正相关性。对于所示的四个数，对应的 p 值 **pval(2,4)** 为零。由于 p 值小于显著性水平 **0.05**，这表明拒绝两列之间不存在相关性的假设。

使用 **corrcoef** 计算 **X** 和 **Y** 之间的相关性。

```
[r,p] = corrcoef(X,Y)
```

```
r = 2×2

    1.0000   -0.0329
   -0.0329    1.0000


p = 2×2

    1.0000    0.7213
    0.7213    1.0000
```

与 **corr** 函数不同，MATLAB® 函数 **corrcoef** 在计算输入矩阵 **X** 和 **Y** 之间的相关性之前，将它们转换为列向量 **X(:)** 和 **Y(:)**。因此，在矩阵 **X** 的第二列和矩阵 **Y** 的第四列之间引入的相关性不再存在，因为这两列位于转换后的列向量的不同部分。

r 的非对角线元素的值（表示 **X** 和 **Y** 之间相关系数）较低。该值表示 **X** 和 **Y** 之间几乎没有相关性。同样，**p** 的非对角线元素的值（表示 p 值）远远高于显著性水平 **0.05**。该值表明没有足够的证据来拒绝 **X** 和 **Y** 之间没有相关性的假设。

检验相关性的备择假设

检验两个矩阵列之间正、负和非零相关性的备择假设。比较每种情况下相关系数的值和 p 值。

生成样本数据。

```
rng('default')
X = randn(50,4);
Y = randn(50,4);
```

在矩阵 **X** 的第一列和矩阵 **Y** 的第四列之间引入正相关性。

```
Y(:,4) = Y(:,4)+0.7*X(:,1);
```

在 **X** 的第二列和 **Y** 的第二列之间引入负相关性。

```
Y(:,2) = Y(:,2)-2*X(:,2);
```

检验相关性大于零的备择假设。

```
[rho,pval] = corr(X,Y,'Tail','right')
```

```
rho = 4×4

    0.0627   -0.1438   -0.0035    0.7060
   -0.1197   -0.8600   -0.0440    0.1984
   -0.1119    0.2210   -0.3433    0.1070
   -0.3526   -0.2224    0.1023    0.0374


pval = 4×4

    0.3327    0.8405    0.5097    0.0000
    0.7962    1.0000    0.6192    0.0836
    0.7803    0.0615    0.9927    0.2298
    0.9940    0.9397    0.2398    0.3982
```

与预期相符，**X** 的第一列和 **Y** 的第四列之间的相关系数 **rho(1,4)** 具有最高的正值，表示这两列之间存在高正相关性。对于所示的四个数，对应的 p 值 **pval(1,4)** 为零，低于显著性水平 **0.05**。这些结果表明拒绝这两列之间不存在相关性的原假设，并得出相关性大于零的结论。

检验相关性小于零的备择假设。

```
[rho,pval] = corr(X,Y,'Tail','left')
```

```
rho = 4×4

    0.0627   -0.1438   -0.0035    0.7060
   -0.1197   -0.8600   -0.0440    0.1984
   -0.1119    0.2210   -0.3433    0.1070
   -0.3526   -0.2224    0.1023    0.0374


pval = 4×4

    0.6673    0.1595    0.4903    1.0000
    0.2038    0.0000    0.3808    0.9164
    0.2197    0.9385    0.0073    0.7702
    0.0060    0.0603    0.7602    0.6018
```

与预期相符，**X** 的第二列和 **Y** 的第二列之间的相关系数 **rho(2,2)** 为绝对值最大的负数 (**-0.86**)，表示这两列之间存在高负相关性。对于所示的四个数，对应的 p 值 **pval(2,2)** 为零，低于显著性水平 **0.05**。同样，这些结果表明拒绝原假设，并得出相关性小于零的结论。

检验相关性不为零的备择假设。

```
[rho,pval] = corr(X,Y)
```

```
rho = 4×4

    0.0627   -0.1438   -0.0035    0.7060
   -0.1197   -0.8600   -0.0440    0.1984
   -0.1119    0.2210   -0.3433    0.1070
   -0.3526   -0.2224    0.1023    0.0374


pval = 4×4

    0.6654    0.3190    0.9807    0.0000
```

```
0.4075   0.0000   0.7615   0.1673
0.4393   0.1231   0.0147   0.4595
0.0120   0.1206   0.4797   0.7964
```

对于所示的四个数，p 值 **pval(1,4)** 和 **pval(2,2)** 均为零。由于 p 值低于显著性水平 **0.05**，因此相关系数 **rho(1,4)** 和 **rho(2,2)** 显著不同于零。因此，拒绝原假设；相关性不是零。

输入参数

X - 输入矩阵
矩阵

输入矩阵，指定为 n×k 矩阵。**X** 的行对应于观测值，而列对应于变量。

示例： **X = randn(10,5)**

数据类型： **single** | **double**

Y - 输入矩阵
矩阵

输入矩阵，当 **X** 指定为 $n \times k_1$ 矩阵时，指定为 $n \times k_2$ 矩阵。**Y** 的行对应于观测值，而列对应于变量。

示例： **Y = randn(20,7)**

数据类型： **single** | **double**

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例： **corr(X,Y,'Type','Kendall','Rows','complete')** 仅使用不包含缺失值的行返回 Kendall tau 相关系数。

Type - 相关性的类型
'Pearson' （默认） | **'Kendall'** | **'Spearman'**

相关性的类型，指定为以逗号分隔的对组，其中包含 **'Type'** 和下列值之一。

值	说明
'Pearson'	“Pearson 线性相关系数” （第 33-33 页）
'Kendall'	“Kendall tau 系数” （第 33-34 页）
'Spearman'	“Spearman rho” （第 33-34 页）

corr 使用 Student t 分布来转换相关性以计算 Pearson 相关性的 p 值。当 **X** 和 **Y** 来自正态分布时，这种相关性是精确的。**corr** 使用精确排列分布（对于小样本）或大样本近似分布计算 Kendall tau 和 Spearman rho 的 p 值。

示例： **'Type','Spearman'**

Rows - 在计算中要使用的行
'all' （默认） | **'complete'** | **'pairwise'**

在计算中要使用的行，指定为以逗号分隔的对组，其中包含 **'Rows'** 和下列值之一。

值	说明
'all'	使用输入的所有行，而不考虑缺失值 (**NaN**)。
'complete'	仅使用没有缺失值的输入行。
'pairwise'	使用 **i** 或 **j** 列中没有缺失值的行计算 **rho(i,j)**。

与 **'pairwise'** 值不同，**'complete'** 值始终生成正定或半正定 **rho**。此外，当输入（**X** 或 **Y**）的行包含缺失值时，**'complete'** 值通常使用较少的观测值来估计 **rho**。

示例：**'Rows','pairwise'**

Tail - 备择假设
'both'（默认）| **'right'** | **'left'**

备择假设，指定为以逗号分隔的对组，其中包含 **'Tail'** 和下表中的值之一。**'Tail'** 指定计算 p 值所针对的备择假设，用于检验无相关性的假设。

值	说明
'both'	检验相关性不为 **0** 的备择假设。
'right'	检验相关性大于 **0** 的备择假设
'left'	检验相关性小于 **0** 的备择假设。

corr 通过将两个单尾 p 值中的较大值加倍，计算双尾检验的 p 值。

示例：**'Tail','left'**

输出参数

rho - 两两线性相关系数
矩阵

两两线性相关系数，以矩阵形式返回。

- 如果您只输入矩阵 **X**，**rho** 是对称的 k×k 矩阵，其中 k 是 **X** 中的列数。条目 **rho(a,b)** 是 **X** 中 a 列和 b 列之间的两两线性相关系数。
- 如果您输入矩阵 **X** 和 **Y**，**rho** 是 $k_1 \times k_2$ 矩阵，其中 k_1 和 k_2 分别是 **X** 和 **Y** 中的列数。条目 **rho(a,b)** 是 **X** 中的 a 列和 **Y** 中的 b 列之间的两两线性相关系数。

pval - p 值
矩阵

p 值，以矩阵形式返回。**pval** 的每个元素均为 **rho** 的对应元素的 p 值。

如果 **pval(a,b)** 较小（小于 **0.05**），则相关性 **rho(a,b)** 显著不同于零。

详细信息

Pearson 线性相关系数

Pearson 线性相关系数是最常用的线性相关系数。对于矩阵 X 中的 X_a 列和矩阵 Y 中的 Y_b 列，其含义为 $\bar{X}_a = \sum_{i=1}^{n} (X_{a,i})/n$，和 $\bar{Y}_b = \sum_{j=1}^{n} (X_{b,j})/n$，Pearson 线性相关系数 rho(a,b) 定义为：

$$rho(a,b) = \frac{\sum_{i=1}^{n}(X_{a,i} - \bar{X}_a)(Y_{b,i} - \bar{Y}_b)}{\left\{\sum_{i=1}^{n}(X_{a,i} - \bar{X}_a)^2 \sum_{j=1}^{n}(Y_{b,j} - \bar{Y}_b)^2\right\}^{1/2}},$$

其中 n 是每列的长度。

相关系数的值的范围是从 **–1** 到 **+1**。值 **–1** 表示完全负相关，而值 **+1** 表示完全正相关。值 **0** 表示列之间没有相关性。

Kendall tau 系数

Kendall tau 系数基于 (i,j) 同序对的计数（其中 i<j），同序是指 $X_{a,i} - X_{a,j}$ 和 $Y_{b,i} - Y_{b,j}$ 具有相同的符号。Kendall tau 方程包括对归一化常量中结值的调整项，通常称为 tau-b。

对于矩阵 X 中的 X_a 列和矩阵 Y 中的 Y_b 列，Kendall tau 系数定义为：

$$\tau = \frac{2K}{n(n-1)},$$

其中 $K = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \xi^*(X_{a,i}, X_{a,j}, Y_{b,i}, Y_{b,j})$，并且

$$\xi^*(X_{a,i}, X_{a,j}, Y_{b,i}, Y_{b,j}) = \begin{cases} 1 & \text{if } (X_{a,i} - X_{a,j})(Y_{b,i} - Y_{b,j}) > 0 \\ 0 & \text{if } (X_{a,i} - X_{a,j})(Y_{b,i} - Y_{b,j}) = 0 \\ -1 & \text{if } (X_{a,i} - X_{a,j})(Y_{b,i} - Y_{b,j}) < 0 \end{cases}.$$

相关系数的值的范围是从 **–1** 到 **+1**。值 **–1** 表示一个列秩序是另一个列秩序的逆，值 **+1** 表示两个列秩序相同。值 **0** 表示列之间没有关系。

Spearman rho

Spearman rho 等效于应用于 X_a 和 Y_b 列的秩序的 “Pearson 线性相关系数” （第 33-33 页）。

如果每个列中的所有秩都不同，该方程可简化为：

$$rho(a,b) = 1 - \frac{6\sum d^2}{n(n^2-1)},$$

其中，d 是两个列的秩之差，n 是每列的长度。

提示

corr(X,Y) 和 MATLAB 函数 **corrcoef(X,Y)** 之间的区别在于 **corrcoef(X,Y)** 返回两个列向量 **X** 和 **Y** 的相关系数的矩阵。如果 **X** 和 **Y** 不是列向量，**corrcoef(X,Y)** 会将它们转换为列向量。

参考

[1] Gibbons, J.D. *Nonparametric Statistical Inference.* 2nd ed. M. Dekker, 1985.

[2] Hollander, M., and D.A. Wolfe. *Nonparametric Statistical Methods*. Wiley, 1973.

[3] Kendall, M.G. *Rank Correlation Methods*. Griffin, 1970.

[4] Best, D.J., and D.E. Roberts. "Algorithm AS 89: The Upper Tail Probabilities of Spearman's rho." *Applied Statistics*, 24:377-379.

扩展功能

tall 数组
对行数太多而无法放入内存的数组进行计算。

此函数支持对无法放入内存的数据使用 tall 数组，但有以下限制：

仅支持 **'Pearson'** 类型。

有关详细信息，请参阅“使用 tall 数组处理无法放入内存的数据”。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
corrcoef | **partialcorr** | **corrcov** | **tiedrank**

在 R2006a 之前推出

ff2n

二水平完全析因设计

语法

```
dFF2 = ff2n(n)
```

说明

dFF2 = ff2n(n) 给出具有 **n** 个因子的二水平完全析因设计的因子设置 **dFF2**。**dFF2** 为 m×**n**，其中 m 是完全析因设计中的处理数。**dFF2** 的每行对应一种处理。每列包含单一因子的设置，两个水平的值分别为 **0** 和 **1**。

示例

```
dFF2 = ff2n(3)
dFF2 =
  0   0   0
  0   0   1
  0   1   0
  0   1   1
  1   0   0
  1   0   1
  1   1   0
  1   1   1
```

另请参阅

fullfact

在 R2006a 之前推出

fitcsvm

训练用于一类和二类分类的支持向量机 (SVM) 分类器

语法

Mdl = fitcsvm(Tbl,ResponseVarName)
Mdl = fitcsvm(Tbl,formula)
Mdl = fitcsvm(Tbl,Y)

Mdl = fitcsvm(X,Y)

Mdl = fitcsvm(___ ,Name,Value)

说明

fitcsvm 基于低维或中维预测变量数据集训练或交叉验证一类和二类（二元）分类的支持向量机 (SVM) 模型。**fitcsvm** 支持使用核函数映射预测变量数据，并支持序列最小优化 (SMO)、迭代单点数据算法 (ISDA) 或 L1 软边距最小化（二次规划目标函数最小化）。

要基于高维数据集（即包含许多预测变量的数据集）训练二类分类线性 SVM 模型，请改用 **fitclinear**。

对于结合使用二类 SVM 模型的多类学习，请使用纠错输出编码 (ECOC)。有关详细信息，请参阅 **fitcecoc**。

要训练 SVM 回归模型，请参阅 **fitrsvm**（适用于低维和中维预测变量数据集）或 **fitrlinear**（适用于高维数据集）。

Mdl = fitcsvm(Tbl,ResponseVarName) 返回训练的支持向量机 (SVM) 分类器 （第 33-65 页） **Mdl**。该分类器使用表 **Tbl** 中包含的样本数据进行训练。**ResponseVarName** 是 **Tbl** 中变量的名称，该变量包含一类或二类分类的类标签。

Mdl = fitcsvm(Tbl,formula) 返回训练的 SVM 分类器。该分类器使用表 **Tbl** 中包含的样本数据进行训练。**formula** 是用于拟合 **Mdl** 的解释模型，该模型由 **Tbl** 中的响应和一部分预测变量构成。

Mdl = fitcsvm(Tbl,Y) 返回训练的 SVM 分类器。该分类器使用表 **Tbl** 中的预测变量和向量 **Y** 中的类标签进行训练。

Mdl = fitcsvm(X,Y) 返回训练的 SVM 分类器，该分类器使用矩阵 **X** 中的预测变量和向量 **Y** 中的一类或两类分类的类标签进行训练。

除了上述语法中的输入参数之外，**Mdl = fitcsvm(___ ,Name,Value)** 还可使用一个或多个名称-值对组参数指定选项。例如，您可以指定交叉验证的类型、误分类的代价以及分数转换函数的类型。

示例

训练 SVM 分类器

加载 Fisher 鸢尾花数据集。删除萼片的长度和宽度以及所有观测到的山鸢尾花。

```
load fisheriris
inds = ~strcmp(species,'setosa');
X = meas(inds,3:4);
y = species(inds);
```

使用经过处理的数据集训练 SVM 分类器。

```
SVMModel = fitcsvm(X,y)
```

```
SVMModel =
  ClassificationSVM
             ResponseName: 'Y'
    CategoricalPredictors: []
              ClassNames: {'versicolor'  'virginica'}
          ScoreTransform: 'none'
         NumObservations: 100
                   Alpha: [24x1 double]
                    Bias: -14.4149
        KernelParameters: [1x1 struct]
          BoxConstraints: [100x1 double]
         ConvergenceInfo: [1x1 struct]
         IsSupportVector: [100x1 logical]
                  Solver: 'SMO'


  Properties, Methods
```

SVMModel 是经过训练的 **ClassificationSVM** 分类器。您可以查看 **SVMModel** 的属性。例如，使用圆点表示法查看类顺序。

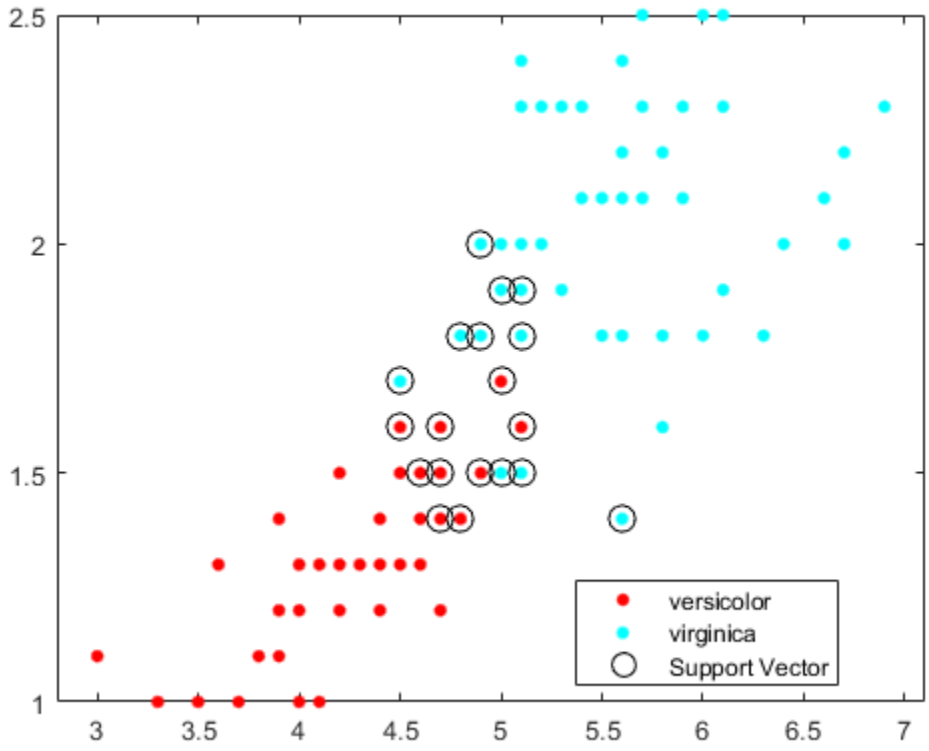
```
classOrder = SVMModel.ClassNames
```

```
classOrder = 2x1 cell
    {'versicolor'}
    {'virginica' }
```

第一个类 (**'versicolor'**) 是负类，第二个类 (**'virginica'**) 是正类。您可以使用 **'ClassNames'** 名称-值对组参数在训练期间更改类顺序。

绘制数据的散点图并圈出支持向量。

```
sv = SVMModel.SupportVectors;
figure
gscatter(X(:,1),X(:,2),y)
hold on
plot(sv(:,1),sv(:,2),'ko','MarkerSize',10)
legend('versicolor','virginica','Support Vector')
hold off
```



支持向量是发生在其估计的类边界之上或之外的观测值。

您可以使用 **'BoxConstraint'** 名称-值对组参数在训练过程中设置框约束来调整边界（从而调整支持向量的数量）。

训练和交叉验证 SVM 分类器

加载 **ionosphere** 数据集。

```
load ionosphere
rng(1); % For reproducibility
```

使用径向基核训练 SVM 分类器。让软件计算核函数的缩放值。对预测变量进行标准化。

```
SVMModel = fitcsvm(X,Y,'Standardize',true,'KernelFunction','RBF',...
    'KernelScale','auto');
```

SVMModel 是经过训练的 **ClassificationSVM** 分类器。

交叉验证 SVM 分类器。默认情况下，软件使用 10 折交叉验证。

```
CVSVMModel = crossval(SVMModel);
```

CVSVMModel 是 **ClassificationPartitionedModel** 交叉验证的分类器。

估计样本外的误分类率。

```
classLoss = kfoldLoss(CVSVMModel)
```

```
classLoss = 0.0484
```

泛化率约为 5%。

使用 SVM 和一类学习检测离群值

通过将所有鸢尾花分配给同一个类来修改 Fisher 的鸢尾花数据集。检测修改后的数据集中的离群值，并确认观测值中离群值的预期比例。

加载 Fisher 鸢尾花数据集。删除花瓣的长度和宽度。将所有鸢尾花都视为来自同一个类。

```
load fisheriris
X = meas(:,1:2);
y = ones(size(X,1),1);
```

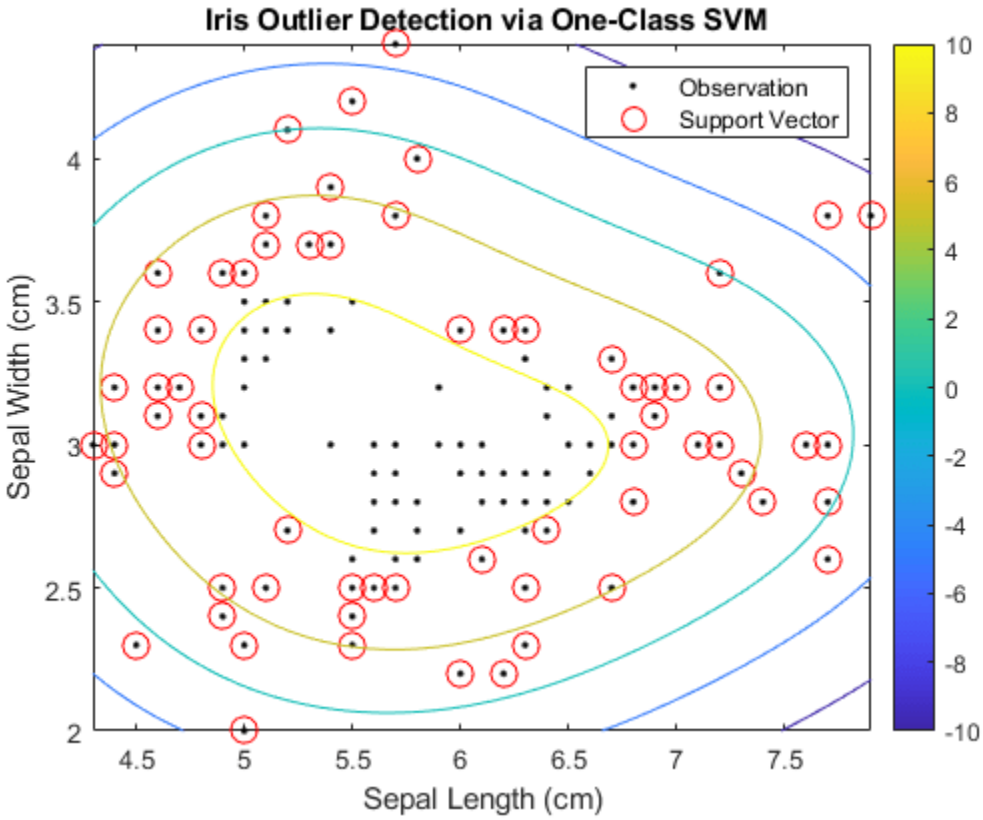
使用修改后的数据集训练 SVM 分类器。假设 5% 的观测值是离群值。对预测变量进行标准化。

```
rng(1);
SVMModel = fitcsvm(X,y,'KernelScale','auto','Standardize',true,...
   'OutlierFraction',0.05);
```

SVMModel 是经过训练的 **ClassificationSVM** 分类器。默认情况下，软件使用高斯核进行一类学习。

绘制观测值和决策边界。标记支持向量和潜在离群值。

```
svInd = SVMModel.IsSupportVector;
h = 0.02; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)),...
   min(X(:,2)):h:max(X(:,2)));
[~,score] = predict(SVMModel,[X1(:),X2(:)]);
scoreGrid = reshape(score,size(X1,1),size(X2,2));

figure
plot(X(:,1),X(:,2),'k.')
hold on
plot(X(svInd,1),X(svInd,2),'ro','MarkerSize',10)
contour(X1,X2,scoreGrid)
colorbar;
title('{\bf Iris Outlier Detection via One-Class SVM}')
xlabel('Sepal Length (cm)')
ylabel('Sepal Width (cm)')
legend('Observation','Support Vector')
hold off
```



将离群值与其余数据分隔的边界出现在围道值为 0 的位置。

验证交叉验证数据中具有负分数的观测值的比例接近 5%。

```
CVSVMModel = crossval(SVMModel);
[~,scorePred] = kfoldPredict(CVSVMModel);
outlierRate = mean(scorePred<0)
```

```
outlierRate = 0.0467
```

使用二类 SVM 计算多个类边界

创建 **fisheriris** 数据集的散点图。将图中网格的坐标点视为来自数据集分布的新观测值，并通过将坐标点分配给数据集中的三个类之一来找出类边界。

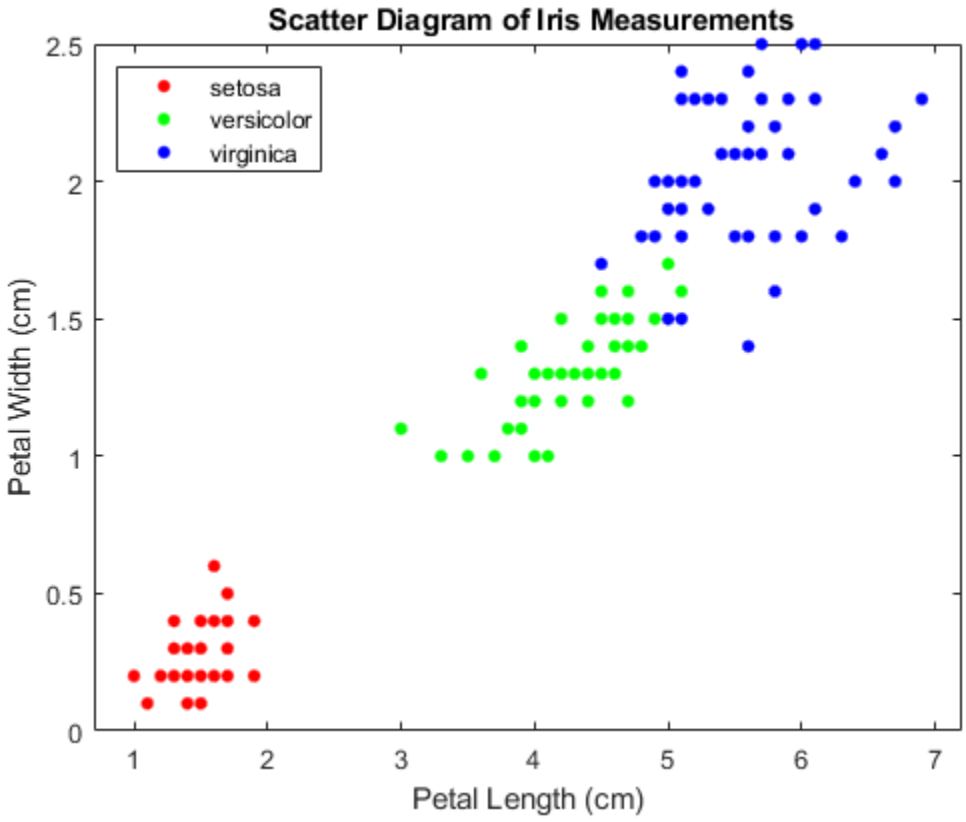
加载 Fisher 鸢尾花数据集。使用花瓣长度和宽度作为预测变量。

```
load fisheriris
X = meas(:,3:4);
Y = species;
```

检查数据的散点图。

```
figure
gscatter(X(:,1),X(:,2),Y);
```

```
h = gca;
lims = [h.XLim h.YLim]; % Extract the x and y axis limits
title('{\bf Scatter Diagram of Iris Measurements}');
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');
legend('Location','Northwest');
```



该数据包含三个类，其中一个类与其他类可线性分离。

对于每个类：

1 创建一个逻辑向量 (`indx`)，指示观测值是否为该类的成员。

2 使用预测变量数据和 `indx` 训练 SVM 分类器。

3 将该分类器存储在元胞数组的一个元胞中。

定义类顺序。

```
SVMModels = cell(3,1);
classes = unique(Y);
rng(1); % For reproducibility

for j = 1:numel(classes)
    indx = strcmp(Y,classes(j)); % Create binary classes for each classifier
    SVMModels{j} = fitcsvm(X,indx,'ClassNames',[false true],'Standardize',true,...
        'KernelFunction','rbf','BoxConstraint',1);
end
```

SVMModels 是 3×1 元胞数组，其中每个元胞包含一个 **ClassificationSVM** 分类器。对于每个元胞，正类分别是 setosa、versicolor 和 virginica。

在图中定义精细网格，并将坐标点视为来自训练数据分布的新观测值。使用每个分类器估计新观测值的分数。

```
d = 0.02;
[x1Grid,x2Grid] = meshgrid(min(X(:,1)):d:max(X(:,1)),...
    min(X(:,2)):d:max(X(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
N = size(xGrid,1);
Scores = zeros(N,numel(classes));

for j = 1:numel(classes)
    [~,score] = predict(SVMModels{j},xGrid);
    Scores(:,j) = score(:,2); % Second column contains positive-class scores
end
```

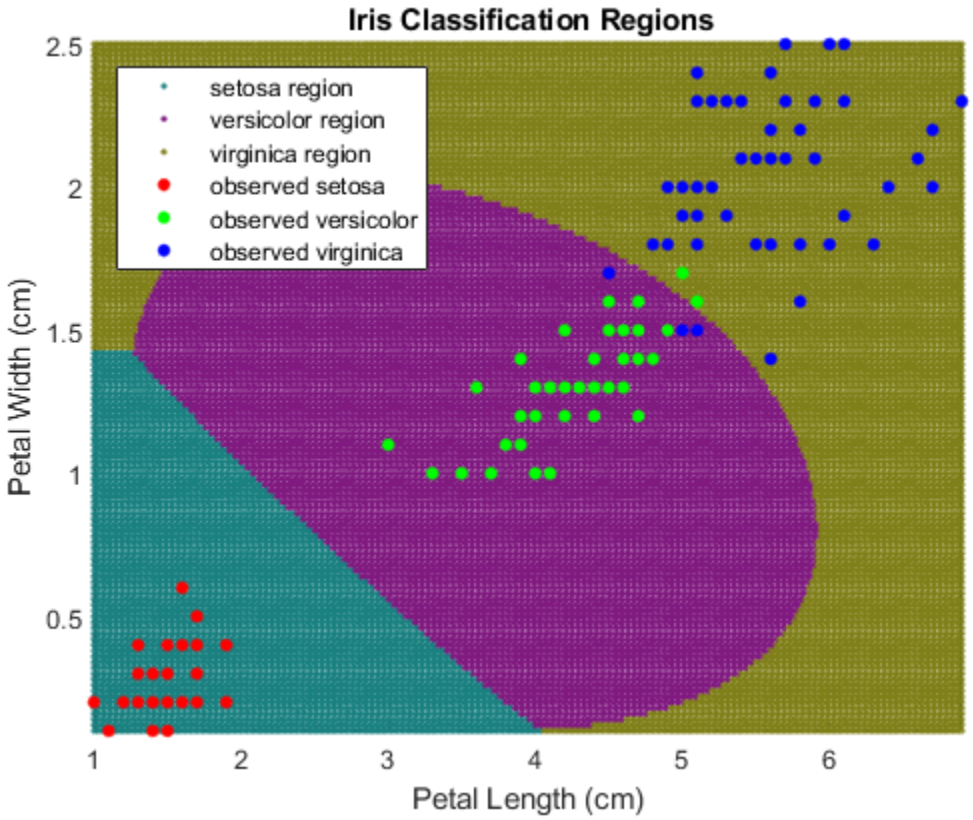
Scores 的每行包含三个分数。分数最高的元素的索引是新类观测值最可能属于的类的索引。

将每个新观测值与给它最高分数的分类器相关联。

```
[~,maxScore] = max(Scores,[],2);
```

基于对应新观测值所属的类，在绘图区域中着色。

```
figure
h(1:3) = gscatter(xGrid(:,1),xGrid(:,2),maxScore,...
    [0.1 0.5 0.5; 0.5 0.1 0.5; 0.5 0.5 0.1]);
hold on
h(4:6) = gscatter(X(:,1),X(:,2),Y);
title('{\bf Iris Classification Regions}');
xlabel('Petal Length (cm)');
ylabel('Petal Width (cm)');
legend(h,{'setosa region','versicolor region','virginica region',...
    'observed setosa','observed versicolor','observed virginica'},...
    'Location','Northwest');
axis tight
hold off
```



优化 SVM 分类器

使用 **fitcsvm** 自动优化超参数。

加载 **ionosphere** 数据集。

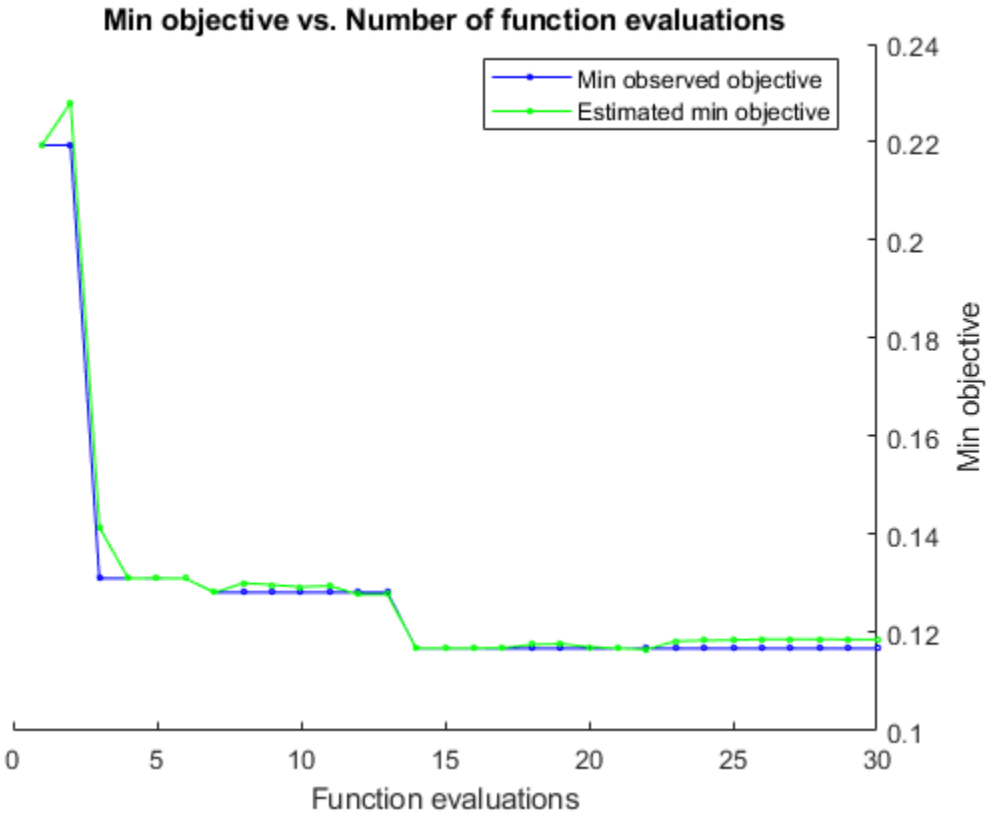
```
load ionosphere
```

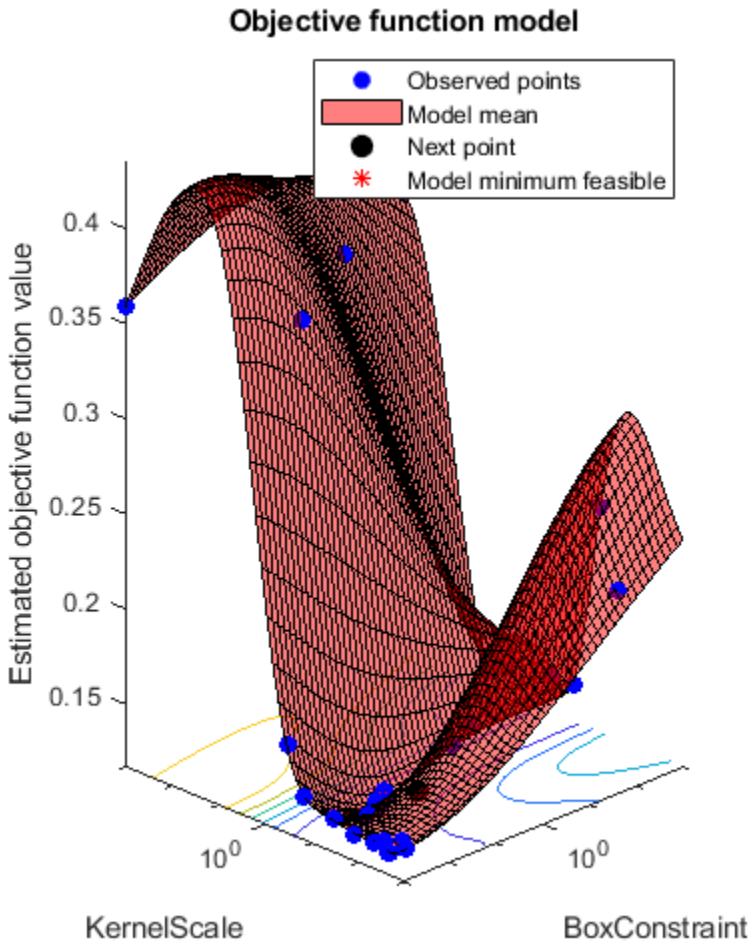
通过使用自动超参数优化，找到最小化五折交叉验证损失的超参数。为了实现可再现性，请设置随机种子并使用 **'expected-improvement-plus'** 采集函数。

```
rng default
Mdl = fitcsvm(X,Y,'OptimizeHyperparameters','auto', ...
   'HyperparameterOptimizationOptions',struct('AcquisitionFunctionName', ...
   'expected-improvement-plus'))
```

```
|=====================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | BoxConstraint|  KernelScale |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|=====================================================================================================|
|    1 | Best   |     0.20513 |     16.181 |     0.20513 |     0.20513 |      64.836 |    0.0015729 |
|    2 | Accept |     0.35897 |    0.094932 |     0.20513 |     0.21471 |     0.036335 |       5.5755 |
|    3 | Best   |     0.13105 |      6.4088 |     0.13105 |     0.14133 |    0.0022147 |    0.0023957 |
|    4 | Accept |     0.35897 |    0.091302 |     0.13105 |     0.13109 |       5.1259 |        98.62 |
|    5 | Accept |     0.13675 |      13.028 |     0.13105 |     0.13111 |    0.0011599 |    0.0010098 |
```

```
|   6 | Accept |    0.13675 |     2.8563 |    0.13105 |    0.13119 |   0.0010151 |   0.0059137 |
|   7 | Accept |     0.1339 |     8.0811 |    0.13105 |    0.13127 |   0.0010281 |   0.0027003 |
|   8 | Accept |    0.13675 |     13.106 |    0.13105 |    0.13232 |    0.016269 |   0.0024597 |
|   9 | Accept |    0.13105 |     10.554 |    0.13105 |    0.13137 |   0.0021526 |   0.0025081 |
|  10 | Best   |    0.12821 |     14.057 |    0.12821 |    0.12841 |   0.0086928 |   0.0010304 |
|  11 | Accept |    0.12821 |     14.173 |    0.12821 |    0.12828 |    0.010039 |   0.0010077 |
|  12 | Accept |    0.13675 |     13.902 |    0.12821 |    0.13162 |   0.0071238 |   0.0010245 |
|  13 | Accept |     0.1339 |     13.12 |    0.12821 |    0.13179 |   0.0050166 |   0.0016385 |
|  14 | Best   |    0.11966 |     14.442 |    0.11966 |    0.12919 |    0.013746 |   0.0010181 |
|  15 | Accept |    0.13105 |     15.207 |    0.11966 |     0.1294 |    0.025222 |    0.001012 |
|  16 | Accept |    0.13105 |     15.241 |    0.11966 |    0.12958 |    0.024019 |    0.001013 |
|  17 | Accept |    0.35897 |     0.11015 |    0.11966 |    0.12962 |   0.0010245 |      994.95 |
|  18 | Accept |    0.13675 |      13.88 |    0.11966 |    0.12925 |      990.28 |     0.36736 |
|  19 | Accept |    0.18234 |     16.368 |    0.11966 |    0.12917 |      949.33 |    0.082849 |
|  20 | Accept |     0.1339 |     5.9526 |    0.11966 |    0.12914 |      979.17 |      1.3107 |
|=====================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   | BoxConstraint|  KernelScale |
|      | result |             | runtime     | (observed)  | (estim.)    |              |              |
|=====================================================================================================|
|  21 | Accept |    0.14245 |      15.28 |    0.11966 |    0.13062 |    0.024598 |   0.0010041 |
|  22 | Accept |    0.13675 |     11.218 |    0.11966 |    0.13058 |         907 |     0.70254 |
|  23 | Accept |    0.35897 |    0.12038 |    0.11966 |    0.13062 |      999.82 |      995.02 |
|  24 | Accept |    0.15385 |    0.11103 |    0.11966 |    0.13042 |    0.001002 |     0.30762 |
|  25 | Accept |    0.12251 |   0.096482 |    0.11966 |    0.12275 |   0.0010105 |    0.056734 |
|  26 | Accept |     0.1339 |    0.16554 |    0.11966 |    0.12251 |    0.021279 |    0.054708 |
|  27 | Accept |    0.12821 |   0.087118 |    0.11966 |    0.12517 |   0.0010127 |    0.064566 |
|  28 | Accept |    0.12821 |    0.11482 |    0.11966 |    0.12557 |   0.0010014 |    0.036667 |
|  29 | Accept |     0.1339 |   0.091185 |    0.11966 |    0.12743 |   0.0010509 |    0.078319 |
|  30 | Accept |    0.12251 |    0.12989 |    0.11966 |    0.12706 |   0.0030057 |    0.028009 |
```


Min objective vs. Number of function evaluations
Min observed objective
Estimated min objective
0.24
0.22
0.2
0.18
0.16
0.14
0.12
0.1
Min objective
0
5
10
15
20
25
30
Function evaluations



Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 273.4325 seconds
Total objective function evaluation time: 234.2696

Best observed feasible point:

BoxConstraint	KernelScale
0.013746	0.0010181

Observed objective function value = 0.11966
Estimated objective function value = 0.13091
Function evaluation time = 14.442

Best estimated feasible point (according to models):

BoxConstraint	KernelScale

```
    0.0010105      0.056734

Estimated objective function value = 0.12706
Estimated function evaluation time = 0.091357

Mdl =
  ClassificationSVM
                ResponseName: 'Y'
       CategoricalPredictors: []
                  ClassNames: {'b'  'g'}
              ScoreTransform: 'none'
             NumObservations: 351
    HyperparameterOptimizationResults: [1x1 BayesianOptimization]
                       Alpha: [117x1 double]
                        Bias: -2.6521
            KernelParameters: [1x1 struct]
              BoxConstraints: [351x1 double]
             ConvergenceInfo: [1x1 struct]
             IsSupportVector: [351x1 logical]
                      Solver: 'SMO'


  Properties, Methods
```

输入参数

Tbl - 样本数据
表

用于训练模型的样本数据，指定为表。**Tbl** 的每行对应一个观测值，每列对应一个预测变量。**Tbl** 也可以包含一个额外的对应于响应变量的列。不允许多列变量和字符向量元胞数组以外的元胞数组。

- 如果 **Tbl** 包含响应变量，并且您要使用 **Tbl** 中的所有其余变量作为预测变量，则使用 **ResponseVarName** 指定响应变量。
- 如果 **Tbl** 包含响应变量，并且您只想使用 **Tbl** 中其余一部分变量作为预测变量，则可使用 **formula** 指定公式。
- 如果 **Tbl** 不包含响应变量，则使用 **Y** 指定响应变量。响应变量的长度和 **Tbl** 中的行数必须相等。

数据类型：**table**

ResponseVarName - 响应变量名称
Tbl 中变量的名称

响应变量名称，指定为 **Tbl** 中变量的名称。

您必须以字符向量或字符串标量指定 **ResponseVarName**。例如，如果响应变量 **Y** 存储为 **Tbl.Y**，则将其指定为 **'Y'**。否则，软件在训练模型时会将 **Tbl** 的所有列（包括 **Y**）视为预测变量。

响应变量必须为分类数组、字符数组或字符串数组、逻辑向量或数值向量，或者字符向量元胞数组。如果 **Y** 是字符数组，则响应变量的每个元素必须对应于数组的一行。

使用 **ClassNames** 名称-值参数来指定类的顺序是很好的做法。

数据类型：**char** | **string**

formula - 响应变量和预测变量子集的解释模型
字符向量 | 字符串标量

由响应变量和部分预测变量构成的解释模型，指定为 **'Y~x1+x2+x3'** 形式的字符向量或字符串标量。在此形式中，**Y** 表示响应变量，**x1**、**x2**、**x3** 表示预测变量。

要将 **Tbl** 中的一部分变量指定为模型训练的预测变量，请使用公式。如果您指定公式，则软件不会使用 **Tbl** 中未出现在 **formula** 中的任何变量。

公式中的变量名称必须既是 **Tbl** 中的变量名称 (**Tbl.Properties.VariableNames**)，又是有效的 MATLAB 标识符。您可以使用 **isvarname** 函数来验证 **Tbl** 中的变量名称。如果变量名称无效，可以使用 **matlab.lang.makeValidName** 函数进行转换。

数据类型： **char** | **string**

Y - 类标签
分类数组 | 字符数组 | 字符串数组 | 逻辑向量 | 数值向量 | 字符向量元胞数组

用于训练 SVM 模型的类标签，指定为分类、字符或字符串数组，逻辑或数值向量，或字符向量元胞数组。

- **Y** 最多只能包含两个不同的类。有关多类学习的信息，请参阅 **fitcecoc**。
- 如果 **Y** 是字符数组，则类标签的每个元素必须对应于数组的一行。
- **Y** 的长度和 **Tbl** 或 **X** 的行数必须相等。
- 使用 **ClassNames** 名称-值对组参数来指定类顺序是很好的做法。

数据类型： **categorical** | **char** | **string** | **logical** | **single** | **double** | **cell**

X - 预测变量数据
数值矩阵

用于训练 SVM 分类器的预测变量数据，指定为数值矩阵。

X 的每行对应一个观测值（也称为实例或示例），每列对应一个预测变量（也称为特征）。

Y 的长度和 **X** 的行数必须相等。

要按预测变量在 **X** 中出现的顺序指定预测变量的名称，请使用 **'PredictorNames'** 名称-值对组参数。

数据类型： **double** | **single**

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例： **fitcsvm(X,Y,'KFold',10,'Cost',[0 2;1 0],'ScoreTransform','sign')** 执行 10 折交叉验证，对假正的罚分是对假负的罚分的两倍，并使用 sign 函数转换分数。

SVM 选项

BoxConstraint - 框约束
1 （默认） | 正标量

框约束 （第 33-64 页），指定为逗号分隔的对组，其中包含 **'BoxConstraint'** 和一个正标量。

对于一类学习，软件始终将框约束设置为 1。

有关 BoxConstraint、Cost、Prior、Standardize 和 Weights 的关系和算法行为的详细信息，请参阅“算法” （第 33-67 页）。

示例： 'BoxConstraint',100

数据类型： double | single

KernelFunction - 核函数
'linear' | 'gaussian' | 'rbf' | 'polynomial' | 函数名称

用于计算 Gram 矩阵 （第 33-64 页）的元素的核函数，指定为以逗号分隔的对组，其中包含 'KernelFunction' 和核函数名称。假设 $G(x_j,x_k)$ 是 Gram 矩阵的元素 (j,k)，其中 x_j 和 x_k 是 p 维向量，表示 X 中的观测值 j 和 k。下表说明支持的核函数名称及其函数形式。

核函数名称	说明	公式
'gaussian' 或 'rbf'	高斯或径向基函数 (RBF) 核，默认用于一类学习	$G(x_j,x_k) = \exp\left(-\|x_j - x_k\|^2\right)$
'linear'	线性核，默认用于二类学习	$G(x_j,x_k) = x_j'x_k$
'polynomial'	多项式核。使用 'PolynomialOrder',q 指定 q 次多项式核。	$G(x_j,x_k) = (1 + x_j'x_k)^q$

您可以通过设置 'KernelFunction','kernel' 来设置自己的核函数，例如 kernel。值 kernel 必须采用以下形式。

```
function G = kernel(U,V)
```

其中：

- U 是 m×p 矩阵。列对应于预测变量，行对应于观测值。
- V 是 n×p 矩阵。列对应于预测变量，行对应于观测值。
- G 是 U 和 V 行的 m×n Gram 矩阵 （第 33-64 页）。

kernel.m 必须在 MATLAB 路径上。

避免对核函数使用泛型名称是很好的做法。例如，调用 sigmoid 核函数 'mysigmoid'，而不是 'sigmoid'。

示例： 'KernelFunction','gaussian'

数据类型： char | string

KernelScale - 核尺度参数
1 （默认） | 'auto' | 正标量

核尺度参数，指定为以逗号分隔的对组，其中包含 'KernelScale' 和 'auto' 或正标量。软件将预测变量矩阵 X 的所有元素除以 KernelScale 的值。然后，软件应用适当的核范数来计算 Gram 矩阵。

- 如果您指定 'auto'，则软件使用启发式过程选择适当的尺度因子。这种启发式过程使用二次抽样，因此不同调用的估计值可能不同。因此，为了重现结果，请在训练前使用 rng 设置随机数种子。
- 如果您指定 KernelScale 和您自己的核函数，例如 'KernelFunction','kernel'，则软件会引发错误。您必须在 kernel 中应用尺度缩放。

示例：'KernelScale','auto'

数据类型：double | single | char | string

PolynomialOrder - 多项式核函数阶
3（默认）| 正整数

多项式核函数阶，指定为以逗号分隔的对组，其中包含 'PolynomialOrder' 和正整数。

如果您设置 'PolynomialOrder' 且 KernelFunction 不是 'polynomial'，则软件会引发错误。

示例：'PolynomialOrder',2

数据类型：double | single

KernelOffset - 核偏移量参数
非负标量

核偏移量参数，指定为以逗号分隔的对组，其中包含 'KernelOffset' 和一个非负标量。

软件将 KernelOffset 添加到 Gram 矩阵的每个元素。

默认值为：

- 0，如果求解器是 SMO（也就是说，您设置 'Solver','SMO'）
- 0.1，如果求解器是 ISDA（也就是说，您设置 'Solver','ISDA'）

示例：'KernelOffset',0

数据类型：double | single

Standardize - 指示是否标准化预测变量数据的标志
false（默认）| true

指示是否标准化预测变量数据的标志，指定为以逗号分隔的对组，其中包含 'Standardize' 和 true (1) 或 false (0)。

如果您设置 'Standardize',true：

- 软件按对应的加权列均值和标准差对每个预测变量（X 或 Tbl）进行中心化并缩放。有关加权标准化的详细信息，请参阅“算法” （第 33-67 页）。MATLAB 不对为分类预测变量生成的虚拟变量列中包含的数据进行标准化。
- 软件使用经过标准化的预测变量训练分类器，但会将未标准化的预测变量以矩阵或表形式存储在分类器属性 X 中。

示例：'Standardize',true

数据类型：logical

Solver - 优化例程
'ISDA' | 'L1QP' | 'SMO'

优化例程，指定为以逗号分隔的对组，其中包含 'Solver' 和下表中的一个值。

值	说明
'ISDA'	迭代单数据算法（请参阅 [30]）

值	说明
'L1QP'	使用 quadprog 通过二次规划实现 L1 软边距最小化。此选项需要 Optimization Toolbox 许可证。有关详细信息，请参阅“二次规划定义” (Optimization Toolbox)。
'SMO'	序列最小优化（请参阅 [17]）

对于二类学习，如果您将 'OutlierFraction' 设置为正值，则默认值为 'ISDA'；否则为 'SMO'。

示例： 'Solver','ISDA'

Alpha - alpha 系数的初始估计值
非负值的数值向量

alpha 系数的初始估计值，指定为以逗号分隔的对组，其中包含 'Alpha' 和非负值的数值向量。Alpha 的长度必须等于 X 的行数。

- 'Alpha' 的每个元素对应于 X 中的一个观测值。
- 'Alpha' 不能包含任何 NaN。
- 如果您指定 'Alpha' 和任一交叉验证名称-值对组参数（'CrossVal'、'CVPartition'、'Holdout'、'KFold' 或 'Leaveout'），软件将返回错误。

如果 Y 包含任何缺失值，则删除与缺失值对应的 Y、X 和 'Alpha' 的所有行。即，输入：

```
idx = ~isundefined(categorical(Y));
Y = Y(idx,:);
X = X(idx,:);
alpha = alpha(idx);
```

然后分别将 Y、X 和 alpha 作为响应、预测变量和初始 alpha 估计值传递。

默认值为：

- 0.5*ones(size(X,1),1)（用于一类学习）
- zeros(size(X,1),1)（用于二类学习）

示例： 'Alpha',0.1*ones(size(X,1),1)

数据类型： double | single

CacheSize - 缓存大小
1000 （默认） | 'maximal' | 正标量

缓存大小，指定为以逗号分隔的对组，其中包含 'CacheSize' 和 'maximal' 或正标量。

如果 CacheSize 是 'maximal'，则软件预留足够的内存来容纳整个 n×n Gram 矩阵 （第 33-64 页）。

如果 CacheSize 是正标量，则软件会预留 CacheSize MB 内存用于训练模型。

示例： 'CacheSize','maximal'

数据类型： double | single | char | string

ClipAlphas - 指定是否限定 alpha 系数的标志
true （默认） | false

指定是否限定 alpha 系数的标志，指定为以逗号分隔的对组，其中包含 **'ClipAlphas'** 和 **true** 或 **false**。

假设观测值 j 的 alpha 系数是 α_j，观测值 j 的框约束是 C_j，j = 1、…、n，其中 n 是训练样本大小。

值	说明
true	在每次迭代中，如果 α_j 接近 0 或接近 C_j，则 MATLAB 分别将 α_j 设置为 0 或 C_j。
false	MATLAB 在优化过程中不更改 alpha 系数。

MATLAB 将 α 的最终值存储在经过训练的 SVM 模型对象的 **Alpha** 属性中。

ClipAlphas 会影响 SMO 和 ISDA 收敛。

示例：**'ClipAlphas',false**

数据类型：**logical**

Nu - 用于一类学习的 ν 参数
0.5（默认）| 正标量

用于“一类学习”（第 33-64 页）的 ν 参数，指定为由 **'Nu'** 和正标量组成的以逗号分隔的对组。**Nu** 必须大于 **0** 且最大为 **1**。

Nu 用于在确保大多数训练样本为正类和尽量减小分数函数权重之间进行权衡。

示例：**'Nu',0.25**

数据类型：**double** | **single**

NumPrint - 优化诊断消息的输出间隔（迭代次数）
1000（默认）| 非负整数

优化诊断消息的输出间隔（迭代次数），指定为由 **'NumPrint'** 和非负整数组成的以逗号分隔的对组。

如果您指定 **'Verbose',1** 和 **'NumPrint',numprint**，则软件会在命令行窗口中每隔 **numprint** 次迭代显示来自 SMO 和 ISDA 的所有优化诊断消息。

示例：**'NumPrint',500**

数据类型：**double** | **single**

OutlierFraction - 训练数据中离群值的预期比例
0（默认）| 区间 [0,1) 中的数值标量

训练数据中离群值的预期比例，指定为以逗号分隔的对组，其中包含 **'OutlierFraction'** 和区间 [0,1) 中的数值标量。

假设您设置 **'OutlierFraction',outlierfraction**，其中 **outlierfraction** 是大于 0 的值。

- 对于二类学习，软件实现稳健学习。换句话说，当优化算法收敛时，软件尝试删除 100***outlierfraction**% 的观测值。删除的观测值对应于幅值较大的梯度。
- 对于一类学习，软件会使用合适的偏差项，使得训练集中 **outlierfraction** 的观测值具有负分数。

示例：**'OutlierFraction',0.01**

数据类型：**double** | **single**

RemoveDuplicates - 指定是否用单一观测值替换重复观测值的标志
false（默认）| **true**

指定是否在训练数据中用单一观测值替换重复观测值的标志，指定为以逗号分隔的对组，其中包含 **'RemoveDuplicates'** 和 **true** 或 **false**。

如果 **RemoveDuplicates** 是 **true**，则 **fitcsvm** 用相同值的单一观测值替换训练数据中的重复观测值。该单一观测值的权重等于对应删除副本的权重总和（请参阅 **Weights**）。

提示 如果您的数据集包含许多重复的观测值，则指定 **'RemoveDuplicates',true** 可以大大缩短收敛时间。

数据类型：**logical**

Verbose - 详细级别
0（默认）| **1** | **2**

详细级别，指定为以逗号分隔的对组，其中包含 **'Verbose'** 和 **0**、**1** 或 **2**。**Verbose** 的值控制软件在命令行窗口中显示的优化信息量，并将这些信息作为结构体保存到 **Mdl.ConvergenceInfo.History** 中。

下表总结了可用的详细级别选项。

值	说明
0	软件不显示或保存收敛信息。
1	软件显示诊断消息，并每隔 **numprint** 次迭代保存收敛条件，其中 **numprint** 是名称-值对组参数 **'NumPrint'** 的值。
2	软件显示诊断消息，并在每次迭代时保存收敛条件。

示例：**'Verbose',1**

数据类型：**double** | **single**

其他分类选项

CategoricalPredictors - 分类预测变量列表
正整数向量 | 逻辑向量 | 字符矩阵 | 字符串数组 | 字符向量元胞数组 | **'all'**

分类预测变量列表，指定为本表中的值之一。

值	说明
正整数向量	向量中的每个条目为一个索引值，对应于预测变量数据中包含分类变量的一列。索引值介于 1 和 **p** 之间，其中 **p** 是用于训练模型的预测变量数目。 如果 **fitcsvm** 使用输入变量的子集作为预测变量，则此函数仅使用此子集对预测变量进行索引。**'CategoricalPredictors'** 值不计算响应变量、观测值权重变量和函数不使用的任何其他变量。
逻辑向量	**true** 条目表示预测变量数据中的对应列是分类变量。向量的长度为 **p**。
字符矩阵	矩阵的每行均为预测变量的名称。名称必须与 **PredictorNames** 中的条目相匹配。系统用空格填充名称以使字符矩阵的每行具有相同的长度。

值	说明
字符串数组或字符向量元胞数组	数组中的每个元素均为预测变量的名称。名称必须与 **PredictorNames** 中的条目相匹配。
'all'	所有预测变量均为分类预测变量。

默认情况下，如果预测变量数据在表 (**Tbl**) 中，并且它是逻辑向量、分类向量、字符数组、字符串数组或字符向量元胞数组，则 **fitcsvm** 假设变量是分类变量。如果预测变量数据是矩阵 (**X**)，**fitcsvm** 假设所有预测变量均为连续的。要将任何其他预测变量识别为分类预测变量，请使用 **'CategoricalPredictors'** 名称-值参数指定它们。

对于已识别的分类预测变量，**fitcsvm** 根据分类变量是无序还是有序，使用两种不同方案创建虚拟变量。对于无序分类变量，**fitcsvm** 为分类变量的每个水平创建一个虚拟变量。对于有序分类变量，**fitcsvm** 创建的虚拟变量数比类别数少一个。有关详细信息，请参阅 “Automatic Creation of Dummy Variables”。

示例：**'CategoricalPredictors','all'**

数据类型：**single** | **double** | **logical** | **char** | **string** | **cell**

ClassNames - 用于训练的类的名称
分类数组 | 字符数组 | 字符串数组 | 逻辑向量 | 数值向量 | 字符向量元胞数组

用于训练的类的名称，指定为分类、字符或字符串数组，逻辑或数值向量，或字符向量元胞数组。**ClassNames** 必须与 **Tbl** 或 **Y** 中的响应变量具有相同的数据类型。

如果 **ClassNames** 是字符数组，则每个元素必须对应于数组的一行。

使用 **ClassNames** 可以：

- 指定训练期间类的顺序。
- 指定对应于类顺序的任何输入或输出参数维度的顺序。例如，使用 **ClassNames** 指定 **Cost** 的维度的顺序或 **predict** 返回的分类分数的列顺序。
- 选择用于训练的类的子集。例如，假设 **Y** 中所有不同类名的集合是 **{'a','b','c'}**。要仅使用来自 **'a'** 和 **'c'** 类的观测值训练模型，请指定 **'ClassNames',{'a','c'}**。

ClassNames 的默认值为 **Tbl** 或 **Y** 中的响应变量的所有不同类名的集合。

示例：**'ClassNames',{'b','g'}**

数据类型：**categorical** | **char** | **string** | **logical** | **single** | **double** | **cell**

Cost - 误分类代价
方阵 | 结构体数组

误分类代价，指定为以逗号分隔的对组，其中包括 **'Cost'** 和方阵或结构体数组。

- 如果您指定方阵 **Cost** 且观测值的真实类是 **i**，则 **Cost(i,j)** 是将一个点分类到 **j** 类的代价。也就是说，行对应于真实类，列对应于预测的类。要为 **Cost** 的对应行和列指定类顺序，则还需指定 **ClassNames** 名称-值对组参数。
- 如果您指定结构体 **S**，则它必须有两个字段：
  - **S.ClassNames**，其中包含类名作为一个变量，变量的数据类型与 **Y** 相同。
  - **S.ClassificationCosts**，其中包含按 **S.ClassNames** 中的顺序排列行和列的代价矩阵

对于二类学习，如果您指定代价矩阵，则软件将通过合并代价矩阵中所述的罚分来更新先验概率。因此，代价矩阵重置为默认值。有关 **BoxConstraint**、**Cost**、**Prior**、**Standardize** 和 **Weights** 的关系和算法行为的详细信息，请参阅“算法” （第 33-67 页）。

默认值为：

- **Cost = 0**（用于一类学习）
- **Cost(i,j) = 1**（如果 **i ~= j**）和 **Cost(i,j) = 0**（如果 **i = j**），用于二类学习

示例： **'Cost',[0,1;2,0]**

数据类型： **double** | **single** | **struct**

PredictorNames - 预测变量名称
唯一名称的字符串数组 | 唯一字符向量元胞数组

预测变量名称，指定为唯一名称字符串数组或唯一字符向量元胞数组。**PredictorNames** 的功能取决于您提供训练数据的方式。

- 如果您提供 **X** 和 **Y**，则可以使用 **PredictorNames** 为 **X** 中的预测变量指定名称。
  - **PredictorNames** 中名称的顺序必须对应于 **X** 中列的顺序。也就是说，**PredictorNames{1}** 是 **X(:,1)** 的名称，**PredictorNames{2}** 是 **X(:,2)** 的名称，依此类推。此外，**size(X,2)** 和 **numel(PredictorNames)** 必须相等。
  - 默认情况下，**PredictorNames** 是 **{'x1','x2',...}**。
- 如果您提供 **Tbl**，则您可以使用 **PredictorNames** 来选择在训练中使用哪些预测变量。也就是说，**fitcsvm** 仅使用 **PredictorNames** 中的预测变量和训练过程中的响应变量。
  - **PredictorNames** 必须为 **Tbl.Properties.VariableNames** 的子集，并且不能包含响应变量的名称。
  - 默认情况下，**PredictorNames** 包含所有预测变量的名称。
  - 最好仅使用 **'PredictorNames'** 或 **formula**（但不能同时使用两者）来指定训练的预测变量。

示例： **'PredictorNames',{'SepalLength','SepalWidth','PetalLength','PetalWidth'}**

数据类型： **string** | **cell**

Prior - 先验概率
'empirical' （默认） | **'uniform'** | 数值向量 | 结构体数组

每个类的先验概率，指定为以逗号分隔的对组，其中包含 **'Prior'** 和下表中的一个值。

值	说明
'empirical'	类先验概率是 **Y** 中的类相对频数。
'uniform'	所有类先验概率等于 1/K，其中 K 是类的数目。
数值向量	向量中的每个元素均为类先验概率。根据 **Mdl.ClassNames** 对元素进行排序，或使用 **ClassNames** 名称-值对组参数指定顺序。软件对元素进行归一化，使其总和为 **1**。

值	说明
结构体	具有以下两个字段的结构体 S： • S.ClassNames 包含类名作为一个变量，变量的数据类型与 Y 相同。 • S.ClassProbs 包含对应先验概率的向量。软件会对向量的元素进行归一化，使其总和为 1。

对于二类学习，如果您指定代价矩阵，则软件将通过合并代价矩阵中所述的罚分来更新先验概率。有关 **BoxConstraint**、**Cost**、**Prior**、**Standardize** 和 **Weights** 的关系和算法行为的详细信息，请参阅 “算法” （第 33-67 页）。

示例： **struct('ClassNames',{{'setosa','versicolor','virginica'}},'ClassProbs',1:3)**

数据类型： **char** | **string** | **double** | **single** | **struct**

ResponseName - 响应变量名称
'Y' （默认） | 字符向量 | 字符串标量

响应变量名称，指定为字符向量或字符串标量。

- 如果您提供 **Y**，则您可以使用 **'ResponseName'** 来指定响应变量的名称。
- 如果您提供 **ResponseVarName** 或 **formula**，则您无法使用 **'ResponseName'**。

示例： **'ResponseName','response'**

数据类型： **char** | **string**

ScoreTransform - 分数转换
'none' （默认） | **'doublelogit'** | **'invlogit'** | **'ismax'** | **'logit'** | 函数句柄 | ...

分数变换，指定为字符向量、字符串标量或函数句柄。

下表总结了可用的字符向量和字符串标量。

值	说明
'doublelogit'	$1/(1 + e^{-2x})$
'invlogit'	$\log(x / (1 - x))$
'ismax'	将分数最高的类的分数设置为 1，并将所有其他类的分数设置为 0
'logit'	$1/(1 + e^{-x})$
'none' 或 **'identity'**	x（无变换）
'sign'	–1 表示 x < 0 0 表示 x = 0 1 表示 x > 0
'symmetric'	2x – 1
'symmetricismax'	将分数最高的类的分数设置为 1，并将所有其他类的分数设置为 –1
'symmetriclogit'	$2/(1 + e^{-x}) - 1$

对于 MATLAB 函数或您定义的函数，请使用其函数句柄进行分数变换。函数句柄必须接受矩阵（原始分数）并返回相同大小（转换后的分数）的矩阵。

示例： 'ScoreTransform','logit'

数据类型： char | string | function_handle

Weights - 观测值权重
数值向量 | Tbl 中变量的名称

观测值权重，指定为以逗号分隔的对组，其中包含 'Weights' 和由正值组成的数值向量或 Tbl 中变量的名称。软件使用 Weights 中的对应值对 X 或 Tbl 的每行中的观测值进行加权。Weights 的大小必须等于 X 或 Tbl 的行数。

如果您将输入数据指定为表 Tbl，则 Weights 可以是包含数值向量的 Tbl 中的变量的名称。在本例中，您必须将 Weights 指定为字符向量或字符串标量。例如，如果权重向量 W 存储为 Tbl.W，则将其指定为 'W'。否则，软件在训练模型时会将 Tbl 的所有列（包括 W）视为预测变量或响应变量。

默认情况下，Weights 是 ones(n,1)，其中 n 是 X 或 Tbl 中的观测值数目。

软件会对 Weights 进行归一化，使其总和等于对应类中的先验概率值。有关 BoxConstraint、Cost、Prior、Standardize 和 Weights 的关系和算法行为的详细信息，请参阅“算法” （第 33-67 页）。

数据类型： double | single | char | string

注意 不能将任何交叉验证名称-值对组参数与 'OptimizeHyperparameters' 名称-值对组参数结合使用。您只能通过使用 'HyperparameterOptimizationOptions' 名称-值对组参数来修改 'OptimizeHyperparameters' 的交叉验证。

交叉验证选项

CrossVal - 指定是否训练交叉验证分类器的标志
'off' （默认） | 'on'

指定是否训练交叉验证分类器的标志，指定为以逗号分隔的对组，其中包含 'Crossval' 和 'on' 或 'off'。

如果您指定 'on'，则软件使用 10 折来训练交叉验证分类器。

您可以使用 CVPartition、Holdout、KFold 或 Leaveout 名称-值对组参数覆盖此交叉验证设置。一次只能使用一个交叉验证名称-值对组参数来创建一个交叉验证模型。

或者，稍后通过将 Mdl 传递给 crossval 来进行交叉验证。

示例： 'Crossval','on'

CVPartition - 交叉验证分区
[] （默认） | cvpartition 分区对象

交叉验证分区，指定为由 cvpartition 创建的 cvpartition 分区对象。该分区对象指定交叉验证的类型以及训练集和验证集的索引。

要创建交叉验证模型，您只能指定以下四个名称-值参数之一：CVPartition、Holdout、KFold 或 Leaveout。

示例： 假设您使用 cvp = cvpartition(500,'KFold',5) 创建一个随机分区，用于对 500 个观测值进行 5 折交叉验证。然后，您可以使用 'CVPartition',cvp 指定交叉验证的模型。

Holdout - 用于留出法验证的数据比例
范围 (0,1) 内的标量值

用于留出法验证的数据比例，指定为 (0,1) 范围内的标量值。如果您指定 **'Holdout',p**，则软件将完成以下步骤：

1 随机选择并预留 **p*100%** 的数据作为验证数据，并使用其余数据训练模型。

2 将经过训练的紧凑模型存储在交叉验证模型的 **Trained** 属性中。

要创建交叉验证模型，您只能指定以下四个名称-值参数之一：**CVPartition**、**Holdout**、**KFold** 或 **Leaveout**。

示例： **'Holdout',0.1**

数据类型： **double** | **single**

KFold - 折的数目
10（默认）| 大于 1 的正整数值

交叉验证模型中使用的折的数目，指定为大于 1 的正整数值。如果您指定 **'KFold',k**，则软件将完成以下步骤：

1 将数据随机分为 **k** 个数据集。

2 对于每个数据集，预留用作验证数据的数据集，并使用其他 **k** – 1 数据集训练模型。

3 将 **k** 个紧凑的经过训练的模型存储在交叉验证模型的 **Trained** 属性中的 **k**×1 元胞向量中。

要创建交叉验证模型，您只能指定以下四个名称-值参数之一：**CVPartition**、**Holdout**、**KFold** 或 **Leaveout**。

示例： **'KFold',5**

数据类型： **single** | **double**

Leaveout - 留一法交叉验证标志
'off'（默认）| **'on'**

留一法交叉验证标志，指定为 **'on'** 或 **'off'**。如果您指定 **'Leaveout','on'**，则对于 n 个观测值中的每一个（其中 n 是不包括缺失观测值的观测值数目，在模型的 **NumObservations** 属性中指定），软件完成以下步骤：

1 预留作为验证数据的观测值，并使用其他 n - 1 个观测值训练模型。

2 将 n 个紧凑的经过训练的模型存储在交叉验证模型的 **Trained** 属性中的 n×1 元胞向量中。

要创建交叉验证模型，您只能指定以下四个名称-值参数之一：**CVPartition**、**Holdout**、**KFold** 或 **Leaveout**。

示例： **'Leaveout','on'**

收敛控制选项

DeltaGradientTolerance - 梯度差的容差
非负标量

通过序列最小优化 (SMO) 或迭代单数据算法 (ISDA) 获得的上违反量和下违反量之间梯度差的容差，指定为由 **'DeltaGradientTolerance'** 和非负标量组成的以逗号分隔的对组。

如果 **DeltaGradientTolerance** 是 **0**，则软件不使用梯度差的容差来检查优化收敛。

默认值为：

- 如果求解器是 SMO（例如，您设置 **'Solver','SMO'**），默认值为 **1e-3**
- 如果求解器是 ISDA（例如，您设置 **'Solver','ISDA'**），默认值 **0**

示例：**'DeltaGradientTolerance',1e-2**

数据类型：**double** | **single**

GapTolerance - 可行性间隙容差
0（默认）| 非负标量

通过 SMO 或 ISDA 获得的可行性间隙容差，指定为由 **'GapTolerance'** 和非负标量组成的以逗号分隔的对组。

如果 **GapTolerance** 是 **0**，则软件不使用可行性间隙容差来检查优化收敛。

示例：**'GapTolerance',1e-2**

数据类型：**double** | **single**

IterationLimit - 数值优化迭代的最大次数
1e6（默认）| 正整数

数值优化迭代的最大次数，指定为以逗号分隔的对组，其中包含 **'IterationLimit'** 和一个正整数。

无论优化例程是否成功收敛，软件都会返回经过训练的模型。**Mdl.ConvergenceInfo** 包含收敛信息。

示例：**'IterationLimit',1e8**

数据类型：**double** | **single**

KKTTolerance - Karush-Kuhn-Tucker 互补条件违规容限
非负标量

Karush-Kuhn-Tucker (KKT) 互补条件 （第 33-64 页）违规容限，指定为由 **'KKTTolerance'** 和非负标量组成的以逗号分隔的对组。

如果 **KKTTolerance** 是 **0**，则软件不使用 KKT 互补条件违规容限来检查优化收敛。

默认值为：

- 如果求解器是 SMO（例如，您设置 **'Solver','SMO'**），默认值为 **0**
- 如果求解器是 ISDA（例如，您设置 **'Solver','ISDA'**），默认值 **1e-3**

示例：**'KKTTolerance',1e-2**

数据类型：**double** | **single**

ShrinkagePeriod - 活动集各次归约之间的迭代次数
0（默认）| 非负整数

活动集各次归约之间的迭代次数，指定为由 **'ShrinkagePeriod'** 和非负整数组成的以逗号分隔的对组。

如果您设置 **'ShrinkagePeriod',0**，则软件不会缩小活动集。

示例：**'ShrinkagePeriod',1000**

数据类型：**double** | **single**

超参数优化选项

OptimizeHyperparameters - 要优化的参数
'none'（默认）| 'auto' | 'all' | 合格参数名称的字符串数组或元胞数组 | optimizableVariable 对象的向量

要优化的参数，指定为以逗号分隔的对组，其中包含 'OptimizeHyperparameters' 和下列值之一：

- 'none' - 不优化。
- 'auto' - 使用 {'BoxConstraint','KernelScale'}。
- 'all' - 优化所有合格参数。
- 合格参数名称的字符串数组或元胞数组。
- optimizableVariable 对象的向量，通常是 hyperparameters 的输出。

优化尝试通过更改参数来最小化 fitcsvm 的交叉验证损失（误差）。有关交叉验证损失的信息，请参阅“Classification Loss”。要控制交叉验证类型和优化的其他方面，请使用 HyperparameterOptimizationOptions 名称-值对组参数。

注意 'OptimizeHyperparameters' 值会覆盖您使用其他名称-值对组参数设置的任何值。例如，将 'OptimizeHyperparameters' 设置为 'auto' 会导致应用 'auto' 值。

fitcsvm 的合格参数包括：

- BoxConstraint - fitcsvm 在正值中搜索，默认情况下对数尺度范围为 [1e-3,1e3]。
- KernelScale - fitcsvm 在正值中搜索，默认情况下对数尺度范围为 [1e-3,1e3]。
- KernelFunction - fitcsvm 在 'gaussian'、'linear' 和 'polynomial' 中搜索。
- PolynomialOrder - fitcsvm 在 [2,4] 范围内的整数中搜索。
- Standardize - fitcsvm 在 'true' 和 'false' 中搜索。

通过传递具有非默认值的 optimizableVariable 对象的向量可设置非默认参数。例如：

```
load fisheriris
params = hyperparameters('fitcsvm',meas,species);
params(1).Range = [1e-4,1e6];
```

将 params 作为 OptimizeHyperparameters 的值传递。

默认情况下，迭代消息会出现在命令行中，绘图会根据优化中的超参数个数来显示。对于优化和绘图，目标函数在回归中是 log(1 + cross-validation loss)，在分类中是误分类率。要控制迭代消息的显示，请设置 'HyperparameterOptimizationOptions' 名称-值对组参数的 Verbose 字段。要控制绘图，请设置 'HyperparameterOptimizationOptions' 名称-值对组参数的 ShowPlots 字段。

有关示例，请参阅“优化 SVM 分类器”（第 33-44 页）。

示例：'auto'

HyperparameterOptimizationOptions - 优化的选项
结构体

优化的选项，指定为由 'HyperparameterOptimizationOptions' 和结构体组成的以逗号分隔的对组。此参数修改 OptimizeHyperparameters 名称-值对组参数的效果。结构体中的所有字段均为可选字段。

字段名称	值	默认值
Optimizer	• **'bayesopt'** - 使用贝叶斯优化。在内部，此设置调用 **bayesopt**。 • **'gridsearch'** - 使用网格搜索，每个维度有 **NumGridDivisions** 个值。 • **'randomsearch'** - 在 **MaxObjectiveEvaluations** 个点中随机搜索。 **'gridsearch'** 基于网格的均匀无放回抽样以随机顺序进行搜索。优化后，您可以使用命令 **sortrows(Mdl.HyperparameterOptimizationResults)** 按网格顺序获得一个表。	**'bayesopt'**
AcquisitionFunctionName	• **'expected-improvement-per-second-plus'** • **'expected-improvement'** • **'expected-improvement-plus'** • **'expected-improvement-per-second'** • **'lower-confidence-bound'** • **'probability-of-improvement'** 其名称包含 **per-second** 的采集函数不会产生可重现的结果，因为优化取决于目标函数的运行时间。其名称包含 **plus** 的采集函数在过度开发某个区域时会更改其行为。有关详细信息，请参阅“Acquisition Function Types”。	**'expected-improvement-per-second-plus'**
MaxObjectiveEvaluations	目标函数计算的最大数量。	对于 **'bayesopt'** 或 **'randomsearch'** 为 **30**，对于 **'gridsearch'** 为整个网格
MaxTime	时间限制，指定为正实数。时间限制（以秒为单位），由 **tic** 和 **toc** 测量。运行时间可以超过 **MaxTime**，因为 **MaxTime** 不会中断函数计算。	**Inf**
NumGridDivisions	用于 **'gridsearch'**，表示每个维度中的值的数量。该值可以是由指定每个维度上值个数的正整数组成的向量，也可以是应用于所有维度的标量。对于分类变量，此字段被忽略。	**10**
ShowPlots	指示是否显示绘图的逻辑值。如果为 **true**，此字段绘制最佳目标函数值对迭代编号的图。如果有一个或两个优化参数，并且如果 **Optimizer** 是 **'bayesopt'**，则 **ShowPlots** 也绘制目标函数对参数的模型图。	**true**
SaveIntermediateResults	指示当 **Optimizer** 为 **'bayesopt'** 时是否保存结果的逻辑值。如果为 **true**，此字段将在每次迭代中覆盖名为 **'BayesoptResults'** 的工作区变量。变量是 **BayesianOptimization** 对象。	**false**

字段名称	值	默认值
Verbose	显示到命令行。 • **0** - 不显示迭代消息 • **1** - 显示迭代消息 • **2** - 显示带额外信息的迭代消息 有关详细信息，请参阅 **bayesopt Verbose** 名称-值对组参数。	**1**
UseParallel	指示是否并行运行贝叶斯优化的逻辑值，这需要 Parallel Computing Toolbox™。由于并行时序的不可再现性，并行贝叶斯优化不一定产生可重现的结果。有关详细信息，请参阅 “Parallel Bayesian Optimization”。	**false**
Repartition	逻辑值，指示是否在每次迭代时对交叉验证进行重新分区。如果为 **false**，优化器将使用单个分区进行优化。 设置为 **true** 通常可得到最稳健的结果，因为此设置考虑了分区噪声。然而，为了获得良好的结果，设置为 **true** 后函数计算次数至少会翻一倍。	**false**
请使用以下三个字段名称之一。		
CVPartition	**cvpartition** 对象，由 **cvpartition** 创建。	如果您没有指定任何交叉验证字段，默认为 **'Kfold',5**
Holdout	**(0,1)** 范围内的标量，表示保留比例。	
Kfold	大于 1 的整数。	

示例： **'HyperparameterOptimizationOptions',struct('MaxObjectiveEvaluations',60)**

数据类型： **struct**

输出参数

Mdl - 经过训练的 SVM 分类模型

ClassificationSVM 模型对象 | **ClassificationPartitionedModel** 交叉验证模型对象

经过训练的 SVM 分类模型，以 **ClassificationSVM** 模型对象或 **ClassificationPartitionedModel** 交叉验证模型对象形式返回。

如果您设置任一名称-值对组参数 **KFold**、**Holdout**、**Leaveout**、**CrossVal** 或 **CVPartition**，则 **Mdl** 是 **ClassificationPartitionedModel** 交叉验证分类器。否则，**Mdl** 是 **ClassificationSVM** 分类器。

要引用 **Mdl** 的属性，请使用圆点表示法。例如，在命令行窗口中输入 **Mdl.Alpha** 以显示经过训练的 Lagrange 乘数。

限制

- **fitcsvm** 为一类或二类学习应用训练 SVM 分类器。要使用就有两个以上类的数据训练 SVM 分类器，请使用 **fitcecoc**。
- **fitcsvm** 支持低维和中维数据集。对于高维数据集，请改用 **fitclinear**。

详细信息

框约束

框约束是参数，它控制对违反边距的观测值施加的最大罚分，这有助于防止过拟合（正则化）。

如果增加框约束，则 SVM 分类器分配的支持向量会更少。然而，增加框约束会导致更长的训练时间。

Gram 矩阵

由 n 个向量 $\{x_1,..,x_n; x_j \in R^p\}$ 组成的 Gram 矩阵是一个 n×n 矩阵，其中元素 (j,k) 定义为 $G(x_j,x_k) = <\phi(x_j),\phi(x_k)>$，即使用核函数 ϕ 的变换后的预测变量的内积。

对于非线性 SVM，算法使用预测变量数据 X 的行形成 Gram 矩阵。对偶问题形式用生成的 Gram 矩阵的对应元素替换 X 中观测值的内积（称为“核方法”）。因此，非线性 SVM 在变换后的预测变量空间中运算以找到分离超平面。

Karush-Kuhn-Tucker (KKT) 互补条件

KKT 互补条件是最优非线性规划解所需的优化约束。

在 SVM 中，对于所有 j = 1、...、n，KKT 互补条件是

$$\begin{cases}\alpha_j\left[y_j f\left(x_j\right) - 1 + \xi_j\right] = 0\\ \xi_j\left(C - \alpha_j\right) = 0\end{cases}$$

，其中 $f\left(x_j\right) = \phi\left(x_j\right)'\beta + b$，ϕ 是核函数（请参阅 Gram 矩阵 （第 33-64 页）），ξ_j 是松弛变量。如果这些类完全可分离，则对于所有 j = 1,...,n，满足 $\xi_j = 0$。

一类学习

一类学习，或无监督 SVM，旨在从高维预测变量空间（而不是原始预测变量空间）的原点分离数据，是一种用于离群值检测的算法。

该算法类似于二类分类的 SVM （第 33-65 页） 的算法。目标是最小化关于 $\alpha_1, \ldots, \alpha_n$ 的对偶表达式

$$0.5\sum_{jk}\alpha_j\alpha_k G(x_j, x_k)$$

，且对于所有 j = 1、...、n，满足

$$\sum\alpha_j = n\nu$$

和 $0 \le \alpha_j \le 1$。$G(x_j,x_k)$ 的值在 Gram 矩阵 （第 33-64 页）的元素 (j,k) 中。

较小的 ν 值会导致较少支持向量，因此决策边界平滑、粗糙。较大的 ν 值会导致较多支持向量，因此决策边界弯曲、灵活。最佳 ν 值应足够大以体现数据的复杂性，同时足够小以避免过度训练。此外，$0 < \nu \le 1$。

有关详细信息，请参阅[5]。

支持向量

支持向量是与 $\alpha_1,...,\alpha_n$ 的严格正估计值对应的观测值。

对于给定训练集，最好选用产生较少支持向量的 SVM 分类器。

用于二类分类的支持向量机

SVM 二类分类算法搜索将数据分成两类的最优超平面。对于可分离类，最优超平面会最大化自身周围的边距（不包含任何观测值的空间），从而在正类和负类间创建边界。对于不可分离的类，目标是相同的，但是，如果有观测值位于其类边界的错误一侧，则算法会相应地对边距长度施加罚分。

线性 SVM 分数函数是

$$f(x) = x'\beta + b,$$

其中：

- x 是观测值（对应于 $\mathbf{X}$ 的一行）。
- 向量 β 包含定义超平面的正交向量的系数（对应于 $\mathbf{Mdl.Beta}$）。对于可分离的数据，最佳边距长度是 $2/\|\beta\|$.
- b 是偏差项（对应于 $\mathbf{Mdl.Bias}$）。

特定系数的 f(x) 的根定义超平面。对于特定的超平面，f(z) 是从 z 点到超平面的距离。

算法搜索最大边距长度，同时将观测值分为正类 (y = 1) 和负类 (y = –1)。

- 对于可分离的类，目标是最小化关于 β 和 b 的 $\|\beta\|$，并且对于所有 j = 1、...、n，满足 $y_j f(x_j) \geq 1$。这是针对可分离类的原问题形式。
- 对于不可分离的类，算法会在遇到跨越类边界的观测值时使用松弛变量 (ξ_j) 对目标函数进行罚分。对于未跨越类边界的观测值，$\xi_j = 0$，否则 $\xi_j \geq 0$。

  目标是最小化关于 β、b 和 ξ_j 的 $0.5\|\beta\|^2 + C\sum\xi_j$，对于所有 j = 1,..,n 和正标量框约束 （第 33-64 页） C，满足 $y_j f(x_j) \geq 1 - \xi_j$ 和 $\xi_j \geq 0$。这是针对不可分离类的原问题形式。

算法采用 Lagrange 乘数方法优化目标，引入 n 个系数 $\alpha_1,...,\alpha_n$（对应于 $\mathbf{Mdl.Alpha}$）。线性 SVM 的对偶问题形式如下：

- 对于可分离的类，最小化关于 $\alpha_1,...,\alpha_n$ 的

  $$0.5\sum_{j=1}^{n}\sum_{k=1}^{n}\alpha_j\alpha_k y_j y_k x_j' x_k - \sum_{j=1}^{n}\alpha_j$$

  ，对于所有 j = 1,...,n，满足 $\sum\alpha_j y_j = 0$，$\alpha_j \geq 0$，且满足 Karush-Kuhn-Tucker (KKT) 互补条件 （第 33-64 页）。
- 对于不可分离的类，目标与可分离的类相同，不同之处是所有 j = 1、...、n 需满足附加条件 $0 \leq \alpha_j \leq C$。

得到的分数函数是

$$\widehat{f}(x) = \sum_{j=1}^{n}\widehat{\alpha}_j y_j x' x_j + \widehat{b}\ .$$

$\widehat{b}$ 是偏差的估计值，$\widehat{\alpha}_j$ 是向量 $\widehat{\alpha}$ 的第 j 个估计值，j = 1,...,n。写为这种形式时，score 函数不再需要原问题形式中的 β 估计值。

SVM 算法使用 $\text{sign}\left(\widehat{f}(z)\right)$. 对新观测值 z 进行分类

在某些情况下，非线性边界对类进行分隔。非线性 SVM 在经过变换的预测变量空间中计算以找到最佳的分离超平面。

非线性 SVM 的对偶问题可表示为关于 α_1、...、α_n 的以下形式

$$0.5\sum_{j=1}^{n}\sum_{k=1}^{n}\alpha_j\alpha_k y_j y_k G(x_j,x_k) - \sum_{j=1}^{n}\alpha_j$$

对于所有 j = 1、...、n，满足 $\sum\alpha_j y_j = 0$，$0 \le \alpha_j \le C$，且满足 KKT 互补条件。$G(x_k,x_j)$ 是 Gram 矩阵 （第 33-64 页）的元素。得到的分数函数是

$$\widehat{f}(x) = \sum_{j=1}^{n}\widehat{\alpha}_j y_j G(x,x_j) + \widehat{b}\,.$$

有关详细信息，请参阅了解支持向量机 （第 18-2 页）、[1] 和 [3]。

提示

- 除非您的数据集很大，否则请始终尝试标准化预测变量（请参阅 **Standardize**）。标准化可使预测变量不受其测量尺度的影响。
- 使用 **KFold** 名称-值对组参数进行交叉验证是很好的做法。交叉验证结果决定 SVM 分类器的泛化能力。
- 对于一类学习：
  - 名称-值对组参数 **Alpha** 的默认设置可能导致较长的训练时间。要加快训练速度，可将 **Alpha** 设置为主要由 **0** 组成的向量。
  - 将名称-值对组参数 **Nu** 设置为更接近 **0** 的值可产生更少的支持向量，从而获得更平滑但粗糙的决策边界。
- 支持向量中的稀疏性是 SVM 分类器的一个理想属性。要减少支持向量的数量，可将 **BoxConstraint** 设置为较大的值。此操作会增加训练时间。
- 为了获得最佳训练时间，请将 **CacheSize** 设置为计算机允许的最大内存限制。
- 如果您期望支持向量比训练集中的观测值少得多，则您可以使用名称-值对组参数 **'ShrinkagePeriod'** 缩小活动集，从而显著加快收敛速度。指定 **'ShrinkagePeriod',1000** 是很好的做法。
- 远离决策边界的重复观测值不影响收敛。然而，决策边界附近出现的少数几个重复的观测值可能会大幅减慢收敛速度。在以下情况下，要加快收敛速度，请指定 **'RemoveDuplicates',true**：
  - 您的数据集包含许多重复的观测值。
  - 您怀疑有几个重复的观测值接近决策边界。

  要在训练期间保留原始数据集，**fitcsvm** 必须临时分别存储两个数据集：原始数据集和没有重复观测值的数据集。因此，如果您为包含少量重复观测值的数据集指定 **true**，则 **fitcsvm** 消耗的内存将接近原始数据的两倍。
- 在训练模型后，您可以生成预测新数据标签的 C/C++ 代码。生成 C/C++ 代码需要 MATLAB Coder™。有关详细信息，请参阅“Introduction to Code Generation”。

算法

- 有关 SVM 二类分类算法的数学公式，请参阅“用于二类分类的支持向量机” （第 33-65 页）和“了解支持向量机” （第 18-2 页）。
- **NaN**、**<undefined>**、空字符向量 (')、空字符串 ("") 和 **<missing>** 值表示缺失值。**fitcsvm** 会删除对应于缺失响应的整行数据。在计算总权重时（请参阅下几项内容），如果某权重所对应的观测值至少具有一个缺失变量，则 **fitcsvm** 忽略该权重。此操作会导致平衡类问题中出现不平衡的先验概率。因此，观测值框约束可能不等于 **BoxConstraint**。
- **fitcsvm** 会删除权重为零或先验概率为零的观测值。
- 对于二类学习，如果您指定代价矩阵 C（请参阅 **Cost**），则软件会通过合并 C 中所述的罚分将类先验概率 p（请参阅 **Prior**）更新为 p_c。

  具体而言，**fitcsvm** 完成以下步骤：

  1 计算 $p_c^* = p'C$.

  2 归一化 p_c^*，使更新后的先验概率总和为 1。

  $$p_c = \frac{1}{\sum_{j=1}^{K} p_{c,j}^*} p_c^* .$$

  K 是类的数量。

  3 将代价矩阵重置为默认值

  $$C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

  4 从训练数据中删除对应于先验概率为零的类的观测值。
- 对于二类学习，**fitcsvm** 将所有观测值权重归一化（请参阅 **Weights**），其总和为 1。然后，该函数将归一化的权重重新归一化，使其总和等于观测值所属类的更新后的先验概率。也就是说，k 类中观测值 j 的总权重是

  $$w_j^* = \frac{w_j}{\sum_{\forall j \in \text{Class } k} w_j} p_{c,k} .$$

  w_j 是观测值 j 的归一化权重；$p_{c、k}$ 是 k 类的更新后的先验概率（请参阅上一项内容）。
- 对于二类学习，**fitcsvm** 为训练数据中的每个观测值指定一个框约束。观测值 j 的框约束公式是

  $$C_j = nC_0 w_j^* .$$

  n 是训练样本大小，C_0 是初始框约束（请参阅 **'BoxConstraint'** 名称-值对组参数），而 w_j^* 是观测值 j 的总权重（请参阅上一项内容）。
- 如果您设置 **'Standardize',true** 和 **'Cost'**、**'Prior'** 或 **'Weights'** 名称-值对组参数，则 **fitcsvm** 使用预测变量的对应加权均值和加权标准差来对预测变量进行标准化。也就是说，**fitcsvm** 使用

  $$x_j^* = \frac{x_j - \mu_j^*}{\sigma_j^*} .$$

  $\mu_j^* = \frac{1}{\sum_k w_k^*} \sum_k w_k^* x_{jk}$. 对预测变量 j ($x_j$) 进行标准化

x_{jk} 是预测变量 j（列）的观测值 k（行）。

$$\left(\sigma_j^*\right)^2 = \frac{v_1}{v_1^2 - v_2}\sum_k w_k^*\left(x_{jk} - \mu_j^*\right)^2.$$

$$v_1 = \sum_j w_j^*.$$

$$v_2 = \sum_j \left(w_j^*\right)^2.$$

- 假设 **p** 是您预期在训练数据中的离群值比例，并且您设置了 **'OutlierFraction',p**。
  - 对于一类学习，软件会训练偏差项，使得训练数据中 100**p**% 的观测值具有负分数。
  - 对于二类学习，软件会实施稳健学习。换句话说，当优化算法收敛时，软件会尝试删除 100**p**% 的观测值。删除的观测值对应于幅值较大的梯度。
- 如果预测变量数据包含分类变量，则软件通常会对这些变量进行完全虚拟变量编码。软件为每个分类变量的每个水平创建一个虚拟变量。
  - **PredictorNames** 属性为每个原始预测变量名称存储一个元素。例如，假设有三个预测变量，其中一个是具有三个水平的分类变量。那么 **PredictorNames** 是 1×3 个字符向量元胞数组，其中包含预测变量的原始名称。
  - **ExpandedPredictorNames** 属性为每个预测变量（包括虚拟变量）存储一个元素。例如，假设有三个预测变量，其中一个是具有三个水平的分类变量。那么 **ExpandedPredictorNames** 就是 1×5 字符向量数组，其中包含预测变量和新虚拟变量的名称。
  - 同样，**Beta** 属性为每个预测变量（包括虚拟变量）存储一个 beta 系数。
  - **SupportVectors** 属性存储支持向量的预测变量值，包括虚拟变量。例如，假设有 m 个支持向量和三个预测变量，其中一个预测变量是具有三个水平的分类变量。那么 **SupportVectors** 就是 n×5 矩阵。
  - **X** 属性将训练数据存储为原始输入，不包括虚拟变量。当输入为表时，**X** 仅包含用作预测变量的列。
- 对于表中指定的预测变量，如果任何变量包含经过排序的（有序）类别，软件会对这些变量使用有序编码。
  - 对于具有 k 个有序水平的变量，软件会创建 k – 1 个虚拟变量。在第 j 个虚拟变量中，前 j 个水平的对应值为 –1；从 j + 1 至 k 的水平的对应值为 +1。
  - 存储在 **ExpandedPredictorNames** 属性中的虚拟变量的名称使用值 +1 指示第一个水平。软件会为虚拟变量另外存储 k – 1 个预测变量名称，包括水平 2、3、...、k 的名称。
- 所有求解器都实现 L1 软边距最小化算法。
- 对于一类学习，软件估计 Lagrange 乘数 α_1、...、α_n，满足

$$\sum_{j=1}^{n} \alpha_j = n\nu.$$

参考

[1] Christianini, N., and J. C. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.

[2] Fan, R.-E., P.-H. Chen, and C.-J. Lin. "Working set selection using second order information for training support vector machines." *Journal of Machine Learning Research*, Vol. 6, 2005, pp. 1889–1918.

[3] Hastie, T., R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*, Second Edition. NY: Springer, 2008.

[4] Kecman V., T. -M. Huang, and M. Vogt. "Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance." *Support Vector Machines: Theory and Applications*. Edited by Lipo Wang, 255–274. Berlin: Springer-Verlag, 2005.

[5] Scholkopf, B., J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. "Estimating the Support of a High-Dimensional Distribution." *Neural Comput.*, Vol. 13, Number 7, 2001, pp. 1443–1471.

[6] Scholkopf, B., and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond, Adaptive Computation and Machine Learning*. Cambridge, MA: The MIT Press, 2002.

扩展功能

自动并行支持
通过使用 Parallel Computing Toolbox™ 自动运行并行计算来加快代码执行。

要执行并行超参数优化，请在调用此函数时使用 **'HyperparameterOptimizationOptions', struct('UseParallel',true)** 名称-值参数。

有关并行超参数优化的详细信息，请参阅 "Parallel Bayesian Optimization"。

有关并行计算的一般信息，请参阅"Run MATLAB Functions with Automatic Parallel Support" (Parallel Computing Toolbox)。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

使用说明和限制：

- **'KernelFunction'** 选项不能指定自定义核。
- **'Solver'** 选项只能设置为 **'SMO'**。
- 不支持 **'OutlierFraction'** 选项。
- **'Alpha'** 选项必须指定可行的起点。
- 不支持 **'OptimizeHyperparameters'** 选项。
- 不支持一类分类。标签必须包含两个不同类。

有关详细信息，请参阅"Run MATLAB Functions on a GPU" (Parallel Computing Toolbox)。

另请参阅
ClassificationSVM | **CompactClassificationSVM** | **ClassificationPartitionedModel** | **predict** | **fitSVMPosterior** | **rng** | **quadprog** | **fitcecoc** | **fitclinear**

主题
"用高斯核训练 SVM 分类器" (第 18-7 页)

“使用自定义核训练 SVM 分类器” （第 18-10 页）
“Optimize Cross-Validated Classifier Using bayesopt”
“使用贝叶斯优化来优化 SVM 分类器拟合” （第 10-2 页）
“了解支持向量机” （第 18-2 页）

在 R2014a 中推出

fitlm

拟合线性回归模型

语法

```
mdl = fitlm(tbl)
mdl = fitlm(X,y)
mdl = fitlm( ___ ,modelspec)
mdl = fitlm( ___ ,Name,Value)
```

说明

mdl = fitlm(tbl) 返回基于表或数据集数组 **tbl** 中变量拟合的线性回归模型。默认情况下，**fitlm** 将最后一个变量作为响应变量。

mdl = fitlm(X,y) 返回基于数据矩阵 **X** 拟合的响应 **y** 的线性回归模型。

mdl = fitlm(___ ,modelspec) 使用上述语法中的任何输入参数组合来定义模型设定。

mdl = fitlm(___ ,Name,Value) 使用一个或多个名称-值对组参数指定附加选项。例如，您可以指定哪些变量是分类变量、执行稳健回归或使用观测值权重。

示例

用矩阵数据拟合线性回归

使用矩阵输入数据集拟合线性回归模型。

加载 **carsmall** 数据集，它是一个矩阵输入数据集。

```
load carsmall
X = [Weight,Horsepower,Acceleration];
```

使用 **fitlm** 拟合线性回归模型。

```
mdl = fitlm(X,MPG)
```

```
mdl =
Linear regression model:
  y ~ 1 + x1 + x2 + x3

Estimated Coefficients:
                Estimate       SE        tStat       pValue
                ________    _________    _______    _________

  (Intercept)     47.977      3.8785       12.37    4.8957e-21
  x1           -0.0065416    0.0011274    -5.8023    9.8742e-08
  x2            -0.042943     0.024313    -1.7663       0.08078
  x3            -0.011583      0.19333    -0.059913      0.95236
```

```
Number of observations: 93, Error degrees of freedom: 89
Root Mean Squared Error: 4.09
R-squared: 0.752,  Adjusted R-Squared: 0.744
F-statistic vs. constant model: 90, p-value = 7.38e-27
```

模型显示包括模型公式、估计的系数和模型汇总统计量。

显示信息的模型公式 **y ~ 1 + x1 + x2 + x3** 对应于 $y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$。

模型显示还显示估计的系数信息，该信息存储在 **Coefficients** 属性中。显示 **Coefficients** 属性。

```
mdl.Coefficients
```

```
ans=4×4 table
                Estimate      SE        tStat       pValue
              ____________ _________ _________ ____________

    (Intercept)      47.977    3.8785     12.37    4.8957e-21
    x1           -0.0065416 0.0011274   -5.8023    9.8742e-08
    x2            -0.042943  0.024313   -1.7663       0.08078
    x3            -0.011583   0.19333  -0.059913       0.95236
```

Coefficient 属性包括以下几列：

- **Estimate** - 模型中每个对应项的系数估计值。例如，常数项 (**intercept**) 的估计值为 47.977。
- **SE** - 系数的标准误差。
- **tStat** - 每个系数的 t 统计量，用于基于对应系数不为零的备择假设来检验对应系数为零的原假设（给出了模型中的其他预测变量）。请注意，**tStat = Estimate/SE**。例如，截距的 t 统计量为 47.977/3.8785 = 12.37。
- **pValue** - 假设检验的 t 统计量的 p 值，该假设检验验证对应系数是否等于零。例如，**x2** 的 t 统计量的 p 值大于 0.05，因此在给定模型中其他项的情况下，该项在 5% 显著性水平上不显著。

模型的汇总统计量如下：

- **Number of observations** - 没有任何 **NaN** 值的行数。例如，**Number of observations** 为 93，因为 **X** 和 **MPG** 中的行数为 100，但 **MPG** 数据向量有六个 **NaN** 值，且 **Horsepower** 数据向量中有另一不同观测值（即不同的行）也为 **NaN** 值。
- **Error degrees of freedom** - n - p，其中 n 是观测值数目，p 是模型中系数的数目，包括截距。例如，该模型有四个预测变量，因此 **Error degrees of freedom** 为 93–4 = 89。
- **Root mean squared error** - 均方误差的平方根，用于估计误差分布的标准差。
- **R-squared** 和 **Adjusted R-squared** - 分别为决定系数和调整决定系数。例如，**R-squared** 值表明，该模型解释了响应变量 **MPG** 中大约 75% 的变异。
- **F-statistic vs. constant model** - 对回归模型进行 F 检验的检验统计量，用于检验该模型的拟合是否显著优于仅包含常数项的退化模型。
- **p-value** - 对模型的 F 检验的 p 值。例如，p 值为 7.3816e-27，说明模型是显著的。

您可以使用 **anova** 函数在模型属性（**NumObservations**、**DFE**、**RMSE** 和 **Rsquared**）中找到这些统计量。

```
anova(mdl,'summary')
```

```
ans=3×5 table
            SumSq    DF    MeanSq      F        pValue
```

```
                ______    __    ______    ______    _________

    Total        6004.8    92    65.269
    Model          4516     3    1505.3    89.987    7.3816e-27
    Residual     1488.8     89    16.728
```

使用表中的数据拟合线性回归

加载样本数据。

```
load carsmall
```

将变量存储在表中。

```
tbl = table(Weight,Acceleration,MPG,'VariableNames',{'Weight','Acceleration','MPG'});
```

显示表的前五行。

```
tbl(1:5,:)
```

```
ans=5×3 table
    Weight    Acceleration    MPG
    ______    ____________    ___

     3504          12         18
     3693        11.5          15
     3436          11         18
     3433          12         16
     3449        10.5          17
```

拟合每加仑英里数 (MPG) 的线性回归模型。使用 Wilkinson 表示法指定模型公式。

```
lm = fitlm(tbl,'MPG~Weight+Acceleration')
```

```
lm =
Linear regression model:
    MPG ~ 1 + Weight + Acceleration

Estimated Coefficients:
                    Estimate        SE         tStat       pValue
                    ________    __________    _______    _________

    (Intercept)        45.155       3.4659     13.028    1.6266e-22
    Weight        -0.0082475    0.00059836    -13.783    5.3165e-24
    Acceleration     0.19694       0.14743     1.3359       0.18493


Number of observations: 94, Error degrees of freedom: 91
Root Mean Squared Error: 4.12
R-squared: 0.743,  Adjusted R-Squared: 0.738
F-statistic vs. constant model: 132, p-value = 1.38e-27
```

此示例中的 **'MPG~Weight+Acceleration'** 模型等效于将模型设定设置为 **'linear'**。例如，

```
lm2 = fitlm(tbl,'linear');
```

如果您对模型设定使用字符向量，但没有指定响应变量，则 **fitlm** 接受 **tbl** 中的最后一个变量作为响应变量，其他变量作为预测变量。

使用指定的模型公式拟合线性回归

使用由 Wilkinson 表示法指定的模型公式拟合线性回归模型。

加载样本数据。

```
load carsmall
```

将变量存储在表中。

```
tbl = table(Weight,Acceleration,Model_Year,MPG,'VariableNames',{'Weight','Acceleration','Model_Year','MPG'});
```

以权重和加速度为预测变量，拟合每加仑英里数 (MPG) 的线性回归模型。

```
lm = fitlm(tbl,'MPG~Weight+Acceleration')
```

```
lm =
Linear regression model:
    MPG ~ 1 + Weight + Acceleration

Estimated Coefficients:
                    Estimate        SE         tStat       pValue
                    ________    __________    _______    __________

    (Intercept)        45.155       3.4659     13.028    1.6266e-22
    Weight         -0.0082475    0.00059836    -13.783    5.3165e-24
    Acceleration      0.19694       0.14743     1.3359       0.18493


Number of observations: 94, Error degrees of freedom: 91
Root Mean Squared Error: 4.12
R-squared: 0.743,  Adjusted R-Squared: 0.738
F-statistic vs. constant model: 132, p-value = 1.38e-27
```

p 值 0.18493 表示 **Acceleration** 对 **MPG** 没有重大影响。

从模型中删除 **Acceleration**，并尝试通过添加预测变量 **Model_Year** 来改进模型。首先将 **Model_Year** 定义为分类变量。

```
tbl.Model_Year = categorical(tbl.Model_Year);
lm = fitlm(tbl,'MPG~Weight+Model_Year')
```

```
lm =
Linear regression model:
    MPG ~ 1 + Weight + Model_Year

Estimated Coefficients:
                    Estimate        SE         tStat       pValue
                    ________    __________    _______    __________

    (Intercept)        40.11       1.5418     26.016    1.2024e-43
    Weight         -0.0066475    0.00042802    -15.531    3.3639e-27
    Model_Year_76      1.9291       0.74761     2.5804      0.011488
```

```
    Model_Year_82       7.9093      0.84975     9.3078    7.8681e-15

Number of observations: 94, Error degrees of freedom: 90
Root Mean Squared Error: 2.92
R-squared: 0.873,  Adjusted R-Squared: 0.868
F-statistic vs. constant model: 206, p-value = 3.83e-40
```

使用 Wilkinson 表示法指定 **modelspec** 使您能够更新模型而无需更改设计矩阵。**fitlm** 仅使用在公式中指定的变量。它还为分类变量 **Model_Year** 创建两个必要的虚拟指示变量。

用项矩阵拟合线性回归

使用项矩阵拟合线性回归模型。

采用表输入时的项矩阵

如果模型变量在表中，则项矩阵中的 **0** 列表示响应变量的位置。

加载 **hospital** 数据集。

```
load hospital
```

将变量存储在表中。

```
t = table(hospital.Sex,hospital.BloodPressure(:,1),hospital.Age,hospital.Smoker, ...
    'VariableNames',{'Sex','BloodPressure','Age','Smoker'});
```

使用项矩阵表示线性模型 **'BloodPressure ~ 1 + Sex + Age + Smoker'**。响应变量位于表的第二列中，因此项矩阵的第二列必须是由 **0** 组成的列，以表示响应变量。

```
T = [0 0 0 0;1 0 0 0;0 0 1 0;0 0 0 1]
```

```
T = 4×4

     0     0     0     0
     1     0     0     0
     0     0     1     0
     0     0     0     1
```

拟合线性模型。

```
mdl1 = fitlm(t,T)
```

```
mdl1 =
Linear regression model:
    BloodPressure ~ 1 + Sex + Age + Smoker

Estimated Coefficients:
                   Estimate      SE       tStat       pValue
                   ________    ________    ______    _________

    (Intercept)      116.14      2.6107     44.485    7.1287e-66
    Sex_Male       0.050106     0.98364    0.050939      0.95948
    Age            0.085276    0.066945      1.2738       0.2058
```

```
    Smoker_1          9.87      1.0346      9.5395    1.4516e-15


Number of observations: 100, Error degrees of freedom: 96
Root Mean Squared Error: 4.78
R-squared: 0.507,  Adjusted R-Squared: 0.492
F-statistic vs. constant model: 33, p-value = 9.91e-15
```

采用矩阵输入时的项矩阵

如果预测变量和响应变量包含在矩阵和列向量中，则必须在项矩阵每一行的末尾包含 **0**，以表示响应变量。

加载 **carsmall** 数据集，并定义预测变量矩阵。

```
load carsmall
X = [Acceleration,Weight];
```

使用项矩阵指定模型 **'MPG ~ Acceleration + Weight + Acceleration:Weight + Weight^2'**。此模型包括变量 **Acceleration** 和 **Weight** 的主效应和双向交互效应项，以及变量 **Weight** 的二阶项。

```
T = [0 0 0;1 0 0;0 1 0;1 1 0;0 2 0]
```

```
T = 5×3

     0     0     0
     1     0     0
     0     1     0
     1     1     0
     0     2     0
```

拟合线性模型。

```
mdl2 = fitlm(X,MPG,T)
```

```
mdl2 =
Linear regression model:
    y ~ 1 + x1*x2 + x2^2

Estimated Coefficients:
                    Estimate        SE        tStat      pValue
                    ________    _________    _______    _________

    (Intercept)       48.906      12.589     3.8847    0.00019665
    x1                0.54418     0.57125    0.95261     0.34337
    x2               -0.012781    0.0060312  -2.1192     0.036857
    x1:x2            -0.00010892  0.00017925 -0.6076      0.545
    x2^2              9.7518e-07  7.5389e-07  1.2935     0.19917


Number of observations: 94, Error degrees of freedom: 89
Root Mean Squared Error: 4.1
R-squared: 0.751,  Adjusted R-Squared: 0.739
F-statistic vs. constant model: 67, p-value = 4.99e-26
```

只有截距和 **x2** 项（对应于 **Weight** 变量）在 5% 显著性水平上是显著的。

具有分类预测变量的线性回归

拟合包含分类预测变量的线性回归模型。对分类预测变量的类别重新排序，以控制模型中的参考水平。然后，使用 **anova** 检验分类变量的显著性。

具有分类预测变量的模型

加载 **carsmall** 数据集，并创建 **MPG** 的线性回归模型作为 **Model_Year** 的函数。要将数值向量 **Model_Year** 视为分类变量，请使用 **'CategoricalVars'** 名称-值对组参数标识预测变量。

```
load carsmall
mdl = fitlm(Model_Year,MPG,'CategoricalVars',1,'VarNames',{'Model_Year','MPG'})
```

```
mdl =
Linear regression model:
   MPG ~ 1 + Model_Year

Estimated Coefficients:
                   Estimate     SE     tStat     pValue
                   ________    ______    ______    __________

   (Intercept)       17.69     1.0328    17.127    3.2371e-30
   Model_Year_76     3.8839    1.4059    2.7625    0.0069402
   Model_Year_82      14.02    1.4369    9.7571    8.2164e-16


Number of observations: 94, Error degrees of freedom: 91
Root Mean Squared Error: 5.56
R-squared: 0.531,  Adjusted R-Squared: 0.521
F-statistic vs. constant model: 51.6, p-value = 1.07e-15
```

显示信息的模型公式 **MPG ~ 1 + Model_Year** 对应于

$$\mathrm{MPG} = \beta_0 + \beta_1 I_{\mathrm{Year}=76} + \beta_2 I_{\mathrm{Year}=82} + \epsilon,$$

其中，$I_{\mathrm{Year}=76}$ 和 $I_{\mathrm{Year}=82}$ 是指示变量，如果 **Model_Year** 的值分别为 76 和 82，则其值为 1。**Model_Year** 变量包含三个不同值，您可以使用 **unique** 函数来进行检查。

```
unique(Model_Year)
```

```
ans = 3×1

   70
   76
   82
```

fitlm 选择 **Model_Year** 中的最小值作为参考水平 (**'70'**)，并创建两个指示变量 $I_{\mathrm{Year}=76}$ 和 $I_{\mathrm{Year}=82}$。该模型仅包括两个指示变量，因为如果该模型包括三个指示变量（每个水平一个）和一个截距项，则设计矩阵变为秩亏矩阵。

具有全指示变量的模型

您可以将 **mdl** 的模型公式解释为一个具有三个指示变量而没有截距项的模型：

$$y = \beta_0 I_{x_1=70} + (\beta_0 + \beta_1) I_{x_1=76} + (\beta_0 + \beta_2) I_{x_2=82} + \epsilon。$$

您也可以通过手动创建指示变量并指定模型公式，创建包含三个指示变量而没有截距项的模型。

```
temp_Year = dummyvar(categorical(Model_Year));
Model_Year_70 = temp_Year(:,1);
Model_Year_76 = temp_Year(:,2);
Model_Year_82 = temp_Year(:,3);
tbl = table(Model_Year_70,Model_Year_76,Model_Year_82,MPG);
mdl = fitlm(tbl,'MPG ~ Model_Year_70 + Model_Year_76 + Model_Year_82 - 1')
```

```
mdl =
Linear regression model:
   MPG ~ Model_Year_70 + Model_Year_76 + Model_Year_82

Estimated Coefficients:
                Estimate     SE      tStat     pValue
                ________    _______    _____    __________

   Model_Year_70     17.69     1.0328    17.127    3.2371e-30
   Model_Year_76    21.574    0.95387    22.617    4.0156e-39
   Model_Year_82     31.71    0.99896    31.743    5.2234e-51


Number of observations: 94, Error degrees of freedom: 91
Root Mean Squared Error: 5.56
```

选择模型中的参考水平

您可以通过修改分类变量中类别的顺序来选择参考水平。首先，创建一个分类变量 **Year**。

```
Year = categorical(Model_Year);
```

使用 **categories** 函数检查类别的顺序。

```
categories(Year)
```

```
ans = 3x1 cell
   {'70'}
   {'76'}
   {'82'}
```

如果您使用 **Year** 作为预测变量，则 **fitlm** 选择第一个类别 **'70'** 作为参考水平。使用 **reordercats** 函数对 **Year** 重新排序。

```
Year_reordered = reordercats(Year,{'76','70','82'});
categories(Year_reordered)
```

```
ans = 3x1 cell
   {'76'}
   {'70'}
   {'82'}
```

Year_reordered 的第一个类别是 **'76'**。创建 **MPG** 为 **Year_reordered** 的函数的线性回归模型。

```
mdl2 = fitlm(Year_reordered,MPG,'VarNames',{'Model_Year','MPG'})
```

```
mdl2 =
Linear regression model:
```

```
    MPG ~ 1 + Model_Year

Estimated Coefficients:
                   Estimate      SE       tStat      pValue
                   ________    _______    _______    _________

    (Intercept)       21.574    0.95387     22.617    4.0156e-39
    Model_Year_70    -3.8839     1.4059    -2.7625     0.0069402
    Model_Year_82     10.136     1.3812     7.3385    8.7634e-11


Number of observations: 94, Error degrees of freedom: 91
Root Mean Squared Error: 5.56
R-squared: 0.531,  Adjusted R-Squared: 0.521
F-statistic vs. constant model: 51.6, p-value = 1.07e-15
```

mdl2 使用 **'76'** 作为参考水平，并包括两个指示变量 $I_{\text{Year = 70}}$ 和 $I_{\text{Year = 82}}$。

计算分类预测变量

mdl2 的模型显示包括每个项的 p 值，以检验对应的系数是否等于零。每个 p 值检查每个指示变量。要将分类变量 **Model_Year** 作为一组指示变量进行检查，请使用 **anova**。使用 **'components'**（默认值）选项返回成分 ANOVA 表，该表包含模型中除常数项之外的每个变量的 ANOVA 统计量。

```
anova(mdl2,'components')
```

```
ans=2×5 table
                SumSq    DF    MeanSq     F       pValue
                _____    __    ______    _____    _________

    Model_Year    3190.1    2    1595.1    51.56    1.0694e-15
    Error         2815.2   91    30.936
```

成分 ANOVA 表包括 **Model_Year** 变量的 p 值，该值小于指示变量的 p 值。

为线性模型指定响应变量和预测变量

对样本数据进行线性回归模型拟合。指定响应变量和预测变量，并且在模型中仅包括成对的交互效应项。

加载样本数据。

```
load hospital
```

对数据进行具有交互效应项的线性模型拟合。指定体重作为响应变量，性别、年龄和吸烟状况作为预测变量。此外，指定性别和吸烟状况是分类变量。

```
mdl = fitlm(hospital,'interactions','ResponseVar','Weight',...
    'PredictorVars',{'Sex','Age','Smoker'},...
    'CategoricalVar',{'Sex','Smoker'})
```

```
mdl =
Linear regression model:
    Weight ~ 1 + Sex*Age + Sex*Smoker + Age*Smoker

Estimated Coefficients:
```

```
                   Estimate     SE      tStat      pValue
                   ________    _____    _______    _________

    (Intercept)         118.7    7.0718     16.785    6.821e-30
    Sex_Male            68.336   9.7153     7.0339    3.3386e-10
    Age              0.31068    0.18531     1.6765     0.096991
    Smoker_1            3.0425   10.446     0.29127      0.77149
    Sex_Male:Age       -0.49094  0.24764    -1.9825     0.050377
    Sex_Male:Smoker_1   0.9509   3.8031     0.25003      0.80312
    Age:Smoker_1       -0.07288  0.26275    -0.27737     0.78211


Number of observations: 100, Error degrees of freedom: 93
Root Mean Squared Error: 8.75
R-squared: 0.898,  Adjusted R-Squared: 0.892
F-statistic vs. constant model: 137, p-value = 6.91e-44
```

根据年龄、吸烟状况或这些因子与患者性别在 5% 显著性水平上的交互效应，患者的体重似乎没有显著差异。

拟合稳健线性回归模型

加载 **hald** 数据集，该数据集测量水泥成分对其硬化热的影响。

```
load hald
```

该数据集包括变量 **ingredients** 和 **heat**。矩阵 **ingredients** 包含水泥中四种化学成分的百分比组成。向量 **heat** 包含每个水泥样本在 180 天后的热硬化值。

对数据进行稳健线性回归模型拟合。

```
mdl = fitlm(ingredients,heat,'RobustOpts','on')
```

```
mdl =
Linear regression model (robust fit):
    y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:
                   Estimate     SE      tStat      pValue
                   ________    _____    _______    _________

    (Intercept)        60.09    75.818    0.79256     0.4509
    x1                1.5753   0.80585     1.9548   0.086346
    x2                0.5322   0.78315    0.67957    0.51596
    x3               0.13346    0.8166    0.16343    0.87424
    x4              -0.12052    0.7672   -0.15709    0.87906


Number of observations: 13, Error degrees of freedom: 8
Root Mean Squared Error: 2.65
R-squared: 0.979,  Adjusted R-Squared: 0.969
F-statistic vs. constant model: 94.6, p-value = 9.03e-07
```

有关详细信息，请参阅主题“Reduce Outlier Effects Using Robust Regression”，该主题将稳健拟合的结果与标准最小二乘拟合进行比较。

使用交叉验证计算均值绝对误差

使用 10 折交叉验证计算回归模型的均值绝对误差。

加载 **carsmall** 数据集。将 **Acceleration** 和 **Displacement** 变量指定为预测变量，将 **Weight** 变量指定为响应变量。

```
load carsmall
X1 = Acceleration;
X2 = Displacement;
y = Weight;
```

创建自定义函数 **regf**（如此示例末尾所示）。此函数将回归模型与训练数据进行拟合，然后基于测试集计算预测的汽车重量。该函数将预测的汽车重量值与实际值进行比较，然后计算均值绝对误差 (MAE) 和根据测试集汽车重量范围调整的 MAE。

注意：如果使用此示例的实时脚本文件，则文件末尾已包含 **regf** 函数。否则，您需要在 .m 文件的末尾创建此函数，或将其作为文件添加到 MATLAB® 路径中。

默认情况下，**crossval** 执行 10 折交叉验证。对于 **X1**、**X2** 和 **y** 中数据的 10 个训练和测试集分区，使用 **regf** 函数计算 MAE 和调整后的 MAE 值。求 MAE 均值和调整后的 MAE 均值。

```
rng('default') % For reproducibility
values = crossval(@regf,X1,X2,y)
```

```
values = 10×2

  319.2261    0.1132
  342.3722    0.1240
  214.3735    0.0902
  174.7247    0.1128
  189.4835    0.0832
  249.4359    0.1003
  194.4210    0.0845
  348.7437    0.1700
  283.1761    0.1187
  210.7444    0.1325
```

```
mean(values)
```

```
ans = 1×2

  252.6701    0.1129
```

以下代码创建函数 **regf**。

```
function errors = regf(X1train,X2train,ytrain,X1test,X2test,ytest)
tbltrain = table(X1train,X2train,ytrain, ...
   'VariableNames',{'Acceleration','Displacement','Weight'});
tbltest = table(X1test,X2test,ytest, ...
   'VariableNames',{'Acceleration','Displacement','Weight'});
mdl = fitlm(tbltrain,'Weight ~ Acceleration + Displacement');
yfit = predict(mdl,tbltest);
```

```
MAE = mean(abs(yfit-tbltest.Weight));
adjMAE = MAE/range(tbltest.Weight);
errors = [MAE adjMAE];
end
```

输入参数

tbl - 输入数据
表 | 数据集数组

输入数据，包括预测变量和响应变量，指定为表或数据集数组。预测变量可以是数值、逻辑值、分类、字符或字符串。响应变量必须为数值或逻辑值。

- 默认情况下，**fitlm** 将最后一个变量作为响应变量，其他变量作为预测变量。
- 要将另外的列设置为响应变量，请使用 **ResponseVar** 名称-值对组参数。
- 要使用列的子集作为预测变量，请使用 **PredictorVars** 名称-值对组参数。
- 要定义模型设定，请使用公式或项矩阵设置 **modelspec** 参数。公式或项矩阵指定哪些列用作预测变量或响应变量。

表中的变量名称不必是有效的 MATLAB 标识符。但是，如果名称无效，则在拟合或调整模型时无法使用公式；例如：

- 您无法使用公式指定 **modelspec**。
- 当您分别使用 **addTerms** 函数或 **removeTerms** 函数时，无法使用公式指定要添加或删除的项。
- 当您使用带名称-值对组参数 **'Lower'** 和 **'Upper'** 的 **step** 或 **stepwiselm** 函数时，则无法使用公式来指定模型的下界和上界。

您可以使用 **isvarname** 函数来验证 **tbl** 中的变量名称。如果变量名称无效，可以使用 **matlab.lang.makeValidName** 函数进行转换。

X - 预测变量
矩阵

预测变量，指定为 n×p 矩阵，其中 n 是观测值数目，p 是预测变量的数目。**X** 的每列表示一个变量，每行表示一个观测值。

默认情况下，模型中有一个常数项，除非您显式删除它，否则不要在 **X** 中包含由 1 组成的列。

数据类型： **single** | **double**

y - 响应变量
向量

响应变量，指定为 n×1 向量，其中 n 是观测值数目。**y** 中的每个条目是对 **X** 的对应行的响应。

数据类型： **single** | **double** | **logical**

modelspec - 模型设定
'linear' （默认） | 命名模型的字符向量或字符串标量 | t×(p + 1) 项矩阵 | **'y ~ terms'** 形式的字符向量或字符串标量公式

模型设定，指定为下列值之一。

- 命名模型的字符向量或字符串标量。

值	模型类型
'constant'	模型只包含一个常数（截距）项。
'linear'	模型包含每个预测变量的截距和线性项。
'interactions'	模型包含每个预测变量的截距、线性项以及不同预测变量对的所有乘积（无平方项）。
'purequadratic'	模型包含每个预测变量的截距项、线性项和平方项。
'quadratic'	模型包含截每个预测变量的截距项、线性项和平方项，以及不同预测变量对组的所有乘积。
'polyijk'	模型是一个多项式，其中具有第一个预测变量的 1 到 **i** 次的所有项，第二个预测变量的 1 到 **j** 次的所有项，依此类推。请使用数字 0 到 9 指定每个预测变量的最大次数。模型包含交互效应项，但是，每个交互效应项的次数不超过指定次数的最大值。例如，**'poly13'** 具有截距和 x_1、x_2、x_2^2、x_2^3、x_1*x_2 和 $x_1*x_2^2$ 项，其中 x_1 和 x_2 分别是第一个和第二个预测变量。

- t×(p + 1) 矩阵或“项矩阵” （第 33-86 页），指定模型中的项，其中 t 是项数，p 是预测变量数，而 + 1 表示响应变量。当预测变量的数目很大并且您要以编程方式生成项时，项矩阵是很方便的。
- 如下形式的字符向量或字符串标量 “公式” （第 33-87 页）：
  'y ~ terms',
  其中 **terms** 位于“Wilkinson 表示法” （第 33-87 页）中。公式中的变量名称必须为 **tbl** 中的变量名称或由 **Varnames** 指定的变量名称。此外，变量名称必须为有效的 MATLAB 标识符。

  软件使用 **tbl** 或 **X** 中各项的顺序来确定拟合模型中各项的顺序。因此，模型中各项的顺序可能不同于指定公式中各项的顺序。

示例： **'quadratic'**

示例： **'y ~ x1 + x2^2 + x1:x2'**

数据类型： **single** | **double** | **char** | **string**

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例： **'Intercept',false,'PredictorVars',[1,3],'ResponseVar',5,'RobustOpts','logistic'** 指定没有常数项的稳健回归模型，其中算法使用带默认调整常量的逻辑加权函数，第一个和第三个变量是预测变量，第五个变量是响应变量。

CategoricalVars - 分类变量列表
字符串数组 | 字符向量元胞数组 | 逻辑或数值索引向量

分类变量列表，指定为以逗号分隔的对组，其中包含 **'CategoricalVars'** 和字符串数组或字符向量元胞数组（包含表或数据集数组 **tbl** 中分类变量的名称）或者逻辑或数值索引向量（指示哪些列是分类变量）。

- 如果数据在表或数据集数组 **tbl** 中，则默认情况下，**fitlm** 将所有分类值、逻辑值、字符数组、字符串数组和字符向量元胞数组视为分类变量。
- 如果数据在矩阵 **X** 中，则 **'CategoricalVars'** 的默认值为空矩阵 **[]**。也就是说，除非您将变量指定为分类变量，否则默认视为非分类变量。

例如，您可以使用以下任一选项将六个变量中的第二个和第三个指定为分类变量：

示例：'CategoricalVars',[2,3]

示例：'CategoricalVars',logical([0 1 1 0 0 0])

数据类型：single | double | logical | string | cell

Exclude - 要排除的观测值
逻辑或数值索引向量

要从拟合中排除的观测值，指定为以逗号分隔的对组，其中包含 'Exclude' 和逻辑或数值索引向量，指示要从拟合中排除哪些观测值。

例如，您可以使用以下任一示例排除 6 个观测值中的 2 个和 3 个。

示例：'Exclude',[2,3]

示例：'Exclude',logical([0 1 1 0 0 0])

数据类型：single | double | logical

Intercept - 常数项的指示符
true （默认） | false

拟合中常数项（截距）的指示符，指定为以逗号分隔的对组，其中包含 'Intercept' 以及 true（表示在模型中包含常数项）或 false（表示从模型中删除常数项）。

仅当使用字符向量或字符串标量而不是公式或矩阵指定模型时，才使用 'Intercept'。

示例：'Intercept',false

PredictorVars - 预测变量
字符串数组 | 字符向量元胞数组 | 逻辑或数值索引向量

拟合中要使用的预测变量，指定为以逗号分隔的对组，其中包含 'PredictorVars' 和字符串数组或字符向量元胞数组（包含表或数据集数组包含 tbl 中变量名称），或是逻辑或数值索引向量（指示哪些列是预测变量）。

字符串值或字符向量应为 tbl 中的名称，或在您使用 'VarNames' 名称-值对组参数指定的名称。

默认值为 X 中的所有变量，或 tbl 中除 ResponseVar 以外的所有变量。

例如，您可以使用以下任一示例将第二个和第三个变量指定为预测变量。

示例：'PredictorVars',[2,3]

示例：'PredictorVars',logical([0 1 1 0 0 0])

数据类型：single | double | logical | string | cell

ResponseVar - 响应变量
tbl 中的最后一列 （默认） | 包含变量名称的字符向量或字符串标量 | 逻辑或数值索引向量

拟合中要使用的响应变量，指定为以逗号分隔的对组，其中包含 'ResponseVar' 和字符向量或字符串标量（包含表或数据集数组 tbl 中变量的名称），或者逻辑或数值索引向量（指示哪个列是响应变量）。在拟合表或数据集数组 tbl 时，通常需要使用 'ResponseVar'。

例如，您可以通过以下方式之一指定第四个变量 yield 作为六个变量的响应。

示例：'ResponseVar','yield'

示例：'ResponseVar',[4]

示例：'ResponseVar',logical([0 0 0 1 0 0])

数据类型：single | double | logical | char | string

RobustOpts - 稳健拟合类型的指示符
'off'（默认）| 'on' | 字符向量 | 字符串标量 | 结构体

要使用的稳健拟合类型的指示符，指定为以逗号分隔的对组，其中包含 'RobustOpts' 和下列值之一。

- 'off' - 没有稳健拟合。fitlm 使用普通最小二乘法。
- 'on' - 使用具有默认调整常量的 'bisquare' 权重函数进行稳健拟合。
- 字符向量或字符串标量 - 下表中稳健拟合权重函数的名称。fitlm 使用表中指定的对应默认调整常量。
- 包含两个字段 RobustWgtFun 和 Tune 的结构体。
  - RobustWgtFun 字段包含下表中稳健拟合权重函数的名称或自定义权重函数的函数句柄。
  - Tune 字段包含调整常量。如果未设置 Tune 字段，fitlm 将使用对应的默认调整常量。

权重函数	说明	默认调整常量
'andrews'	w = (abs(r)<pi) .* sin(r) ./ r	1.339
'bisquare'	w = (abs(r)<1) .* (1 - r.^2).^2（也称为双权）	4.685
'cauchy'	w = 1 ./ (1 + r.^2)	2.385
'fair'	w = 1 ./ (1 + abs(r))	1.400
'huber'	w = 1 ./ max(1, abs(r))	1.345
'logistic'	w = tanh(r) ./ r	1.205
'ols'	普通最小二乘法（无加权函数）	无
'talwar'	w = 1 * (abs(r)<1)	2.795
'welsch'	w = exp(-(r.^2))	2.985
函数句柄	自定义权重函数，接受缩放残差的向量 r，并返回与 r 大小相同的权重向量	1

  - 使用内置权重函数的默认调整常量得到的系数估计值的统计效率是普通最小二乘估计值的 95%，前提是响应具无离群值的正态分布。降低调整常量会增加分配给大残差的降权；增加调整常量会降低分配给大残差的降权。
  - 权重函数中的值 r 为

    r = resid/(tune*s*sqrt(1–h))，

    其中，resid 是上一次迭代的残差向量，tune 是调整常量，h 是最小二乘拟合的杠杆值向量，s 是误差项的标准差的估计值，由下式给出

    s = MAD/0.6745。

    MAD 是残差与其中位数的中位数绝对偏差。常量 0.6745 使正态分布的估计无偏。如果 X 有 p 个列，软件在计算中位数时会排除最小的 p 个绝对偏差。

对于稳健拟合，**fitlm** 使用 M 估计来构成估计方程，并使用 “Iteratively Reweighted Least Squares” (IRLS) 方法求解它们。

示例： **'RobustOpts','andrews'**

VarNames - 变量的名称
{'x1','x2',...,'xn','y'} （默认） | 字符串数组 | 字符向量元胞数组

变量的名称，指定为以逗号分隔的对组，其中前面包含 **'VarNames'** 和字符串数组或字符向量元胞数组（包含 **X** 列的名称），最后包含响应变量 **y** 的名称。

'VarNames' 不适用于表或数据集数组中的变量，因为这些变量已有名称。

变量名称不必是有效的 MATLAB 标识符。但是，如果名称无效，则在拟合或调整模型时无法使用公式；例如：

- 当您分别使用 **addTerms** 函数或 **removeTerms** 函数时，无法使用公式指定要添加或删除的项。
- 当您使用带名称-值对组参数 **'Lower'** 和 **'Upper'** 的 **step** 或 **stepwiselm** 函数时，则无法使用公式来指定模型的下界和上界。

在指定 **'VarNames',varNames** 之前，您可以使用 **isvarname** 函数验证 **varNames** 中的变量名称。如果变量名称无效，可以使用 **matlab.lang.makeValidName** 函数进行转换。

示例： **'VarNames',{'Horsepower','Acceleration','Model_Year','MPG'}**

数据类型： **string** | **cell**

Weights - 观测值权重
ones(n,1) （默认） | 非负标量值的 n×1 向量

观测值权重，指定为以逗号分隔的对组，其中包含 **'Weights'** 和非负标量值的 n×1 向量，其中 n 是观测值数目。

数据类型： **single** | **double**

输出参数

mdl - 线性模型
LinearModel 对象

表示对数据响应的最小二乘拟合的线性模型，以 **LinearModel** 对象形式返回。

如果 **'RobustOpts'** 名称-值对组的值不是 **[]** 或 **'ols'**，则模型不是最小二乘拟合，而是使用稳健拟合函数。

详细信息

项矩阵

项矩阵 **T** 是 t×(p + 1) 矩阵，用于指定模型中的项，其中 t 是项数，p 是预测变量数，而 + 1 表示响应变量。**T(i,j)** 的值是变量 **j** 在项 **i** 中的指数。

例如，假设一个输入包括三个预测变量 **x1**、**x2** 和 **x3** 以及响应变量 **y**，顺序为 **x1**、**x2**、**x3** 和 **y**。**T** 的每行表示一个项：

- **[0 0 0 0]** - 常数项或截距
- **[0 1 0 0]** - **x2**；等效于 **x1^0 * x2^1 * x3^0**
- **[1 0 1 0]** - **x1*x3**
- **[2 0 0 0]** - **x1^2**
- **[0 1 2 0]** - **x2*(x3^2)**

每项末尾的 **0** 表示响应变量。通常，项矩阵中由零组成的列向量表示响应变量的位置。如果您在矩阵和列向量中包含有预测变量和响应变量，则必须在每行的最后一列中包含 **0**，以表示响应变量。

公式

模型设定的公式是 **'y ~ terms'** 形式的字符向量或字符串标量。

- **y** 是响应名称。
- **terms** 使用 Wilkinson 表示法表示模型中的预测变量项。

要表示预测变量和响应变量，请使用表输入 **tbl** 的变量名称或使用 **VarNames** 指定的变量名称。**VarNames** 的默认值为 **{'x1','x2',...,'xn','y'}**。

例如：

- **'y ~ x1 + x2 + x3'** 指定一个具有截距的三变量线性模型。
- **'y ~ x1 + x2 + x3 – 1'** 指定一个无截距的三变量线性模型。请注意，默认情况下，公式包含常数（截距）项。要从模型中排除常数项，必须在公式中包括 **–1**。

公式包含常数项，除非您使用 **–1** 显式删除它。

Wilkinson 表示法

Wilkinson 表示法描述模型中出现的项。该表示法涉及模型中出现的项，而不涉及这些项的乘数（系数）。

Wilkinson 表示法使用下列符号：

- **+** 表示包含下一个变量。
- **–** 表示不包含下一个变量。
- **:** 定义交互效应，即项的乘积。
- ***** 定义交互效应和所有低阶项。
- **^** 求预测变量的幂，与使用 ***** 重复相乘效果一样，因此 **^** 也包括低阶项。
- **()** 对项进行分组。

下表显示 Wilkinson 表示法的典型示例。

Wilkinson 表示法	标准表示法中的项
1	常数（截距）项
x1^k，其中 **k** 是正整数	**x1**、$\mathbf{x1^2}$、...、$\mathbf{x1^k}$
x1 + x2	**x1**、**x2**
x1*x2	**x1**、**x2**、**x1*x2**

Wilkinson 表示法	标准表示法中的项
x1:x2	仅限 **x1*x2**
–x2	不包括 **x2**
x1*x2 + x3	**x1**、**x2**、**x3**、**x1*x2**
x1 + x2 + x3 + x1:x2	**x1**、**x2**、**x3**、**x1*x2**
x1*x2*x3 – x1:x2:x3	**x1**、**x2**、**x3**、**x1*x2**、**x1*x3**、**x2*x3**
x1*(x2 + x3)	**x1**、**x2**、**x3**、**x1*x2**、**x1*x3**

有关详细信息，请参阅“Wilkinson Notation”。

提示

- 要访问 **LinearModel** 对象 **mdl** 的模型属性，您可以使用圆点表示法。例如，**mdl.Residuals** 返回模型的原始、Pearson、Student 和标准化残差值的表。
- 在训练模型后，您可以生成预测新数据的响应的 C/C++ 代码。生成 C/C++ 代码需要 MATLAB Coder。有关详细信息，请参阅“Introduction to Code Generation”。

算法

- 主拟合算法是 QR 分解。对于稳健拟合，**fitlm** 使用 M 估计来构成估计方程，并使用 “Iteratively Reweighted Least Squares” (IRLS) 方法求解它们。
- **fitlm** 对分类预测变量的处理如下：
  - 具有 L 个水平（类别）的分类预测变量的模型包括 L – 1 个指示变量。模型使用第一个类别作为参考水平，因此它不包括该参考水平的指示变量。如果分类预测变量的数据类型是 **categorical**，则您可以使用 **categories** 检查类别的顺序，并使用 **reordercats** 对类别重新排序，以自定义参考水平。有关创建指示变量的更多详细信息，请参阅“Automatic Creation of Dummy Variables”。
  - **fitlm** 将一组 L – 1 个指示变量视为单一变量。如果要将这些指示变量分别视为不同的预测变量，可使用 **dummyvar** 手动创建指示变量。然后，在拟合模型时可使用这些指示变量，但对应于分类变量参考水平的一个变量除外。对于分类预测变量 **X**，如果您指定 **dummyvar(X)** 的所有列和一个截距项作为预测变量，则设计矩阵会变为秩亏矩阵。
  - 连续预测变量和具有 L 个水平的分类预测变量之间的交互效应项为连续预测变量与 L – 1 个指示变量的按元素乘积。
  - 分别具有 L 个和 M 个水平的两个分类预测变量之间的交互效应项为 (L – 1)*(M – 1) 个指示变量，包括两个分类预测变量水平的所有可能组合。
  - 您不能为分类预测变量指定高阶项，因为指示变量的平方等于其本身。
- **fitlm** 将 **tbl**、**X** 和 **Y** 中的 **NaN**、**''**（空字符向量）、**""**（空字符串）、**<missing>** 和 **<undefined>** 值视为缺失值。**fitlm** 在拟合中不使用具有缺失值的观测值。拟合后的模型的 **ObservationInfo** 属性指示 **fitlm** 是否在拟合中使用每个观测值。

替代功能

- 为了减少在高维数据集上的计算时间，可以使用 **fitrlinear** 函数拟合线性回归模型。
- 要正则化回归，请使用 **fitrlinear**、**lasso**、**ridge** 或 **plsregress**。
  - **fitrlinear** 使用 LASSO 或岭回归对高维数据集的回归进行正则化。

- **lasso** 使用 LASSO 或弹性网删除线性回归中多余的预测变量。
- **ridge** 使用岭回归对具有相关项的回归进行正则化。
- **plsregress** 使用偏最小二乘对具有相关项的回归进行正则化。

参考

[1] DuMouchel, W. H., and F. L. O'Brien. "Integrating a Robust Option into a Multiple Regression Computing Environment." *Computer Science and Statistics: Proceedings of the 21st Symposium on the Interface*. Alexandria, VA: American Statistical Association, 1989.

[2] Holland, P. W., and R. E. Welsch. "Robust Regression Using Iteratively Reweighted Least-Squares." *Communications in Statistics: Theory and Methods*, *A6*, 1977, pp. 813–827.

[3] Huber, P. J. *Robust Statistics*. Hoboken, NJ: John Wiley & Sons, Inc., 1981.

[4] Street, J. O., R. J. Carroll, and D. Ruppert. "A Note on Computing Robust Regression Estimates via Iteratively Reweighted Least Squares." *The American Statistician*. Vol. 42, 1988, pp. 152–154.

扩展功能

tall 数组
对行数太多而无法放入内存的数组进行计算。

此函数支持对无法放入内存的数据使用 tall 数组，但有一些限制。

- 如果 **fitlm** 的任何输入参数均为 tall 数组，则所有其他输入也必须为 tall 数组。这包括随 **'Weights'** 和 **'Exclude'** 名称-值对组提供的非空变量。
- tall 数组不支持 **'RobustOpts'** 名称-值对组。
- 对于 tall 数据，**fitlm** 返回 **CompactLinearModel** 对象，该对象包含与 **LinearModel** 对象相同的大多数属性。主要区别在于紧凑对象对内存要求很敏感。紧凑对象不包括包含数据的属性，也不包括与数据大小相同的数组。紧凑对象不包含以下 **LinearModel** 属性：

  - **Diagnostics**
  - **Fitted**
  - **ObservationInfo**
  - **ObservationNames**
  - **Residuals**
  - **Steps**
  - **Variables**

  您可以使用以下方法基于 **LM = fitlm(X,Y)** 返回的紧凑对象直接计算残差：

  ```
  RES = Y - predict(LM,X);
  S = LM.RMSE;
  histogram(RES,linspace(-3*S,3*S,51))
  ```

- 如果 **CompactLinearModel** 对象缺失包含分类因子的低阶项：

  - 不支持 **plotEffects** 和 **plotInteraction** 方法。

- 不支持具有 **'components'** 选项的 **anova** 方法。

有关详细信息，请参阅“使用 tall 数组处理无法放入内存的数据”。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
LinearModel | **predict** | **stepwiselm** | **fitrlinear**

主题
“What Is a Linear Regression Model?”
“Linear Regression”
“Linear Regression Workflow”
“Train Linear Regression Model”
“Predict or Simulate Responses to New Data”
“Examine Quality and Adjust Fitted Model”
“Linear Regression with Categorical Covariates”
“Reduce Outlier Effects Using Robust Regression”
“Stepwise Regression”

在 R2013b 中推出

fitdist

对数据进行概率分布对象拟合

语法

```
pd = fitdist(x,distname)
pd = fitdist(x,distname,Name,Value)

[pdca,gn,gl] = fitdist(x,distname,'By',groupvar)
[pdca,gn,gl] = fitdist(x,distname,'By',groupvar,Name,Value)
```

说明

pd = fitdist(x,distname) 通过对列向量 **x** 中的数据进行 **distname** 指定的分布拟合，创建概率分布对象。

pd = fitdist(x,distname,Name,Value) 使用一个或多个名称-值对组参数指定的附加选项创建概率分布对象。例如，您可以为迭代拟合算法指示删失数据或指定控制参数。

[pdca,gn,gl] = fitdist(x,distname,'By',groupvar) 基于分组变量 **groupvar** 对 **x** 中的数据进行 **distname** 指定的分布拟合，以创建概率分布对象。它返回拟合后的概率分布对象的元胞数组 **pdca**、组标签的元胞数组 **gn** 以及分组变量水平的元胞数组 **gl**。

[pdca,gn,gl] = fitdist(x,distname,'By',groupvar,Name,Value) 使用一个或多个名称-值对组参数指定的附加选项返回上述输出参数。例如，您可以为迭代拟合算法指示删失数据或指定控制参数。

示例

对数据进行正态分布拟合

加载样本数据。创建包含患者体重数据的向量。

```
load hospital
x = hospital.Weight;
```

通过对数据进行正态分布拟合来创建正态分布对象。

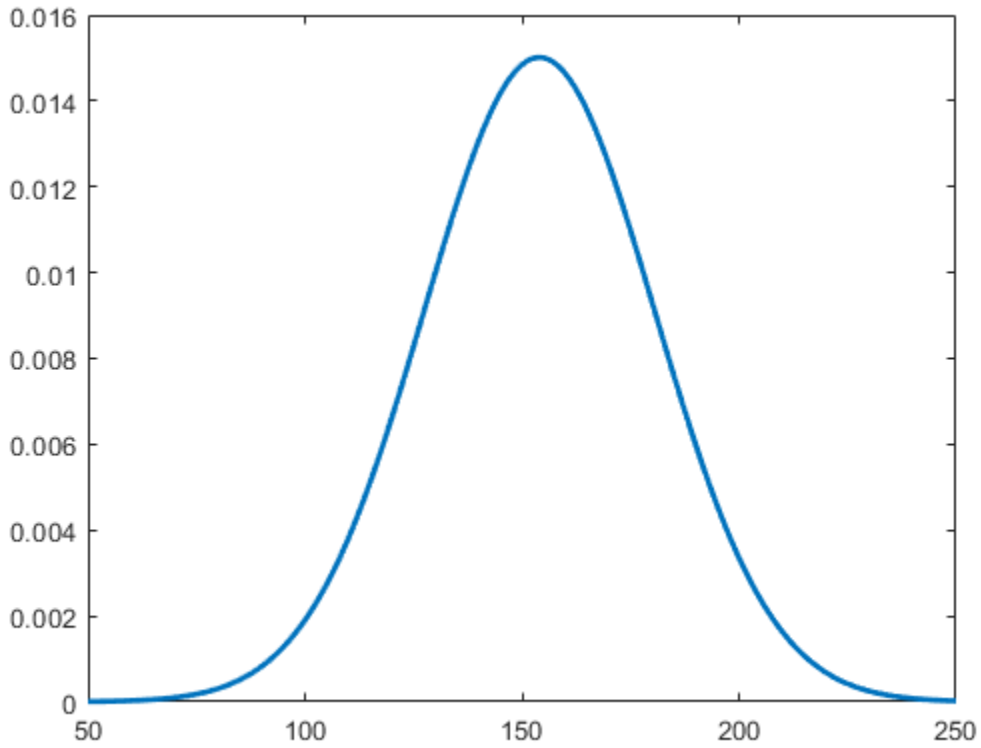
```
pd = fitdist(x,'Normal')
```

```
pd =
  NormalDistribution

  Normal distribution
     mu =     154   [148.728, 159.272]
  sigma = 26.5714   [23.3299, 30.8674]
```

参数估计值旁边的区间是分布参数的 95% 置信区间。

绘制分布的 pdf。

```
x_values = 50:1:250;
y = pdf(pd,x_values);
plot(x_values,y,'LineWidth',2)
```



对数据进行核分布拟合

加载样本数据。创建包含患者体重数据的向量。

```
load hospital
x = hospital.Weight;
```

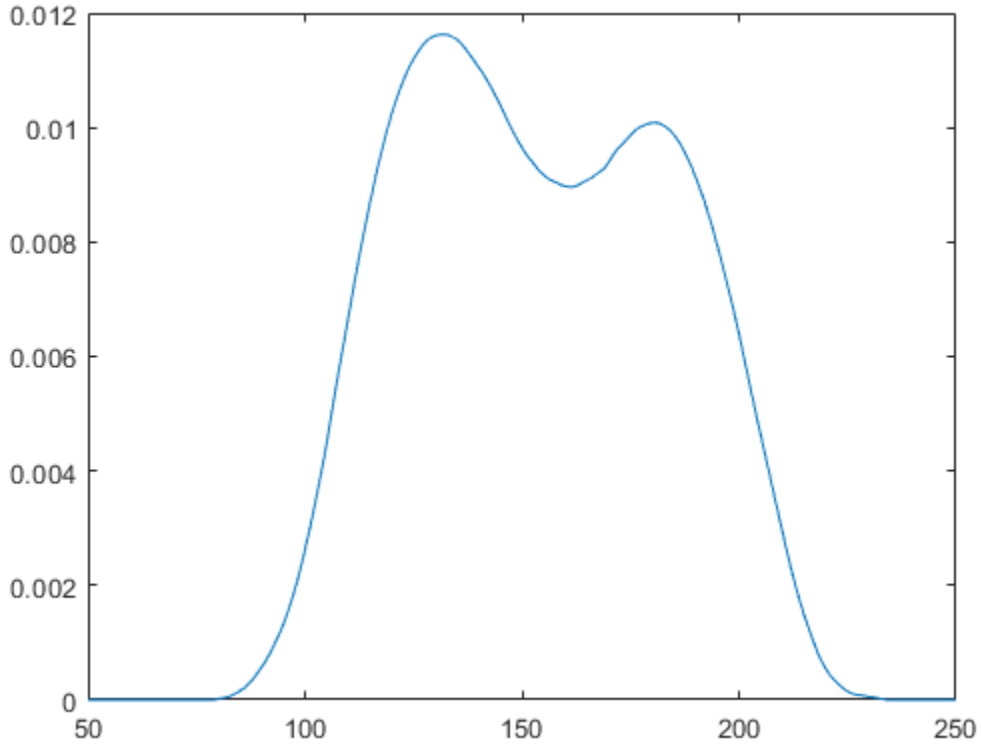
通过对数据进行核分布拟合来创建核分布对象。使用 Epanechnikov 核函数。

```
pd = fitdist(x,'Kernel','Kernel','epanechnikov')
```

```
pd =
  KernelDistribution

    Kernel = epanechnikov
    Bandwidth = 14.3792
    Support = unbounded
```

绘制分布的 pdf。

```
x_values = 50:1:250;
y = pdf(pd,x_values);
plot(x_values,y)
```



对分组数据进行正态分布拟合

加载样本数据。创建包含患者体重数据的向量。

```
load hospital
x = hospital.Weight;
```

通过对按患者性别分组的数据进行正态分布拟合来创建正态分布对象。

```
gender = hospital.Sex;
[pdca,gn,gl] = fitdist(x,'Normal','By',gender)
```

```
pdca=1×2 cell array
   {1x1 prob.NormalDistribution}    {1x1 prob.NormalDistribution}


gn = 2x1 cell
   {'Female'}
   {'Male'  }


gl = 2x1 cell
   {'Female'}
```

```
{'Male' }
```

元胞数组 **pdca** 包含两个概率分布对象，分别对应每个性别组。元胞数组 **gn** 包含两个组标签。元胞数组 **gl** 包含两个组水平。

查看元胞数组 **pdca** 中的各个分布，比较各性别的均值 **mu** 和标准差 **sigma**。

```
female = pdca{1}  % Distribution for females
```

```
female =
  NormalDistribution

  Normal distribution
     mu = 130.472   [128.183, 132.76]
  sigma = 8.30339   [6.96947, 10.2736]
```

```
male = pdca{2}  % Distribution for males
```

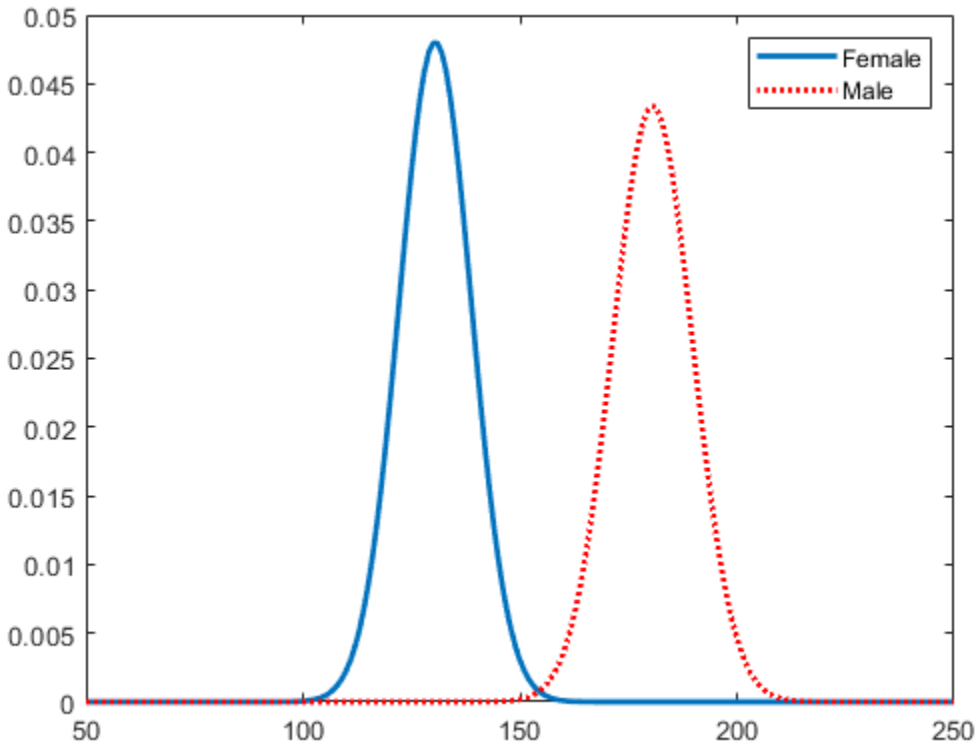
```
male =
  NormalDistribution

  Normal distribution
     mu = 180.532   [177.833, 183.231]
  sigma = 9.19322   [7.63933, 11.5466]
```

计算每个分布的 pdf。

```
x_values = 50:1:250;
femalepdf = pdf(female,x_values);
malepdf = pdf(male,x_values);
```

对 pdf 绘图，以直观地比较各性别的体重分布。

```
figure
plot(x_values,femalepdf,'LineWidth',2)
hold on
plot(x_values,malepdf,'Color','r','LineStyle',':','LineWidth',2)
legend(gn,'Location','NorthEast')
hold off
```



对分组数据进行核分布拟合

加载样本数据。创建包含患者体重数据的向量。

```
load hospital
x = hospital.Weight;
```

通过对按患者性别分组的数据进行核分布拟合来创建核分布对象。使用三角核函数。

```
gender = hospital.Sex;
[pdca,gn,gl] = fitdist(x,'Kernel','By',gender,'Kernel','triangle');
```

查看元胞数组 **pdca** 中的每个分布，以了解每个性别的核分布。

```
female = pdca{1}  % Distribution for females
```

```
female =
  KernelDistribution

    Kernel = triangle
    Bandwidth = 4.25894
    Support = unbounded
```

```
male = pdca{2}  % Distribution for males
```

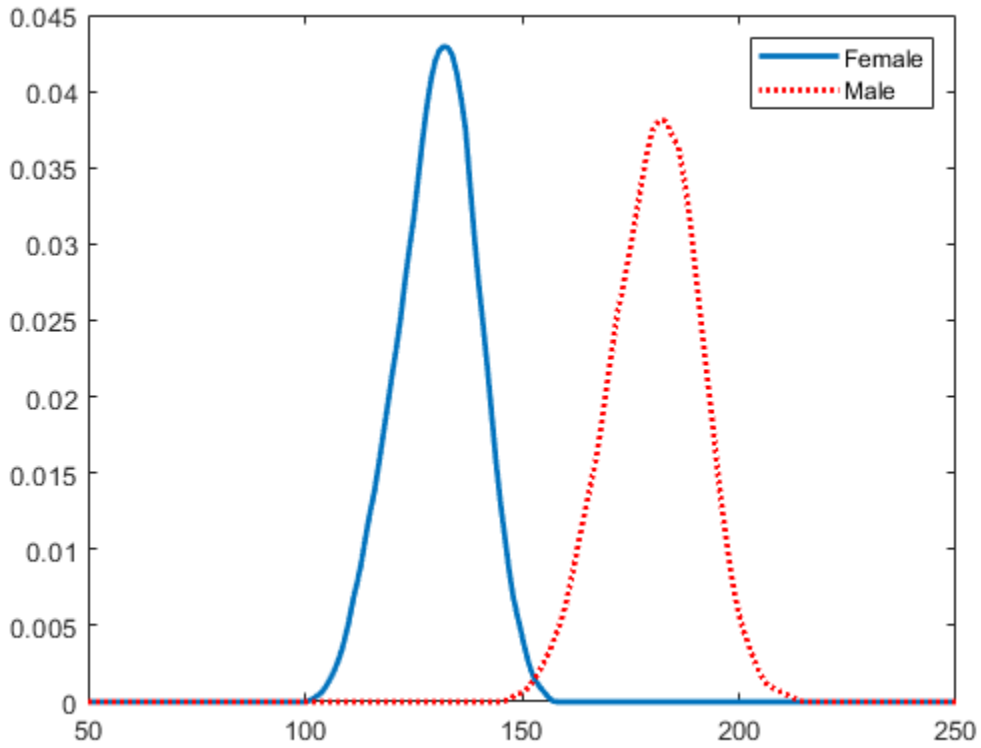
```
male =
  KernelDistribution

    Kernel = triangle
    Bandwidth = 5.08961
    Support = unbounded
```

计算每个分布的 pdf。

```
x_values = 50:1:250;
femalepdf = pdf(female,x_values);
malepdf = pdf(male,x_values);
```

对 pdf 绘图，以直观地比较各性别的体重分布。

```
figure
plot(x_values,femalepdf,'LineWidth',2)
hold on
plot(x_values,malepdf,'Color','r','LineStyle',':','LineWidth',2)
legend(gn,'Location','NorthEast')
hold off
```



输入参数

x - 输入数据
列向量

输入数据，指定为列向量。**fitdist** 忽略 **x** 中的 **NaN** 值。此外，删失向量或频数向量中的任何 **NaN** 值都会导致 **fitdist** 忽略 **x** 中的对应值。

数据类型： **double**

distname - 分布名称
字符向量 | 字符串标量

分布名称，指定为下列字符向量或字符串标量之一。**distname** 指定的分布决定返回的概率分布对象的类型。

分布名称	说明	分布对象
'Beta'	beta 分布	**BetaDistribution**
'Binomial'	二项分布	**BinomialDistribution**
'BirnbaumSaunders'	Birnbaum-Saunders 分布	**BirnbaumSaundersDistribution**
'Burr'	Burr 分布	**BurrDistribution**
'Exponential'	指数分布	**ExponentialDistribution**
'ExtremeValue'	极值分布	**ExtremeValueDistribution**
'Gamma'	gamma 分布	**GammaDistribution**
'GeneralizedExtremeValue'	广义极值分布	**GeneralizedExtremeValueDistribution**
'GeneralizedPareto'	广义帕累托分布	**GeneralizedParetoDistribution**
'HalfNormal'	半正态分布	**HalfNormalDistribution**
'InverseGaussian'	逆高斯分布	**InverseGaussianDistribution**
'Kernel'	核分布	**KernelDistribution**
'Logistic'	逻辑分布	**LogisticDistribution**
'Loglogistic'	对数逻辑分布	**LoglogisticDistribution**
'Lognormal'	对数正态分布	**LognormalDistribution**
'Nakagami'	Nakagami 分布	**NakagamiDistribution**
'NegativeBinomial'	负二项分布	**NegativeBinomialDistribution**
'Normal'	正态分布	**NormalDistribution**
'Poisson'	泊松分布	**PoissonDistribution**
'Rayleigh'	瑞利分布	**RayleighDistribution**
'Rician'	莱斯分布	**RicianDistribution**
'Stable'	稳定分布	**StableDistribution**
'tLocationScale'	t 位置尺度分布	**tLocationScaleDistribution**
'Weibull'	Weibull 分布	**WeibullDistribution**

groupvar - 分组变量
分类数组 | 逻辑或数值向量 | 字符数组 | 字符串数组 | 字符向量元胞数组

分组变量，指定为分类数组、逻辑或数值向量、字符数组、字符串数组或字符向量元胞数组。分组变量中的每个唯一值定义一个组。

例如，如果 **Gender** 是字符向量元胞数组，其值为 **'Male'** 和 **'Female'**，则您可以使用 **Gender** 作为分组变量，按性别对数据进行分布拟合。

通过指定包含分组变量的元胞数组，可以使用多个分组变量。所有指定分组变量的值相同的观测值会放在同一个组中。

例如，如果 **Smoker** 是逻辑向量，其中值为 **0** 表示非吸烟者，值为 **1** 表示吸烟者，则指定元胞数组 **{Gender,Smoker}** 会将观测值分为四组：男性吸烟者、男性非吸烟者、女性吸烟者和女性非吸烟者。

示例： **{Gender,Smoker}**

数据类型： **categorical** | **logical** | **single** | **double** | **char** | **string** | **cell**

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例： **fitdist(x,'Kernel','Kernel','triangle')** 使用三角核函数对 **x** 中的数据进行核分布对象拟合。

Censoring - 删失数据的逻辑标志
0 （默认） | 逻辑值向量

指示删失数据的逻辑标志，指定为以逗号分隔的对组，其中包含 **'Censoring'** 和与输入向量 **x** 大小相同的逻辑值向量。当 **x** 中的对应元素是右删失观测值时，该值为 **1**，当对应元素是精确观测值时，该值为 **0**。默认值为由 **0** 组成的向量，表示所有观测值均为准确的。

fitdist 会忽略此删失向量中的任何 **NaN** 值。此外，**x** 或频率向量中的任何 **NaN** 值都会导致 **fitdist** 忽略删失向量中的对应值。

仅当 **distname** 是 **'BirnbaumSaunders'**、**'Burr'**、**'Exponential'**、**'ExtremeValue'**、**'Gamma'**、**'InverseGaussian'**、**'Kernel'**、**'Logistic'**、**'Loglogistic'**、**'Lognormal'**、**'Nakagami'**、**'Normal'**、**'Rician'**、**'tLocationScale'** 或 **'Weibull'** 时，此参数才有效。

数据类型： **logical**

Frequency - 观测值频率
1 （默认） | 非负整数值向量

观测值频率，指定为以逗号分隔的对组，其中包含 **'Frequency'** 和与输入向量 **x** 大小相同的非负整数值向量。频率向量的每个元素指定 **x** 中对应元素的频率。默认值为由 **1** 组成的向量，表示 **x** 中的每个值仅出现一次。

fitdist 忽略此频率向量中的任何 **NaN** 值。此外，**x** 或删失向量中的任何 **NaN** 值都会导致 **fitdist** 忽略频率向量中的对应值。

数据类型： **single** | **double**

Options - 控制参数
结构体

迭代拟合算法的控制参数，指定为以逗号分隔的对组，其中包含 **'Options'** 和您使用 **statset** 创建的结构体。

数据类型： **struct**

NTrials - 试验次数
正整数值

二项分布的试验次数，指定为由 **'NTrials'** 和正整数值组成的以逗号分隔的对组。您必须将 **distname** 指定为 **'Binomial'** 才能使用此选项。

数据类型： **single** | **double**

Theta - 阈值参数
0 （默认） | 标量值

广义帕累托分布的阈值参数，指定为由 **'Theta'** 和标量值组成的以逗号分隔的对组。您必须将 **distname** 指定为 **'GeneralizedPareto'** 才能使用此选项。

数据类型： **single** | **double**

mu - 位置参数
0 （默认） | 标量值

半正态分布的位置参数，指定为由 **'mu'** 和标量值组成的以逗号分隔的对组。您必须将 **distname** 指定为 **'HalfNormal'** 才能使用此选项。

数据类型： **single** | **double**

Kernel - 核平滑器类型
'normal' （默认） | **'box'** | **'triangle'** | **'epanechnikov'**

核平滑器类型，指定为以逗号分隔的对组，其中包含 **'Kernel'** 和以下项之一：

- **'normal'**
- **'box'**
- **'triangle'**
- **'epanechnikov'**

您必须将 **distname** 指定为 **'Kernel'** 才能使用此选项。

Support - 核密度支持
'unbounded' （默认） | **'positive'** | 二元素向量

核密度支持，指定为以逗号分隔的对组，其中包含 **'Support'** 和 **'unbounded'**、**'positive'** 或二元素向量。

'unbounded'	密度可以取任意实数。
'positive'	密度仅限于正值。

您也可以指定二元素向量，给定支持密度的有限下限和上限。

您必须将 **distname** 指定为 **'Kernel'** 才能使用此选项。

数据类型： **single** | **double** | **char** | **string**

Width - 核平滑窗口的带宽
标量值

核平滑窗口的带宽，指定为由 **'Width'** 和标量值组成的以逗号分隔的对组。**fitdist** 使用的默认值为估计正态密度的最佳值，但您可能希望选择较小的值来显示一些特征，比如多个众数。您必须将 **distname** 指定为 **'Kernel'** 才能使用此选项。

数据类型： **single | double**

输出参数

pd - 概率分布
概率分布对象

概率分布，以概率分布对象形式返回。**distname** 指定的分布决定返回的概率分布对象的类的类型。有关 **distname** 值和对应概率分布对象的列表，请参阅 **distname**。

pdca - 概率分布对象
元胞数组

由 **distname** 指定类型的概率分布对象，以元胞数组形式返回。有关 **distname** 值和对应概率分布对象的列表，请参阅 **distname**。

gn - 组标签
字符向量元胞数组

组标签，以字符向量元胞数组形式返回。

gl - 分组变量水平
字符向量元胞数组

分组变量水平，以字符向量元胞数组形式返回，每个分组变量对应于其中包含的一列。

算法

fitdist 函数使用最大似然估计来拟合大多数分布。两个例外是带有未删失数据的正态分布和对数正态分布。

- 对于未删失的正态分布，sigma 参数的估计值是方差的无偏估计值的平方根。
- 对于未删失的对数正态分布，sigma 参数的估计值是数据对数的方差的无偏估计值的平方根。

替代功能

App

分布拟合器打开一个图形用户界面，以便您从工作区导入数据，并以交互方式对该数据进行概率分布拟合。然后，您可以将分布作为概率分布对象保存到工作区。使用 **distributionFitter** 打开分布拟合器，或点击 Apps 选项卡上的“分布拟合器”。

参考

[1] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Vol. 1, Hoboken, NJ: Wiley-Interscience, 1993.

[2] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Vol. 2, Hoboken, NJ: Wiley-Interscience, 1994.

[3] Bowman, A. W., and A. Azzalini. *Applied Smoothing Techniques for Data Analysis*. New York: Oxford University Press, 1997.

扩展功能

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

- 支持的语法包括：

  ```
  pd = fitdist(x,distname)
  pd = fitdist(x,distname,Name,Value)
  ```

  代码生成不支持包含分组变量 **'By',groupvar** 和相关输出参数 **pdca**、**gn** 和 **gl** 的语法。
- **fitdist** 支持 beta 分布、指数分布、极值分布、对数正态分布、正态分布和 Weibull 分布的代码生成。
  - **distname** 的值可以是 **'Beta'**、**'Exponential'**、**'ExtremeValue'**、**'Lognormal'**、**'Normal'** 或 **'Weibull'**。
  - **distname** 的值必须为编译时常量。
- **x**、**'Censoring'**、**'Frequency'** 的值不能包含 **NaN** 值。
- 代码生成会忽略 beta 分布的 **'Frequency'** 值，所以您需要手动将重复值添加到 **x** 中（而不是通过指定 **'Frequency'** 值），以便 **x** 中的值具有所需的频数。
- 代码生成不支持这些输入参数：**groupvar**、**NTrials**、**Theta**、**mu**、**Kernel**、**Support** 和 **Width**。
- 名称-值对组参数中的名称必须为编译时常量。
- **pd** 的下列对象函数支持代码生成：**cdf**、**icdf**、**iqr**、**mean**、**median**、**pdf**、**std**、**truncate** 和 **var**。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “Code Generation for Probability Distribution Objects”。

另请参阅

makedist | **distributionFitter** | **paramci** | **histfit** | **mle**

主题
“Working with Probability Distributions”
“Supported Distributions”

在 R2009a 中推出

fullfact

完全析因设计

语法

dFF = fullfact(levels)

说明

dFF = fullfact(levels) 给出具有 n 个因子的完全析因设计的因子设置 **dFF**，其中每个因子的水平数由长度为 n 的向量 **levels** 给出。**dFF** 是 m×n，其中 m 是完全析因设计中的处理数。**dFF** 的每行对应一种处理。每列都包含单个因子的设置，即从 1 到水平数的整数值。

示例

下面生成一个八次试验的完全析因设计，第一个因子有两个水平，第二个因子有四个水平：

```
dFF = fullfact([2 4])
dFF =
  1   1
  2   1
  1   2
  2   2
  1   3
  2   3
  1   4
  2   4
```

另请参阅

ff2n

在 R2006a 之前推出

gline

以交互方式将线添加到绘图

语法

```
gline(h)
gline
hline = gline(...)
```

说明

gline(h) 允许您通过点击两个端点处的指针，用句柄 **h** 在图窗中绘制线段。伸缩线条跟踪指针移动。

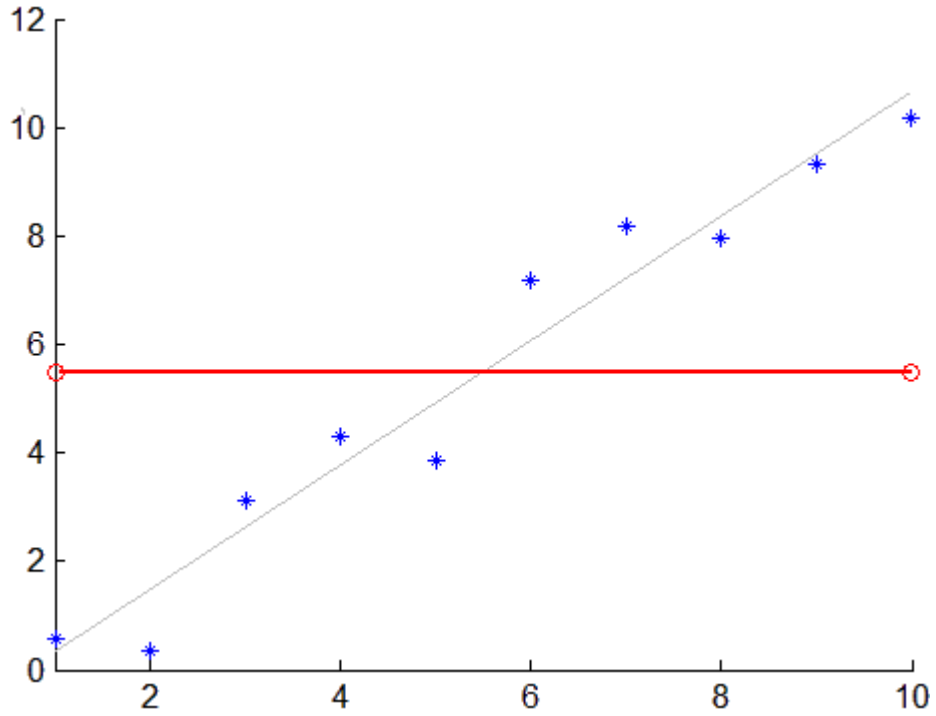
没有输入参数的 **gline** 默认为 **h = gcf** 并在当前图窗中绘制。

hline = gline(...) 返回该线的句柄 **hline**。

示例

使用 **gline** 连接绘图中的两点：

```
x = 1:10;

y = x + randn(1,10);
scatter(x,y,25,'b','*')

lsline

mu = mean(y);
hold on
plot([1 10],[mu mu],'ro')

hline = gline; % Connect circles
set(hline,'Color','r')
```



另请参阅

refline | refcurve | lsline

在 R2006a 之前推出

histfit

具有分布拟合的直方图

语法

histfit(data)
histfit(data,nbins)
histfit(data,nbins,dist)

histfit(ax, ___)

h = histfit(___)

说明

histfit(data) 绘制 **data** 中的值的直方图并拟合正态密度函数，直方图的 bin 个数等于 **data** 中元素个数的平方根。

histfit(data,nbins) 使用 **nbins** 个 bin 绘制直方图，并拟合正态密度函数。

histfit(data,nbins,dist) 使用 **nbins** 个 bin 绘制直方图，并根据 **dist** 指定的分布拟合密度函数。

histfit(ax, ___) 使用 **Axes** 对象 **ax** 指定的绘图坐标区。将 **ax** 指定为第一个输入参数，后跟先前语法中的任意输入参数组合。

h = histfit(___) 返回句柄向量 **h**，其中 **h(1)** 是直方图的句柄，**h(2)** 是密度曲线的句柄。

示例

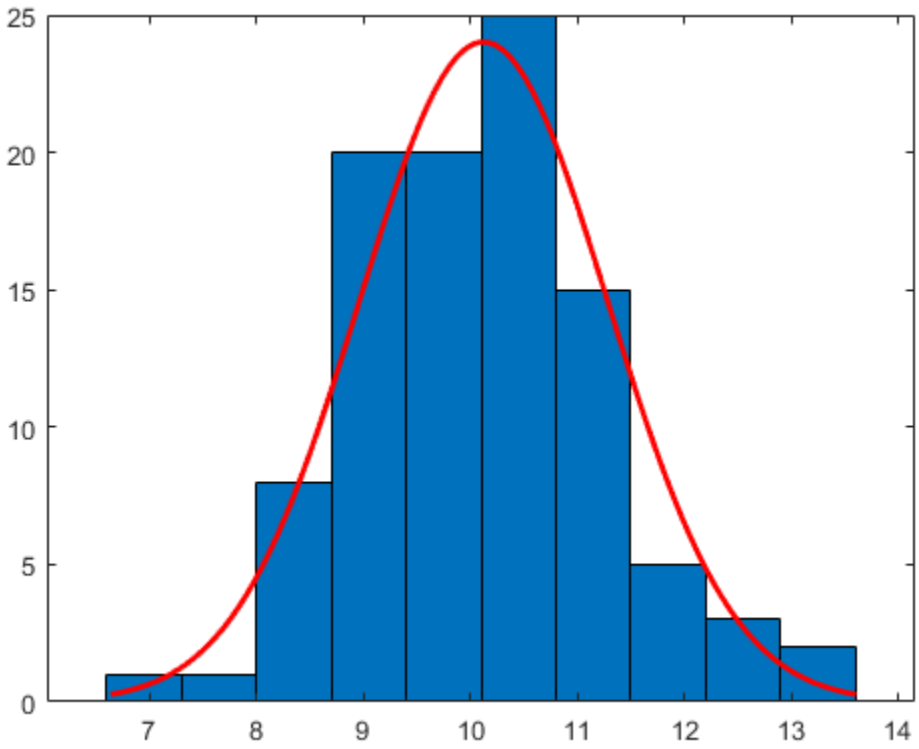
具有正态分布拟合的直方图

用均值 10 和方差 1 从正态分布生成大小为 100 的样本。

```
rng default; % For reproducibility
r = normrnd(10,1,100,1);
```

构建具有正态分布拟合的直方图。

```
histfit(r)
```



histfit 使用 **fitdist** 对数据进行分布拟合。使用 **fitdist** 获得在拟合中使用的参数。

```
pd = fitdist(r,'Normal')
```

```
pd =
  NormalDistribution

  Normal distribution
      mu = 10.1231    [9.89244, 10.3537]
   sigma =  1.1624    [1.02059, 1.35033]
```

参数估计值旁边的区间是分布参数的 95% 置信区间。

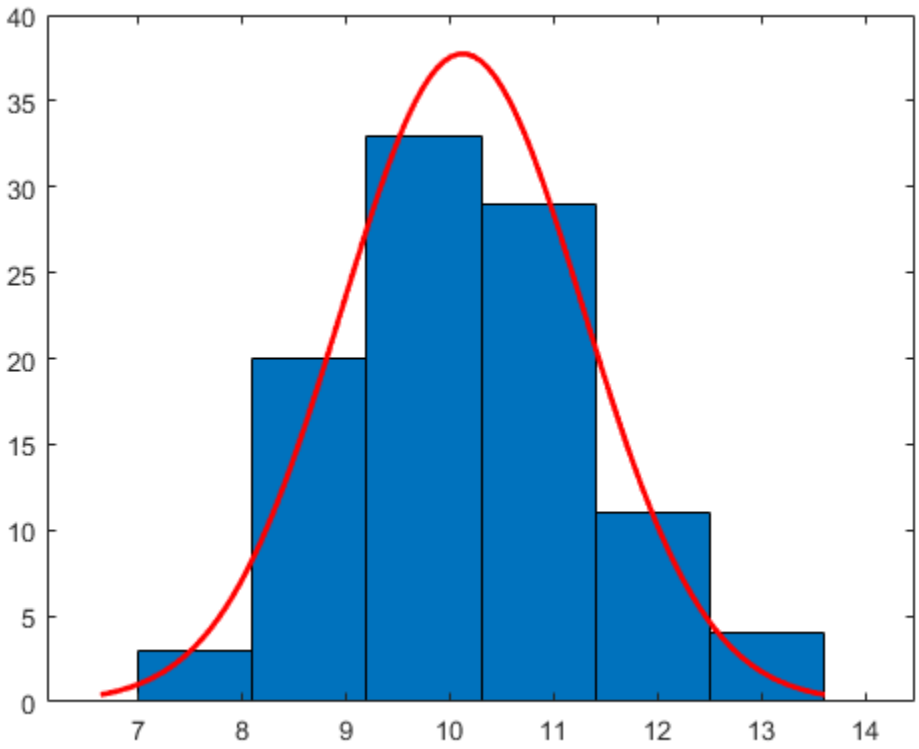
给定 bin 数的直方图

用均值 10 和方差 1 从正态分布生成大小为 100 的样本。

```
rng default; % For reproducibility
r = normrnd(10,1,100,1);
```

使用六个 bin 构造具有正态分布拟合的直方图。

```
histfit(r,6)
```



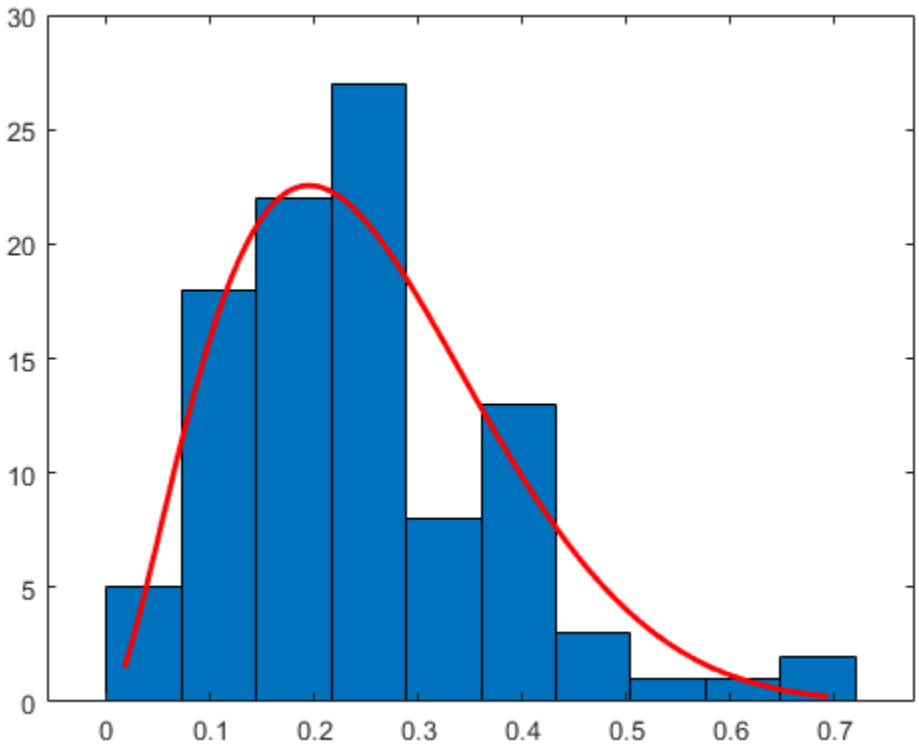
具有指定分布拟合的直方图

使用参数 (3,10) 从 beta 分布生成大小为 100 的样本。

```
rng default;  % For reproducibility
b = betarnd(3,10,100,1);
```

使用 10 个 bin 构造具有 beta 分布拟合的直方图。

```
histfit(b,10,'beta')
```



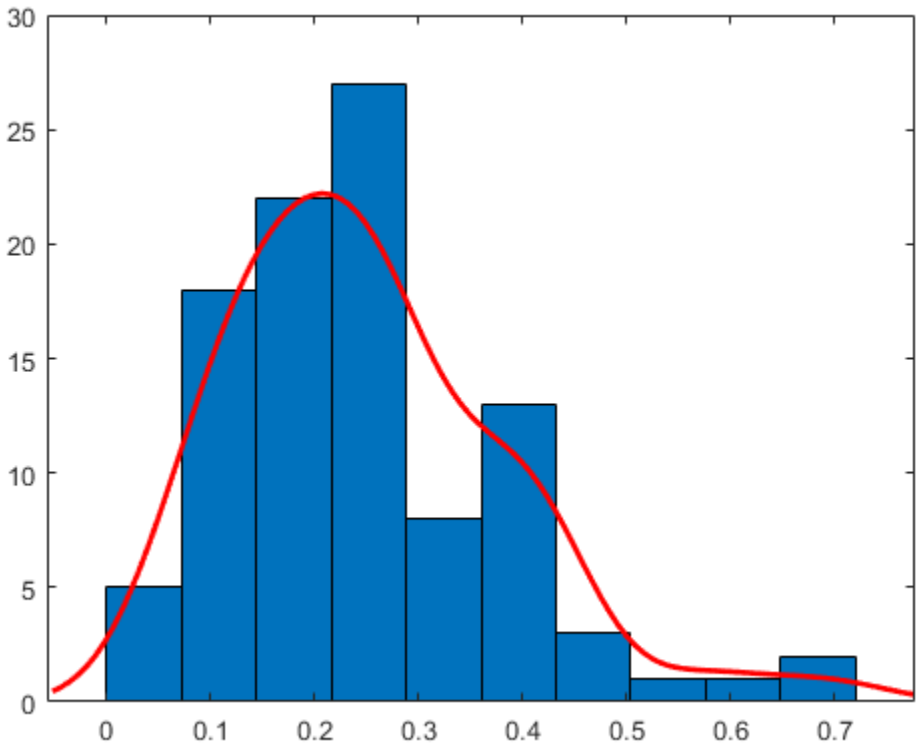
具有核平滑函数拟合的直方图

使用参数 (3,10) 从 beta 分布生成大小为 100 的样本。

```
rng default;  % For reproducibility
b = betarnd(3,10,[100,1]);
```

使用 10 个 bin 构造具有平滑函数拟合的直方图。

```
histfit(b,10,'kernel')
```



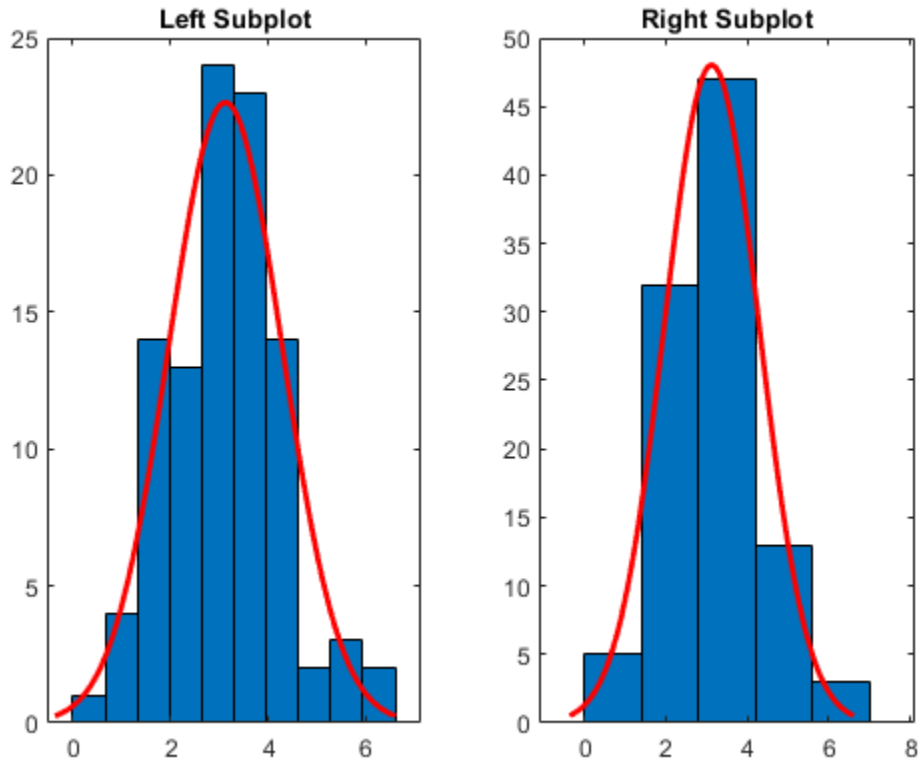
为具有分布拟合的直方图指定坐标区

用均值 3 和方差 1 从正态分布生成大小为 100 的样本。

```
rng('default') % For reproducibility
r = normrnd(3,1,100,1);
```

创建一个包含两个子图的图窗，并以 ax1 和 ax2 形式返回 Axes 对象。通过引用对应的 Axes 对象，在每个坐标区中创建一个具有正态分布拟合的直方图。在左侧子图中，绘制一个具有 10 个 bin 的直方图。在右侧子图中，绘制一个具有 5 个 bin 的直方图。通过将对应的 Axes 对象传递给 title 函数，为每个绘图添加标题。

```
ax1 = subplot(1,2,1); % Left subplot
histfit(ax1,r,10,'normal')
title(ax1,'Left Subplot')

ax2 = subplot(1,2,2); % Right subplot
histfit(ax2,r,5,'normal')
title(ax2,'Right Subplot')
```



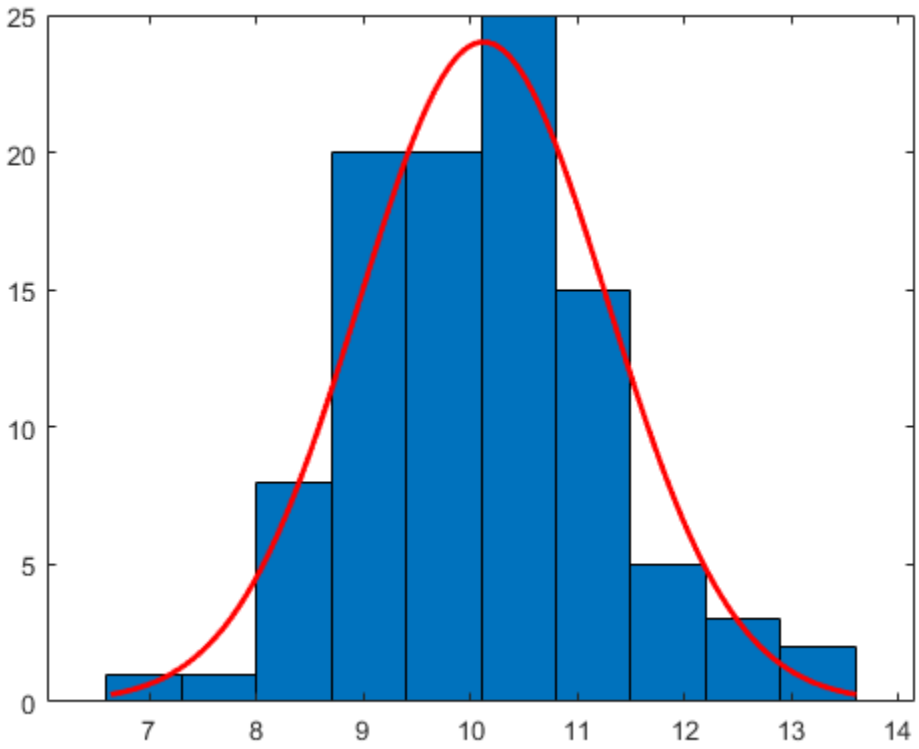
具有分布拟合的直方图的句柄

用均值 10 和方差 1 从正态分布生成大小为 100 的样本。

```
rng default % for reproducibility
r = normrnd(10,1,100,1);
```

构建具有正态分布拟合的直方图。

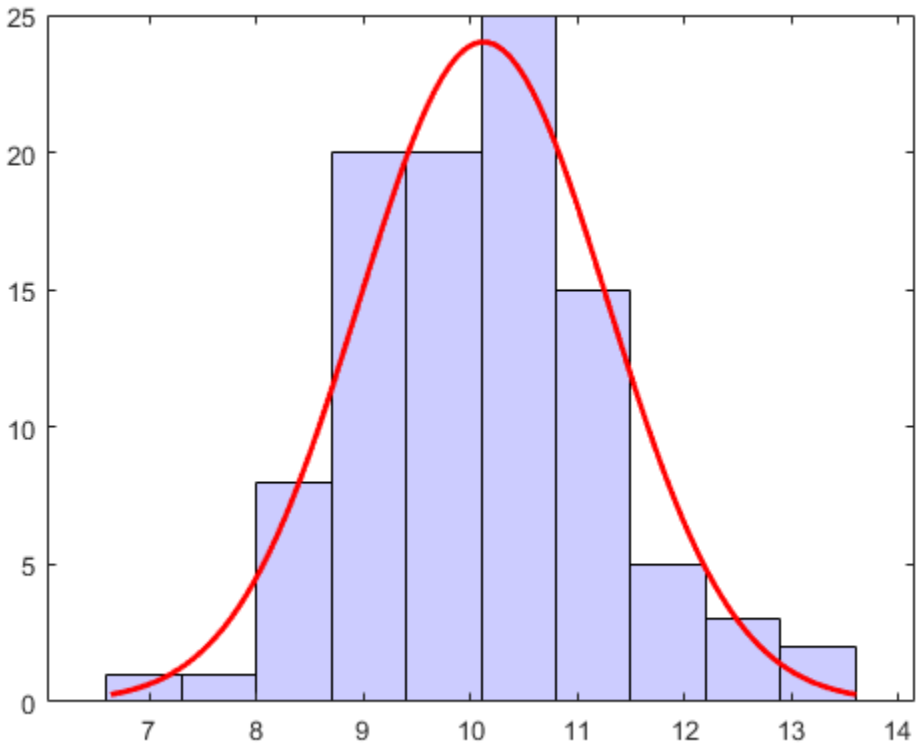
```
h = histfit(r,10,'normal')
```




```
h =
  2x1 graphics array:

  Bar
  Line
```

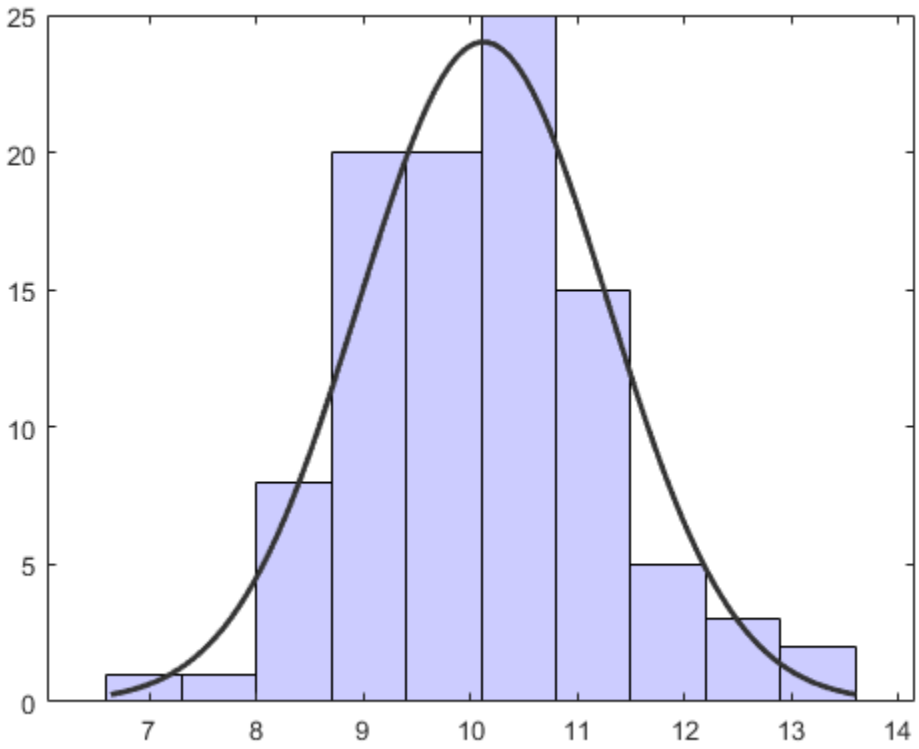
更改直方图的条形颜色。

```
h(1).FaceColor = [.8 .8 1];
```



更改密度曲线的颜色。

```
h(2).Color = [.2 .2 .2];
```



输入参数

data - 输入数据
向量

输入数据，指定为向量。

示例： `data = [1.5 2.5 4.6 1.2 3.4]`

示例： `data = [1.5 2.5 4.6 1.2 3.4]'`

数据类型： `double` | `single`

nbins - bin 个数
正整数 | [ ]

直方图的 bin 个数，指定为正整数。默认值为 `data` 中元素数的平方根，向上舍入。在拟合分布时，使用 [ ] 表示使用默认的 bin 个数。

示例： `y = histfit(x,8)`

示例： `y = histfit(x,10,'gamma')`

示例： `y = histfit(x,[ ],'weibull')`

数据类型： `double` | `single`

dist - 要拟合的分布
'normal'（默认）| 字符向量 | 字符串标量

要与直方图拟合的分布，指定为字符向量或字符串标量。下表显示支持的分布。

dist	说明
'beta'	beta
'birnbaumsaunders'	Birnbaum-Saunders
'burr'	Burr 类型 XII
'exponential'	指数
'extreme value' 或 'ev'	极值
'gamma'	gamma
'generalized extreme value' 或 'gev'	广义极值
'generalized pareto' 或 'gp'	广义帕累托（阈值 0）
'inversegaussian'	逆高斯
'logistic'	逻辑
'loglogistic'	对数逻辑
'lognormal'	对数正态
'nakagami'	Nakagami
'negative binomial' 或 'nbin'	负二项
'normal'	正态
'poisson'	泊松
'rayleigh'	瑞利
'rician'	莱斯
'tlocationscale'	t 位置尺度
'weibull' 或 'wbl'	Weibull
'kernel'	非参数化核平滑分布。密度在覆盖 data 数据范围的 100 个等距点处进行计算。它最适合连续分布的样本。

ax - 绘图坐标区
Axes 对象

绘图的坐标区，指定为 Axes 对象。如果未指定 ax，则 histfit 使用当前坐标区创建绘图。有关创建 Axes 对象的详细信息，请参阅 axes。

输出参数

h - 绘图的句柄
绘图句柄

以向量形式返回的绘图句柄，其中 h(1) 是直方图句柄，h(2) 是密度曲线句柄。histfit 对密度进行归一化，使曲线下的总面积与直方图的总面积相匹配。

算法

histfit 使用 **fitdist** 对数据进行分布拟合。使用 **fitdist** 获得在拟合中使用的参数。

另请参阅

histogram | **normfit** | **distributionFitter** | **fitdist** | **paramci**

在 R2006a 之前推出

hougen

Hougen-Watson 模型

语法

yhat = hougen(beta,x)

说明

yhat = hougen(beta,x) 函数根据参数向量 **beta** 和数据矩阵 **X** 返回响应速率的预测值 **yhat**。**beta** 必须有 5 个元素，**X** 必须有 3 列。

hougen 是 **rsmdemo** 的工具函数。

模型形式为：

$$\widehat{y} = \frac{\beta_1 x_2 - x_3/\beta_5}{1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_3}$$

参考文献

[1] Bates, D. M., and D. G. Watts. *Nonlinear Regression Analysis and Its Applications*. Hoboken, NJ: John Wiley & Sons, Inc., 1988.

另请参阅

rsmdemo

在 R2006a 之前推出

kmeans

k 均值聚类

语法

```
idx = kmeans(X,k)
idx = kmeans(X,k,Name,Value)
[idx,C] = kmeans( ___ )
[idx,C,sumd] = kmeans( ___ )
[idx,C,sumd,D] = kmeans( ___ )
```

说明

idx = kmeans(X,k) 执行 k 均值聚类 （第 33-130 页），以将 n×p 数据矩阵 **X** 的观测值划分为 **k** 个聚类，并返回包含每个观测值的簇索引的 n×1 向量 (**idx**)。**X** 的行对应于点，列对应于变量。

默认情况下，**kmeans** 使用欧几里德距离平方度量，并用 k-means++ 算法 （第 33-130 页）进行簇中心初始化。

idx = kmeans(X,k,Name,Value) 进一步按一个或多个 **Name,Value** 对组参数所指定的附加选项返回簇索引。

例如，指定余弦距离、使用新初始值重复聚类的次数或使用并行计算的次数。

[idx,C] = kmeans(___) 在 **k**×p 矩阵 **C** 中返回 **k** 个簇质心的位置。

[idx,C,sumd] = kmeans(___) 在 **k**×1 向量 **sumd** 中返回簇内的点到质心距离的总和。

[idx,C,sumd,D] = kmeans(___) 在 n×**k** 矩阵 **D** 中返回每个点到每个质心的距离。

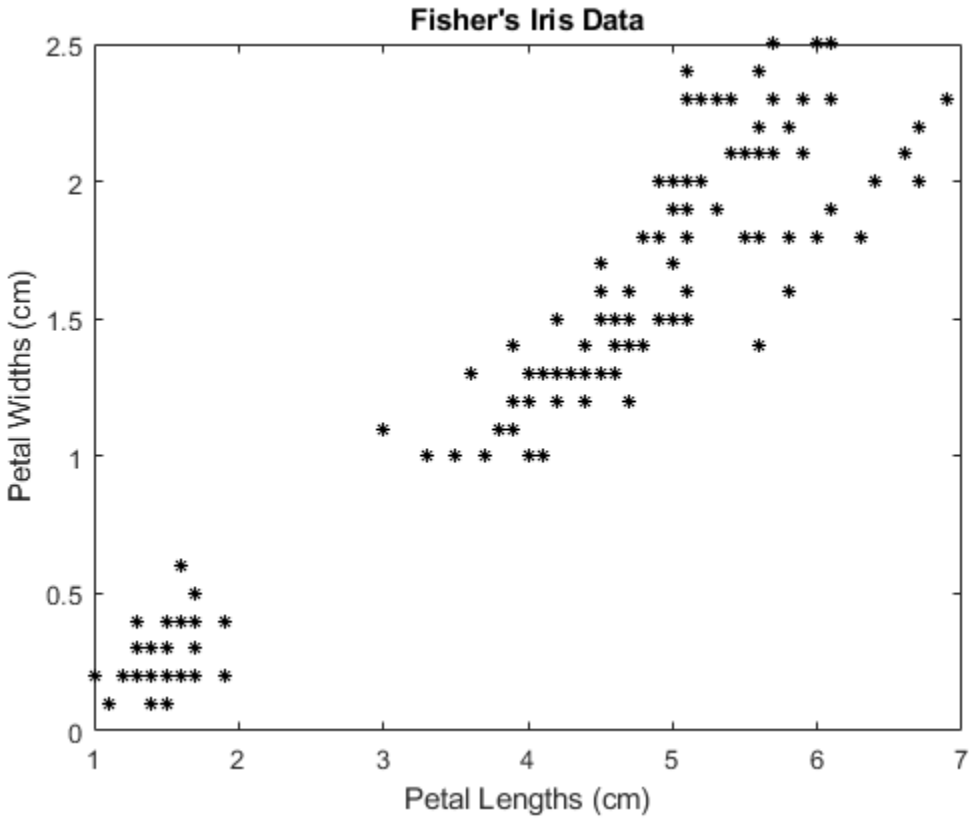
示例

训练 k 均值聚类算法

对数据进行 k 均值聚类，然后绘制簇区域。

加载 Fisher 鸢尾花数据集。使用花瓣长度和宽度作为预测变量。

```
load fisheriris
X = meas(:,3:4);

figure;
plot(X(:,1),X(:,2),'k*','MarkerSize',5);
title 'Fisher''s Iris Data';
xlabel 'Petal Lengths (cm)';
ylabel 'Petal Widths (cm)';
```



较大的簇似乎被分成两个区域：一个较低方差区域，一个较高方差区域。这可能表明较大的簇是两个重叠的簇。

对数据进行聚类。指定 k = 3 个簇。

```
rng(1); % For reproducibility
[idx,C] = kmeans(X,3);
```

idx 是与 **X** 中的观测值对应的预测簇索引的向量。**C** 是包含最终质心位置的 3×2 矩阵。

使用 **kmeans** 计算从每个质心到网格上各点的距离。为此，将质心 (**C**) 和网格上的点传递给 **kmeans**，并实现算法的一次迭代。

```
x1 = min(X(:,1)):0.01:max(X(:,1));
x2 = min(X(:,2)):0.01:max(X(:,2));
[x1G,x2G] = meshgrid(x1,x2);
XGrid = [x1G(:),x2G(:)]; % Defines a fine grid on the plot

idx2Region = kmeans(XGrid,3,'MaxIter',1,'Start',C);
```

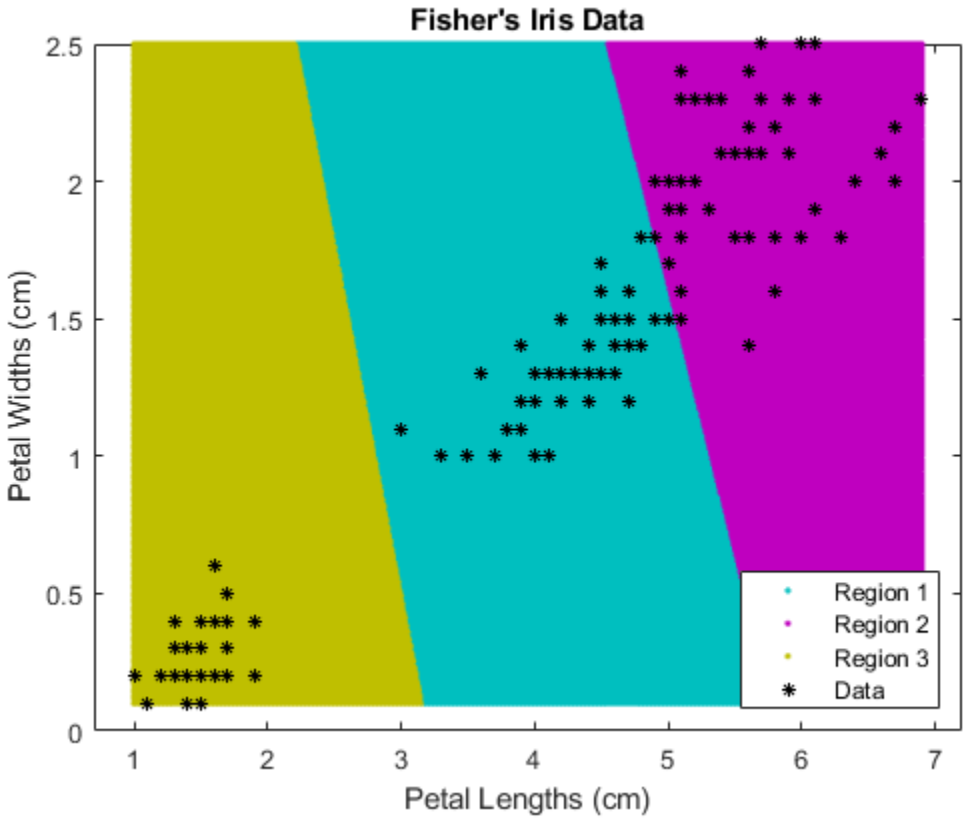
```
Warning: Failed to converge in 1 iterations.
```

```
   % Assigns each node in the grid to the closest centroid
```

kmeans 显示一条警告，指出算法未收敛，这是您应预料到的，因为软件只实现了一次迭代。

绘制簇区域。

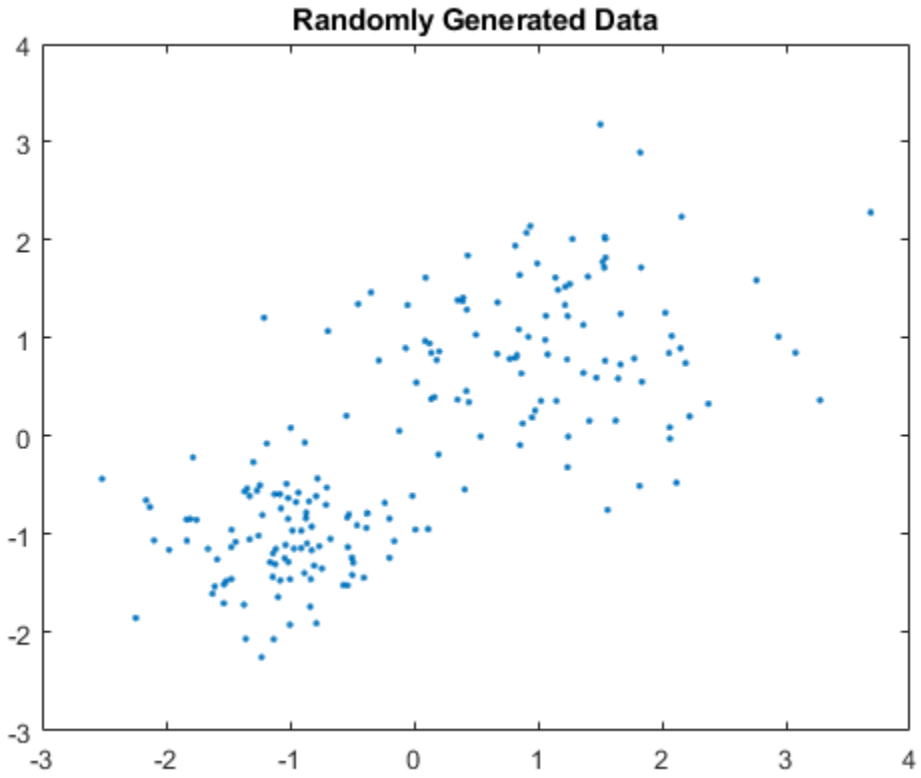
```
figure;
gscatter(XGrid(:,1),XGrid(:,2),idx2Region,...
    [0,0.75,0.75;0.75,0,0.75;0.75,0.75,0],'..');
hold on;
plot(X(:,1),X(:,2),'k*','MarkerSize',5);
title 'Fisher''s Iris Data';
xlabel 'Petal Lengths (cm)';
ylabel 'Petal Widths (cm)';
legend('Region 1','Region 2','Region 3','Data','Location','SouthEast');
hold off;
```



将数据分成两个簇

随机生成样本数据。

```
rng default; % For reproducibility
X = [randn(100,2)*0.75+ones(100,2);
    randn(100,2)*0.5-ones(100,2)];

figure;
plot(X(:,1),X(:,2),'.');
title 'Randomly Generated Data';
```



数据中似乎有两个簇。

将数据分成两个簇，并从五个初始化中选择最佳排列。显示最终输出。

```
opts = statset('Display','final');
[idx,C] = kmeans(X,2,'Distance','cityblock',...
    'Replicates',5,'Options',opts);
```

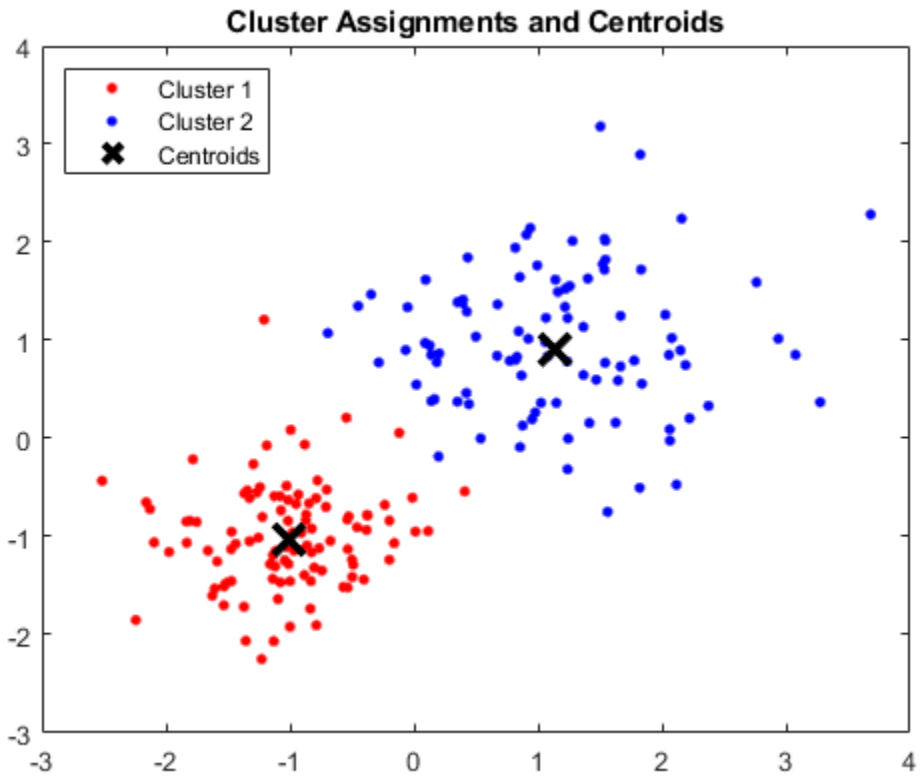
```
Replicate 1, 3 iterations, total sum of distances = 201.533.
Replicate 2, 5 iterations, total sum of distances = 201.533.
Replicate 3, 3 iterations, total sum of distances = 201.533.
Replicate 4, 3 iterations, total sum of distances = 201.533.
Replicate 5, 2 iterations, total sum of distances = 201.533.
Best total sum of distances = 201.533
```

默认情况下，软件使用 k-means++ 分别对每次重复进行初始化。

绘制簇和簇质心。

```
figure;
plot(X(idx==1,1),X(idx==1,2),'r.','MarkerSize',12)
hold on
plot(X(idx==2,1),X(idx==2,2),'b.','MarkerSize',12)
plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
legend('Cluster 1','Cluster 2','Centroids',...
       'Location','NW')
```

```
title 'Cluster Assignments and Centroids'
hold off
```



通过将 **idx** 传递给 **silhouette**，您可以确定簇之间的分离程度。

使用并行计算对数据进行聚类

对大型数据集进行聚类可能需要大量时间，尤其是在您使用在线更新（默认设置）时。如果您拥有 Parallel Computing Toolbox™ 许可证并设置了并行计算的选项，则 **kmeans** 将并行运行每个聚类任务（或副本）。而且，如果 **Replicates** > 1，则并行计算会减少收敛时间。

从高斯混合模型中随机生成一个大型数据集。

```
Mu = bsxfun(@times,ones(20,30),(1:20)'); % Gaussian mixture mean
rn30 = randn(30,30);
Sigma = rn30'*rn30; % Symmetric and positive-definite covariance
Mdl = gmdistribution(Mu,Sigma); % Define the Gaussian mixture distribution

rng(1); % For reproducibility
X = random(Mdl,10000);
```

Mdl 是一个 30 维 **gmdistribution** 模型，包含 20 个分量。**X** 是一个 10000×30 矩阵，其数据出自 **Mdl** 模型。

指定并行计算的选项。

```
stream = RandStream('mlfg6331_64');  % Random number stream
options = statset('UseParallel',1,'UseSubstreams',1,...
    'Streams',stream);
```

RandStream 的输入参数 **'mlfg6331_64'** 指定使用乘法滞后 Fibonacci 生成器算法。**options** 是一个结构体数组，其字段指定控制估算的选项。

对数据进行 k 均值聚类。指定数据中有 k = 20 个簇，并增加迭代次数。通常，目标函数包含局部最小值。指定 10 个副本以帮助找到更低的局部最小值。

```
tic; % Start stopwatch timer
[idx,C,sumd,D] = kmeans(X,20,'Options',options,'MaxIter',10000,...
    'Display','final','Replicates',10);
```

```
Starting parallel pool (parpool) using the 'local' profile ...
connected to 6 workers.
Replicate 5, 72 iterations, total sum of distances = 7.73161e+06.
Replicate 1, 64 iterations, total sum of distances = 7.72988e+06.
Replicate 3, 68 iterations, total sum of distances = 7.72576e+06.
Replicate 4, 84 iterations, total sum of distances = 7.72696e+06.
Replicate 6, 82 iterations, total sum of distances = 7.73006e+06.
Replicate 7, 40 iterations, total sum of distances = 7.73451e+06.
Replicate 2, 194 iterations, total sum of distances = 7.72953e+06.
Replicate 9, 105 iterations, total sum of distances = 7.72064e+06.
Replicate 10, 125 iterations, total sum of distances = 7.72816e+06.
Replicate 8, 70 iterations, total sum of distances = 7.73188e+06.
Best total sum of distances = 7.72064e+06
```

```
toc % Terminate stopwatch timer
```

```
Elapsed time is 61.915955 seconds.
```

命令行窗口指示有六个工作进程可用。您的系统中的工作进程数量可能会有所不同。命令行窗口显示迭代次数和每个副本的最终目标函数值。输出参数包含副本 9 的结果，因为它的总距离最低。

将新数据分配给现有簇并生成 C/C++ 代码

kmeans 执行 k 均值聚类以将数据划分为 k 个簇。当您有要进行聚类的新数据集时，可以使用 **kmeans** 创建包含现有数据和新数据的新簇。**kmeans** 函数支持 C/C++ 代码生成，因此您可以生成接受训练数据并返回聚类结果的代码，然后将代码部署到设备上。在此工作流中，您必须传递训练数据，训练数据有可能相当大。为了节省设备上的内存，您可以分别使用 **kmeans** 和 **pdist2** 来分离训练和预测。

使用 **kmeans** 在 MATLAB® 中创建簇，并在生成的代码中使用 **pdist2** 将新数据分配给现有簇。对于代码生成，定义接受簇质心位置和新数据集的入口函数，并返回最近邻簇的索引。然后，为入口函数生成代码。

生成 C/C++ 代码需要 MATLAB® Coder™。

执行 k 均值聚类

使用三种分布生成训练数据集。

```
rng('default') % For reproducibility
X = [randn(100,2)*0.75+ones(100,2);
```

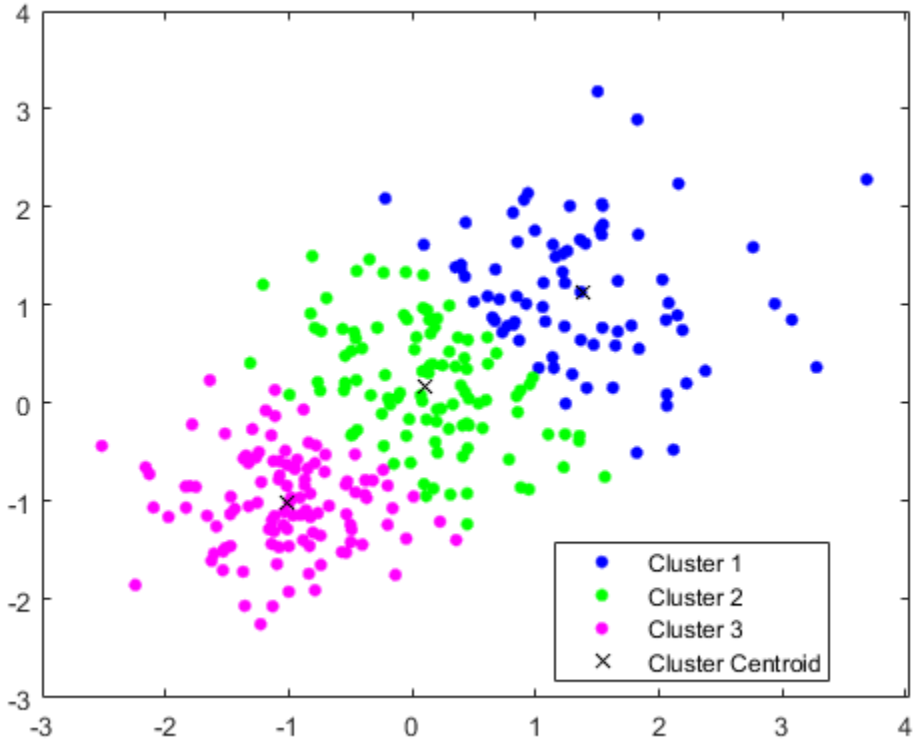
```
   randn(100,2)*0.5-ones(100,2);
   randn(100,2)*0.75];
```

使用 **kmeans** 将训练数据分成三个簇。

```
[idx,C] = kmeans(X,3);
```

绘制簇和簇质心。

```
figure
gscatter(X(:,1),X(:,2),idx,'bgm')
hold on
plot(C(:,1),C(:,2),'kx')
legend('Cluster 1','Cluster 2','Cluster 3','Cluster Centroid')
```



将新数据分配给现有簇

生成测试数据集。

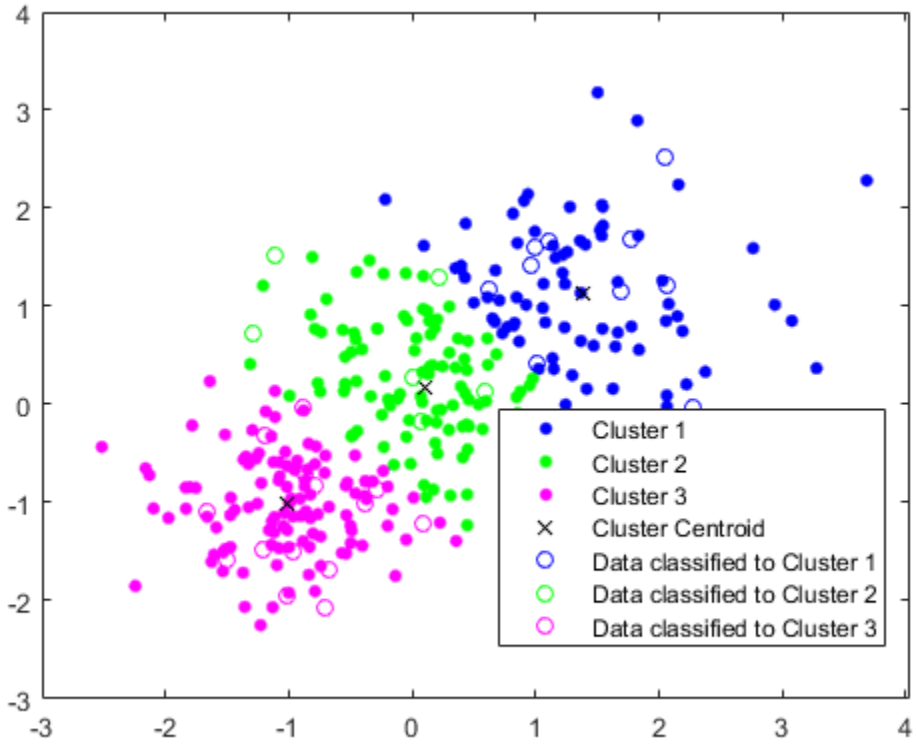
```
Xtest = [randn(10,2)*0.75+ones(10,2);
   randn(10,2)*0.5-ones(10,2);
   randn(10,2)*0.75];
```

使用现有簇对测试数据集进行分类。使用 **pdist2** 找到距离每个测试数据点最近的质心。

```
[~,idx_test] = pdist2(C,Xtest,'euclidean','Smallest',1);
```

使用 **idx_test** 和 **gscatter** 绘制测试数据并对测试数据加标签。

```
gscatter(Xtest(:,1),Xtest(:,2),idx_test,'bgm','ooo')
legend('Cluster 1','Cluster 2','Cluster 3','Cluster Centroid', ...
   'Data classified to Cluster 1','Data classified to Cluster 2', ...
   'Data classified to Cluster 3')
```



生成代码

生成将新数据分配给现有簇的 C 代码。请注意，生成 C/C++ 代码需要 MATLAB® Coder™。

定义名为 **findNearestCentroid** 的入口函数，该函数接受质心位置和新数据，然后使用 **pdist2** 找到最近的簇。

在入口函数的函数签名后面添加 **%#codegen** 编译器指令（即 pragma），以指示您要为此 MATLAB 算法生成代码。添加此指令指示 MATLAB 代码分析器帮助您诊断和修复在代码生成过程中可能导致错误的违规。

```
type findNearestCentroid % Display contents of findNearestCentroid.m
```

```
function idx = findNearestCentroid(C,X) %#codegen
[~,idx] = pdist2(C,X,'euclidean','Smallest',1); % Find the nearest centroid
```

注意：如果您点击位于此页右上角的按钮，并在 MATLAB® 中打开此示例，则 MATLAB® 将打开示例文件夹。该文件夹包括入口函数文件。

使用 **codegen** (MATLAB Coder) 生成代码。由于 C 和 C++ 是静态类型语言，因此必须在编译时确定入口函数中所有变量的属性。要指定 **findNearestCentroid** 的输入的数据类型和数组大小，请使用 **-args** 选项传递表示具有特定数据类型和数组大小的值集的 MATLAB 表达式。有关详细信息，请参阅 “Specify Variable-Size Arguments for Code Generation” 。

```
codegen findNearestCentroid -args {C,Xtest}

Code generation successful.
```

codegen 生成 MEX 函数 **findNearestCentroid_mex**，扩展名因平台而异。

验证生成的代码。

```
myIndx = findNearestCentroid(C,Xtest);
myIndex_mex = findNearestCentroid_mex(C,Xtest);
verifyMEX = isequal(idx_test,myIndx,myIndex_mex)

verifyMEX = logical
   1
```

isequal 返回逻辑值 1 (**true**)，这意味着所有输入都相等。这一比较结果确认 **pdist2** 函数、**findNearestCentroid** 函数和 MEX 函数均返回相同的索引。

您还可以使用 GPU Coder™ 生成优化的 CUDA® 代码。

```
cfg = coder.gpuConfig('mex');
codegen -config cfg findNearestCentroid -args {C,Xtest}
```

有关代码生成的详细信息，请参阅“General Code Generation Workflow”。有关 GPU Coder 的详细信息，请参阅“Get Started with GPU Coder” (GPU Coder) 和“Supported Functions” (GPU Coder)。

输入参数

X - 数据

数值矩阵

数据，指定为数值矩阵。**X** 的行对应于观测值，而列对应于变量。

如果 **X** 是数值向量，则 **kmeans** 将其视为 n×1 数据矩阵，而不管其方向如何。

该软件将 **X** 中的 **NaN** 视为缺失数据，并删除 **X** 中包含至少一个 **NaN** 的任何行。删除 **X** 的行会缩小样本大小。对于输出参数 **idx** 中的对应值，**kmeans** 函数返回 **NaN**。

数据类型：**single** | **double**

k - 簇的数量

正整数

数据中的簇数，指定为一个正整数。

数据类型：**single** | **double**

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例：**'Distance','cosine','Replicates',10,'Options',statset('UseParallel',1)** 指定使用余弦距离，以不同起始值重复 **10** 次聚类，并使用并行计算。

Display - 要显示的输出的级别
'off' （默认） | **'final'** | **'iter'**

要在命令行窗口中显示的输出的级别，指定为以逗号分隔的对组，其中包含 **'Display'** 和以下选项之一：

- **'final'** - 显示最终迭代的结果
- **'iter'** - 显示每次迭代的结果
- **'off'** - 不显示任何内容

示例： **'Display','final'**

Distance - 距离度量
'sqeuclidean' （默认） | **'cityblock'** | **'cosine'** | **'correlation'** | **'hamming'**

p 维空间中的距离度量，用于最小化距离和，指定为以逗号分隔的对组，其中包含 **'Distance'** 和 **'sqeuclidean'**、**'cityblock'**、**'cosine'**、**'correlation'** 或 **'hamming'**。

对于支持的各种距离度量，**kmeans** 分别以不同方式计算质心簇。下表总结了可用的距离度量。在公式中，x 是一个观测值（即 **X** 的一个行），c 是一个质心（一个行向量）。

距离度量	说明	公式
'sqeuclidean'	欧几里德距离的平方（默认值）。每个质心是该簇中各点的均值。	$d(x,c) = (x-c)(x-c)'$
'cityblock'	绝对差之和，即 L1 距离。每个质心是该簇中各点的按成分中位数。	$d(x,c) = \sum_{j=1}^{p} \left\lvert x_j - c_j \right\rvert$
'cosine'	1 减去点之间夹角的余弦值（视为向量）。每个质心是该簇中各点的均值（在将这些点归一化为单位欧几里德长度之后）。	$d(x,c) = 1 - \frac{xc'}{\sqrt{(xx')(cc')}}$
'correlation'	1 减去点之间的样本相关性（视为值序列）。每个质心是该簇中各点的按成分均值（在将这些点中心化并归一化为零均值和单位标准差后）。	其中 $d(x,c) = 1 - \frac{\left(x-\overrightarrow{\bar{x}}\right)\left(c-\overrightarrow{\bar{c}}\right)'}{\sqrt{\left(x-\overrightarrow{\bar{x}}\right)\left(x-\overrightarrow{\bar{x}}\right)'}\sqrt{\left(c-\overrightarrow{\bar{c}}\right)\left(c-\overrightarrow{\bar{c}}\right)'}},$ • $\overrightarrow{\bar{x}} = \frac{1}{p}\left(\sum_{j=1}^{p} x_j\right)\overrightarrow{1}_p$ • $\overrightarrow{\bar{c}} = \frac{1}{p}\left(\sum_{j=1}^{p} c_j\right)\overrightarrow{1}_p$ • $\overrightarrow{1}_p$ 是 p 个 1 组成的行向量。

距离度量	说明	公式
'hamming'	此度量仅适用于二类数据。 它是相异位所占的比例。每个质心是该簇中各点的按成分中位数。	$$d(x,y)=\frac{1}{p}\sum_{j=1}^{p}I\{x_j\neq y_j\},$$ 其中 I 是指示函数。

示例： **'Distance','cityblock'**

EmptyAction - 在簇丢失所有成员观测值时应采取的操作
'singleton' （默认） | **'error'** | **'drop'**

在簇丢失其所有成员观测值时应采取的操作，指定为以逗号分隔的对组，其中包含 **'EmptyAction'** 和以下选项之一。

值	说明
'error'	将空簇视为错误。
'drop'	删除所有变空的簇。**kmeans** 将 **C** 和 **D** 中对应的返回值设置为 **NaN**。
'singleton'	创建一个新簇，其中包含一个距其质心最远的点（默认值）。

示例： **'EmptyAction','error'**

MaxIter - 最大迭代次数
100 （默认） | 正整数

最大迭代次数，指定为以逗号分隔的对组，其中包含 **'MaxIter'** 和一个正整数。

示例： **'MaxIter',1000**

数据类型： **double** | **single**

OnlinePhase - 在线更新标志
'off' （默认） | **'on'**

在线更新标志，指定为以逗号分隔的对组，其中包含 **'OnlinePhase'** 和 **'off'** 或 **'on'**。

如果 **OnlinePhase** 是 **on**，则 **kmeans** 除了执行批量更新阶段之外，还会执行在线更新阶段。对于大型数据集来说，在线阶段可能很耗时，但可以保证解是距离标准的局部最小值。换句话说，软件会找到一个数据分区，在此分区中，将任一点移至不同簇都会增加总距离。

示例： **'OnlinePhase','on'**

Options - 控制迭代算法以最小化拟合标准的选项
[] （默认） | **statset** 返回的结构体数组

控制迭代算法以最小化拟合标准的选项，指定为以逗号分隔的对组，其中包含 **'Options'** 和 **statset** 返回的结构体数组。结构体数组中受支持的字段指定用于控制迭代算法的选项。

下表总结了支持的字段。请注意，支持的字段需要 Parallel Computing Toolbox。

字段	说明
'Streams'	**RandStream** 对象或此类对象的元胞数组。如果您没有指定 **Streams**，**kmeans** 将使用默认流。如果您指定 **Streams**，请使用单个对象，除非满足以下所有条件： • 您有一个打开的并行池。 • **UseParallel** 是 **true**。 • **UseSubstreams** 是 **false**。 在这种情况下，请使用与并行池大小相同的元胞数组。如果并行池未打开，则 **Streams** 必须提供一个随机数流。
'UseParallel'	• 如果为 **true** 且 **Replicates** > 1，则 **kmeans** 以并行方式对每次重复实现 k 均值算法。 • 如果未安装 Parallel Computing Toolbox，则计算以串行模式进行。默认值为 **false**，表示串行计算。
'UseSubstreams'	设置为 **true** 将以可重现的方式执行并行计算。默认值为 **false**。要以可重现方式执行计算，请将 **Streams** 设置为允许子流的类型：**'mlfg6331_64'** 或 **'mrg32k3a'**。

为确保更具预测性的结果，请在调用 **kmeans** 并设置 **'Options',statset('UseParallel',1)** 之前，使用 **parpool** 并显式创建并行池。

示例： **'Options',statset('UseParallel',1)**

数据类型： **struct**

Replicates - 使用新初始簇质心位置重复聚类的次数
1（默认）| 正整数

使用新初始簇质心位置重复聚类的次数，指定为由 **'Replicates'** 和整数组成的以逗号分隔的对组。**kmeans** 返回 **sumd** 最低的解。

您可以通过提供三维数组作为 **'Start'** 名称-值对组参数的值来隐式设置 **'Replicates'**。

示例： **'Replicates',5**

数据类型： **double** | **single**

Start - 一种选择初始簇质心位置的方法
'plus'（默认）| **'cluster'** | **'sample'** | **'uniform'** | 数值矩阵 | 数值数组

一种选择初始簇质心位置（即种子）的方法，指定为以逗号分隔的对组，其中包含 **'Start'** 以及 **'cluster'**、**'plus'**、**'sample'**、**'uniform'**、一个数值矩阵，或一个数值数组。下表总结了选择种子的可用选项。

值	说明
'cluster'	当 **X** 的一个随机 10% 子样本中的观测值数目多于 **k** 个时，对该子样本执行初步聚类阶段。此初步阶段本身是使用 **'sample'** 进行初始化的。 如果随机 10% 子样本中的观测值数目少于 **k** 个，则软件从 **X** 中随机选择 **k** 个观测值。
'plus'（默认值）	实现用于簇中心初始化的 k-means++ 算法 （第 33-130 页），依算法选择 **k** 个种子。
'sample'	从 **X** 中随机选择 **k** 个观测值。
'uniform'	从 **X** 范围内随机均匀选择 **k** 个点。Hamming 距离无效。
数值矩阵	包含质心起始位置的 **k**×p 矩阵。**Start** 的行对应于种子。软件根据 **Start** 的第一个维度推断 **k**，因此您可以为 **k** 传入 **[]**。
数值数组	包含质心起始位置的 **k**×p×r 数组。每页的行对应于种子。第三个维度调用聚类例程的重复。第 j 页包含副本 j 的种子集。软件根据第三个维度的大小推断副本的数量（由 **'Replicates'** 名称-值对组参数指定）。

示例： **'Start','sample'**

数据类型： **char** | **string** | **double** | **single**

输出参数

idx - 簇索引
数值列向量

簇索引，以数值列向量形式返回。**idx** 的行数与 **X** 的行数一样多，每行都表示对应观测值的簇分配。

C - 簇质心位置
数值矩阵

簇质心位置，以数值矩阵形式返回。**C** 是 **k**×p 矩阵，其中第 j 行是簇 j 的质心。

sumd - 簇内的点到质心距离的总和
数值列向量

簇内的点到质心距离的总和，以数值列向量形式返回。**sumd** 是 **k**×1 向量，其中元素 j 是簇 j 内的点到质心距离的总和。默认情况下，**kmeans** 使用欧几里德距离平方（请参阅 **'Distance'** 度量）。

D - 从每个点到每个质心的距离
数值矩阵

从每个点到每个质心的距离，以数值矩阵形式返回。**D** 是 n×**k** 矩阵，其中元素 (j,m) 是从观测值 j 到质心 m 的距离。默认情况下，**kmeans** 使用欧几里德距离平方（请参阅 **'Distance'** 度量）。

详细信息

k 均值聚类

k 均值聚类，即 Lloyd 算法 [2]，是一种迭代数据划分算法，它将 n 个观测值分配给由质心定义的 k 个簇之一，其中 k 是在算法开始之前选择的。

算法的执行如下：

1 选择 k 个初始簇中心（质心）。例如，随机选择 k 个观测值（通过使用 **'Start','sample'**）或使用 k-means ++ 算法 （第 33-130 页）进行簇中心初始化（默认值）。

2 计算所有观测值到每个质心的点到簇质心的距离。

3 有两种方法可以继续（由 **OnlinePhase** 指定）：

- 批量更新 - 将每个观测值分配给离质心最近的簇。
- 在线更新 - 只要将观测值重新分配给另一质心可减少簇内点到质心距离平方和的总和，就对该观测值执行此分配。

有关详细信息，请参阅“算法” （第 33-131 页）。

4 计算每个簇中观测值的平均值，以获得 k 个新质心位置。

5 重复步骤 2 到 4，直到簇分配不变，或达到最大迭代次数。

k-means++ 算法

k-means++ 算法使用启发式方法找到 k 均值聚类的质心种子。根据 Arthur 和 Vassilvitskii 的研究·[1]，k-means++ 改进了 Lloyd 算法的运行时间和最终解的质量。

k-means++ 算法按如下方式选择种子，假设簇数为 k。

1 从数据集 X 中随机均匀选择一个观测值。所选观测值是第一个质心，表示为 c_1。

2 计算从每个观测值到 c_1 的距离。将 c_j 和观测值 m 之间的距离表示为 $d(x_m, c_j)$。

3 从 X 中随机选择下一个质心 c_2，概率为

$$\frac{d^2(x_m, c_1)}{\sum_{j=1}^{n} d^2(x_j, c_1)}.$$

4 要选择中心 j，请执行以下操作：

a 计算从每个观测值到每个质心的距离，并将每个观测值分配给其最近的质心。

b 对于 m = 1、...、n 和 p = 1、...、j - 1，从 X 中随机选择质心 j，概率为

$$\frac{d^2(x_m, c_p)}{\sum_{\{h;\, x_h \in C_p\}} d^2(x_h, c_p)},$$

其中 C_p 是最接近质心 c_p 的所有观测值的集合，而 x_m 属于 C_p。

也就是说，选择每个后续中心时，其选择概率与它到已选最近中心的距离成比例。

5 重复步骤 4，直到选择了 k 个质心。

Arthur 和 Vassilvitskii·[1] 通过对几个簇方向的模拟研究证明，在计算簇内点到质心距离平方和时，k-means++ 相比 Lloyd 算法能更快地收敛至更低的总和。

算法

- **kmeans** 使用两阶段迭代算法来最小化点到质心的距离总和，该距离总和覆盖所有 **k** 个簇。

  1 第一阶段使用批量更新，其中每次迭代都包括一次性将点重新分配给其最近邻的簇质心，然后重新计算簇质心。此阶段有时候不会收敛至局部最小值解。也就是说，将任一点移至不同簇都会增加总距离的数据分区。尤其是对小数据集来说，更容易出现这种情况。批量更新阶段速度很快，但在该阶段逼近的解可能只能作为第二阶段的起点。

  2 第二阶段使用在线更新，即只要重新分配点可减少距离的总和，则进行重新分配，并在每次重新分配后重新计算簇质心。此阶段中的每次迭代都对所有点进行一次遍历。此阶段会收敛至一个局部最小值，尽管可能存在总距离更低的其他局部最小值。一般情况下，需要穷尽选择起始点才能求解全局最小值，但是，使用具有随机起始点的几个副本通常也会得到全局最小值解。
- 如果 **Replicates** = r > 1 且 **Start** 是 **plus**（默认值），则软件根据 k-means++ 算法 （第 33-130 页）选择 r 个可能不同的种子集。
- 如果您在 **Options** 中启用 **UseParallel** 选项且 **Replicates** > 1，则每个工作进程会以并行方式选择种子和进行聚类。

参考

[1] Arthur, David, and Sergi Vassilvitskii. “K-means++: The Advantages of Careful Seeding.” *SODA ‘07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2007, pp. 1027–1035.

[2] Lloyd, Stuart P. “Least Squares Quantization in PCM.” *IEEE Transactions on Information Theory*. Vol. 28, 1982, pp. 129–137.

[3] Seber, G. A. F. *Multivariate Observations*. Hoboken, NJ: John Wiley & Sons, Inc., 1984.

[4] Spath, H. *Cluster Dissection and Analysis: Theory, FORTRAN Programs, Examples*. Translated by J. Goldschmidt. New York: Halsted Press, 1985.

扩展功能

tall 数组
对行数太多而无法放入内存的数组进行计算。

使用说明和限制：

- 支持的语法包括：
  - **idx = kmeans(X,k)**
  - **[idx,C] = kmeans(X,k)**
  - **[idx,C,sumd] = kmeans(X,k)**
  - **[___] = kmeans(___,Name,Value)**
- 支持的名称-值对组参数以及用法差异（如有）如下：
  - **'Display'** - 默认值为 **'iter'**。

- **'MaxIter'**
- **'Options'** - 对于 **statset** 创建的结构体数组，只支持 **'TolFun'** 字段。**'TolFun'** 的默认值为 **1e-4**。**kmeans** 函数使用 **'TolFun'** 的值作为簇内的点到质心距离总和的终止容差。例如，您可以指定 **'Options',statset('TolFun',1e-8)**。
- **'Replicates'**
- **'Start'** - 仅支持 **'plus'**、**'sample'** 以及数值数组。

有关详细信息，请参阅“使用 tall 数组处理无法放入内存的数据”。

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

- 如果 **Start** 方法使用随机选择，初始质心簇位置可能与 MATLAB 不匹配。
- 如果 **X** 中的行数是固定的，代码生成不会删除包含 **NaN** 的 **X** 的行。
- **C** 中的簇质心位置可能与 MATLAB 中的顺序不同。在这种情况下，**idx** 中的簇索引也会相应地有所不同。
- 如果您提供 **Display**，其值必须为 **'off'**。
- 如果您提供 **Streams**，它必须为空，且 **UseSubstreams** 必须为 **false**。
- 当您将 **UseParallel** 选项设置为 **true** 时：
  - 即使 **Replicates** 是 **1**，某些计算也可以并行执行。对于大型数据集，当 **Replicates** 是 **1** 时，请考虑将 **UseParallel** 选项设置为 **true**。
  - **kmeans** 使用 **parfor** 创建在支持的共享内存多核平台上并行运行的循环。并行运行的循环可能比在单个线程上运行的循环更快。如果您的编译器不支持 Open Multiprocessing (OpenMP) 应用程序接口，或您禁用 OpenMP 库，则 MATLAB Coder 会将 **parfor** 循环视为 **for** 循环。要查找受支持的编译器，请参阅 **https://www.mathworks.com/support/compilers/current_release/**。
- 为了节省生成代码要部署到的目标设备上的内存，您可以分别使用 **kmeans** 和 **pdist2** 来分离训练和预测。使用 **kmeans** 在 MATLAB 中创建簇，并在生成的代码中使用 **pdist2** 将新数据分配给现有簇。对于代码生成，定义接受簇质心位置和新数据集的入口函数，并返回最近邻簇的索引。然后，为入口函数生成代码。有关示例，请参阅“将新数据分配给现有簇并生成 C/C++ 代码”（第 33-122 页）。
- 从 R2020a 开始，**kmeans** 在生成的独立 C/C++ 代码中返回整数类型 (**int32**) 索引，而不是双精度索引。因此，当您使用单精度输入时，该函数允许更严格的单精度支持。对于 MEX 代码生成，该函数仍返回双精度索引以匹配 MATLAB 行为。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “General Code Generation Workflow”。

自动并行支持
通过使用 Parallel Computing Toolbox™ 自动运行并行计算来加快代码执行。

要并行运行，请在调用此函数时指定 **'Options'** 名称-值参数，并使用 **statset** 将 options 结构体的 **'UseParallel'** 字段设置为 **true**。

例如：**'Options',statset('UseParallel',true)**

有关并行计算的详细信息，请参阅“Run MATLAB Functions with Automatic Parallel Support” (Parallel Computing Toolbox)。

GPU 数组

通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅

linkage | **clusterdata** | **silhouette** | **parpool** | **statset** | **gmdistribution** | **kmedoids**

主题

“Compare k-Means Clustering Solutions”
“Introduction to k-Means Clustering”

在 R2006a 之前推出

lhsnorm

来自正态分布的拉丁超立方样本

语法

```
X = lhsnorm(mu,sigma,n)
X = lhsnorm(mu,sigma,n,flag)
[X,Z] = lhsnorm(...)
```

说明

X = lhsnorm(mu,sigma,n) 返回 n×p 矩阵 **X**，其中包含大小为 **n** 的拉丁超立方样本，该样本来自具有均值向量 **mu** 和协方差矩阵 **sigma** 的 p 维多元正态分布。

X 类似于从多元正态分布抽取的随机样本，只是每列的边缘分布经过调整，使得其样本边缘分布接近其理论正态分布。

X = lhsnorm(mu,sigma,n,flag) 控制样本中的平滑处理量。如果 **flag** 是 **'off'**，则每列在概率等级上都有等距的点。换句话说，每列均为值 **G(0.5/n), G(1.5/n), ..., G(1-0.5/n)** 的排列，其中 **G** 是该列边缘分布的逆正态累积分布。如果 **flag** 是 **'on'**（默认值），则每列都有在概率等级上均匀分布的点。例如，您可以使用在区间 **(0/n,1/n)** 上具有均匀分布的值来代替 **0.5/n**。

[X,Z] = lhsnorm(...) 还返回 **Z**，即调整边距以获得 **X** 之前的原始多元正态样本。

参考文献

[1] Stein, M. “Large sample properties of simulations using latin hypercube sampling.” *Technometrics*. Vol. 29, No. 2, 1987, pp. 143–151. Correction, Vol. 32, p. 367.

另请参阅

lhsdesign | **mvnrnd**

在 R2006a 之前推出

nlinfit

非线性回归

语法

```
beta = nlinfit(X,Y,modelfun,beta0)
beta = nlinfit(X,Y,modelfun,beta0,options)
beta = nlinfit(___,Name,Value)
[beta,R,J,CovB,MSE,ErrorModelInfo] = nlinfit(___)
```

说明

beta = nlinfit(X,Y,modelfun,beta0) 使用 **modelfun** 指定的模型，返回一个向量，其中包含 **Y** 中的响应对 **X** 中的预测变量的非线性回归的估计系数。它使用迭代最小二乘估计来估计系数，初始值由 **beta0** 指定。

beta = nlinfit(X,Y,modelfun,beta0,options) 使用结构体 **options** 中的算法控制参数来拟合非线性回归。您可以返回上述语法中的任何输出参数。

beta = nlinfit(___,Name,Value) 使用由一个或多个名称-值对组参数指定的附加选项。例如，您可以指定观测值权重或非常量误差模型。您可以使用上述语法中的任何输入参数。

[beta,R,J,CovB,MSE,ErrorModelInfo] = nlinfit(___) 还返回残差 **R**、**modelfun** 的 Jacobian 矩阵 **J**、估计系数的估计方差-协方差矩阵 **CovB**、误差项的方差估计 **MSE** 以及包含误差模型细节的结构体 **ErrorModelInfo**。

示例

使用默认选项的非线性回归模型

加载样本数据。

```
S = load('reaction');
X = S.reactants;
y = S.rate;
beta0 = S.beta;
```

使用 **beta0** 中的初始值对速率数据进行 Hougen-Watson 模型拟合。

```
beta = nlinfit(X,y,@hougen,beta0)
```

```
beta = 5×1

    1.2526
    0.0628
    0.0400
    0.1124
    1.1914
```

使用稳健选项的非线性回归

从非线性回归模型 $y = b_1 + b_2 \cdot exp\{-b_3 x\} + \epsilon$ 生成样本数据，其中 b_1、b_2、b_3 是系数，误差项呈正态分布，均值为 0，标准差为 0.1。

```
modelfun = @(b,x)(b(1)+b(2)*exp(-b(3)*x));

rng('default') % for reproducibility
b = [1;3;2];
x = exprnd(2,100,1);
y = modelfun(b,x) + normrnd(0,0.1,100,1);
```

设置稳健拟合选项。

```
opts = statset('nlinfit');
opts.RobustWgtFun = 'bisquare';
```

使用稳健拟合选项拟合非线性模型。

```
beta0 = [2;2;2];
beta = nlinfit(x,y,modelfun,beta0,opts)
```

```
beta = 3×1

   1.0041
   3.0997
   2.1483
```

使用观测值权重的非线性回归

加载样本数据。

```
S = load('reaction');
X = S.reactants;
y = S.rate;
beta0 = S.beta;
```

指定一个包含已知观测值权重的向量。

```
W = [8 2 1 6 12 9 12 10 10 12 2 10 8]';
```

使用指定的观测值权重对速率数据进行 Hougen-Watson 模型拟合。

```
[beta,R,J,CovB] = nlinfit(X,y,@hougen,beta0,'Weights',W);
beta
```

```
beta = 5×1

   2.2068
   0.1077
   0.0766
   0.1818
```

```
    0.6516
```

显示系数标准误差。

```
sqrt(diag(CovB))
```

```
ans = 5×1

    2.5721
    0.1251
    0.0950
    0.2043
    0.7735
```

使用权重函数句柄的非线性回归

加载样本数据。

```
S = load('reaction');
X = S.reactants;
y = S.rate;
beta0 = S.beta;
```

指定一个指向观测值权重的函数句柄。该函数接受模型拟合值作为输入，并返回权重向量。

```
 a = 1; b = 1;
 weights = @(yhat) 1./((a + b*abs(yhat)).^2);
```

使用指定的观测值权重函数对速率数据进行 Hougen-Watson 模型拟合。

```
[beta,R,J,CovB] = nlinfit(X,y,@hougen,beta0,'Weights',weights);
beta
```

```
beta = 5×1

    0.8308
    0.0409
    0.0251
    0.0801
    1.8261
```

显示系数标准误差。

```
sqrt(diag(CovB))
```

```
ans = 5×1

    0.5822
    0.0297
    0.0197
    0.0578
    1.2810
```

使用非常量误差模型的非线性回归

加载样本数据。

```
S = load('reaction');
X = S.reactants;
y = S.rate;
beta0 = S.beta;
```

使用合并的误差模型对速率数据进行 Hougen-Watson 模型拟合。

```
[beta,R,J,CovB,MSE,ErrorModelInfo] = nlinfit(X,y,@hougen,beta0,'ErrorModel','combined');
beta
```

```
beta = 5×1

   1.2526
   0.0628
   0.0400
   0.1124
   1.1914
```

显示误差模型信息。

```
ErrorModelInfo
```

```
ErrorModelInfo = struct with fields:
          ErrorModel: 'combined'
     ErrorParameters: [0.1517 5.6783e-08]
      ErrorVariance: [function_handle]
                MSE: 1.6245
       ScheffeSimPred: 6
       WeightFunction: 0
        FixedWeights: 0
  RobustWeightFunction: 0
```

输入参数

X - 预测变量
矩阵

非线性回归函数的预测变量，指定为矩阵。通常，**X** 是预测变量（自变量）的设计矩阵，**Y** 中每个值对应该矩阵中的一个行，每个预测变量对应该矩阵中的一个列。但 **X** 其实可以是 **modelfun** 能接受的任何数组。

数据类型：`single` | `double`

Y - 响应值
向量

用于拟合非线性回归函数的响应值（因变量），指定为与 **X** 行数相同的向量。

数据类型： **single** | **double**

modelfun - 非线性回归模型函数
函数句柄

非线性回归模型函数，指定为函数句柄。**modelfun** 必须按顺序接受两个输入参数 - 系数向量和数组 **X**，并返回由拟合响应值组成的向量。

例如，要指定 **hougen** 非线性回归函数，请使用函数句柄 **@hougen**。

数据类型： **function_handle**

beta0 - 初始系数值
向量

最小二乘估计算法的初始系数值，指定为向量。

注意 不理想的起始值可能会导致具有较大残差的解。

数据类型： **single** | **double**

options - 估计算法选项
使用 **statset** 创建的结构体

估计算法选项，指定为您使用 **statset** 创建的结构体。以下 **statset** 参数适用于 **nlinfit**。

DerivStep - 有限差分梯度的相对差
eps^(1/3) （默认） | 正标量值 | 向量

有限差分梯度计算的相对差，指定为正标量值，或与 **beta** 大小相同的向量。使用向量为每个系数指定不同的相对差。

Display - 输出显示级别
'off' （默认） | **'iter'** | **'final'**

估算期间的输出显示级别，指定为 **'off'**、**'iter'** 或 **'final'** 之一。如果您指定 **'iter'**，每次迭代都会显示输出。如果指定 **'final'**，将在最终迭代后显示输出。

FunValCheck - 是否检查无效值的指示符
'on' （默认） | **'off'**

是否检查目标函数中的无效值（如 **NaN** 或 **Inf**）的指示符，指定为 **'on'** 或 **'off'**。

MaxIter - 最大迭代次数
100 （默认） | 正整数

估计算法的最大迭代次数，指定为正整数。迭代继续进行，直到估计值在收敛容差内，或达到 **MaxIter** 指定的最大迭代次数。

RobustWgtFun - 权重函数
字符向量 | 字符串标量 | 函数句柄 | **[]**

稳健拟合的权重函数，指定为有效的字符向量、字符串标量或函数句柄。

注意 使用观测值权重 **W** 时，**RobustWgtFun** 的值必须为 **[]**。

下表说明了可能的字符向量和字符串标量。用 r 表示归一化残差，w 表示稳健权重。如果表达式 x 的计算结果为 true，则指示函数 [x] 等于 1，否则等于 0。

权重函数	方程	默认调整常量
''（默认值）	没有稳健拟合	—
'andrews'	$w = I[\lvert r\rvert < \pi] \times \sin(r)/r$	1.339
'bisquare'	$w = I[\lvert r\rvert < 1] \times \left(1 - r^2\right)^2$	4.685
'cauchy'	$w = 1/\left(1 + r^2\right)$	2.385
'fair'	$w = 1/(1 + \lvert r\rvert)$	1.400
'huber'	$w = 1/\max(1, \lvert r\rvert)$	1.345
'logistic'	$w = \tanh(r)/r$	1.205
'talwar'	$w = I[\lvert r\rvert < 1]$	2.795
'welsch'	$w = \exp\left\{-r^2\right\}$	2.985

您也可以指定函数句柄，该句柄接受归一化残差向量作为输入，并返回稳健权重向量作为输出。如果使用函数句柄，必须提供 **Tune** 常量。

Tune - 调整常量
正标量值

稳健拟合的调整常量，指定为正标量值。调整常量用于在应用稳健权重函数之前归一化残差。默认调整常量取决于 **RobustWgtFun** 指定的函数。

如果使用函数句柄指定 **RobustWgtFun**，则必须为 **Tune** 指定值。

TolFun - 基于残差平方和的终止容差
1e-8（默认）| 正标量值

残差平方和的终止容差，指定为正标量值。迭代继续进行，直到估计值在收敛容差内，或达到 **MaxIter** 指定的最大迭代次数。

TolX - 基于估计系数的终止容差
1e-8（默认）| 正标量值

基于估计系数 **beta** 的终止容差，指定为正标量值。迭代继续进行，直到估计值在收敛容差内，或达到 **MaxIter** 指定的最大迭代次数。

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例：**'ErrorModel','proportional','ErrorParameters',0.5** 指定比例误差模型，误差参数估计的初始值为 0.5

ErrorModel - 误差项的形式
'constant' （默认） | 'proportional' | 'combined'

误差项的形式，指定为以逗号分隔的对组，其中包含 'ErrorModel'，以及指示误差模型的 'constant'、'proportional' 或 'combined'。每个模型在定义误差时，都用到了一个标准的零均值单位方差变量 e，以及下列独立部分：函数值 f，还有 a 和 b 这两个参数中的一个或两个。

'constant'（默认值）	$y = f + ae$
'proportional'	$y = f + bfe$
'combined'	$y = f + (a + b\|f\|)e$

使用 Weights 时，唯一允许的误差模型是 'constant'。

注意 使用 'constant' 以外的误差模型时，options.RobustWgtFun 的值必须为 []。

ErrorParameters - 误差模型参数的初始估计值
1 或 [1,1] （默认） | 标量值 | 二元素向量

所选 ErrorModel 中误差模型参数的初始估计值，指定为以逗号分隔的对组，其中包含 'ErrorParameters' 和一个标量值或一个二元素向量。

误差模型	参数	默认值
'constant'	a	1
'proportional'	b	1
'combined'	a, b	[1,1]

例如，如果 'ErrorModel' 的值为 'combined'，您可以为 a 指定起始值 1，为 b 指定起始值 2，如下所示。

示例： 'ErrorParameters',[1,2]

使用 Weights 时，您只能使用 'constant' 误差模型。

注意 使用 'constant' 以外的误差模型时，options.RobustWgtFun 的值必须为 []。

数据类型： double | single

Weights - 观测值权重
向量 | 函数句柄

观测值权重，指定为以逗号分隔的对组，其中包含 'Weights' 和一个正实数权重向量或一个函数句柄。如果您希望减少某些观测值对拟合模型的影响，则可使用观测值权重来降低其权重。

- 如果 W 是向量，则它的大小必须与 Y 相同。
- 如果 W 是函数句柄，则它必须接受预测响应值向量作为输入，并返回正实数权重向量作为输出。

注意 当您使用观测值权重时，options.RobustWgtFun 的值必须为 []。

数据类型： **double** | **single** | **function_handle**

输出参数

beta - 估计的回归系数
向量

估计的回归系数，以向量形式返回。**beta** 中的元素数等于 **beta0** 中的元素数。

用 $f(X_i, \mathbf{b})$ 表示 **modelfun** 指定的非线性函数，其中 $\mathbf{x}_i$ 是观测值 i 的预测变量，i = 1,...,N，**b** 是回归系数。**beta** 中返回的系数向量将以下加权最小二乘方程最小化：

$$\sum_{i=1}^{N} w_i[y_i - f(\mathbf{x}_i, \mathbf{b})]^2.$$

对于未加权的非线性回归，所有权重项都等于 1。

R - 残差
向量

拟合模型的残差，以向量形式返回。

- 如果使用名称-值对组参数 **Weights** 指定观测值权重，则 **R** 包含加权残差 （第 33-143 页）。
- 如果使用名称-值对组参数 **ErrorModel** 指定 **'constant'** 以外的误差模型，则不能再将 **R** 解释为模型拟合残差。

J - Jacobian 矩阵
矩阵

非线性回归模型 **modelfun** 的 Jacobian 矩阵，以 N×p 矩阵形式返回，其中 N 是观测值数目，p 是估计系数的数目。

- 如果使用名称-值对组参数 **Weights** 指定观测值权重，则 **J** 是加权模型函数 Jacobian 矩阵 （第 33-143 页）。
- 如果使用名称-值对组参数 **ErrorModel** 指定 **'constant'** 以外的误差模型，则不能再将 **J** 解释为模型函数 Jacobian 矩阵。

CovB - 估计的方差-协方差矩阵
矩阵

在拟合系数 **beta** 的基础上估计的方差-协方差矩阵，以 p×p 矩阵形式返回，其中 p 是估计系数的数目。如果模型 Jacobian 矩阵 **J** 具有满列秩，则 **CovB = inv(J'*J)*MSE**，其中 **MSE** 是均方误差。

MSE - 均方误差
标量值

拟合模型的均方误差 (MSE)，以标量值形式返回。MSE 是误差项的方差的估计值。如果模型 Jacobian 矩阵 **J** 具有满列秩，则 **MSE = (R'*R)/(N-p)**，其中 **N** 是观测值数目，**p** 是估计系数的数目。

ErrorModelInfo - 关于误差模型拟合的信息
结构体

关于误差模型拟合的信息，以具有以下字段的结构体形式返回：

ErrorModel	所选误差模型
ErrorParameters	估计误差参数
ErrorVariance	函数句柄，它接受 N×p 矩阵 **X** 并使用估计误差模型返回 N×1 误差方差向量
MSE	均方误差
ScheffeSimPred	使用估计误差模型时用于联合预测区间的 Scheffé 参数
WeightFunction	如果您以前在 **nlinfit** 中使用过自定义权重函数，则逻辑值为 **true**
FixedWeights	如果您以前在 **nlinfit** 中使用过固定权重，则逻辑值为 **true**
RobustWeightFunction	如果您以前在 **nlinfit** 中使用过稳健拟合，则逻辑值为 **true**

详细信息

加权残差

加权残差是残差与对应观测值权重的平方根相乘的结果。

给定估计回归系数 **b**, 时，观测值 i 的残差是

$$r_i = y_i - f(\mathbf{x}_i, \mathbf{b}),$$

其中 y_i 是观测到的响应，$f(\mathbf{x}_i, \mathbf{b})$ 是预测变量 $\mathbf{x}_i$. 处的拟合响应

当您用权重 w_i（i = 1、...、N）拟合加权非线性回归时，**nlinfit** 返回加权残差，

$$r_i^* = \sqrt{w_i}(y_i - f(\mathbf{x}_i, \mathbf{b})) .$$

加权模型函数 Jacobian 矩阵

加权模型函数 Jacobian 矩阵是非线性模型 Jacobian 矩阵与观测值权重矩阵的平方根相乘的结果。

给定估计回归系数 **b**,，非线性函数 $f(\mathbf{x}_i, \mathbf{b})$ 的估计模型 Jacobian 矩阵 **J**, 具有元素

$$\mathbf{J}_{ij} = \frac{\partial f(\mathbf{x}_i, \mathbf{b})}{\partial b_j},$$

其中 b_j 是 **b** . 的第 j 个元素

当您用对角权重矩阵 **W**,**nlinfit** 拟合加权非线性回归时，返回加权 Jacobian 矩阵，

$$\mathbf{J}^* = \mathbf{W}^{1/2}\mathbf{J} .$$

提示

- 要基于预测生成误差估计，请使用可选输出参数 **R**、**J**、**CovB** 或 **MSE** 作为 **nlpredci** 的输入。
- 要基于估计系数 **beta** 生成误差估计，请使用可选输出参数 **R**、**J**、**CovB** 或 **MSE** 作为 **nlparci** 的输入。
- 如果您使用稳健拟合选项 **RobustWgtFun**，则必须使用 **CovB**（可能还需要 **MSE**）作为 **nlpredci** 或 **nlparci** 的输入，以确保在计算置信区间时考虑稳健拟合。

算法

- **nlinfit** 将 **Y** 或 **modelfun(beta0,X)** 中的 **NaN** 值视为缺失数据，并忽略对应的观测值。
- 对于非稳健估计，**nlinfit** 使用 Levenberg-Marquardt 非线性最小二乘算法 [1]。
- 对于稳健估计，**nlinfit** 使用“Iteratively Reweighted Least Squares”的算法（[2]、[3]）。在每次迭代中，基于前一次迭代中每个观测值的残差重新计算稳健权重。这些权重会对离群值进行降权，从而降低它们对拟合的影响。迭代一直持续到权重收敛为止。
- 为观测值权重指定函数句柄时，权重取决于拟合模型。在这种情况下，**nlinfit** 使用迭代广义最小二乘算法来拟合非线性回归模型。

参考

[1] Seber, G. A. F., and C. J. Wild. *Nonlinear Regression*. Hoboken, NJ: Wiley-Interscience, 2003.

[2] DuMouchel, W. H., and F. L. O'Brien. “Integrating a Robust Option into a Multiple Regression Computing Environment.” *Computer Science and Statistics: Proceedings of the 21st Symposium on the Interface*. Alexandria, VA: American Statistical Association, 1989.

[3] Holland, P. W., and R. E. Welsch. “Robust Regression Using Iteratively Reweighted Least-Squares.” *Communications in Statistics: Theory and Methods*, *A6*, 1977, pp. 813–827.

另请参阅

fitnlm | **nlparci** | **nlpredci** | **nlintool**

主题

“Nonlinear Regression”

在 R2006a 之前推出

normcdf

正态累积分布函数

语法

```
p = normcdf(x)
p = normcdf(x,mu)
p = normcdf(x,mu,sigma)

[p,pLo,pUp] = normcdf(x,mu,sigma,pCov)
[p,pLo,pUp] = normcdf(x,mu,sigma,pCov,alpha)

___ = normcdf( ___ ,'upper')
```

说明

p = normcdf(x) 返回标准正态分布的累积分布函数 (cdf)，在 **x** 中的值处计算函数值。

p = normcdf(x,mu) 返回具有均值 **mu** 和单位标准差的正态分布的 cdf，在 **x** 中的值处计算函数值。

p = normcdf(x,mu,sigma) 返回具有均值 **mu** 和标准差 **sigma** 的正态分布的 cdf，在 **x** 中的值处计算函数值。

当 **mu** 和 **sigma** 为估计值时，**[p,pLo,pUp] = normcdf(x,mu,sigma,pCov)** 还返回 **p** 的 95% 置信边界 **[pLo,pUp]**。**pCov** 是估计参数的协方差矩阵。

[p,pLo,pUp] = normcdf(x,mu,sigma,pCov,alpha) 指定置信区间 **[pLo,pUp]** 的置信水平为 **100(1–alpha)%**。

___ = normcdf(___ ,'upper') 使用可更精确计算极值上尾概率的算法返回在 **x** 中的值处计算的 cdf 的补码。**'upper'** 可以跟在上述语法中的任何输入参数之后。

示例

标准正态分布 cdf

计算标准正态分布中的观测值落在区间 **[–1 1]** 上的概率。

```
p = normcdf([-1 1]);
p(2)-p(1)
```

```
ans = 0.6827
```

一个正态分布中大约 68% 的观测值落在均值为 0 的一个标准差内。

正态分布 cdf

计算均值为 **mu** 和标准差为 **sigma** 的正态分布在 **x** 中的值处计算的 cdf 值。

```
x = [-2,-1,0,1,2];
mu = 2;
sigma = 1;
p = normcdf(x,mu,sigma)
```

```
p = 1×5

   0.0000    0.0013    0.0228    0.1587    0.5000
```

计算具有不同均值参数的各种正态分布在零处计算的 cdf 值。

```
mu = [-2,-1,0,1,2];
sigma = 1;
p = normcdf(0,mu,sigma)
```

```
p = 1×5

   0.9772    0.8413    0.5000    0.1587    0.0228
```

正态 cdf 值的置信区间

计算正态分布参数的最大似然估计 (MLE)，然后计算对应 cdf 值的置信区间。

从均值为 5、标准差为 2 的正态分布中生成 1000 个正态随机数。

```
rng('default') % For reproducibility
n = 1000; % Number of samples
x = normrnd(5,2,n,1);
```

使用 **mle** 计算分布参数（均值和标准差）的 MLE。

```
phat = mle(x)
```

```
phat = 1×2

   4.9347    1.9969
```

```
muHat = phat(1);
sigmaHat = phat(2);
```

使用 **normlike** 估计分布参数的协方差。如果您传递 MLE 和用于估计 MLE 的样本，则函数 **normlike** 返回渐近协方差矩阵的逼近。

```
[~,pCov] = normlike([muHat,sigmaHat],x)
```

```
pCov = 2×2

   0.0040   -0.0000
  -0.0000    0.0020
```

计算在零处的 cdf 值及其 95% 置信区间。

```
[p,pLo,pUp] = normcdf(0,muHat,sigmaHat,pCov)
```

```
p = 0.0067

pLo = 0.0047

pUp = 0.0095
```

p 是使用参数为 **muHat** 和 **sigmaHat** 的正态分布的 cdf 值。考虑到使用 **pCov** 的 **muHat** 和 **sigmaHat** 的不确定性，区间 **[pLo,pUp]** 是在 0 处计算的 cdf 的 95% 置信区间。95% 置信区间意味着 **[pLo,pUp]** 包含真实 cdf 值的概率为 0.95。

互补 cdf（尾分布）

确定标准正态分布中的观测值落入 **[10,Inf]** 区间的概率。

```
p1 = 1 - normcdf(10)

p1 = 0
```

normcdf(10) 接近 1，因此 **p1** 变为 0。指定 **'upper'** 以便 **normcdf** 更准确地计算极值上尾概率。

```
p2 = normcdf(10,'upper')

p2 = 7.6199e-24
```

您还可以使用 **'upper'** 计算右尾 p 值。

使用函数句柄检验正态分布

使用概率分布函数 **normcdf** 作为卡方拟合优度检验中的函数句柄 (**chi2gof**)。

检验原假设，即输入向量 **x** 中的样本数据来自正态分布，其参数 μ 和 σ 分别等于样本数据的均值 (**mean**) 和标准差 (**std**)。

```
rng('default') % For reproducibility
x = normrnd(50,5,100,1);
h = chi2gof(x,'cdf',{@normcdf,mean(x),std(x)})

h = 0
```

返回的结果 **h = 0** 表明 **chi2gof** 在默认的 5% 显著性水平上未拒绝原假设。

输入参数

x - 用于计算 cdf 的值

标量值 | 标量值组成的数组

用于计算 cdf 的值，指定为标量值或标量值组成的数组。

如果您指定 **pCov** 来计算置信区间 **[pLo,pUp]**，则 **x** 必须为标量值。

要在多个值处计算 cdf，请使用数组指定 **x**。要计算多个分布的 cdf，请使用数组指定 **mu** 和 **sigma**。如果输入参数 **x**、**mu** 和 **sigma** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**normcdf**

将每个标量输入扩展为与数组输入大小相同的常量数组。**p** 中的每个元素是由 **mu** 和 **sigma** 中对应元素指定的分布的 cdf 值，其值在 **x** 中对应元素处进行计算。

示例： **[-1,0,3,4]**

数据类型： **single** | **double**

mu - 均值
0 （默认） | 标量值 | 标量值组成的数组

正态分布的均值，指定为标量值或由标量值组成的数组。

如果您指定 **pCov** 来计算置信区间 **[pLo,pUp]**，则 **mu** 必须为标量值。

要在多个值处计算 cdf，请使用数组指定 **x**。要计算多个分布的 cdf，请使用数组指定 **mu** 和 **sigma**。如果输入参数 **x**、**mu** 和 **sigma** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**normcdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。**p** 中的每个元素是由 **mu** 和 **sigma** 中对应元素指定的分布的 cdf 值，其值在 **x** 中对应元素处进行计算。

示例： **[0 1 2; 0 1 2]**

数据类型： **single** | **double**

sigma - 标准差
1 （默认） | 非负标量值 | 非负标量值组成的数组

正态分布的标准差，指定为非负标量值或由非负标量值组成的数组。

如果 **sigma** 为零，则输出 **p** 为 0 或 1。如果 **x** 小于 **mu**，则 **p** 为 0，否则为 1。

如果您指定 **pCov** 来计算置信区间 **[pLo,pUp]**，则 **sigma** 必须为标量值。

要在多个值处计算 cdf，请使用数组指定 **x**。要计算多个分布的 cdf，请使用数组指定 **mu** 和 **sigma**。如果输入参数 **x**、**mu** 和 **sigma** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**normcdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。**p** 中的每个元素是由 **mu** 和 **sigma** 中对应元素指定的分布的 cdf 值，其值在 **x** 中对应元素处进行计算。

示例： **[1 1 1; 2 2 2]**

数据类型： **single** | **double**

pCov - 估计值的协方差
2×2 数值矩阵

mu 和 **sigma** 估计值的协方差，指定为 2×2 矩阵。

如果指定 **pCov** 来计算置信区间 **[pLo,pUp]**，则 **x**、**mu** 和 **sigma** 必须为标量值。

您可以使用 **mle** 来估计 **mu** 和 **sigma**，并使用 **normlike** 来估计 **mu** 和 **sigma** 的协方差。有关示例，请参阅“正态 cdf 值的置信区间” （第 33-146 页）。

数据类型： **single** | **double**

alpha - 显著性水平
0.05 （默认） | (0,1) 范围内的标量

置信区间的显著性水平，指定为范围 (0,1) 内的标量。置信水平是 **100(1–alpha)%**，其中 **alpha** 是置信区间不包含 true 值的概率。

示例： **0.01**

数据类型： **single** | **double**

输出参数

p - cdf 值
标量值 | 标量值组成的数组

在 **x** 中的值处计算的 cdf 值，以标量值或标量值数组的形式返回。在经过任何必要的标量扩展后，**p** 的大小与 **x**、**mu** 和 **sigma** 相同。**p** 中的每个元素是由 **mu** 和 **sigma** 中对应元素指定的分布的 cdf 值，其值在 **x** 中对应元素处进行计算。

pLo - p 的置信边界下限
标量值 | 标量值组成的数组

p 的置信边界下限，以标量值或标量值数组的形式返回。**pLo** 的大小与 **p** 相同。

pUp - p 的置信边界上限
标量值 | 标量值组成的数组

p 的置信边界上限，以标量值或标量值数组的形式返回。**pUp** 的大小与 **p** 相同。

详细信息

正态分布

正态分布是双参数曲线族。第一个参数 μ 是均值。第二个参数 σ 是标准差。

标准正态分布具有零均值和单位标准差。

正态累积分布函数 (cdf) 表示为

$$p = F(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-(t-\mu)^2}{2\sigma^2}} dt, \quad \text{for } x \in \mathbb{R}.$$

p 是参数为 μ 和 σ 的正态分布中的一个观测值落入 (-∞,x] 区间的概率。

算法

- **normcdf** 函数使用补余误差函数 **erfc**。**normcdf** 和 **erfc** 之间的关系是

  $$\text{normcdf}(x) = \frac{1}{2}\text{erfc}\left(-\frac{x}{\sqrt{2}}\right).$$

  补余误差函数 **erfc(x)** 定义为

  $$\text{erfc}(x) = 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2} dt.$$

- **normcdf** 函数使用 delta 方法计算 **p** 的置信边界。**normcdf(x,mu,sigma)** 等效于 **normcdf((x–mu)/sigma,0,1)**。因此，**normcdf** 函数通过 delta 方法使用 **mu** 和 **sigma** 的协方差矩阵来估计 **(x–mu)/sigma** 的方差，并使用此方差的估计值来计算 **(x–mu)/sigma** 的置信边界。然后，该函数将边界

转换为 **p** 的尺度。当您从大样本中估计 **mu**、**sigma** 和 **pCov** 时，计算的界限会给出所需的大致置信水平。

替代功能

- **normcdf** 是正态分布特有的函数。Statistics and Machine Learning Toolbox 还提供泛型函数 **cdf**，它支持各种概率分布。要使用 **cdf**，请创建一个 **NormalDistribution** 概率分布对象，并将该对象作为输入参数传递，或指定概率分布名称及其参数。请注意，分布特有的函数 **normcdf** 比泛型函数 **cdf** 的执行速度要快。
- 使用 **Probability Distribution Function** App 为概率分布创建累积分布函数 (cdf) 或概率密度函数 (pdf) 的交互图。

参考

[1] Abramowitz, M., and I. A. Stegun. *Handbook of Mathematical Functions*. New York: Dover, 1964.

[2] Evans, M., N. Hastings, and B. Peacock. *Statistical Distributions*. 2nd ed., Hoboken, NJ: John Wiley & Sons, Inc., 1993.

扩展功能

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅

cdf | **normpdf** | **norminv** | **normfit** | **normlike** | **NormalDistribution** | **erfc**

主题
“正态分布” (第 B-2 页)

在 R2006a 之前推出

normpdf

正态概率密度函数

语法

```
y = normpdf(x)
y = normpdf(x,mu)
y = normpdf(x,mu,sigma)
```

说明

y = normpdf(x) 返回标准正态分布的概率密度函数 (pdf)，在 **x** 中的值处计算函数值。

y = normpdf(x,mu) 返回具有均值 **mu** 和单位标准差的正态分布的 pdf，在 **x** 中的值处计算函数值。

y = normpdf(x,mu,sigma) 返回具有均值 **mu** 和标准差 **sigma** 的正态分布的 pdf，在 **x** 中的值处计算函数值。

示例

标准正态分布 pdf

计算标准正态分布在 **x** 中的值处的 pdf 值。

```
x = [-2,-1,0,1,2];
y = normpdf(x)
```

```
y = 1×5

   0.0540    0.2420    0.3989    0.2420    0.0540
```

正态分布 pdf

计算均值为 **mu**、标准差为 **sigma** 的正态分布在 **x** 中的值处计算的 pdf 值。

```
x = [-2,-1,0,1,2];
mu = 2;
sigma = 1;
y = normpdf(x,mu,sigma)
```

```
y = 1×5

   0.0001    0.0044    0.0540    0.2420    0.3989
```

计算具有不同均值参数的各种正态分布在零处计算的 pdf 值。

```
mu = [-2,-1,0,1,2];
sigma = 1;
y = normpdf(0,mu,sigma)
```

```
y = 1×5

   0.0540    0.2420    0.3989    0.2420    0.0540
```

输入参数

x - 用于计算 pdf 的值
标量值 | 标量值组成的数组

计算 pdf 时所基于的值，指定为标量值或标量值组成的数组。

要在多个值处计算 pdf，请使用数组指定 **x**。要计算多个分布的 pdf，请使用数组指定 **mu** 和 **sigma**。如果输入参数 **x**、**mu** 和 **sigma** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**normpdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。**y** 中的每个元素是由 **mu** 和 **sigma** 中对应元素指定的分布的 pdf 值，其值在 **x** 中对应元素处进行计算。

示例：**[-1,0,3,4]**

数据类型：**single** | **double**

mu - 均值
0 （默认） | 标量值 | 标量值组成的数组

正态分布的均值，指定为标量值或由标量值组成的数组。

要在多个值处计算 pdf，请使用数组指定 **x**。要计算多个分布的 pdf，请使用数组指定 **mu** 和 **sigma**。如果输入参数 **x**、**mu** 和 **sigma** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**normpdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。**y** 中的每个元素是由 **mu** 和 **sigma** 中对应元素指定的分布的 pdf 值，其值在 **x** 中对应元素处进行计算。

示例：**[0 1 2; 0 1 2]**

数据类型：**single** | **double**

sigma - 标准差
1 （默认） | 正标量值 | 正标量值组成的数组

正态分布的标准差，指定为正标量值或正标量值组成的数组。

要在多个值处计算 pdf，请使用数组指定 **x**。要计算多个分布的 pdf，请使用数组指定 **mu** 和 **sigma**。如果输入参数 **x**、**mu** 和 **sigma** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**normpdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。**y** 中的每个元素是由 **mu** 和 **sigma** 中对应元素指定的分布的 pdf 值，其值在 **x** 中对应元素处进行计算。

示例：**[1 1 1; 2 2 2]**

数据类型：**single** | **double**

输出参数

y - pdf 值
标量值 | 标量值组成的数组

在 **x** 中的值处计算的 pdf 值，以标量值或标量值数组的形式返回。在经过任何必要的标量扩展后，**y** 的大小与 **x**、**mu** 和 **sigma** 相同。**y** 中的每个元素是由 **mu** 和 **sigma** 中对应元素指定的分布的 pdf 值，其值在 **x** 中对应元素处进行计算。

详细信息

正态分布

正态分布是双参数曲线族。第一个参数 μ 是均值。第二个参数 σ 是标准差。

标准正态分布具有零均值和单位标准差。

正态概率密度函数 (pdf) 是

$$y = f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad \text{for } x \in \mathbb{R}.$$

似然函数是被视为参数函数的 pdf。最大似然估计 (MLE) 是最大化 **x** 的固定值的似然函数的参数估计。

替代功能

- **normpdf** 是正态分布特有的函数。Statistics and Machine Learning Toolbox 还提供泛型函数 **pdf**，它支持各种概率分布。要使用 **pdf**，请创建一个 **NormalDistribution** 概率分布对象，并将该对象作为输入参数传递，或指定概率分布名称及其参数。请注意，分布特有的函数 **normpdf** 比泛型函数 **pdf** 的执行速度要快。
- 使用 **Probability Distribution Function** App 为概率分布创建累积分布函数 (cdf) 或概率密度函数 (pdf) 的交互图。

参考

[1] Evans, M., N. Hastings, and B. Peacock. *Statistical Distributions*. 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc., 1993.

扩展功能

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅

pdf | normcdf | norminv | normrnd | mvnpdf | NormalDistribution | normspec

主题

“正态分布” (第 B-2 页)

在 R2006a 之前推出

normrnd

正态随机数

语法

```
r = normrnd(mu,sigma)
r = normrnd(mu,sigma,sz1,...,szN)
r = normrnd(mu,sigma,sz)
```

说明

r = normrnd(mu,sigma) 从均值参数为 **mu** 和标准差参数为 **sigma** 的正态分布中生成随机数。

r = normrnd(mu,sigma,sz1,...,szN) 生成正态随机数数组，其中 **sz1,...,szN** 指示每个维度的大小。

r = normrnd(mu,sigma,sz) 生成正态随机数数组，其中向量 **sz** 指定 **size(r)**。

示例

生成正态随机数

生成一个标准正态分布的随机值。

```
rng('default') % For reproducibility
r = normrnd(0,1)
```

```
r = 0.5377
```

重置随机数生成器

保存随机数生成器的当前状态。然后基于均值为 3 标准差为 10 的正态分布创建由正态随机数组成的 1×5 向量。

```
s = rng;
r = normrnd(3,10,[1,5])
```

```
r = 1×5

   8.3767   21.3389  -19.5885   11.6217    6.1877
```

将随机数生成器的状态恢复为 **s**，然后创建一个由随机数组成的新的 1×5 向量。值与之前相同。

```
rng(s);
r1 = normrnd(3,10,[1,5])
```

```
r1 = 1×5
```

```
   8.3767   21.3389  -19.5885   11.6217    6.1877
```

根据现有数组克隆大小

创建一个由正态分布的随机数组成并且大小与现有数组相同的矩阵。

```
A = [3 2; -2 1];
sz = size(A);
R = normrnd(0,1,sz)
```

```
R = 2×2

   0.5377   -2.2588
   1.8339    0.8622
```

您可以将前两行代码合并成一行。

```
R = normrnd(1,0,size(A));
```

输入参数

mu - 均值
标量值 | 标量值组成的数组

正态分布的均值，指定为标量值或由标量值组成的数组。

要从多个分布中生成随机数，请使用数组指定 **mu** 和 **sigma**。如果 **mu** 和 **sigma** 均为数组，则数组大小必须相同。如果 **mu** 或 **sigma** 是标量，则 **normrnd** 会将标量参数扩展为与另一个参数大小相同的常量数组。**r** 中的每个元素均是从 **mu** 和 **sigma** 中对应元素所指定的分布中生成的随机数。

示例：`[0 1 2; 0 1 2]`

数据类型：**single** | **double**

sigma - 标准差
非负标量值 | 非负标量值组成的数组

正态分布的标准差，指定为非负标量值或由非负标量值组成的数组。

如果 **sigma** 为零，则输出 **r** 始终等于 **mu**。

要从多个分布中生成随机数，请使用数组指定 **mu** 和 **sigma**。如果 **mu** 和 **sigma** 均为数组，则数组大小必须相同。如果 **mu** 或 **sigma** 是标量，则 **normrnd** 会将标量参数扩展为与另一个参数大小相同的常量数组。**r** 中的每个元素均是从 **mu** 和 **sigma** 中对应元素所指定的分布中生成的随机数。

示例：`[1 1 1; 2 2 2]`

数据类型：**single** | **double**

sz1,...,szN - 每个维度的大小（作为单独参数）
整数

每个维度的大小，指定为整数。例如，指定 **5,3,2** 会从概率分布生成一个由随机数组成的 5×3×2 数组。

如果 **mu** 或 **sigma** 是数组，则在进行任何必要的标量扩展后，指定的维度 **sz1,...,szN** 必须与 **mu** 和 **sigma** 的公共维度相匹配。**sz1,...,szN** 的默认值为公共维度。

- 如果您指定单一值 **sz1**，则 **r** 是大小为 **sz1×sz1** 的方阵。
- 如果任一维度的大小是 **0** 或负数，则 **r** 是空数组。
- 对于第二个维度以上的维度，**normrnd** 会忽略大小为 1 的尾部维度。例如，指定 **3,1,1,1** 会生成由随机数组成的 3×1 向量。

示例：**5,3,2**

数据类型：**single** | **double**

sz - 每个维度的大小（作为行向量）
由整数组成的行向量

每个维度的大小，指定为由整数组成的行向量。例如，指定 **[5,3,2]** 会从概率分布生成一个由随机数组成的 5×3×2 数组。

如果 **mu** 或 **sigma** 是数组，则在进行任何必要的标量扩展后，指定的维度 **sz** 必须与 **mu** 和 **sigma** 的公共维度相匹配。**sz** 的默认值为公共维度。

- 如果您指定单一值 **[sz1]**，则 **r** 是大小为 **sz1×sz1** 的方阵。
- 如果任一维度的大小是 **0** 或负数，则 **r** 是空数组。
- 对于第二个维度以上的维度，**normrnd** 会忽略大小为 1 的尾部维度。例如，指定 **[3,1,1,1]** 会生成由随机数组成的 3×1 向量。

示例：**[5,3,2]**

数据类型：**single** | **double**

输出参数

r - 正态随机数
标量值 | 标量值组成的数组

正态随机数，以标量值或标量值数组的形式返回，其维度由 **sz1,...,szN** 或 **sz** 指定。**r** 中的每个元素均是从 **mu** 和 **sigma** 中对应元素所指定的分布中生成的随机数。

替代功能

- **normrnd** 是正态分布特有的函数。Statistics and Machine Learning Toolbox 还提供泛型函数 **random**，它支持各种概率分布。要使用 **random**，请创建一个 **NormalDistribution** 概率分布对象，并将该对象作为输入参数传递，或指定概率分布名称及其参数。请注意，分布特有的函数 **normrnd** 比泛型函数 **random** 的执行速度要快。
- 使用 **randn** 生成标准正态分布随机数。
- 要以交互方式生成随机数，请使用 **randtool**，它是用于生成随机数的用户界面。

参考

[1] Marsaglia, G, and W. W. Tsang. "A Fast, Easily Implemented Method for Sampling from Decreasing or Symmetric Unimodal Density Functions." *SIAM Journal on Scientific and Statistical Computing*. Vol. 5, Number 2, 1984, pp. 349–359.

[2] Evans, M., N. Hastings, and B. Peacock. *Statistical Distributions*. 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc., 1993.

扩展功能

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

如果下列任一情况成立，则生成的代码可能返回与 MATLAB 不同的数字序列：

- 输出是非标量。
- 输入参数对分布无效。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “General Code Generation Workflow”。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
random | mvnrnd | lognrnd | NormalDistribution | randn | normcdf | normpdf

主题
“Random Number Generation”
“正态分布” (第 B-2 页)

在 R2006a 之前推出

pca

原始数据的主成分分析

语法

coeff = pca(X)
coeff = pca(X,Name,Value)
[coeff,score,latent] = pca(___)
[coeff,score,latent,tsquared] = pca(___)
[coeff,score,latent,tsquared,explained,mu] = pca(___)

说明

coeff = pca(X) 返回 n×p 数据矩阵 **X** 的主成分系数，也称为载荷。**X** 的行对应于观测值，列对应于变量。系数矩阵是 p×p 矩阵。**coeff** 的每列包含一个主成分的系数，并且这些列按成分方差的降序排列。默认情况下，**pca** 将数据中心化，并使用奇异值分解 (SVD) 算法。

coeff = pca(X,Name,Value) 使用由一个或多个 **Name,Value** 对组参数指定的用于计算和处理特殊数据类型的附加选项，返回上述语法中的任何输出参数。

例如，您可以指定 **pca** 返回的主成分数或使用 SVD 以外的其他算法。

[coeff,score,latent] = pca(___) 还在 **score** 中返回主成分分数，在 **latent** 中返回主成分方差。您可以使用上述语法中的任何输入参数。

主成分分数是 **X** 在主成分空间中的表示。**score** 的行对应于观测值，列对应于成分。

主成分方差是 **X** 的协方差矩阵的特征值。

[coeff,score,latent,tsquared] = pca(___) 还返回 **X** 中每个观测值的 Hotelling T 方统计量。

[coeff,score,latent,tsquared,explained,mu] = pca(___) 还返回 **explained**（即每个主成分解释的总方差的百分比）和 **mu**（即 **X** 中每个变量的估计均值）。

示例

数据集的主成分

加载样本数据集。

```
load hald
```

原料数据有 4 个变量的 13 个观测值。

找出原料数据的主成分。

```
coeff = pca(ingredients)
```

```
coeff = 4×4
```

```
   -0.0678   -0.6460    0.5673    0.5062
   -0.6785   -0.0200   -0.5440    0.4933
    0.0290    0.7553    0.4036    0.5156
    0.7309   -0.1085   -0.4684    0.4844
```

coeff 的行包含四个原料变量的系数，其列对应于四个主成分。

存在缺失数据时的 PCA

当数据集中存在缺失值时，计算主成分系数。

加载样本数据集。

```
load imports-85
```

数据矩阵 **X** 在第 3 列至第 15 列中有 13 个连续变量：轴距、长度、宽度、高度、整备重量、引擎大小、缸径、冲程、压缩比、马力、峰值转速、城市油耗和高速公路油耗。变量缸径和冲程在第 56 行至第 59 行中缺失四个值，变量马力和峰值转速在第 131 行和第 132 行中缺失两个值。

执行主成分分析。

```
coeff = pca(X(:,3:15));
```

默认情况下，**pca** 执行 **'Rows','complete'** 名称-值对组参数指定的操作。此选项在计算前会先删除包含 **NaN** 值的观测值。然后再将 **NaN** 行重新插入 **score** 和 **tsquared** 中的对应位置，即第 56 至 59、131 和 132 行。

使用 **'pairwise'** 执行主成分分析。

```
coeff = pca(X(:,3:15),'Rows','pairwise');
```

在本例中，**pca** 使用 **X** 的列 i 或 j 中没有 **NaN** 值的行来计算协方差矩阵的 (i,j) 元素。请注意，得到的协方差矩阵可能不是正定矩阵。当 **pca** 使用的算法是特征值分解时，此选项适用。当您没有指定算法时，如此示例中所示，**pca** 会将其设置为 **'eig'**。如果您需要 **'svd'** 作为算法并使用 **'pairwise'** 选项，则 **pca** 返回警告消息，将算法设置为 **'eig'** 并继续。

如果您使用 **'Rows','all'** 名称-值对组参数，**pca** 将终止，因为此选项假设数据集中没有缺失值。

```
coeff = pca(X(:,3:15),'Rows','all');
```

```
Error using pca (line 180)
Raw data contains NaN missing value while 'Rows' option is set to 'all'. Consider using 'complete' or pairwise' op
```

加权 PCA

执行主成分分析时，使用变量的逆方差作为权重。

加载样本数据集。

```
load hald
```

使用原料数据的逆方差作为变量权重执行主成分分析。

```
 [wcoeff,~,latent,~,explained] = pca(ingredients,...
'VariableWeights','variance')
```

```
wcoeff = 4×4

  -2.7998   2.9940  -3.9736   1.4180
  -8.7743  -6.4411   4.8927   9.9863
   2.5240  -3.8749  -4.0845   1.7196
   9.1714   7.5529   3.2710  11.3273


latent = 4×1

   2.2357
   1.5761
   0.1866
   0.0016


explained = 4×1

  55.8926
  39.4017
   4.6652
   0.0406
```

请注意，系数矩阵 **wcoeff** 不是正交矩阵。

计算正交系数矩阵。

```
coefforth = inv(diag(std(ingredients)))* wcoeff
```

```
coefforth = 4×4

  -0.4760   0.5090  -0.6755   0.2411
  -0.5639  -0.4139   0.3144   0.6418
   0.3941  -0.6050  -0.6377   0.2685
   0.5479   0.4512   0.1954   0.6767
```

检查新系数矩阵 **coefforth** 的正交性。

```
 coefforth*coefforth'
```

```
ans = 4×4

   1.0000   0.0000  -0.0000   0.0000
   0.0000   1.0000  -0.0000  -0.0000
  -0.0000  -0.0000   1.0000   0.0000
   0.0000  -0.0000   0.0000   1.0000
```

对缺失数据使用 ALS 进行 PCA

当数据中有缺失值时，使用交替最小二乘法 (ALS) 算法计算主成分。

加载样本数据。

```
load hald
```

原料数据有 4 个变量的 13 个观测值。

使用 ALS 算法执行主成分分析，并显示成分系数。

```
[coeff,score,latent,tsquared,explained] = pca(ingredients);
coeff
```

```
coeff = 4×4

   -0.0678   -0.6460    0.5673    0.5062
   -0.6785   -0.0200   -0.5440    0.4933
    0.0290    0.7553    0.4036    0.5156
    0.7309   -0.1085   -0.4684    0.4844
```

随机引入缺失值。

```
y = ingredients;
rng('default'); % for reproducibility
ix = random('unif',0,1,size(y))<0.30;
y(ix) = NaN
```

```
y = 13×4

    7   26    6  NaN
    1   29   15   52
  NaN  NaN    8   20
   11   31  NaN   47
    7   52    6   33
  NaN   55  NaN  NaN
  NaN   71  NaN    6
    1   31  NaN   44
    2  NaN  NaN   22
   21   47    4   26
    ⋮
```

NaN 指示，现在约有 30% 的数据包含缺失值。

使用 ALS 算法执行主成分分析，并显示成分系数。

```
[coeff1,score1,latent,tsquared,explained,mu1] = pca(y,...
'algorithm','als');
coeff1
```

```
coeff1 = 4×4

   -0.0362    0.8215   -0.5252    0.2190
   -0.6831   -0.0998    0.1828    0.6999
    0.0169    0.5575    0.8215   -0.1185
    0.7292   -0.0657    0.1261    0.6694
```

显示估计的均值。

```
mu1
```

```
mu1 = 1×4

   8.9956   47.9088    9.0451   28.5515
```

重新构造观测到的数据。

```
t = score1*coeff1' + repmat(mu1,13,1)
```

```
t = 13×4

   7.0000   26.0000    6.0000   51.5250
   1.0000   29.0000   15.0000   52.0000
  10.7819   53.0230    8.0000   20.0000
  11.0000   31.0000   13.5500   47.0000
   7.0000   52.0000    6.0000   33.0000
  10.4818   55.0000    7.8328   17.9362
   3.0982   71.0000   11.9491    6.0000
   1.0000   31.0000   -0.5161   44.0000
   2.0000   53.7914    5.7710   22.0000
  21.0000   47.0000    4.0000   26.0000
     ⋮
```

ALS 算法估计数据中的缺失值。

另一种比较结果的方法是找出系数向量占据的两个空间之间的角度。使用 ALS 计算完整数据和含缺失值的数据的系数之间的角度。

```
subspace(coeff,coeff1)
```

```
ans = 9.1336e-16
```

这是一个很小的值。这表明，如果您在没有缺失数据的情况下使用 **pca** 和 **'Rows','complete'** 名称-值对组参数，而在有缺失数据的情况下使用 **pca** 和 **'algorithm','als'** 名称-值对组参数，结果彼此接近。

使用 **'Rows','complete'** 名称-值对组参数执行主成分分析，并显示成分系数。

```
[coeff2,score2,latent,tsquared,explained,mu2] = pca(y,...
'Rows','complete');
coeff2
```

```
coeff2 = 4×3

  -0.2054    0.8587    0.0492
  -0.6694   -0.3720    0.5510
   0.1474   -0.3513   -0.5187
   0.6986   -0.0298    0.6518
```

在本例中，**pca** 删除包含缺失值的行，而 **y** 只有四个行没有缺失值。**pca** 仅返回三个主成分。您无法使用 **'Rows','pairwise'** 选项，因为协方差矩阵不是半正定矩阵，**pca** 返回错误消息。

计算完整数据与整行删除缺失值数据（当 **'Rows','complete'** 时）的系数之间的角度。

```
subspace(coeff(:,1:3),coeff2)
```

```
ans = 0.3576
```

这两个空间之间的角度要大得多。这表明这两个结果不同。

显示估计的均值。

```
mu2
```

```
mu2 = 1×4

   7.8889   46.9091    9.8750   29.6000
```

在本例中，均值只是 y 的样本均值。

重新构造观测到的数据。

```
score2*coeff2'
```

```
ans = 13×4

    NaN     NaN     NaN     NaN
 -7.5162 -18.3545  4.0968  22.0056
    NaN     NaN     NaN     NaN
    NaN     NaN     NaN     NaN
 -0.5644  5.3213 -3.3432  3.6040
    NaN     NaN     NaN     NaN
    NaN     NaN     NaN     NaN
    NaN     NaN     NaN     NaN
    NaN     NaN     NaN     NaN
 12.8315 -0.1076 -6.3333 -3.7758
    ⋮
```

这表明删除包含 NaN 值的行的效果不如 ALS 算法。当数据有太多缺失值时，使用 ALS 更好。

主成分系数、分数和方差

计算主成分的系数、分数和方差。

加载样本数据集。

```
load hald
```

原料数据有 4 个变量的 13 个观测值。

计算原料数据的成分的主成分系数、分数和方差。

```
[coeff,score,latent] = pca(ingredients)
```

```
coeff = 4×4

 -0.0678  -0.6460   0.5673   0.5062
 -0.6785  -0.0200  -0.5440   0.4933
  0.0290   0.7553   0.4036   0.5156
  0.7309  -0.1085  -0.4684   0.4844
```

```
score = 13×4

   36.8218   -6.8709   -4.5909    0.3967
   29.6073    4.6109   -2.2476   -0.3958
  -12.9818   -4.2049    0.9022   -1.1261
   23.7147   -6.6341    1.8547   -0.3786
   -0.5532   -4.4617   -6.0874    0.1424
  -10.8125   -3.6466    0.9130   -0.1350
  -32.5882    8.9798   -1.6063    0.0818
   22.6064   10.7259    3.2365    0.3243
   -9.2626    8.9854   -0.0169   -0.5437
   -3.2840  -14.1573    7.0465    0.3405
      ⋮
```

```
latent = 4×1

  517.7969
   67.4964
   12.4054
    0.2372
```

score 的每列对应一个主成分。向量 **latent** 存储四个主成分的方差。

重新构造中心化的原料数据。

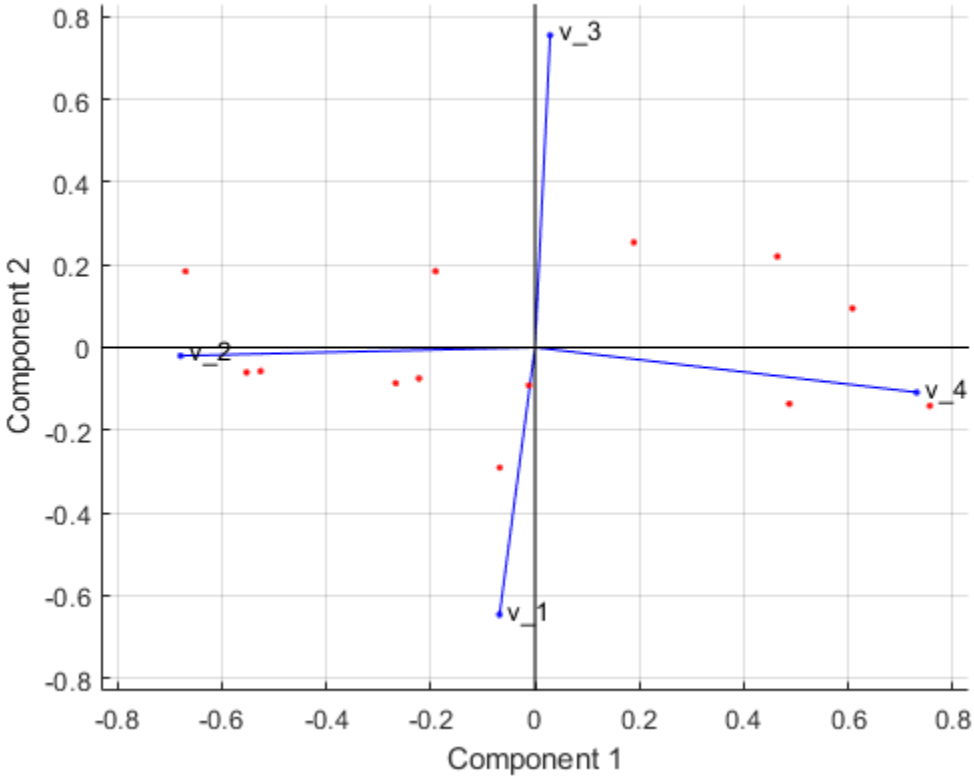
```
Xcentered = score*coeff'
```

```
Xcentered = 13×4

   -0.4615  -22.1538   -5.7692   30.0000
   -6.4615  -19.1538    3.2308   22.0000
    3.5385    7.8462   -3.7692  -10.0000
    3.5385  -17.1538   -3.7692   17.0000
   -0.4615    3.8462   -5.7692    3.0000
    3.5385    6.8462   -2.7692   -8.0000
   -4.4615   22.8462    5.2308  -24.0000
   -6.4615  -17.1538   10.2308   14.0000
   -5.4615    5.8462    6.2308   -8.0000
   13.5385   -1.1538   -7.7692   -4.0000
      ⋮
```

Xcentered 中的新数据是将原始原料数据对应列减去列均值进行中心化后所得的结果。

在单一图中可视化每个变量的正交主成分系数和每个观测值的主成分分数。

```
biplot(coeff(:,1:2),'scores',score(:,1:2),'varlabels',{'v_1','v_2','v_3','v_4'});
```



所有四个变量在此双标图中都用向量来表示，向量的方向和长度指示每个变量对图中两个主成分的贡献。例如，位于水平轴上的第一个主成分对于第三个和第四个变量具有正系数。因此，向量 v_3 和 v_4 映射到图的右半部分。第一个主成分中最大的系数是第四个，对应于变量 v_4。

第二个主成分位于垂直轴上，对于变量 v_1、v_2 和 v_4 具有负系数，对于变量 v_3 具有正系数。

该二维双标图还包含 13 个观测值的对应点，点在图中的坐标指示了每个观测值的两个主成分的分数。例如，绘图靠近左边缘的点对第一个主成分的分数最低。这些点基于最大分数值和最大系数长度进行了缩放，因此只能从图中确定其相对位置。

T 方统计量

计算 Hotelling T 方统计量值。

加载样本数据集。

```
load hald
```

原料数据有 4 个变量的 13 个观测值。

执行主成分分析并请求 T 方值。

```
[coeff,score,latent,tsquared] = pca(ingredients);
tsquared
```

```
tsquared = 13×1

   5.6803
   3.0758
   6.0002
   2.6198
   3.3681
   0.5668
   3.4818
   3.9794
   2.6086
   7.4818
     ⋮
```

仅请求前两个主成分，并计算请求的主成分在降维空间中的 T 方值。

```
[coeff,score,latent,tsquared] = pca(ingredients,'NumComponents',2);
tsquared
```

```
tsquared = 13×1

   5.6803
   3.0758
   6.0002
   2.6198
   3.3681
   0.5668
   3.4818
   3.9794
   2.6086
   7.4818
     ⋮
```

请注意，即使您指定了降维的成分空间，**pca** 仍会使用所有四个成分计算完整空间中的 T 方值。

降维空间中的 T 方值对应于降维空间中的马氏距离。

```
tsqreduced = mahal(score,score)
```

```
tsqreduced = 13×1

   3.3179
   2.0079
   0.5874
   1.7382
   0.2955
   0.4228
   3.2457
   2.6914
   1.3619
   2.9903
     ⋮
```

通过求完整空间中的 T 方值和降维空间中的马氏距离之差来计算丢弃空间中的 T 方值。

```
tsqdiscarded = tsquared - tsqreduced
```

```
tsqdiscarded = 13×1

   2.3624
   1.0679
   5.4128
   0.8816
   3.0726
   0.1440
   0.2362
   1.2880
   1.2467
   4.4915
      ⋮
```

主成分解释的变异性百分比

计算主成分解释的变异性百分比。显示主成分空间中的数据表示。

加载样本数据集。

```
load imports-85
```

数据矩阵 **X** 在第 3 列至第 15 列中有 13 个连续变量：轴距、长度、宽度、高度、整备重量、引擎大小、缸径、冲程、压缩比、马力、峰值转速、城市油耗和高速公路油耗。

计算由这些变量的主成分解释的变异性百分比。

```
[coeff,score,latent,tsquared,explained] = pca(X(:,3:15));
```

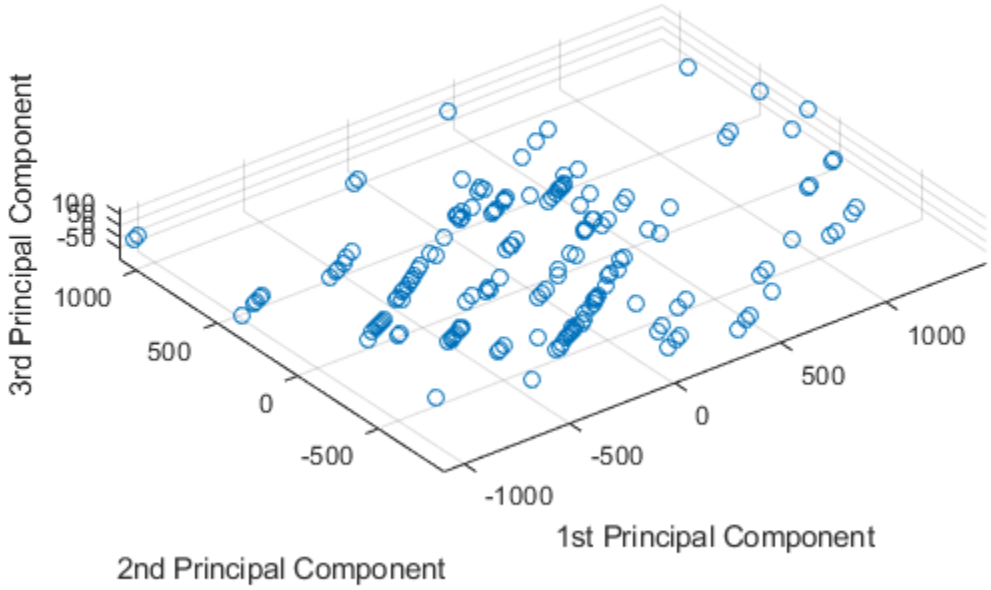
```
explained
```

```
explained = 13×1

  64.3429
  35.4484
   0.1550
   0.0379
   0.0078
   0.0048
   0.0013
   0.0011
   0.0005
   0.0002
      ⋮
```

前三个成分解释所有变异性的 99.95%。

可视化前三个主成分的空间中的数据表示。

```
scatter3(score(:,1),score(:,2),score(:,3))
axis equal
xlabel('1st Principal Component')
ylabel('2nd Principal Component')
zlabel('3rd Principal Component')
```



数据在第一主成分轴上具有最大变异性。在第一个轴的所有可能选择中，这是有可能得到的最大方差。在剩余可供第二主成分轴使用的所有可能选择中，该选择具有最大变异性。第三主成分轴具有第三大变异性，它显著小于在第二主成分轴上的变异性。第四至第十三主成分轴不值得研究，因为它们只解释数据中所有变异性的 0.05%。

要跳过任何输出，您可以在对应的元素中使用 ~。例如，如果您不想得到 T 方值，请指定

```
[coeff,score,latent,~,explained] = pca(X(:,3:15));
```

将 PCA 应用于新数据并生成 C/C++ 代码

找出一个数据集的主成分，并将 PCA 应用于另一个数据集。当您有用于机器学习模型的训练数据集和测试数据集时，此过程非常有用。例如，您可以使用 PCA 对训练数据集进行预处理，然后训练模型。要使用测试数据集测试训练模型，您需要将从训练数据获得的 PCA 转换应用于测试数据集。

此示例还说明如何生成 C/C++ 代码。由于 **pca** 支持代码生成，因此您可以使用训练数据集生成执行 PCA 的代码，并将 PCA 应用于测试数据集。然后将代码部署到设备上。在此工作流中，您必须传递训练数据，训练数据有可能相当大。为了节省设备上的内存，您可以将训练和预测分离。在 MATLAB® 中使用 **pca**，并通过部署到设备上的生成代码对新数据应用 PCA。

生成 C/C++ 代码需要 MATLAB® Coder™。

将 PCA 应用于新数据

使用 **readtable** 将数据集加载到表中。数据集在文件 **CreditRating_Historical.dat** 中，其中包含历史信用评分数据。

```
creditrating = readtable('CreditRating_Historical.dat');
creditrating(1:5,:)
```

```
ans=5×8 table
   ID    WC_TA   RE_TA   EBIT_TA   MVE_BVTD   S_TA   Industry   Rating
  _____  _____  _____  _______  ________  _____  ________  _______

  62394  0.013  0.104  0.036    0.447     0.142     3      {'BB' }
  48608  0.232  0.335  0.062    1.969     0.281     8      {'A'  }
  42444  0.311  0.367  0.074    1.935     0.366     1      {'A'  }
  48631  0.194  0.263  0.062    1.017     0.228     4      {'BBB'}
  43768  0.121  0.413  0.057    3.647     0.466     12     {'AAA'}
```

第一列是每个观测值的 ID，最后一列是评分。将第二列至第七列指定为预测变量数据，并将最后一列 (**Rating**) 指定为响应。

```
X = table2array(creditrating(:,2:7));
Y = creditrating.Rating;
```

使用前 100 个观测值作为测试数据，其余的作为训练数据。

```
XTest = X(1:100,:);
XTrain = X(101:end,:);
YTest = Y(1:100);
YTrain = Y(101:end);
```

计算训练数据集 **XTrain** 的主成分。

```
[coeff,scoreTrain,~,~,explained,mu] = pca(XTrain);
```

此代码返回四个输出：**coeff**、**scoreTrain**、**explained** 和 **mu**。使用 **explained**（解释方差占总方差的百分比）计算解释至少 95% 变异性所需的成分的数目。使用 **coeff**（主成分系数）和 **mu**（**XTrain** 的估计均值）将 PCA 应用于测试数据集。在训练模型时，请使用 **scoreTrain**（主成分分数）而不是 **XTrain**。

显示由主成分解释的变异性百分比。

```
explained
```

```
explained = 6×1

  58.2614
  41.2606
   0.3875
   0.0632
   0.0269
   0.0005
```

前两个成分解释了 95% 以上的变异性。计算解释至少 95% 变异性所需的成分的数目。

```
idx = find(cumsum(explained)>95,1)
```

```
idx = 2
```

使用前两个成分训练分类树。

```
scoreTrain95 = scoreTrain(:,1:idx);
mdl = fitctree(scoreTrain95,YTrain);
```

mdl 是一个 **ClassificationTree** 模型。

要对测试集使用训练模型，您需要使用从训练数据集获得的 PCA 来转换测试数据集。通过从 **XTest** 中减去 **mu** 再乘以 **coeff** 获得测试数据集的主成分分数。只需要前两个成分的分数，因此使用前两个系数 **coeff(:,1:idx)**。

```
scoreTest95 = (XTest-mu)*coeff(:,1:idx);
```

将经过训练的模型 **mdl** 和转换后的测试数据集 **scoreTest** 传递给 **predict** 函数，以预测测试集的评分。

```
YTest_predicted = predict(mdl,scoreTest95);
```

生成代码

生成代码，这些代码将 PCA 应用于数据并使用经过训练的模型预测评分。请注意，生成 C/C++ 代码需要 MATLAB® Coder™。

使用 **saveLearnerForCoder** 将分类模型保存到文件 **myMdl.mat** 中。

```
saveLearnerForCoder(mdl,'myMdl');
```

定义名为 **myPCAPredict** 的入口函数，该函数接受测试数据集 (**XTest**) 和 PCA 信息（**coeff** 和 **mu**），并返回测试数据的评分。

在入口函数的函数签名后面添加 **%#codegen** 编译器指令（即 pragma），以指示您要为此 MATLAB 算法生成代码。添加此指令指示 MATLAB 代码分析器帮助您诊断和修复在代码生成过程中可能导致错误的违规。

```
function label = myPCAPredict(XTest,coeff,mu) %#codegen
% Transform data using PCA
scoreTest = bsxfun(@minus,XTest,mu)*coeff;

% Load trained classification model
mdl = loadLearnerForCoder('myMdl');
% Predict ratings using the loaded model
label = predict(mdl,scoreTest);
```

myPCAPredict 使用 **coeff** 和 **mu** 将 PCA 应用于新数据，然后使用转换后的数据预测评分。这样，您就不必传递有可能特别大的训练数据。

注意：如果您点击位于此页右上角的按钮，并在 MATLAB® 中打开此示例，则 MATLAB® 将打开示例文件夹。该文件夹包括入口函数文件。

使用 **codegen** (MATLAB Coder) 生成代码。由于 C 和 C++ 是静态类型语言，因此必须在编译时确定入口函数中所有变量的属性。要指定数据类型和精确的输入数组大小，请使用 **-args** 选项传递表示具有特定数据类型和数组大小的值集的 MATLAB® 表达式。如果在编译时观测值数目未知，您也可以使用 **coder.typeof** (MATLAB Coder) 将输入指定为可变大小。有关详细信息，请参阅 “Specify Variable-Size Arguments for Code Generation”。

```
codegen myPCAPredict -args {coder.typeof(XTest,[Inf,6],[1,0]),coeff(:,1:idx),mu}
```

```
Code generation successful.
```

codegen 生成 MEX 函数 **myPCAPredict_mex**，扩展名因平台而异。

验证生成的代码。

```
YTest_predicted_mex = myPCAPredict_mex(XTest,coeff(:,1:idx),mu);
isequal(YTest_predicted,YTest_predicted_mex)
```

```
ans = logical
   1
```

isequal 返回逻辑值 1 (**true**)，这意味着所有输入都相等。比较结果证实，**mdl** 的 **predict** 函数和 **myPCAPredict_mex** 函数返回相同的评分。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “Code Generation and Classification Learner App”。后者说明如何使用分类学习器执行 PCA 和训练模型，以及如何生成基于训练模型预测新数据标签的 C/C++ 代码。

输入参数

X - 输入数据

矩阵

计算主成分时所基于的输入数据，指定为 n×p 矩阵。**X** 的行对应于观测值，列对应于变量。

数据类型： **single** | **double**

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例： **'Algorithm','eig','Centered',false,'Rows','all','NumComponents',3** 指定 **pca** 使用特征值分解算法，不将数据中心化，使用所有观测值，并仅返回前三个主成分。

Algorithm - 主成分算法

'svd' （默认） | **'eig'** | **'als'**

pca 用于执行主成分分析的主成分算法，指定为以逗号分隔的对组，其中包含 **'Algorithm'** 和以下项之一。

值	说明
'svd'	默认值。**X** 的奇异值分解 (SVD)。
'eig'	协方差矩阵的特征值分解 (EIG)。当观测值数目 n 超过变量的数目 p 时，EIG 算法比 SVD 更快，但不太准确，因为协方差的条件数是 **X** 的条件数的平方。

值	说明
'als'	交替最小二乘 (ALS) 算法。此算法通过将 **X** 分解为 n×k 左因子矩阵 L 和 p×k 右因子矩阵 R 来计算最佳秩 k 逼近，其中 k 是主成分的数量。分解使用从随机初始值开始的迭代方法。 ALS 能够更好地处理缺失值。它倾向于采用成对删除 (**'Rows','pairwise'**)，而不是采用整行删除 (**'Rows','complete'**) 处理缺失值。它可以很好地处理随机缺失少量数据的数据集，但对于稀疏数据集可能表现不佳。

示例： **'Algorithm','eig'**

Centered - 是否中心化列的指示符
true （默认） | **false**

是否中心化列的指示符，指定为以逗号分隔的对组，其中包含 **'Centered'** 和下列逻辑表达式之一。

值	说明
true	默认值。**pca** 通过在计算奇异值分解或特征值分解之前减去列均值，将 **X** 中心化。如果 **X** 包含 **NaN** 缺失值，则使用 **mean(X,'omitnan')** 计算所有可用数据的均值。您可以使用 **score*coeff'** 重新构造中心化的数据。
false	在这种情况下，**pca** 不会对数据进行中心化。您可以使用 **score*coeff'** 重新构造原始数据。

示例： **'Centered',false**

数据类型： **logical**

Economy - 是否精简大小输出的指示符
true （默认） | **false**

当自由度 （第 33-176 页） d 小于变量数 p 时，是否精简大小输出的指示符，指定为以逗号分隔的对组，其中包含 **'Economy'** 和下列逻辑表达式之一。

值	说明
true	默认值。**pca** 仅返回 **latent** 的前 d 个元素以及 **coeff** 和 **score** 的对应列。 当变量的数目 p 远大于 d 时，此选项可以显著加快执行速度。
false	**pca** 返回 **latent** 的所有元素。对应于 **latent** 中零元素的 **coeff** 和 **score** 的列是零。

请注意，当 d < p 时，**score(:,d+1:p)** 和 **latent(d+1:p)** 必须为零，且 **coeff(:,d+1:p)** 的列将定义与 **X** 正交的方向。

示例： **'Economy',false**

数据类型： **logical**

NumComponents - 请求的成分的数目
变量的数目 （默认） | 整数标量

请求的成分的数目，指定为以逗号分隔的对组，其中包含 **'NumComponents'** 和满足 0 < k ≤ p 的整数标量 k，其中 p 是 **X** 中原始变量的数目。指定此项时，**pca** 返回 **coeff** 和 **score** 的前 k 个列。

示例： **'NumComponents',3**

数据类型： **single** | **double**

Rows - 要对 NaN 值采取的操作
'complete' （默认） | **'pairwise'** | **'all'**

要对数据矩阵 **X** 中的 **NaN** 值采取的操作，指定为以逗号分隔的对组，其中包含 **'Rows'** 和以下项之一。

值	说明
'complete'	默认值。在计算前会先删除值为 **NaN** 的观测值。待计算后再将这些 **NaN** 行重新插入 **score** 和 **tsquared** 中的对应位置。
'pairwise'	此选项仅适用于算法为 **'eig'** 的情况。如果您没有与 **'pairwise'** 一起指定算法，则 **pca** 会将算法设置为 **'eig'**。如果您指定 **'svd'** 作为算法并指定了选项 **'Rows','pairwise'**，则 **pca** 会返回警告消息，提示您将算法设置为 **'eig'**，然后再继续。 当您指定 **'Rows','pairwise'** 选项时，**pca** 使用 **X** 的列 i 或 j 中没有 **NaN** 值的行来计算协方差矩阵的 (i,j) 元素。 请注意，得到的协方差矩阵可能不是正定矩阵。在这种情况下，**pca** 会以一条错误消息终止。
'all'	**X** 不应包含缺失值。**pca** 使用所有数据，如果发现任何 **NaN** 值，则终止。

示例： **'Rows','pairwise'**

Weights - 观测值权重
1 （默认） | 行向量

观测值权重，指定为以逗号分隔的对组，其中包含 **'Weights'**，以及一个包含所有正元素的长度为 n 的向量。

数据类型： **single** | **double**

VariableWeights - 变量权重
行向量 | **'variance'**

变量权重 （第 33-176 页），指定为以逗号分隔的对组，其中包含 **'VariableWeights'** 和以下项之一。

值	说明
行向量	包含所有正元素的长度为 p 的向量。
'variance'	变量权重是样本方差的倒数。如果您还使用 **'Weights'** 为观测值分配权重，则变量权重变为加权样本方差的倒数。 如果同时 **'Centered'** 设置为 **true**，则会对数据矩阵 **X** 进行中心化和标准化处理。在这种情况下，**pca** 将根据相关矩阵返回主成分。

示例： **'VariableWeights','variance'**

数据类型： **single** | **double** | **char** | **string**

Coeff0 - 系数的初始值
由随机值组成的矩阵 （默认） | p×k 矩阵

系数矩阵 **coeff** 的初始值，指定为以逗号分隔的对组，由 **'Coeff0'** 和 p×k 矩阵组成，其中 p 是变量的数目，k 是请求的主成分的数目。

注意 仅当 **'algorithm'** 是 **'als'** 时，才能使用此名称-值对组。

数据类型： **single** | **double**

Score0 - 分数的初始值
由随机值组成的矩阵 （默认） | k×m 矩阵

分数矩阵 **score** 的初始值，指定为以逗号分隔的对组，由 **'Score0'** 和 n×k 矩阵组成，其中 n 是观测值的数目，k 是请求的主成分的数目。

注意 仅当 **'algorithm'** 是 **'als'** 时，才能使用此名称-值对组。

数据类型： **single** | **double**

Options - 迭代的选项
结构体

迭代的选项，指定为以逗号分隔的对组，其中包含 **'Options'** 和由 **statset** 函数创建的结构体。**pca** 在 options 结构体中使用以下字段。

字段名称	说明
'Display'	显示输出的级别。选项包括 **'off'**、**'final'** 和 **'iter'**。
'MaxIter'	允许的最大步数。默认值为 1000。与优化设置不同，达到 **MaxIter** 值即视为收敛。
'TolFun'	用来指定代价函数的终止容差的正数。默认值为 1e-6。
'TolX'	正数，用来指定 ALS 算法中左因子矩阵和右因子矩阵的元素中相对变化的收敛阈值。默认值为 1e-6。

注意 仅当 **'algorithm'** 是 **'als'** 时，才能使用此名称-值对组。

您可以更改这些字段的值，并使用 **'Options'** 名称-值对组参数在 **pca** 中指定新结构体。

示例： **opt = statset('pca'); opt.MaxIter = 2000; coeff = pca(X,'Options',opt);**

数据类型： **struct**

输出参数

coeff - 主成分系数
矩阵

主成分系数，以 p×p 矩阵形式返回。**coeff** 的每列都包含一个主成分的系数。这些列按成分方差 **latent** 递减的顺序排列。

score - 主成分分数
矩阵

主成分分数，以矩阵形式返回。**score** 的行对应于观测值，列对应于成分。

latent - 主成分方差
列向量

主成分方差，即 **X** 的协方差矩阵的特征值，以列向量形式返回。

tsquared - Hotelling T 方统计量
列向量

“Hotelling T 方统计量” （第 33-176 页），它是每个观测值的标准化分数的平方和，以列向量形式返回。

explained - 解释方差占总方差的百分比
列向量

每个主成分解释方差占总方差的百分比，以列向量形式返回。

mu - 估计的均值
行向量

当 **Centered** 设置为 **true** 时，**X** 中变量的估计的均值以行向量形式返回。当 **Centered** 是 **false** 时，软件不计算均值，并返回零向量。

详细信息

Hotelling T 方统计量

Hotelling T 方统计量是每个观测值离数据集中心的多元距离的统计度量。

即使您请求的成分数目少于变量的数目，**pca** 也会使用所有主成分来计算 T 方统计量（在完整空间中计算它）。如果您需要计算降维空间或丢弃空间中的 T 方统计量，请执行以下操作之一：

- 要获得降维空间中的 T 方统计量，请使用 **mahal(score,score)**。
- 要获得丢弃空间中的 T 方统计量，请首先使用 **[coeff,score,latent,tsquared] = pca(X,'NumComponents',k,...)** 计算 T 方统计量，再使用 **tsqreduced = mahal(score,score)** 计算降维空间中的 T 方统计量，然后取差值：**tsquared - tsqreduced**。

自由度

如果数据经过中心化，则自由度 d 等于 n - 1，否则等于 n，其中：

- 如果您使用 **'Rows','complete'**，则 n 是不包含任何 **NaN** 的行的数目。
- 如果您使用 **'Rows','pairwise'**，则 n 是不包含 **NaN** 的行数最多的列对组中不包含任何 **NaN** 的行的数目。

变量权重

请注意，当使用变量权重时，系数矩阵不是正交矩阵。假设您使用的变量权重向量称为 **varwei**，**pca** 返回的主成分系数向量为 **wcoeff**。则您可以使用转换 **diag(sqrt(varwei))*wcoeff** 来计算正交系数。

算法

pca 函数施加符号约定，强制 **coefs** 的每列中具有最大幅值的元素为正值。更改系数向量的符号并不更改其含义。

参考

[1] Jolliffe, I. T. *Principal Component Analysis*. 2nd ed., Springer, 2002.

[2] Krzanowski, W. J. *Principles of Multivariate Analysis*. Oxford University Press, 1988.

[3] Seber, G. A. F. *Multivariate Observations*. Wiley, 1984.

[4] Jackson, J. E. A. *User's Guide to Principal Components*. Wiley, 1988.

[5] Roweis, S. "EM Algorithms for PCA and SPCA." *In Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems*. Vol.10 (NIPS 1997), Cambridge, MA, USA: MIT Press, 1998, pp. 626–632.

[6] Ilin, A., and T. Raiko. "Practical Approaches to Principal Component Analysis in the Presence of Missing Values." *J. Mach. Learn. Res.*. Vol. 11, August 2010, pp. 1957–2000.

扩展功能

tall 数组
对行数太多而无法放入内存的数组进行计算。

此函数支持对无法放入内存的数据使用 tall 数组，但有一些限制。

- **pca** 通过计算协方差矩阵并使用内存 **pcacov** 函数计算主成分，直接处理 tall 数组。
- 支持的语法包括：
  - **coeff = pca(X)**
  - **[coeff,score,latent] = pca(X)**
  - **[coeff,score,latent,explained] = pca(X)**
  - **[coeff,score,latent,tsquared] = pca(X)**
  - **[coeff,score,latent,tsquared,explained] = pca(X)**
- 不支持名称-值对组参数。

有关详细信息，请参阅“使用 tall 数组处理无法放入内存的数据”。

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

- 当 **'Algorithm'** 是 **'als'** 时，**'Options'** 的 **'Display'** 值将被忽略。
- **'Weights'** 和 **'VariableWeights'** 名称-值对组参数的值必须为实数。
- **'Economy'** 名称-值对组参数的值必须为编译时常量。例如，要在生成的代码中使用 **'Economy',false** 名称-值对组参数，请在 **codegen** 的 **-args** 值中包括 **{coder.Constant('Economy'),coder.Constant(false)}**。
- 名称-值对组参数中的名称必须为编译时常量。
- 生成的代码始终以列向量形式返回第五个输出 **explained**。
- 生成的代码始终以行向量形式返回第六个输出 **mu**。

- 如果 **mu** 为空，**pca** 会将 **mu** 以 1×0 数组形式返回。**pca** 不会将 **mu** 转换为 0×0 空数组。
- 生成的代码不会将包含所有 **NaN** 值的输入矩阵 **X** 视为特例。输出维数与对应的有限输入项项数相当。
- 为了节省部署生成代码的设备上的内存，可以将训练（基于输入数据构造 PCA 成分）和预测（执行 PCA 转换）分离。在 MATLAB 中构造 PCA 成分。然后，定义入口函数，该函数使用 **pca** 的输出，即主成分系数 (**coeff**) 和估计均值 (**mu**) 执行 PCA 转换。最后，为入口函数生成代码。有关示例，请参阅 “将 PCA 应用于新数据并生成 C/C++ 代码” （第 33-169 页）。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “General Code Generation Workflow”。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

使用说明和限制：

- 您无法将名称-值参数 **'Algorithm'** 指定为 **'als'**。
- 默认的主成分算法是 SVD（名称-值参数 **'Algorithm','svd'**）。SVD 算法在 GPU 而不是 CPU 上执行时，速度很难有所提升。将名称-值参数 **'Algorithm'** 指定为 **'eig'**，以加快 GPU 上的计算速度。

有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
barttest | **biplot** | **canoncorr** | **factoran** | **pcacov** | **pcares** | **rotatefactors** | **ppca**

主题
“Analyze Quality of Life in U.S. Cities Using PCA”
“主成分分析 (PCA)” （第 15-2 页）

在 R2012b 中推出

pdf

包： prob

概率密度函数

语法

```
y = pdf('name',x,A)
y = pdf('name',x,A,B)
y = pdf('name',x,A,B,C)
y = pdf('name',x,A,B,C,D)

y = pdf(pd,x)
```

说明

y = pdf('name',x,A) 返回由 **'name'** 和分布参数 **A** 指定的单参数分布族的概率密度函数 (pdf)，在 **x** 中的值处计算函数值。

y = pdf('name',x,A,B) 返回由 **'name'** 以及分布参数 **A** 和 **B** 指定的双参数分布族的 pdf，在 **x** 中的值处计算函数值。

y = pdf('name',x,A,B,C) 返回由 **'name'** 以及分布参数 **A**、**B** 和 **C** 指定的三参数分布族的 pdf，在 **x** 中的值处计算函数值。

y = pdf('name',x,A,B,C,D) 返回由 **'name'** 以及分布参数 **A**、**B**、**C** 和 **D** 指定的四参数分布族的 pdf，在 **x** 中的值处计算函数值。

y = pdf(pd,x) 返回概率分布对象 **pd** 的 pdf，在 **x** 中的值处计算函数值。

示例

计算正态分布 pdf

创建均值 μ 等于 0、标准差 σ 等于 1 的标准正态分布对象。

```
mu = 0;
sigma = 1;
pd = makedist('Normal','mu',mu,'sigma',sigma);
```

定义输入向量 x 以包含用于计算 pdf 的值。

```
x = [-2 -1 0 1 2];
```

计算标准正态分布在 x 中的值处的 pdf 值。

```
y = pdf(pd,x)
```

```
y = 1×5
```

```
    0.0540    0.2420    0.3989    0.2420    0.0540
```

y 中的每个值对应于输入向量 x 中的一个值。例如，在值 x 等于 1 处，y 中对应的 pdf 值等于 0.2420。

或者，不用创建概率分布对象，也可以计算此 pdf 值。使用 **pdf** 函数，再使用同样的 μ 和 σ 参数值指定一个标准正态分布。

```
y2 = pdf('Normal',x,mu,sigma)
```

```
y2 = 1×5
```

```
    0.0540    0.2420    0.3989    0.2420    0.0540
```

pdf 值与使用概率分布对象计算的值相同。

计算泊松分布 pdf

创建一个泊松分布对象，使用的速率参数 λ 等于 2。

```
lambda = 2;
pd = makedist('Poisson','lambda',lambda);
```

定义输入向量 x 以包含用于计算 pdf 的值。

```
x = [0 1 2 3 4];
```

计算泊松分布在 x 中的值处的 pdf 值。

```
y = pdf(pd,x)
```

```
y = 1×5
```

```
    0.1353    0.2707    0.2707    0.1804    0.0902
```

y 中的每个值对应于输入向量 x 中的一个值。例如，在值 x 等于 3 处，y 中对应的 pdf 值等于 0.1804。

或者，不用创建概率分布对象，也可以计算此 pdf 值。使用 **pdf** 函数，并使用相同的速率参数值指定泊松分布，λ。

```
y2 = pdf('Poisson',x,lambda)
```

```
y2 = 1×5
```

```
    0.1353    0.2707    0.2707    0.1804    0.0902
```

pdf 值与使用概率分布对象计算的值相同。

绘制标准正态分布的 pdf

创建一个标准正态分布对象。

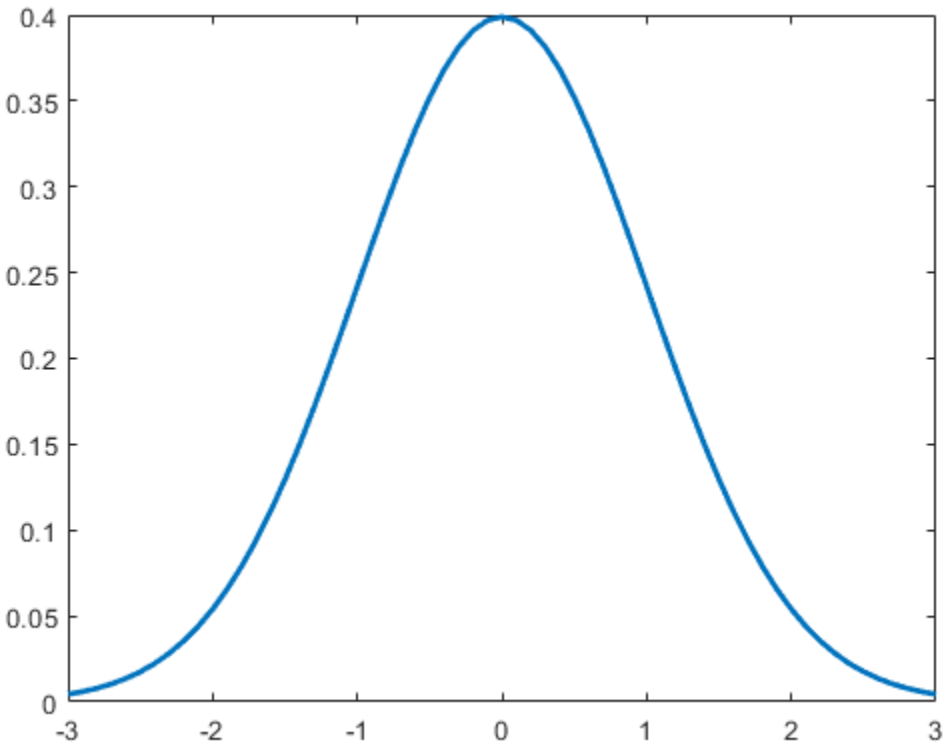
```
pd = makedist('Normal')

pd =
  NormalDistribution

  Normal distribution
     mu = 0
    sigma = 1
```

指定 x 值并计算 pdf。

```
x = -3:.1:3;
pdf_normal = pdf(pd,x);
```

绘制 pdf。

```
plot(x,pdf_normal,'LineWidth',2)
```



绘制 Weibull 分布的 pdf

创建 Weibull 概率分布对象。

```
pd = makedist('Weibull','a',5,'b',2)

pd =
  WeibullDistribution
```

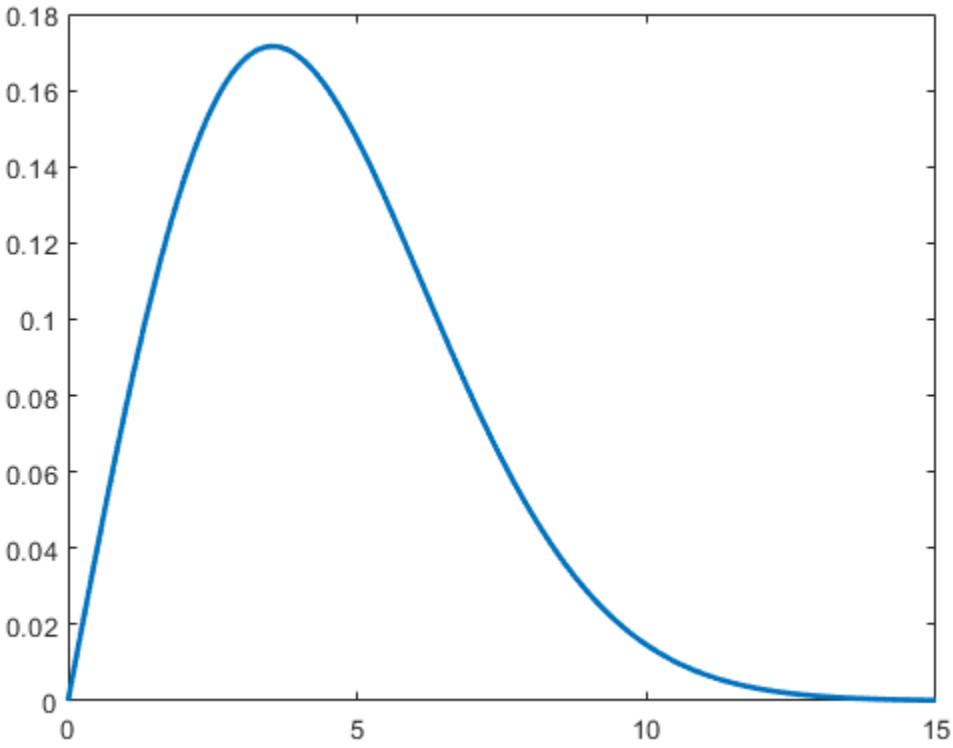
```
Weibull distribution
  A = 5
  B = 2
```

指定 x 值并计算 pdf。

```
x = 0:.1:15;
y = pdf(pd,x);
```

绘制 pdf。

```
plot(x,y,'LineWidth',2)
```



输入参数

'name' - 概率分布名称

概率分布名称的字符向量或字符串标量

概率分布名称，指定为下表中的概率分布名称之一。

'name'	分布	输入参数 A	输入参数 B	输入参数 C	输入参数 D
'Beta'	“Beta Distribution”	a 第一个形状参数	b 第二个形状参数	—	—
'Binomial'	“Binomial Distribution”	n 试验次数	p 每次试验成功的概率	—	—
'BirnbaumSaunders'	“Birnbaum-Saunders Distribution”	β 尺度参数	γ 形状参数	—	—
'Burr'	“Burr Type XII Distribution”	α 尺度参数	c 第一个形状参数	k 第二个形状参数	—
'Chisquare'	“Chi-Square Distribution”	ν 自由度	—	—	—
'Exponential'	“Exponential Distribution”	μ 均值	—	—	—
'Extreme Value'	“Extreme Value Distribution”	μ 位置参数	σ 尺度参数	—	—
'F'	“F Distribution”	ν_1 分子自由度	ν_2 分母自由度	—	—
'Gamma'	“Gamma Distribution”	a 形状参数	b 尺度参数	—	—
'Generalized Extreme Value'	“Generalized Extreme Value Distribution”	k 形状参数	σ 尺度参数	μ 位置参数	—
'Generalized Pareto'	“Generalized Pareto Distribution”	k 尾部指数（形状）参数	σ 尺度参数	μ 阈值（位置）参数	—
'Geometric'	“Geometric Distribution”	p 概率参数	—	—	—
'HalfNormal'	“Half-Normal Distribution”	μ 位置参数	σ 尺度参数	—	—
'Hypergeometric'	“Hypergeometric Distribution”	m 总体的大小	k 总体中具有所需特征的项数	n 抽取的样本数量	—
'InverseGaussian'	“Inverse Gaussian Distribution”	μ 尺度参数	λ 形状参数	—	—
'Logistic'	“Logistic Distribution”	μ 均值	σ 尺度参数	—	—
'LogLogistic'	“Loglogistic Distribution”	μ 对数值的均值	σ 对数值的尺度参数	—	—
'Lognormal'	“Lognormal Distribution”	μ 对数值的均值	σ 对数值的标准差	—	—
'Nakagami'	“Nakagami Distribution”	μ 形状参数	ω 尺度参数	—	—
'Negative Binomial'	“Negative Binomial Distribution”	r 成功次数	p 单个试验的成功概率	—	—
'Noncentral F'	“Noncentral F Distribution”	ν_1 分子自由度	ν_2 分母自由度	δ 非中心参数	—
'Noncentral t'	“Noncentral t Distribution”	ν 自由度	δ 非中心参数	—	—
'Noncentral Chi-square'	“Noncentral Chi-Square Distribution”	ν 自由度	δ 非中心参数	—	—
'Normal'	“正态分布” (第 B-2 页)	μ 均值	σ 标准差	—	—
'Poisson'	“泊松分布” (第 B-13 页)	λ 均值	—	—	—
'Rayleigh'	“Rayleigh Distribution”	b 尺度参数	—	—	—
'Rician'	“Rician Distribution”	s 非中心参数	σ 尺度参数	—	—

'name'	分布	输入参数 A	输入参数 B	输入参数 C	输入参数 D
'Stable'	“Stable Distribution”	α 第一个形状参数	β 第二个形状参数	γ 尺度参数	δ 位置参数
'T'	“Student's t Distribution”	ν 自由度	—	—	—
'tLocationScale'	“t Location-Scale Distribution”	μ 位置参数	σ 尺度参数	ν 形状参数	—
'Uniform'	“Uniform Distribution (Continuous)”	a 下部端点（最小值）	b 上部端点（最大值）	—	—
'Discrete Uniform'	“Uniform Distribution (Discrete)”	n 最大可观测值	—	—	—
'Weibull'	“Weibull Distribution”	a 尺度参数	b 形状参数	—	—

示例： 'Normal'

x - 用于计算 pdf 的值
标量值 | 标量值组成的数组

计算 pdf 时所基于的值，指定为标量值或标量值组成的数组。

如果输入参数 x、A、B、C 和 D 中的一个或多个是数组，则数组大小必须相同。在这种情况下，pdf 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 'name' 了解每个分布的 A、B、C 和 D 的定义。

示例： [-1,0,3,4]

数据类型： single | double

A - 第一概率分布参数
标量值 | 标量值组成的数组

第一概率分布参数，指定为标量值或标量值组成的数组。

如果输入参数 x、A、B、C 和 D 中的一个或多个是数组，则数组大小必须相同。在这种情况下，pdf 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 'name' 了解每个分布的 A、B、C 和 D 的定义。

数据类型： single | double

B - 第二概率分布参数
标量值 | 标量值组成的数组

第二概率分布参数，指定为标量值或标量值组成的数组。

如果输入参数 x、A、B、C 和 D 中的一个或多个是数组，则数组大小必须相同。在这种情况下，pdf 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 'name' 了解每个分布的 A、B、C 和 D 的定义。

数据类型： single | double

C - 第三概率分布参数
标量值 | 标量值组成的数组

第三概率分布参数，指定为标量值或标量值组成的数组。

如果输入参数 **x**、**A**、**B**、**C** 和 **D** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**pdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 **'name'** 了解每个分布的 **A**、**B**、**C** 和 **D** 的定义。

数据类型：**single** | **double**

D - 第四概率分布参数
标量值 | 标量值组成的数组

第四概率分布参数，指定为标量值或标量值组成的数组。

如果输入参数 **x**、**A**、**B**、**C** 和 **D** 中的一个或多个是数组，则数组大小必须相同。在这种情况下，**pdf** 将每个标量输入扩展为与数组输入大小相同的常量数组。请参阅 **'name'** 了解每个分布的 **A**、**B**、**C** 和 **D** 的定义。

数据类型：**single** | **double**

pd - 概率分布
概率分布对象

概率分布，指定为使用下表中的函数或 App 创建的概率分布对象。

函数或 App	说明
makedist	使用指定的参数值创建一个概率分布对象。
fitdist	对样本数据进行概率分布对象拟合。
分布拟合器	使用交互式分布拟合器对样本数据进行概率分布拟合，并将拟合的对象导出到工作区。
paretotails	创建一个分段分布对象，它在尾部具有广义帕累托分布。

输出参数

y - pdf 值
标量值 | 标量值组成的数组

pdf 值，以标量值或标量值数组的形式返回。在经过任何必要的标量扩展后，**y** 的大小与 **x** 相同。**y** 中的每个元素均为由分布参数（**A**、**B**、**C** 和 **D**）中的对应元素指定或由概率分布对象 (**pd**) 指定的分布的 pdf 值，其值在 **x** 中的对应元素处进行计算。

替代功能

- **pdf** 是泛型函数，它按名称 **'name'** 或概率分布对象 **pd** 接受分布。使用分布特有的函数更快，例如正态分布特有的 **normpdf**，二项分布特有的 **binopdf**。有关特定于分布的函数的列表，请参阅“Supported Distributions”。
- 使用 **Probability Distribution Function** App 为概率分布创建累积分布函数 (cdf) 或概率密度函数 (pdf) 的交互图。

扩展功能

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

- 输入参数 **'name'** 必须为编译时常量。例如，要使用正态分布，请将 **coder.Constant('Normal')** 包含在 **codegen** 的 **-args** 值中。
- 输入参数 **pd** 可以是 beta、指数、极值、对数正态、正态和 Weibull 分布的拟合概率分布对象。通过对来自 **fitdist** 函数的样本数据进行概率分布拟合，创建 **pd**。有关示例，请参阅“Code Generation for Probability Distribution Objects”。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “General Code Generation Workflow”。

另请参阅

cdf | **icdf** | **mle** | **random** | **makedist** | **fitdist** | **分布拟合器** | **paretotails**

主题

“Working with Probability Distributions”
“Supported Distributions”

在 R2006a 之前推出

pdist

成对观测值之间的两两距离

语法

```
D = pdist(X)
D = pdist(X,Distance)
D = pdist(X,Distance,DistParameter)
```

说明

D = pdist(X) 返回 **X** 中成对观测值之间的欧几里德距离。

D = pdist(X,Distance) 使用 **Distance** 指定的方法返回距离。

D = pdist(X,Distance,DistParameter) 使用 **Distance** 和 **DistParameter** 指定的方法返回距离。仅当 **Distance** 是 **'seuclidean'**、**'minkowski'** 或 **'mahalanobis'** 时，您才能指定 **DistParameter**。

示例

计算欧几里德距离并将距离向量转换为矩阵

计算成对观测值之间的欧几里德距离，并使用 **squareform** 将距离向量转换为矩阵。

创建包含三个观测值和两个变量的矩阵。

```
rng('default') % For reproducibility
X = rand(3,2);
```

计算欧几里德距离。

```
D = pdist(X)
```

```
D = 1×3

    0.2954    1.0670    0.9448
```

两两距离按 (2,1)、(3,1)、(3,2) 顺序排列。通过使用 **squareform**，您可以轻松定位观测值 **i** 和 **j** 之间的距离。

```
Z = squareform(D)
```

```
Z = 3×3

         0    0.2954    1.0670
    0.2954         0    0.9448
    1.0670    0.9448         0
```

squareform 返回一个对称矩阵，其中 **Z(i,j)** 对应于观测值 **i** 和 **j** 之间的两两距离。例如，您可以找到观测值 2 和 3 之间的距离。

```
Z(2,3)
```

```
ans = 0.9448
```

将 Z 传递给 squareform 函数，以重现 pdist 函数的输出。

```
y = squareform(Z)
```

```
y = 1×3

   0.2954    1.0670    0.9448
```

squareform 的输出 y 和 pdist 的输出 D 是相同的。

计算 Minkowski 距离

创建包含三个观测值和两个变量的矩阵。

```
rng('default') % For reproducibility
X = rand(3,2);
```

使用默认指数 2 计算 Minkowski 距离。

```
D1 = pdist(X,'minkowski')
```

```
D1 = 1×3

   0.2954    1.0670    0.9448
```

用指数 1 计算 Minkowski 距离，它等于 City block 距离。

```
D2 = pdist(X,'minkowski',1)
```

```
D2 = 1×3

   0.3721    1.5036    1.3136
```

```
D3 = pdist(X,'cityblock')
```

```
D3 = 1×3

   0.3721    1.5036    1.3136
```

使用自定义距离函数计算涉及缺失元素的两两距离

定义一个忽略 NaN 值坐标的自定义距离函数，并使用该自定义距离函数计算两两距离。

创建包含三个观测值和两个变量的矩阵。

```
rng('default') % For reproducibility
X = rand(3,2);
```

假设第一个观测值的第一个元素缺失。

```
X(1,1) = NaN;
```

计算欧几里德距离。

```
D1 = pdist(X)
```

```
D1 = 1×3

    NaN     NaN    0.9448
```

如果观测值 i 或 j 包含 NaN 值，函数 pdist 为 i 和 j 之间的两两距离返回 NaN。因此，D1(1) 和 D1(2)，即 (2,1) 和 (3,1) 之间的两两距离，是 NaN 值。

定义一个自定义距离函数 naneucdist，该函数忽略 NaN 值的坐标，并返回欧几里德距离。

```
function D2 = naneucdist(XI,XJ)
%NANEUCDIST Euclidean distance ignoring coordinates with NaNs
n = size(XI,2);
sqdx = (XI-XJ).^2;
nstar = sum(~isnan(sqdx),2); % Number of pairs that do not contain NaNs
nstar(nstar == 0) = NaN; % To return NaN if all pairs include NaNs
D2squared = sum(sqdx,2,'omitnan').*n./nstar; % Correction for missing coordinates
D2 = sqrt(D2squared);
```

将函数句柄作为输入参数传递给 pdist，以使用 naneucdist 计算该距离。

```
D2 = pdist(X,@naneucdist)
```

```
D2 = 1×3

  0.3974    1.1538    0.9448
```

输入参数

X - 输入数据

数值矩阵

输入数据，指定为大小是 m×n 的数值矩阵。行对应于单个观测值，列对应单个变量。

数据类型： single | double

Distance - 距离度量

字符向量 | 字符串标量 | 函数句柄

距离度量，指定为字符向量、字符串标量或函数句柄，如下表中所述。

值	说明
'euclidean'	欧几里德距离（默认值）。

值	说明
'squaredeuclidean'	欧几里德距离的平方。（此选项仅用于提高效率。它不满足三角不等式。）
'seuclidean'	标准化的欧几里德距离。每个观测值间坐标差都通过除以标准差 S = std(X,'omitnan') 中的对应元素来缩放。使用 DistParameter 为 S 指定另一个值。
'mahalanobis'	基于 X 的样本协方差 C = cov(X,'omitrows') 计算的马氏距离。使用 DistParameter 为 C 指定另一个值，其中矩阵 C 是对称正定矩阵。
'cityblock'	City block 距离。
'minkowski'	Minkowski 距离。默认指数是 2。使用 DistParameter 指定其他指数 P，其中 P 是表示指数的正标量值。
'chebychev'	Chebychev 距离（最大坐标差）。
'cosine'	1 减去点之间夹角的余弦值（视为向量）。
'correlation'	1 减去点之间的样本相关性（视为值序列）。
'hamming'	Hamming 距离，即相异坐标所占的百分比。
'jaccard'	1 减去 Jaccard 系数，即非零相异坐标所占的百分比。
'spearman'	1 减去样本观测值之间的 Spearman 秩相关（视为值序列）。

@distfun	自定义距离函数句柄。距离函数的形式如下
	
	`function D2 = distfun(ZI,ZJ)`
	`% calculation of distance`
	`...`
	
	其中
	
	• ZI 是包含单个观测值的 1×n 向量。
	• ZJ 是包含多个观测值的 m2×n 矩阵。distfun 必须接受具有任意数目的观测值的矩阵 ZJ。
	• D2 是距离的 m2×1 向量，D2(k) 是观测值 ZI 和 ZJ(k,:) 之间的距离。
	
	对于非稀疏数据，使用内置距离函数计算距离通常比使用函数句柄更快。

有关定义，请参阅“距离度量” （第 33-191 页）。

当您使用 'seuclidean'、'minkowski' 或 'mahalanobis' 时，您可以指定额外的输入参数 DistParameter 来控制这些度量。您也可以像使用其他度量一样来使用这些度量，但这种情况下使用的是 DistParameter 的默认值。

示例： 'minkowski'

DistParameter - 距离度量参数值
正标量 | 数值向量 | 数值矩阵

距离度量参数值，指定为正标量、数值向量或数值矩阵。仅当您将 Distance 指定为 'seuclidean'、'minkowski' 或 'mahalanobis' 时，此参数才有效。

- 如果 Distance 是 'seuclidean'，DistParameter 是对应于每个维度的缩放因子的向量，指定为正向量。默认值为 std(X,'omitnan')。

- 如果 **Distance** 是 **'minkowski'**，**DistParameter** 是 Minkowski 距离的指数，指定为正标量。默认值为 2。
- 如果 **Distance** 是 **'mahalanobis'**，**DistParameter** 是协方差矩阵，指定为数值矩阵。默认值为 **cov(X,'omitrows')**。**DistParameter** 必须是对称正定矩阵。

示例：**'minkowski',3**

数据类型：**single** | **double**

输出参数

D - 两两距离
数值行向量

两两距离，以长度为 m(m–1)/2 的数值行向量形式返回，对应于成对观测值，其中 m 是 **X** 中的观测值数目。

距离按 (2,1)、(3,1)、...、(m,1)、(3,2)、...、(m,2)、...、(m,m–1) 顺序排列，即按列向排列 m×m 距离矩阵的左下三角元素。观测值 i 和 j 之间的两两距离对应于 D((i-1)*(m-i/2)+j-i)，其中 i≤j。

您可以使用 **squareform** 函数将 **D** 转换为对称矩阵。**Z = squareform(D)** 返回 m×m 矩阵，其中 **Z(i,j)** 对应于观测值 i 和 j 之间的两两距离。

如果观测值 i 或 j 包含 **NaN**，则对于内置距离函数，**D** 中的对应值为 **NaN**。

D 通常在聚类或多维尺度分析中用作相异度矩阵。有关详细信息，请参阅“Hierarchical Clustering”以及 **cmdscale**、**cophenet**、**linkage**、**mdscale** 和 **optimalleaforder** 的函数参考页。这些函数接受 **D** 作为输入参数。

详细信息

距离度量

距离度量是定义两个观测值之间距离的函数。**pdist** 支持各种距离度量：欧几里德距离、标准化的欧几里德距离、马氏距离、City block 距离、Minkowski 距离、Chebychev 距离、余弦距离、相关性距离、Hamming 距离、Jaccard 距离和 Spearman 距离。

给定 m×n 数据矩阵 **X**，它被视为 m 个 (1×n) 行向量 x_1、x_2、...、x_m，向量 x_s 和 x_t 之间的各种距离定义如下：

- 欧几里德距离

  $$d_{st}^2 = (x_s - x_t)(x_s - x_t)'.$$

  欧几里德距离是 Minkowski 距离的特例，其中 p = 2。
- 标准化的欧几里德距离

  $$d_{st}^2 = (x_s - x_t)V^{-1}(x_s - x_t)',$$

  其中 V 是 n×n 对角矩阵，它的第 j 个对角线元素是 $(S(j))^2$，其中 S 是对应于每个维度的缩放因子的向量。
- 马氏距离

$$d_{st}^2 = (x_s - x_t)C^{-1}(x_s - x_t)',$$

其中 C 是协方差矩阵。

- City block 距离

$$d_{st} = \sum_{j=1}^{n} \left| x_{sj} - x_{tj} \right|.$$

City block 距离是 Minkowski 距离的特例，其中 p = 1。

- Minkowski 距离

$$d_{st} = \sqrt[p]{\sum_{j=1}^{n} \left| x_{sj} - x_{tj} \right|^p}.$$

对于特例 p = 1，Minkowski 距离即 City block 距离。对于特例 p = 2，Minkowski 距离即欧几里德距离。对于特例 p = ∞，Minkowski 距离即 Chebychev 距离。

- Chebychev 距离

$$d_{st} = \max_j \left\{ \left| x_{sj} - x_{tj} \right| \right\}.$$

Chebychev 距离是 Minkowski 距离的特例，其中 p = ∞。

- 余弦距离

$$d_{st} = 1 - \frac{x_s x'_t}{\sqrt{(x_s x'_s)(x_t x'_t)}}.$$

- 相关性距离

$$d_{st} = 1 - \frac{(x_s - \bar{x}_s)(x_t - \bar{x}_t)'}{\sqrt{(x_s - \bar{x}_s)(x_s - \bar{x}_s)'}\sqrt{(x_t - \bar{x}_t)(x_t - \bar{x}_t)'}},$$

其中

$\bar{x}_s = \frac{1}{n}\sum_j x_{sj}$ 且 $\bar{x}_t = \frac{1}{n}\sum_j x_{tj}$。

- Hamming 距离

$$d_{st} = \left(\#\left(x_{sj} \neq x_{tj}\right)/n \right).$$

- Jaccard 距离

$$d_{st} = \frac{\#\left[\left(x_{sj} \neq x_{tj}\right) \cap \left(\left(x_{sj} \neq 0\right) \cup \left(x_{tj} \neq 0\right)\right)\right]}{\#\left[\left(x_{sj} \neq 0\right) \cup \left(x_{tj} \neq 0\right)\right]}.$$

- Spearman 距离

$$d_{st} = 1 - \frac{(r_s - \bar{r}_s)(r_t - \bar{r}_t)'}{\sqrt{(r_s - \bar{r}_s)(r_s - \bar{r}_s)'}\sqrt{(r_t - \bar{r}_t)(r_t - \bar{r}_t)'}},$$

其中

- r_{sj} 是在 x_{1j}、x_{2j}、...x_{mj} 上所取的 x_{sj} 的秩，如 **tiedrank** 计算所得。
- r_s 和 r_t 是 x_s 和 x_t 的基于坐标轴的秩向量，即 $r_s = (r_{s1}, r_{s2}, \ldots r_{sn})$。
- $\bar{r}_s = \frac{1}{n}\sum_j r_{sj} = \frac{(n+1)}{2}$。
- $\bar{r}_t = \frac{1}{n}\sum_j r_{tj} = \frac{(n+1)}{2}$。

扩展功能

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

- 距离输入参数值 (**Distance**) 必须为编译时常量。例如，要使用 Minkowski 距离，请将 **coder.Constant('Minkowski')** 包含在 **codegen** 的 **-args** 值中。
- 距离输入参数值 (**Distance**) 不能为自定义距离函数。
- **pdist** 的生成代码使用 **parfor** 在生成的代码中创建可在受支持的共享内存多核平台上并行运行的循环。如果您的编译器不支持 Open Multiprocessing (OpenMP) 应用程序接口，或您禁用 OpenMP 库，则 MATLAB Coder 会将 **parfor** 循环视为 **for** 循环。要查找受支持的编译器，请参阅 **https://www.mathworks.com/support/compilers/current_release/**。要禁用 OpenMP 库，请将配置对象的 **EnableOpenMP** 属性设置为 **false**。有关详细信息，请参阅 **coder.CodeConfig**。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “General Code Generation Workflow”。

GPU 代码生成
使用 GPU Coder™ 为 NVIDIA® GPU 生成 CUDA® 代码。

使用说明和限制：

- 优化的 CUDA 代码支持的距离输入参数值 (**Distance**) 包括 **'euclidean'**、**'squaredeuclidean'**、**'seuclidean'**、**'cityblock'**、**'minkowski'**、**'chebychev'**、**'cosine'**、**'correlation'**、**'hamming'** 和 **'jaccard'**。
- **Distance** 不能为自定义距离函数。
- **Distance** 必须为编译时常量。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

使用说明和限制：

- 距离输入参数值 (**Distance**) 不能为自定义距离函数。

有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
cluster | **clusterdata** | **cmdscale** | **cophenet** | **dendrogram** | **inconsistent** | **linkage** | **pdist2** | **silhouette** | **squareform**

主题

“Choose Cluster Analysis Method”
“Hierarchical Clustering”

在 R2006a 之前推出

poissfit

泊松参数估计

语法

lambdahat = poissfit(data)
[lambdahat,lambdaci] = poissfit(data)
[lambdahat,lambdaci] = poissfit(data,alpha)

说明

lambdahat = poissfit(data) 基于给定数据 **data** 返回泊松分布参数 λ 的最大似然估计 (MLE) 值。

[lambdahat,lambdaci] = poissfit(data) 还在 **lambdaci** 中给出 95% 的置信区间。

[lambdahat,lambdaci] = poissfit(data,alpha) 给出 **100(1 - alpha)**% 的置信区间。例如，**alpha** = **0.001** 得出 99.9% 的置信区间。

样本均值是 λ 的 MLE。

$$\widehat{\lambda} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

示例

```
r = poissrnd(5,10,2);
[l,lci] = poissfit(r)
l =
   7.4000    6.3000
lci =
   5.8000    4.8000
   9.1000    7.9000
```

另请参阅

mle | **poisspdf** | **poisscdf** | **poissinv** | **poisstat** | **poissrnd**

主题
“泊松分布” (第 B-13 页)

在 R2006a 之前推出

prctile

数据集的百分位数

语法

```
Y = prctile(X,p)
Y = prctile(X,p,'all')
Y = prctile(X,p,dim)
Y = prctile(X,p,vecdim)
Y = prctile( ___ ,'Method',method)
```

说明

Y = prctile(X,p) 根据区间 [0,100] 中的百分比 **p** 返回数据向量或数组 **X** 中元素的百分位数。

- 如果 **X** 是向量，则 **Y** 是标量或向量，向量长度等于所请求百分位数的个数 (**length(p)**)。**Y(i)** 包含第 **p(i)** 个百分位数。
- 如果 **X** 是矩阵，则 **Y** 是行向量或矩阵，其中 **Y** 的行数等于所请求百分位数的个数 (**length(p)**)。**Y** 的第 **i** 行包含 **X** 的每一列的第 **p(i)** 个百分位数。
- 对于多维数组 （第 33-204 页），**prctile** 在 **X** 的第一个非单一维度 （第 33-204 页）上进行运算。

Y = prctile(X,p,'all') 返回 **X** 的所有元素的百分位数。

Y = prctile(X,p,dim) 返回运算维度 **dim** 上的百分位数。

Y = prctile(X,p,vecdim) 基于向量 **vecdim** 所指定的维度返回百分位数。例如，如果 **X** 是矩阵，则 **prctile(X,50,[1 2])** 返回 **X** 的所有元素的第 50 个百分位数，因为矩阵的每个元素都包含在由维度 1 和 2 定义的数组切片中。

Y = prctile(___ ,'Method',method) 使用上述任一语法中的输入参数组合，根据 **method** 的值，返回精确或近似百分位数。

示例

数据向量的百分位数

生成大小为 10 的数据集。

```
rng('default'); % for reproducibility
x = normrnd(5,2,1,10)
```

```
x = 1×10

   6.0753   8.6678   0.4823   6.7243   5.6375   2.3846   4.1328   5.6852  12.1568  10.5389
```

计算第 42 个百分位数。

```
Y = prctile(x,42)
```

```
Y = 5.6709
```

所有值的百分位数

计算数组中所有值的百分位数。

创建 3×5×2 数组 **X**。

```
X = reshape(1:30,[3 5 2])
```

```
X =
X(:,:,1) =

    1    4    7   10   13
    2    5    8   11   14
    3    6    9   12   15


X(:,:,2) =

   16   19   22   25   28
   17   20   23   26   29
   18   21   24   27   30
```

计算 **X** 的元素的第 40 个和第 60 个百分位数。

```
Y = prctile(X,[40 60],'all')
```

```
Y = 2×1

   12.5000
   18.5000
```

Y(1) 是 **X** 的第 40 个百分位数，**Y(2)** 是 **X** 的第 60 个百分位数。

数据矩阵的百分位数

沿数据矩阵的行和列计算与指定百分比对应的百分位数。

生成 5×5 数据矩阵。

```
X = (1:5)'*(2:6)
```

```
X = 5×5

    2    3    4    5    6
    4    6    8   10   12
    6    9   12   15   18
    8   12   16   20   24
   10   15   20   25   30
```

沿 **X** 的列计算第 25、50 和 75 个百分位数。

```
Y = prctile(X,[25 50 75],1)
```

```
Y = 3×5

   3.5000   5.2500   7.0000   8.7500  10.5000
   6.0000   9.0000  12.0000  15.0000  18.0000
   8.5000  12.7500  17.0000  21.2500  25.5000
```

Y 的每一列对应于 **X** 的每一列上的各个百分位数。例如，具有元素（4、8、12、16、20）的 **X** 的第三列的第 25、50 和 75 个百分位数分别为 7、12 和 17。**Y = prctile(X,[25 50 75])** 返回相同的百分位数矩阵。

沿 **X** 的行计算第 25、50 和 75 个百分位数。

```
Y = prctile(X,[25 50 75],2)
```

```
Y = 5×3

   2.7500   4.0000   5.2500
   5.5000   8.0000  10.5000
   8.2500  12.0000  15.7500
  11.0000  16.0000  21.0000
  13.7500  20.0000  26.2500
```

Y 的每一行对应于 **X** 的每一行上的各个百分位数。例如，具有元素（2、3、4、5、6）的 **X** 的第一行的第 25、50 和 75 个百分位数分别为 2.75、4 和 5.25。

多维数组的百分位数

同时沿多个维度计算一个多维数组的百分位数。

创建 3×5×2 数组 **X**。

```
X = reshape(1:30,[3 5 2])
```

```
X =
X(:,:,1) =

   1   4   7  10  13
   2   5   8  11  14
   3   6   9  12  15


X(:,:,2) =

  16  19  22  25  28
  17  20  23  26  29
  18  21  24  27  30
```

通过将维度 1 和 2 指定为运算维度，计算 **X** 的每页的第 40 和 60 个百分位数。

```
Ypage = prctile(X,[40 60],[1 2])
```

```
Ypage =
Ypage(:,:,1) =

    6.5000
    9.5000


Ypage(:,:,2) =

   21.5000
   24.5000
```

例如，**Ypage(1,1,1)** 是 **X** 的第一页的第 40 个百分位数，**Ypage(2,1,1)** 是 **X** 的第一页的第 60 个百分位数。

通过将维度 1 和 3 指定为运算维度，计算每个 **X(:,i,:)** 切片中元素的第 40 和 60 个百分位数。

```
Ycol = prctile(X,[40 60],[1 3])
```

```
Ycol = 2×5

    2.9000    5.9000    8.9000   11.9000   14.9000
   16.1000   19.1000   22.1000   25.1000   28.1000
```

例如，**Ycol(1,4)** 是 **X(:,4,:)** 中元素的第 40 个百分位数，**Ycol(2,4)** 是 **X(:,4,:)** 中元素的第 60 个百分位数。

tall 向量中与给定百分比对应的百分位数

计算 tall 列向量中与给定百分比对应的精确和近似百分位数。

当您对 tall 数组执行计算时，MATLAB® 使用并行池（如有 Parallel Computing Toolbox™ 则默认使用并行池）或本地 MATLAB 会话。如果您有 Parallel Computing Toolbox 但要使用本地 MATLAB 会话运行该示例，请使用 **mapreducer** 函数更改全局执行环境。

```
mapreducer(0)
```

为 **airlinesmall** 数据集创建数据存储。将 **'NA'** 值视为缺失数据，以便 **datastore** 用 **NaN** 值替换它们。指定使用 **ArrTime** 变量。

```
ds = datastore('airlinesmall.csv','TreatAsMissing','NA',...
   'SelectedVariableNames','ArrTime');
```

基于数据存储创建一个 tall 表，并将该 tall 表中的数据提取到一个 tall 向量中。

```
t = tall(ds) % Tall table
```

```
t =

  Mx1 tall table

    ArrTime
```

```
  735
 1124
 2218
 1431
  746
 1547
 1052
 1134
  :
  :
```

```
x = t{:,:}   % Tall vector
```

```
x =

  Mx1 tall double column vector

      735
     1124
     2218
     1431
      746
     1547
     1052
     1134
      :
      :
```

计算 **x** 的第 50 个精确百分位数。由于 **x** 是 tall 列向量，**p** 是标量，因此 **prctile** 默认情况下返回精确百分位数的值。

```
p = 50;
yExact = prctile(x,p)
```

```
yExact =

  tall double

    ?
```

计算 x 的第 50 个近似百分位数。指定 **'Method','approximate'** 使用基于 “T-digest” （第 33-204 页） 的逼近算法计算百分位数。

```
yApprox = prctile(x,p,'Method','approximate')
```

```
yApprox =

  MxNx... tall double array

    ?   ?   ?   ...
    ?   ?   ?   ...
    ?   ?   ?   ...
    :   :   :
    :   :   :
```

计算 tall 数组并使用 **gather** 将结果传入内存中。

```
[yExact,yApprox] = gather(yExact,yApprox)

Evaluating tall expression using the Local MATLAB Session:
- Pass 1 of 4: Completed in 1.2 sec
- Pass 2 of 4: Completed in 0.47 sec
- Pass 3 of 4: Completed in 0.65 sec
- Pass 4 of 4: Completed in 0.68 sec
Evaluation completed in 4 sec

yExact = 1522

yApprox = 1.5220e+03
```

近似百分位数和精确百分位数的值在精确到所示的四位数字时相同。

tall 矩阵中不同维度上的百分位数

沿不同维度计算 tall 矩阵中与指定百分比对应的精确和近似百分位数。

当您对 tall 数组执行计算时，MATLAB® 使用并行池（如有 Parallel Computing Toolbox™ 则默认使用并行池）或本地 MATLAB 会话。如果您有 Parallel Computing Toolbox 但要使用本地 MATLAB 会话运行该示例，请使用 **mapreducer** 函数更改全局执行环境。

```
mapreducer(0)
```

创建包含 **airlinesmall** 数据集中的变量子集的 tall 矩阵 **X**。有关从 tall 数组中提取数据的步骤的详细信息，请参阅“tall 向量中与给定百分比对应的百分位数” （第 33-199 页）。

```
varnames = {'ArrDelay','ArrTime','DepTime','ActualElapsedTime'}; % Subset of variables in the data set
ds = datastore('airlinesmall.csv','TreatAsMissing','NA',...
   'SelectedVariableNames',varnames); % Datastore
t = tall(ds);     % Tall table
X = t{:,varnames} % Tall matrix

X =

  Mx4 tall double matrix

       8      735      642       53
       8     1124     1021       63
      21     2218     2055       83
      13     1431     1332       59
       4      746      629       77
      59     1547     1446       61
       3     1052      928       84
      11     1134      859      155
       :        :        :        :
       :        :        :        :
```

当在非一维度上运算时，**prctile** 函数仅计算精确百分位数，因此它可以使用基于排序的算法（请参阅“算法” （第 33-206 页））高效地执行计算，而不是使用基于 “T-digest” （第 33-204 页） 的逼近算法。

计算 **X** 中第二个维度上的第 25、50 和 75 个精确百分位数。

```
p = [25 50 75]; % Vector of percentages
Yexact = prctile(X,p,2)
```

```
Yexact =

  MxNx... tall double array

    ?   ?   ?   ...
    ?   ?   ?   ...
    ?   ?   ?   ...
    :   :   :
    :   :   :
```

当函数在第一个维度上进行运算并且 **p** 是百分比向量时，您必须使用基于 T-digest 的逼近算法来计算百分位数。使用基于排序的算法计算 tall 数组中第一个维度上的百分位数会是密集型计算。

计算 **X** 中第一个维度上的第 25、50 和 75 个近似百分位数。由于默认维度为 1，您不需要为 **dim** 指定值。

```
Yapprox = prctile(X,p,'Method','approximate')
```

```
Yapprox =

  MxNx... tall double array

    ?   ?   ?   ...
    ?   ?   ?   ...
    ?   ?   ?   ...
    :   :   :
    :   :   :
```

计算 tall 数组并使用 **gather** 将结果传入内存中。

```
[Yexact,Yapprox] = gather(Yexact,Yapprox);
```

```
Evaluating tall expression using the Local MATLAB Session:
- Pass 1 of 1: Completed in 4 sec
Evaluation completed in 4.9 sec
```

显示 **X** 中第二个维度上的第 25、50 和 75 个精确百分位数的前五行。

```
Yexact(1:5,:)
```

```
ans = 5×3
10^3 ×

    0.0305    0.3475    0.6885
    0.0355    0.5420    1.0725
    0.0520    1.0690    2.1365
    0.0360    0.6955    1.3815
    0.0405    0.3530    0.6875
```

矩阵 **Yexact** 的每行都包含 **X** 中对应行的这三个百分位数。例如，**30.5**、**347.5** 和 **688.5** 分别是 **X** 中第一行的第 25、50 和 75 个百分位数。

显示 **X** 中第一个维度上的第 25、50 和 75 个近似百分位数。

```
Yapprox
```

```
Yapprox = 3×4
10^3 ×

  -0.0070    1.1150    0.9321    0.0700
        0    1.5221    1.3350    0.1020
   0.0110    1.9180    1.7400    0.1510
```

矩阵 **Yapprox** 的每列对应于矩阵 **X** 中每列的这三个百分位数。例如，**Yapprox** 的第一列包含元素 (–7, 0, 11)，表示 **X** 中第一列的百分位数。

输入参数

X - 输入数据

向量 | 数组

输入数据，指定为向量或数组。

数据类型： **double** | **single**

p - 百分比

标量 | 向量

计算百分位数所用的百分比，指定为 0 到 100 范围内的标量或由此种标量组成的向量。

示例： 25

示例： [25, 50, 75]

数据类型： **double** | **single**

dim - 维度

正整数

要计算 **X** 的百分位数的维度，指定为正整数。例如，对于矩阵 **X**，当 **dim** = 1 时，**prctile** 返回 **X** 的列的百分位数；当 **dim** = 2 时，**prctile** 返回 **X** 的行的百分位数。对于多维数组 **X**，**Y** 的第 **dim** 个维度的长度等于 **p** 的长度。

数据类型： **double** | **single**

vecdim - 维度向量

正整数向量

维度向量，指定为正整数向量。**vecdim** 的每个元素表示输入数组 **X** 的一个维度。输出 **Y** 在最小指定运算维度（即维度 **min(vecdim)**）上长度为 **length(p)**，在其余每个运算维度上长度为 1。**X** 和 **Y** 的其他维度长度相同。

例如，假设有 2×3×3 数组 **X**，**p = [20 40 60 80]**。在本例中，**prctile(X,p,[1 2])** 返回数组，数组的每页都包含 **X** 的对应页中元素的第 20、40、60 和 80 个百分位数。由于 1 和 2 是运算维度，**min([1 2]) = 1** 且 **length(p) = 4**，因此输出是 4×1×3 数组。

数据类型： **single** | **double**

method - 百分位数的计算方法

'exact' （默认） | **'approximate'**

百分位数的计算方法，指定为 **'exact'** 或 **'approximate'**。默认情况下，**prctile** 通过实现基于排序的算法 （第 33-206 页）返回精确百分位数。您也可以为 **prctile** 指定 **'method','approximate'**，以通过实现基于 T-Digest （第 33-204 页） 的算法返回近似百分位数。

数据类型： **char** | **string**

输出参数

Y - 百分位数
标量 | 数组

数据向量或数组的百分位数，返回为标量或包含一个或多个百分位数的值的数组。

- 如果 **X** 是向量，则 **Y** 是标量或长度为所请求百分位数的个数 (**length(p)**) 的向量。**Y(i)** 包含第 **p(i)** 个百分位数。
- 如果 **X** 是维度为 d 的数组，则 **Y** 是一个数组，其中最小运算维度的长度等于所请求百分位数的个数 (**length(p)**)。

详细信息

多维数组

多维数组是具有两个以上维度的数组。例如，如果 X 是 1×3×4 数组，则 X 是三维数组。

非单一维度

数组的非单一维度是其大小不等于 1 的维度。数组的第一个非单一维度是第一个满足非单一条件的维度。例如，如果 **X** 是 1×1×2×4 数组，则第三个维度是 **X** 的第一个非单一维度。

线性插值

线性插值使用线性多项式计算 $y_i = f(x_i)$，即基础函数 Y = f(X) 在向量或数组 x 中的点处的值。如有数据点 (x_1, y_1) 和 (x_2, y_2)，其中 $y_1 = f(x_1)$ 且 $y_2 = f(x_2)$，则线性插值根据下式计算 x_1 和 x_2 之间给定 x 处的 y = f(x)：

$$y = f(x) = y_1 + \frac{(x - x_1)}{(x_2 - x_1)}(y_2 - y_1).$$

同样，如果第 100(1.5/n) 个百分位数是 $y_{1.5/n}$ 且第 100(2.5/n) 个百分位数是 $y_{2.5/n}$，则线性插值根据下式计算第 100(2.3/n) 个百分位数 $y_{2.3/n}$：

$$y\frac{2.3}{n} = y\frac{1.5}{n} + \frac{\left(\frac{2.3}{n} - \frac{1.5}{n}\right)}{\left(\frac{2.5}{n} - \frac{1.5}{n}\right)}\left(y\frac{2.5}{n} - y\frac{1.5}{n}\right).$$

T-digest

T-digest[2] （第 33-206 页） 是一种概率数据结构体，它是数据集的经验累积分布函数 (CDF) 的稀疏表示。T-digest 可用于根据在线或分布式数据计算基于秩的统计量（例如百分位数和分位数）的逼近值，所采用的方式支持可控精度，特别是在数据分布的尾部附近。

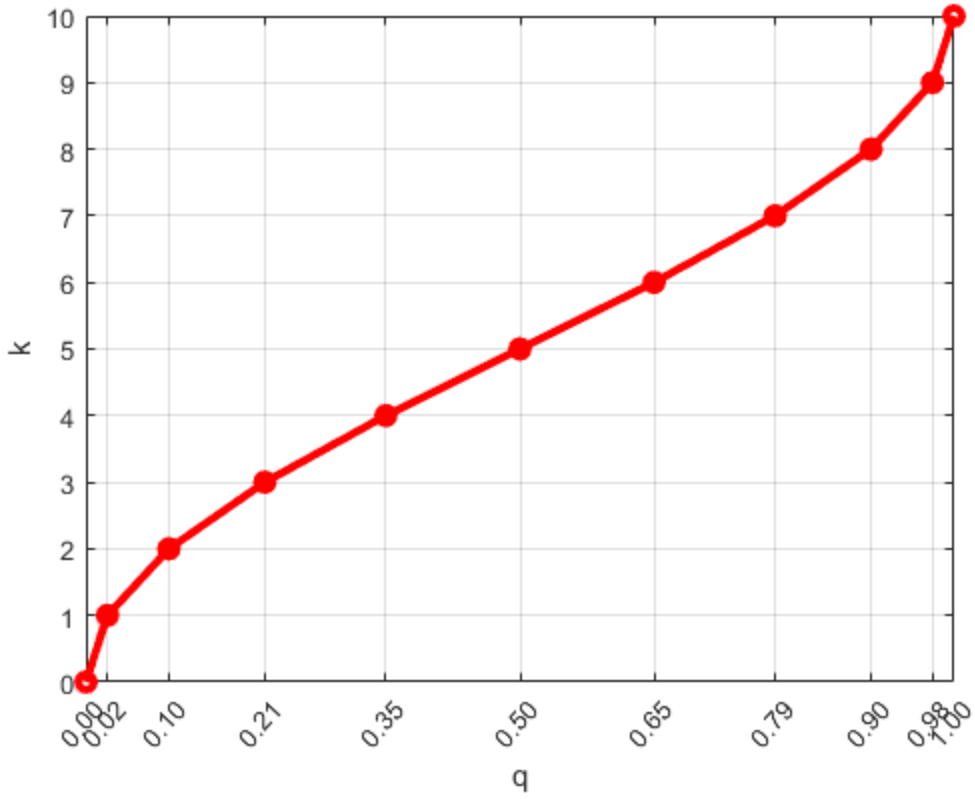
对于分布在不同分区中的数据，T-digest 会单独计算每个数据分区的分位数估计值（和百分位数估计值），然后合并这些估计值，同时保持恒定的内存界限和恒定的计算相对精度（对于第 q 个分位数，为 $q(1 - q)$）。由于这些原因，T-digest 在处理 tall 数组时非常实用。

要估计分布在不同分区中的数组的分位数，首先在数据的每个分区中构建一个 T-digest。T-digest 对分区中的数据进行聚类，并通过质心值和表示对聚类有贡献的样本数量的累积权重来汇总每个聚类。T-digest 使用大簇（质心间距大）表示 $\mathbf{q = 0.5}$ 附近的 CDF 区域，使用小簇（质心间距小）表示 $\mathbf{q = 0}$ 或 $\mathbf{q = 1}$ 附近的 CDF 区域。

T-digest 通过使用缩放函数控制簇大小，该函数使用压缩参数 δ 将分位数 q 映射到索引 k。即

$$k(q,\delta) = \delta \cdot \left(\frac{\sin^{-1}(2q-1)}{\pi} + \frac{1}{2} \right),$$

，其中映射 k 呈单调形态，最小值为 k(0,δ) = 0，最大值为 k(1,δ) = δ。下图显示了 δ = 10 的缩放函数。



缩放函数将分位数 q 转换为缩放因子 k，以便在 q 中给出可变大小的步。因此，簇大小不相等（中心分位数附近的簇较大，$\mathbf{q = 0}$ 或 $\mathbf{q = 1}$ 附近的簇较小）。较小的簇使得数据边缘附近的准确度更高。

要用具有权重和位置的新观测值更新 T-digest，请找到距离新观测值最近的簇。然后，添加权重，并基于加权平均值更新簇质心，前提是更新后的簇权重不超过大小限制。

您可以合并来自每个数据分区的独立 T-digest，方法是对这些 T-digest 求并集，并合并其质心。要合并 T-digest，请首先对来自所有独立 T-digest 的簇按簇权重降序排序。然后，合并满足大小限制的相邻簇，以形成一个新 T-digest。

一旦形成表示完整数据集的 T-digest，就可以估计 T-digest 中每个簇的端点（或边界），然后使用每个簇的端点之间的插值来计算准确的分位数估计值。

算法

对于 n 元素向量 **X**，**prctile** 使用基于排序的算法返回百分位数，如下所示：

1. **X** 中经过排序的元素被视为第 100(0.5/n) 个、第 100(1.5/n) 个、...、第 100([n – 0.5]/n) 个百分位数。例如：
   - 对于包含五个元素的数据向量，例如 {6, 3, 2, 10, 1}，排序后的元素 {1, 2, 3, 6, 10} 分别对应于第 10、30、50、70 和 90 个百分位数。
   - 对于包含六个元素的数据向量，例如 {6, 3, 2, 10, 8, 1}，排序后的元素 {1, 2, 3, 6, 8, 10} 分别对应于第 (50/6)、(150/6)、(250/6)、(350/6)、(450/6) 和 (550/6) 个百分位数。
2. **prctile** 使用线性插值 （第 33-204 页）计算与 100(0.5/n) 和 100([n – 0.5]/n) 之间的百分比对应的百分位数。
3. **prctile** 将 **X** 中元素的最小值或最大值赋给与该范围之外的百分比对应的百分位数。

prctile 将 **NaN** 视为缺失值，并将其删除。

参考

[1] Langford, E. “Quartiles in Elementary Statistics”，*Journal of Statistics Education*. Vol. 14, No. 3, 2006.

[2] Dunning, T., and O. Ertl. “Computing Extremely Accurate Quantiles Using T-Digests.” August 2017.

扩展功能

tall 数组
对行数太多而无法放入内存的数组进行计算。

使用说明和限制：

- 仅当 **X** 是 tall 列向量时，**Y = prctile(X,p)** 才会返回精确百分位数（使用基于排序的算法 （第 33-206 页））。
- 仅当满足下列条件之一时，**Y = prctile(X,p,dim)** 才会返回精确百分位数：
  - **X** 是 tall 列向量。
  - **X** 是 tall 数组，**dim** 不是 **1**。例如，**prctile(X,p,2)** 返回 tall 数组 **X** 中各行的精确百分位数。

  如果 **X** 是 tall 数组，**dim** 是 **1**，则您必须指定 **'Method','approximate'** 使用基于 T-digest （第 33-204 页） 的逼近算法来计算百分位数。例如，**prctile(X,p,1,'Method','approximate')** 返回 tall 数组 **X** 中各列的近似百分位数。
- 仅当满足下列条件之一时，**Y = prctile(X,p,vecdim)** 才会返回精确百分位数：
  - **X** 是 tall 列向量。
  - **X** 是 tall 数组，且 **vecdim** 不包含 **1**。例如，如果 **X** 是 3×5×2 数组，则 **prctile(X,p,[2,3])** 将返回每个 **X(i,:,:)** 切片中元素的精确百分位数。
  - **X** 是 tall 数组，且 **vecdim** 包含 **1** 和 **X** 的所有非单一维度 （第 33-204 页）。例如，如果 **X** 是 10×1×4 数组，则 **prctile(X,p,[1 3])** 将返回 **X(:,1,:)** 中元素的精确百分位数。

如果 **X** 是 tall 数组，**vecdim** 包含 **1** 但不包含 **X** 的所有非单一维度，则必须指定 **'Method','approximate'** 以使用逼近算法。例如，如果 **X** 是 10×1×4 数组，您可以使用 **prctile(X,p,[1 2],'Method','approximate')** 来计算 **X** 的每页的近似百分位数。

有关详细信息，请参阅“tall 数组”。

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

- 不支持 **'all'** 和 **vecdim** 输入参数。
- 不支持 **'Method'** 名称-值对组参数。
- **dim** 输入参数必须为编译时常量。
- 如果未指定 **dim** 输入参数，则生成的代码中的工作（或运算）维度可能不同。因此，可能会出现运行时错误。有关详细信息，请参阅“Automatic dimension restriction” (MATLAB Coder)。
- 如果输出 **Y** 是向量，则当以下所有条件都成立时，**Y** 的方向不同于 MATLAB：
  - 您未提供 **dim**。
  - **X** 在编译时是可变大小数组而不是可变大小向量，但 **X** 在运行时是向量。
  - 向量 **X** 的方向与向量 **p** 的方向不匹配。

  在这种情况下，输出 **Y** 匹配 **X** 的方向，而不是 **p** 的方向。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “General Code Generation Workflow”。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

使用说明和限制：

- 不支持 **'all'** 和 **vecdim** 输入参数。
- 不支持 **'Method'** 名称-值对组参数。

有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
quantile | **median** | **iqr**

主题
“Quantiles and Percentiles”

在 R2006a 之前推出

randsample

随机样本

语法

```
y = randsample(n,k)
y = randsample(population,k)
y = randsample( ___ ,replacement)
y = randsample(n,k,true,w)
y = randsample(population,k,true,w)
y = randsample(s, ___ )
```

说明

y = randsample(n,k) 返回从整数 **1** 到 **n** 中无放回随机均匀抽取的 **k** 个值。

y = randsample(population,k) 返回一个向量，其中包含从向量 **population** 的值中无放回随机均匀抽取的 **k** 个值。

y = randsample(___ ,replacement) 在 **replacement** 为 **true** 时返回一个有放回抽取的样本，在 **replacement** 为 **false** 时返回一个无放回抽取的样本。您可以在上述任一语法中的输入参数组合之后指定 **replacement**。

y = randsample(n,k,true,w) 使用长度为 **n** 的非负权重向量 **w** 来确定整数 **i** 被选为 **y** 的输入项的概率。

y = randsample(population,k,true,w) 使用与向量 **population** 长度相同的非负权重向量 **w** 来确定值 **population(i)** 被选为 **y** 的输入项的概率。

y = randsample(s, ___) 使用流 **s** 来生成随机数。选项 **s** 可以位于上述任一语法中的输入参数之前。**s** 是 **RandStream** 类的成员。

示例

从范围中抽取唯一值

从整数 **1** 到 **10** 中抽取一个值。

```
n = 10;
x = randsample(n,1)
```

```
x = 9
```

从总体向量中抽样

创建随机种子以实现结果的可再现性。

```
s = RandStream('mlfg6331_64');
```

从向量 **[10:20]** 中抽取一个值。

```
population = 10:20;
y = randsample(s,population,1)
```

```
y = 17
```

为指定的概率生成随机序列

创建随机数流以实现可再现性。

```
s = RandStream('mlfg6331_64');
```

根据指定的概率，从序列 **ACGT** 中有放回地随机选择 48 个字符。

```
R = randsample(s,'ACGT',48,true,[0.15 0.35 0.35 0.15])
```

```
R =
'GGCGGCGCAAGGCGCCGGACCTGGCTGCACGCCGTTCCCTGCTACTCG'
```

设置随机数流

创建随机数流以实现可再现性。

```
s = RandStream('mlfg6331_64');
```

从整数 **1:10** 中有放回地抽取五个值。

```
y = randsample(s,10,5,true)
```

```
y = 5×1

     7
     8
     5
     7
     8
```

输入参数

n - 范围的上限
正整数

抽样范围（**1** 到 **n**）的上限，指定为正整数。默认情况下，**randsample** 从 **1** 到 **n** 范围内的值中无放回随机均匀抽样。

数据类型： **single** | **double**

population - 输入数据
向量

要抽样的输入数据，指定为向量。默认情况下，**randsample** 从 **population** 的值中无放回随机均匀抽样。**y** 的方向（行或列）与 **population** 的方向相同。

如果 **population** 是仅包含非负整数值的数值向量，而 **population** 的长度可以是 1，则使用 **y = population(randsample(length(population),k))** 而不是 **y = randsample(population,k)**。

示例： **y = randsample([50:100],20)** 返回一个向量，其中包含从 **population** 向量（由整数 **50** 到 **100** 组成）中无放回随机均匀抽取的 **20** 个值。

数据类型： **single** | **double** | **logical** | **char** | **string** | **categorical**

k - 样本数量
正整数

样本数量，指定为正整数。

示例： **randsample(20,10)** 返回一个向量，其中包含从整数 **1** 到 **20** 中无放回随机均匀抽取的 **10** 个值。

数据类型： **single** | **double**

replacement - 有放回抽样的指示符
false （默认） | true

有放回抽样的指示符，指定为 **false** 或 **true**。

示例： **randsample(10,2,true)** 返回从整数 **1** 到 **10** 中有放回抽取的两个值。

数据类型： **logical**

w - 抽样权重
ones(n,1) （默认） | 非负标量值向量

抽样权重，指定为非负标量值向量。**w** 的长度必须等于要抽样的整数范围或 **population** 的长度。向量 **w** 必须有至少一个正值。如果 **w** 包含负值或 **NaN** 值，**randsample** 将显示一条错误消息。**randsample** 函数以与 **w(i)/sum(w)** 成正比的概率进行抽样。通常，**w** 是概率向量。**randsample** 函数仅支持为有放回抽样指定权重。

示例： **[0.1 0.5 0.35 0.46]**

数据类型： **single** | **double**

s - 随机数流
MATLAB 默认随机数流 （默认） | **RandStream**

随机数流，指定为 MATLAB 默认随机数流或 **RandStream**。有关详细信息，请参阅 “创建和控制随机数流” 。

示例： **s = RandStream('mlfg6331_64')** 使用乘法滞后 Fibonacci 生成器算法创建随机数流。

输出参数

y - 样本
向量 | 标量

样本，以向量或标量形式返回。

- 如果 **k = 1**，则 **y** 是标量。

- 如果 $\mathbf{k > 1}$，则 $\mathbf{y}$ 是 k×1 向量。

提示

要对数据进行随机抽样，无论是有放回还是无放回，请使用 **datasample**。

扩展功能

C/C++ 代码生成
使用 MATLAB® Coder™ 生成 C 代码和 C++ 代码。

使用说明和限制：

- 当您进行无放回抽样时，输出值的顺序可能与 MATLAB 中的顺序不匹配。
- 代码生成不支持随机数流输入参数 **s**。

有关代码生成的详细信息，请参阅 “Introduction to Code Generation” 和 “General Code Generation Workflow”。

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
datasample | **rand** | **randperm** | **RandStream**

在 R2006a 之前推出

raylfit

瑞利参数估计值

语法

raylfit(data,alpha)
[phat,pci] = raylfit(data,alpha)

说明

raylfit(data,alpha) 基于向量 **data** 中的给定数据返回瑞利分布参数的最大似然估计值。

[phat,pci] = raylfit(data,alpha) 基于给定数据返回最大似然估计值和 100(1 - **alpha**)% 置信区间。可选参数 **alpha** 的默认值为 0.05，对应于 95% 的置信区间。

另请参阅

mle | **raylpdf** | **raylcdf** | **raylinv** | **raylstat** | **raylrnd**

主题
"Rayleigh Distribution"

在 R2006a 之前推出

refline

将参考线添加到绘图中

语法

```
refline(m,b)
refline(coeffs)
refline
refline(ax, ___ )
hline = refline( ___ )
```

说明

refline(m,b) 在当前坐标区中添加一条具有斜率 **m** 和截距 **b** 的参考线。

refline(coeffs) 将由向量 **coeffs** 的元素定义的线添加到图窗中。

没有输入参数的 **refline** 等效于 **lsline**。

refline(ax, ___) 使用上述任一语法中的输入参数，向 **ax** 所指定坐标区中的图上添加一条参考线。

hline = refline(___) 使用上述任一语法中的输入参数，返回参考线对象 **hline**。在创建参考线后，使用 **hline** 修改其属性。有关属性列表，请参阅 Line Properties。

示例

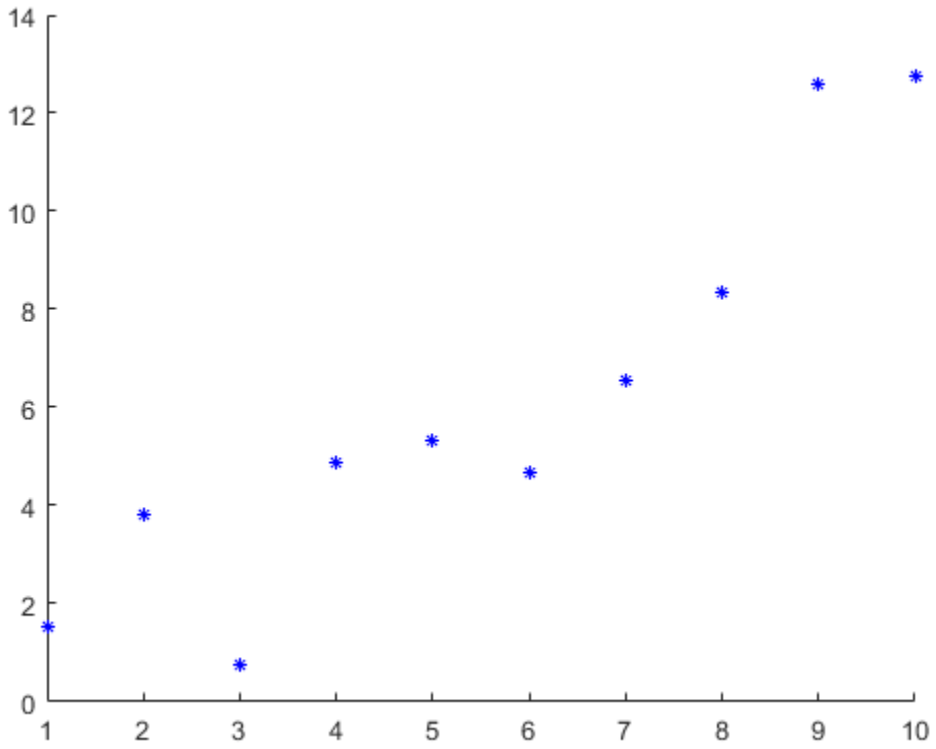
在均值处添加参考线

为自变量 **x** 和因变量 **y** 生成样本数据。

```
x = 1:10;
y = x + randn(1,10);
```

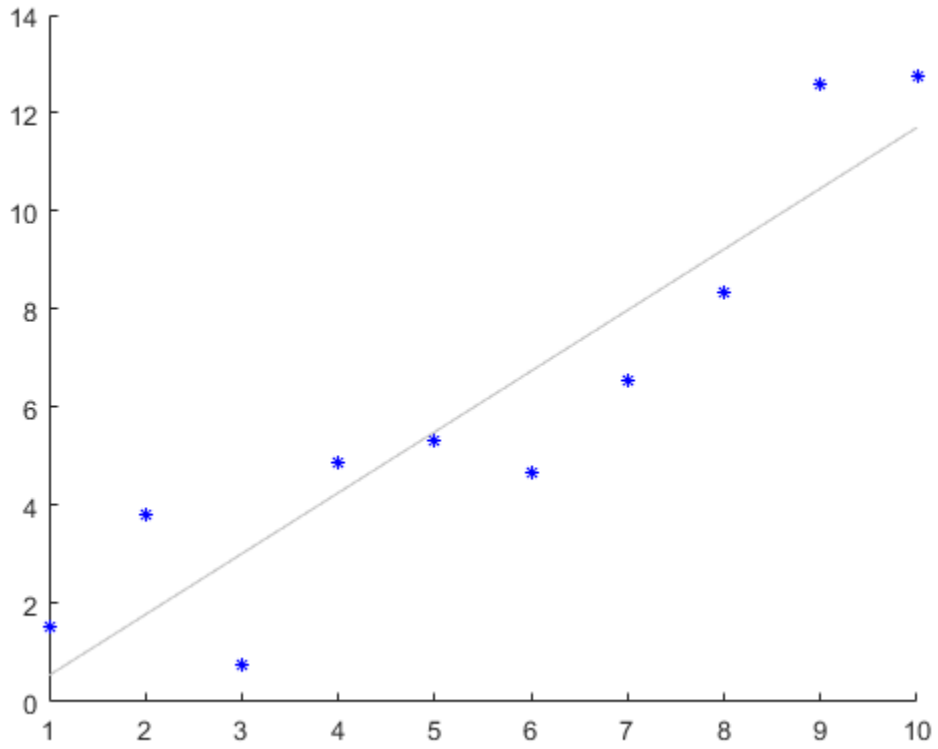
创建 **x** 和 **y** 的散点图。

```
scatter(x,y,25,'b','*')
```



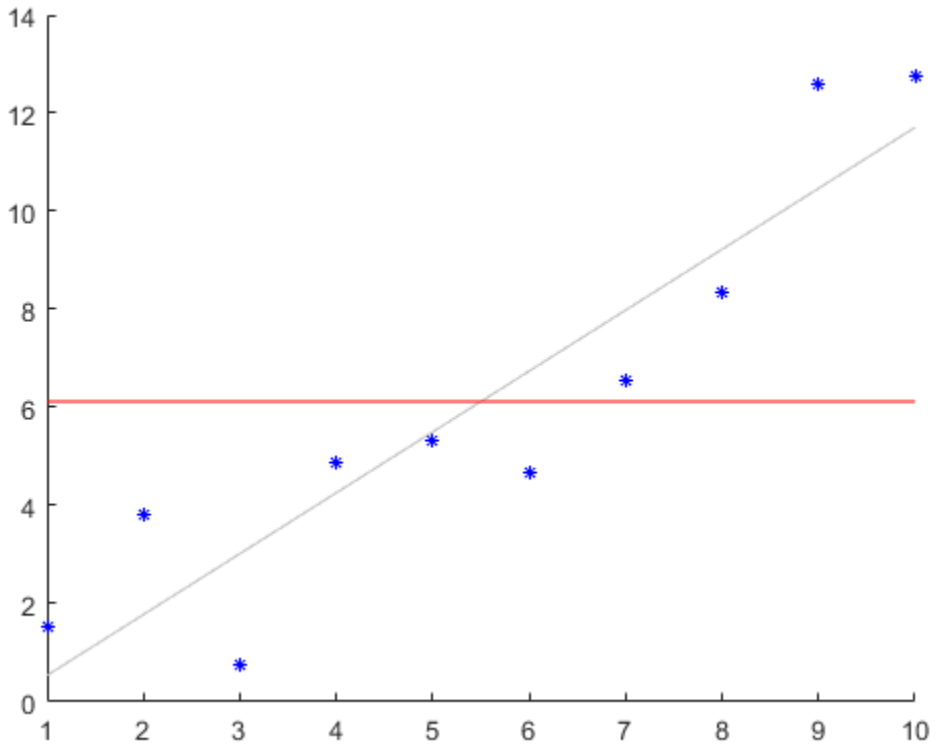
在散点图上叠加一条最小二乘线。

refline



在散点图的均值处添加一条参考线。

```
mu = mean(y);
hline = refline([0 mu]);
hline.Color = 'r';
```



红线是数据均值处的参考线。

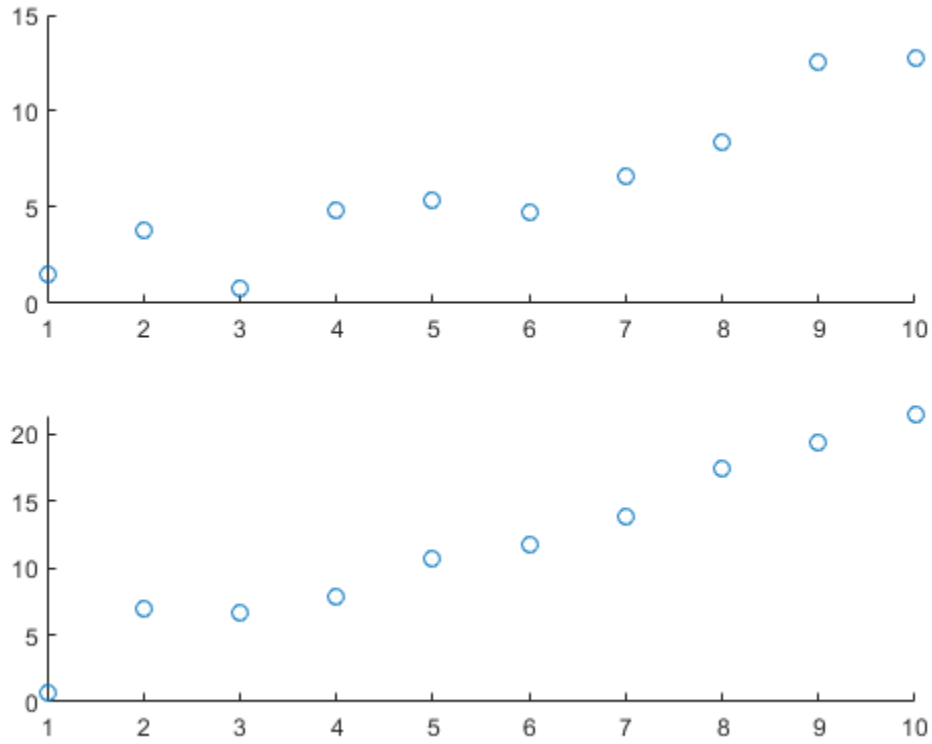
指定要添加最小二乘线和参考线的坐标区

定义用于绘图的 x 变量和两个不同 y 变量。

```
rng default  % For reproducibility
x = 1:10;
y1 = x + randn(1,10);
y2 = 2*x + randn(1,10);
```

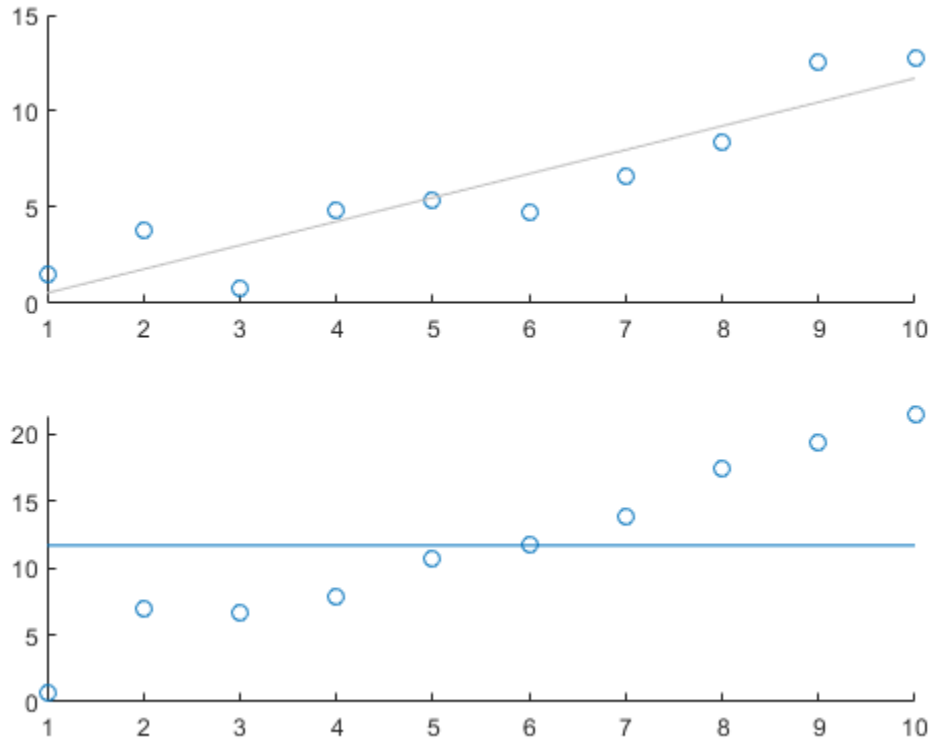
将 **ax1** 定义为图窗的上半部分，**ax2** 定义为图窗的下半部分。使用 **y1** 在顶部坐标区中创建第一个散点图，使用 **y2** 在底部坐标区中创建第二个散点图。

```
figure
ax1 = subplot(2,1,1);
ax2 = subplot(2,1,2);

scatter(ax1,x,y1)
scatter(ax2,x,y2)
```



在顶部绘图上叠加一条最小二乘线，在底部绘图上 y2 值的均值处叠加一条参考线。

```
lsline(ax1) % This is equivalent to refline(ax1)

mu = mean(y2);
refline(ax2,[0 mu])
```



输入参数

m - 参考线的斜率
数值标量

参考线的斜率，指定为数值标量。函数使用 **m** 来定义线

y = m*x + b.

示例： **refline(-1,1)**

数据类型： **single | double**

b - 参考线的截距
数值标量

参考线的截距，指定为数值标量。函数使用 **b** 来定义线

y = m*x + b.

示例： **refline(2,-10)**

数据类型： **single | double**

coeffs - 线性系数
长度为 2 的数值向量

线性系数，指定为长度为 2 的数值向量。**coeffs** 包含由下式定义的线的系数

y = coeffs(1)*x + coeffs(2).

示例： **refline([-1,2])**

数据类型： **single** | **double**

ax - 目标坐标区
gca（默认） | **axes** 对象

目标坐标区，指定为 **axes** 对象。如果不指定坐标区，且当前坐标区是笛卡尔坐标区，则 **refline** 函数将使用当前坐标区。

输出参数

hline - 一个或多个参考线对象
标量 | 向量

一个或多个参考线对象，以标量或向量的形式返回。这些对象是唯一标识符，可用于查询和修改特定参考线的属性。有关属性列表，请参阅 Chart Line。

另请参阅

refcurve | **lsline** | **gline**

在 R2006a 之前推出

regress

多元线性回归

语法

```
b = regress(y,X)
[b,bint] = regress(y,X)
[b,bint,r] = regress(y,X)
[b,bint,r,rint] = regress(y,X)
[b,bint,r,rint,stats] = regress(y,X)
[ ___ ] = regress(y,X,alpha)
```

说明

b = regress(y,X) 返回向量 **b**，其中包含向量 **y** 中的响应对矩阵 **X** 中的预测变量的多元线性回归的系数估计值。要计算具有常数项（截距）的模型的系数估计值，请在矩阵 **X** 中包含一个由 1 构成的列。

[b,bint] = regress(y,X) 还返回系数估计值的 95% 置信区间的矩阵 **bint**。

[b,bint,r] = regress(y,X) 还返回由残差组成的向量 **r**。

[b,bint,r,rint] = regress(y,X) 还返回矩阵 **rint**，其中包含可用于诊断离群值的区间。

[b,bint,r,rint,stats] = regress(y,X) 还返回向量 **stats**，其中包含 R^2 统计量、F 统计量及其 p 值，以及误差方差的估计值。矩阵 **X** 必须包含一个由 1 组成的列，以便软件正确计算模型统计量。

[___] = regress(y,X,alpha) 使用 **100*(1-alpha)**% 置信水平来计算 **bint** 和 **rint**。您可以指定上述任一语法中的输出参数组合。

示例

估计多元线性回归系数

加载 **carsmall** 数据集。确定权重和马力作为预测变量，里程作为响应。

```
load carsmall
x1 = Weight;
x2 = Horsepower;   % Contains NaN data
y = MPG;
```

计算具有交互效应项的线性模型的回归系数。

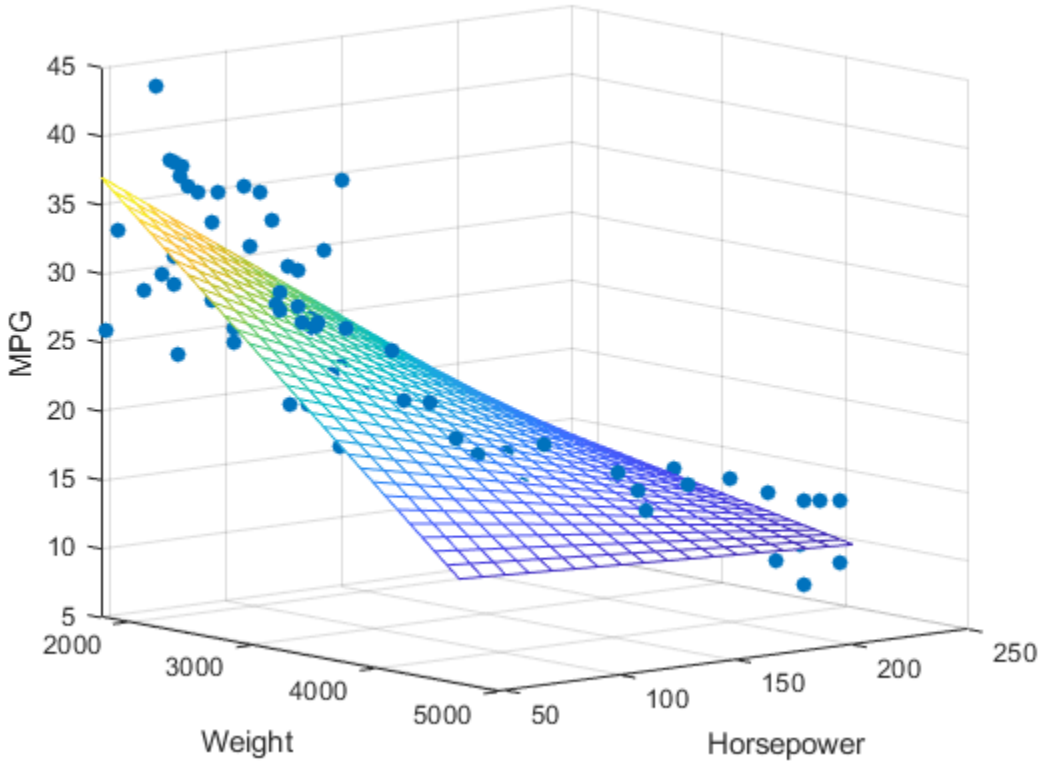
```
X = [ones(size(x1)) x1 x2 x1.*x2];
b = regress(y,X)   % Removes NaN data
```

```
b = 4×1

   60.7104
   -0.0102
   -0.1882
```

```
    0.0000
```

对数据和模型绘图。

```
scatter3(x1,x2,y,'filled')
hold on
x1fit = min(x1):100:max(x1);
x2fit = min(x2):10:max(x2);
[X1FIT,X2FIT] = meshgrid(x1fit,x2fit);
YFIT = b(1) + b(2)*X1FIT + b(3)*X2FIT + b(4)*X1FIT.*X2FIT;
mesh(X1FIT,X2FIT,YFIT)
xlabel('Weight')
ylabel('Horsepower')
zlabel('MPG')
view(50,10)
hold off
```



使用残差诊断离群值

加载 **examgrades** 数据集。

```
load examgrades
```

使用最后一次考试分数作为响应数据，前两次考试分数作为预测变量数据。

```
y = grades(:,5);
X = [ones(size(grades(:,1))) grades(:,1:2)];
```

用 alpha = 0.01 执行多元线性回归。

```
[~,~,r,rint] = regress(y,X,0.01);
```

通过计算不包含 0 的残差区间 **rint** 来诊断离群值。

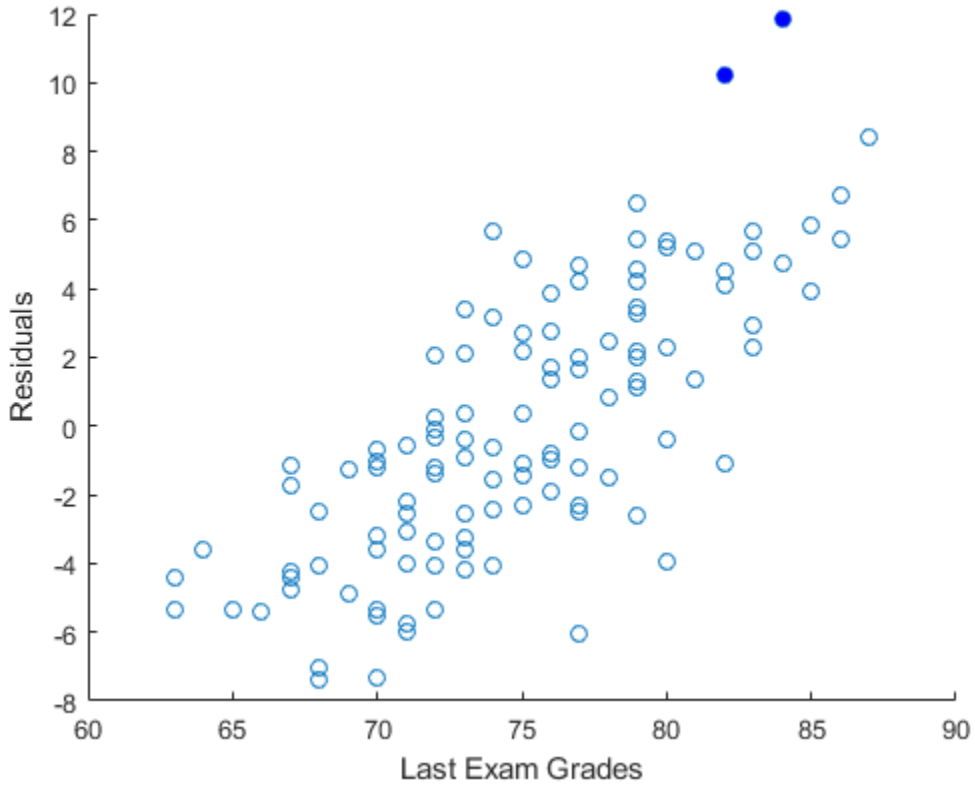
```
contain0 = (rint(:,1)<0 & rint(:,2)>0);
idx = find(contain0==false)
```

```
idx = 2×1

    53
    54
```

观测值 **53** 和 **54** 是可能的离群值。

创建残差的散点图。填充与离群值对应的点。

```
hold on
scatter(y,r)
scatter(y(idx),r(idx),'b','filled')
xlabel("Last Exam Grades")
ylabel("Residuals")
hold off
```



确定线性回归关系的显著性

加载 **hald** 数据集。使用 **heat** 作为响应变量，使用 **ingredients** 作为预测变量数据。

```
load hald
y = heat;
X1 = ingredients;
x1 = ones(size(X1,1),1);
X = [x1 X1];    % Includes column of ones
```

执行多元线性回归并生成模型统计量。

```
[~,~,~,~,stats] = regress(y,X)
```

```
stats = 1×4

   0.9824  111.4792    0.0000    5.9830
```

由于 R^2 值 **0.9824** 接近于 1，p 值 **0.0000** 小于 0.05 的默认显著性水平，因此响应 **y** 和 **X** 中的预测变量之间存在显著的线性回归关系。

输入参数

y - 响应数据
数值向量

响应数据，指定为 n×1 数值向量。**y** 的行对应于不同观测值。**y** 的行数必须与 **X** 的行数相同。

数据类型：**single** | **double**

X - 预测变量数据
数值矩阵

预测变量数据，指定为 n×p 数值矩阵。**X** 的行对应于各个观测值，列对应于预测变量。**X** 的行数必须与 **y** 的行数相同。

数据类型：**single** | **double**

alpha - 显著性水平
0.05（默认）| 正标量

显著性水平，指定为正标量。**alpha** 必须介于 0 和 1 之间。

数据类型：**single** | **double**

输出参数

b - 多元线性回归的系数估计值
数值向量

多元线性回归的系数估计值，以数值向量形式返回。**b** 是 p×1 向量，其中 p 是 **X** 中预测变量的数目。如果 **X** 的列是线性相关的，**regress** 会将 **b** 的最大元素数设置为零。

数据类型： **double**

bint - 系数估计值的置信边界下限和置信边界上限
数值矩阵

系数估计值的置信边界下限和置信边界上限，以数值矩阵形式返回。**bint** 是 p×2 矩阵，其中 p 是 **X** 中预测变量的数目。**bint** 的第一列包含每个系数估计值的置信边界下限；第二列包含置信边界上限。如果 **X** 的列是线性相关的，则 **regress** 为 **bint** 中对应于 **b** 中零元素的元素返回零。

数据类型： **double**

r - 残差
数值向量

残差，以数值向量形式返回。**r** 是 n×1 向量，其中 n 是 **X** 中的观测值数目或行数。

数据类型： **single** | **double**

rint - 用于诊断离群值的区间
数值矩阵

用于诊断离群值的区间，以数值矩阵形式返回。**rint** 是 n×2 矩阵，其中 n 是 **X** 中的观测值数目或行数。如果观测值 **i** 的 **rint(i,:)** 区间不包含零，则对应的残差大于 **100*(1-alpha)**% 的新观测值的预期残差，即表明存在离群值。有关详细信息，请参阅“算法” （第 33-224 页）。

数据类型： **single** | **double**

stats - 模型统计量
数值向量

模型统计量，以数值向量形式返回，包括 R^2 统计量、F 统计量及其 p 值，以及误差方差的估计值。

- **X** 必须包含一个由 1 组成的列，以便在模型中包含常数项。F 统计量及其 p 值是在此假设下计算的，对于没有常量的模型则不正确。
- F 统计量是对回归模型的 F 检验的检验统计量。F 检验寻找响应变量和预测变量之间的显著线性回归关系。
- 对于没有常量的模型，R^2 统计量可能是负数，表明模型不适用于数据。

数据类型： **single** | **double**

提示

- **regress** 将 **X** 或 **y** 中的 **NaN** 值视为缺失值。**regress** 在回归拟合中忽略具有缺失值的观测值。

算法

残差区间

在线性模型中，观测到的 **y** 的值及其残差是随机变量。残差呈现具有零均值的正态分布，但视预测变量值的不同而具有不同方差。为了将残差放在可比较的尺度上，**regress** 将残差 Student 化。也就是说，**regress** 将残差除以独立于其值的标准差估计值。Student 化残差具有已知自由度的 t 分布。**rint** 返回的各个区间是这些 t 分布的 **100*(1-alpha)**% 置信区间移至以残差为中心的结果。

替代功能

在您只需要函数的输出参数以及要在循环中多次重复拟合模型时，**regress** 非常有用。如果您需要进一步研究拟合后的回归模型，请使用 **fitlm** 或 **stepwiselm** 创建线性回归模型对象 **LinearModel**。**LinearModel** 对象提供的功能比 **regress** 更多。

- 使用 **LinearModel** 的属性来研究拟合线性回归模型。对象属性包括关于系数估计值、汇总统计量、拟合方法和输入数据的信息。
- 使用 **LinearModel** 的对象函数来预测响应以及修改、计算和可视化线性回归模型。
- 与 **regress** 不同，**fitlm** 函数不要求输入数据包含一个由 1 组成的列。由 **fitlm** 创建的模型始终包含截距项，除非您使用 **'Intercept'** 名称-值对组参数指定不包含它。
- 使用 **LinearModel** 的属性和对象函数，您可以在 **regress** 的输出中找到信息。

regress 的输出	**LinearModel 中的等效值**
b	请查看 **Coefficients** 属性的 **Estimate** 列。
bint	请使用 **coefCI** 函数。
r	请查看 **Residuals** 属性的 **Raw** 列。
rint	不支持。在这种情况下，请使用 Student 化残差（**Residuals** 属性）和观测值诊断（**Diagnostics** 属性）来查找离群值。
stats	请查看命令行窗口中的模型显示。您可以使用 **anova** 函数以及在模型属性（**MSE** 和 **Rsquared**）中找到这些统计量。

参考

[1] Chatterjee, S., and A. S. Hadi. "Influential Observations, High Leverage Points, and Outliers in Linear Regression." *Statistical Science*. Vol. 1, 1986, pp. 379–416.

另请参阅

LinearModel | **fitlm** | **stepwiselm** | **mvregress** | **rcoplot**

主题

"Interpret Linear Regression Results"
"Linear Regression Workflow"

在 R2006a 之前推出

ttest

单样本和配对样本 t 检验

语法

```
h = ttest(x)

h = ttest(x,y)
h = ttest(x,y,Name,Value)

h = ttest(x,m)
h = ttest(x,m,Name,Value)

[h,p] = ttest( ___ )
[h,p,ci,stats] = ttest( ___ )
```

说明

h = ttest(x) 使用单样本 t 检验 （第 33-230 页）返回原假设的检验决策，该原假设假定 **x** 中的数据来自均值等于零且方差未知的正态分布。备择假设是总体分布的均值不等于零。如果检验在 5% 的显著性水平上拒绝原假设，则结果 **h** 为 **1**，否则为 **0**。

h = ttest(x,y) 使用配对样本 t 检验返回针对原假设的检验决策，该原假设假定 **x – y** 中的数据来自均值等于零且方差未知的正态分布。

h = ttest(x,y,Name,Value) 返回配对样本 t 检验的检验决策，其中使用由一个或多个名称-值对组参数指定附加选项。例如，您可以更改显著性水平或进行单侧检验。

h = ttest(x,m) 返回针对原假设的检验决策，该原假设假定 **x** 中的数据来自均值为 **m** 且方差未知的正态分布。备择假设是均值不为 **m**。

h = ttest(x,m,Name,Value) 返回单样本 t 检验的检验决策，其中使用一个或多个名称-值对组参数指定附加选项。例如，您可以更改显著性水平或进行单侧检验。

[h,p] = ttest(___) 还使用上述语法组中的任何输入参数返回检验的 p 值 **p**。

[h,p,ci,stats] = ttest(___) 还返回 **x**（对于配对 t 检验则为 **x – y**）的均值的置信区间 **ci**，以及包含检验统计量信息的结构体 **stats**。

示例

均值等于零的 t 检验

加载样本数据。创建一个包含股票收益数据第三列的向量。

```
load stockreturns
x = stocks(:,3);
```

检验原假设，即样本数据来自均值等于零的总体。

```
[h,p,ci,stats] = ttest(x)

h = 1

p = 0.0106

ci = 2×1

   -0.7357
   -0.0997


stats = struct with fields:
    tstat: -2.6065
       df: 99
       sd: 1.6027
```

返回值 **h = 1** 表示 **ttest** 在 5% 显著性水平上拒绝了原假设。

不同显著性水平的 t 检验

加载样本数据。创建一个包含股票收益数据第三列的向量。

```
load stockreturns
x = stocks(:,3);
```

检验原假设，即样本数据来自在 1% 显著性水平上均值等于零的总体。

```
h = ttest(x,0,'Alpha',0.01)

h = 0
```

返回值 **h = 0** 表示 **ttest** 在 1% 显著性水平上未拒绝原假设。

配对样本 t 检验

加载样本数据。创建包含数据矩阵第一列和第二列的向量，以表示学生在两次考试中的成绩。

```
load examgrades
x = grades(:,1);
y = grades(:,2);
```

检验原假设，即数据向量 **x** 和 **y** 之间的成对差异的均值等于零。

```
[h,p] = ttest(x,y)

h = 0

p = 0.9805
```

返回值 **h = 0** 表明 **ttest** 在默认的 5% 显著性水平上未拒绝原假设。

不同显著性水平上的配对样本 t 检验

加载样本数据。创建包含数据矩阵第一列和第二列的向量，以表示学生在两次考试中的成绩。

```
load examgrades
x = grades(:,1);
y = grades(:,2);
```

检验原假设，即数据向量 x 和 y 之间的成对差异在 1% 显著性水平上的均值等于零。

```
[h,p] = ttest(x,y,'Alpha',0.01)
```

```
h = 0
```

```
p = 0.9805
```

返回值 h = 0 表明 ttest 在 1% 显著性水平上未拒绝原假设。

假设均值的 t 检验

加载样本数据。创建包含学生考试成绩数据的第一列的向量。

```
load examgrades
x = grades(:,1);
```

检验原假设，即样本数据来自均值 m = 75 的分布。

```
h = ttest(x,75)
```

```
h = 0
```

返回值 h = 0 表明 ttest 在 5% 显著性水平上未拒绝原假设。

单侧 t 检验

加载样本数据。创建包含学生考试成绩数据的第一列的向量。

```
load examgrades
x = grades(:,1);
```

检验原假设，即数据来自均值等于 65 的总体，而备择假设的均值大于 65。

```
h = ttest(x,65,'Tail','right')
```

```
h = 1
```

返回值 h = 1 表明 ttest 在 5% 显著性水平上拒绝了原假设，而支持备择假设，即数据来自均值大于 65 的总体。

输入参数

x - 样本数据
向量 | 矩阵 | 多维数组

样本数据，指定为向量、矩阵或多维数组 （第 33-231 页）。**ttest** 对每列执行单独的 t 检验并返回结果向量。如果指定 **y** 样本数据，**x** 和 **y** 必须大小相同。

数据类型： **single** | **double**

y - 样本数据
向量 | 矩阵 | 多维数组

样本数据，指定为向量、矩阵或多维数组 （第 33-231 页）。如果指定 **y** 样本数据，**x** 和 **y** 必须大小相同。

数据类型： **single** | **double**

m - 假设的总体均值
0 （默认） | 标量值

假设的总体均值，指定为标量值。

数据类型： **single** | **double**

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例： **'Tail','right','Alpha',0.01** 在 1% 显著性水平上进行右尾假设检验。

Alpha - 显著性水平
0.05 （默认） | 范围 (0,1) 内的标量值

假设检验的显著性水平，指定为以逗号分隔的对组，其中包含 **'Alpha'** 和范围 (0,1) 内的一个标量值。

示例： **'Alpha',0.01**

数据类型： **single** | **double**

Dim - 维度
第一个非单一维度 （默认） | 正整数值

用于检验均值的输入矩阵的维度，指定为以逗号分隔的对组，其中包含 **'Dim'** 和一个正整数值。例如，指定 **'Dim',1** 检验列均值，而 **'Dim',2** 检验行均值。

示例： **'Dim',2**

数据类型： **single** | **double**

Tail - 备择假设的类型
'both' （默认） | **'right'** | **'left'**

要计算的备择假设的类型，指定为以逗号分隔的对组，其中包含 **'Tail'** 和以下项之一：

- **'both'** - 检验总体均值不为 **m** 的备择假设。

- **'right'** - 检验总体均值大于 **m** 的备择假设。
- **'left'** - 检验总体均值小于 **m** 的备择假设。

ttest 根据指定的备择假设检验总体均值为 **m** 的原假设。

示例： **'Tail','right'**

输出参数

h - 假设检验结果
1 | **0**

假设检验结果，返回为 **1** 或 **0**。

- 如果 **h = 1**，这表明在 **Alpha** 显著性水平上拒绝原假设。
- 如果 **h = 0**，这表明未能在 **Alpha** 显著性水平上拒绝原假设。

p - p 值
范围 [0,1] 内的标量值

检验的 p 值，以 [0,1] 范围内的标量值形式返回。**p** 是观测到的检验统计量与原假设下观测到的值一样极端或更极端的概率。**p** 值较小会让人对原假设的有效性产生怀疑。

ci - 置信区间
向量

真实总体均值的置信区间，以二元素向量形式返回，其中包含 100 × (1 – **Alpha**)% 置信区间的上下边界。

stats - 检验统计量
结构体

检验统计量，以包含以下内容的结构体形式返回：

- **tstat** - 检验统计量的值。
- **df** - 检验的自由度。
- **sd** - 估计的总体标准差。对于配对 t 检验，**sd** 是 **x** – **y** 的标准差。

详细信息

单样本 t 检验

单样本 t 检验是当总体标准差未知时位置参数的参数化检验。

检验统计量的计算公式为

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}},$$

，其中 $\bar{x}$ 是样本均值，**μ** 是假设的总体均值，s 是样本标准差，n 是样本大小。在原假设下，检验统计量具有 Student t 分布和 n – 1 个自由度。

多维数组

多维数组有两个以上的维度。例如，如果 **x** 是 1×3×4 数组，则 **x** 是三维数组。

第一个非单一维度

第一个非单一维度是其大小不等于 1 的数组的第一个维度。例如，如果 **x** 是 1×2×3×4 数组，则第二个维度是 **x** 的第一个非单一维度。

提示

- 使用 **sampsizepwr** 计算：
  - 对应于指定检验功效和参数值的样本大小；
  - 给定真实参数值时，特定样本大小的检验功效；
  - 可用指定的样本大小和检验功效检测的参数值。

扩展功能

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
ztest | **ttest2** | **sampsizepwr**

在 R2006a 之前推出

ttest2

双样本 t 检验

语法

h = ttest2(x,y)
h = ttest2(x,y,Name,Value)
[h,p] = ttest2(___)
[h,p,ci,stats] = ttest2(___)

说明

h = ttest2(x,y) 使用双样本 t 检验 （第 33-235 页）返回原假设的检验决策，该原假设假定向量 **x** 和 **y** 中的数据来自均值相等、方差相同但未知的正态分布的独立随机样本。备择假设是 **x** 和 **y** 中的数据来自均值不相等的总体。如果检验在 5% 的显著性水平上拒绝原假设，则结果 **h** 为 **1**，否则为 **0**。

h = ttest2(x,y,Name,Value) 返回针对双样本 t 的检验决策，该检验使用由一个或多个名称-值对组参数指定的附加选项。例如，您可以更改显著性水平或进行无需假设方差齐性的检验。

[h,p] = ttest2(___) 还使用上述语法中的任何输入参数返回检验的 p 值 **p**。

[h,p,ci,stats] = ttest2(___) 还返回总体均值差的置信区间 **ci**，以及包含检验统计量信息的结构体 **stats**。

示例

等均值的双样本 t 检验

加载数据集。创建包含数据矩阵第一列和第二列的向量，以表示学生在两次考试中的成绩。

```
load examgrades
x = grades(:,1);
y = grades(:,2);
```

检验原假设，即两个数据样本来自均值相等的总体。

```
[h,p,ci,stats] = ttest2(x,y)
```

```
h = 0

p = 0.9867

ci = 2×1

   -1.9438
    1.9771


stats = struct with fields:
    tstat: 0.0167
       df: 238
```

```
    sd: 7.7084
```

返回值 **h = 0** 表明 **ttest2** 在默认的 5% 显著性水平上未拒绝原假设。

不假设方差齐性的等均值的 t 检验

加载数据集。创建包含数据矩阵第一列和第二列的向量，以表示学生在两次考试中的成绩。

```
load examgrades
x = grades(:,1);
y = grades(:,2);
```

检验原假设，即两个数据向量来自均值相等的总体，而不假设总体还具有方差齐性。

```
[h,p] = ttest2(x,y,'Vartype','unequal')
```

```
h = 0
```

```
p = 0.9867
```

返回值 **h = 0** 表明 **ttest2** 在默认的 5% 显著性水平上未拒绝原假设，即使没有假设方差齐性也是如此。

输入参数

x - 样本数据
向量 | 矩阵 | 多维数组

样本数据，指定为向量、矩阵或多维数组。**ttest2** 将 **NaN** 值视为缺失数据，并忽略它们。

- 如果 **x** 和 **y** 指定为向量，则它们不需要长度相同。
- 如果将 **x** 和 **y** 指定为矩阵，则它们必须具有相同的列数。**ttest2** 对每列执行单独的 t 检验并返回结果向量。
- 如果将 **x** 和 **y** 指定为多维数组 （第 33-236 页），则除了第一个非单一维度 （第 33-236 页）外，其他维度必须具有相同的大小。

数据类型： **single** | **double**

y - 样本数据
向量 | 矩阵 | 多维数组

样本数据，指定为向量、矩阵或多维数组。**ttest2** 将 **NaN** 值视为缺失数据，并忽略它们。

- 如果 **x** 和 **y** 指定为向量，则它们不需要长度相同。
- 如果将 **x** 和 **y** 指定为矩阵，则它们必须具有相同的列数。**ttest2** 对每列执行单独的 t 检验并返回结果向量。
- 如果将 **x** 和 **y** 指定为多维数组 （第 33-236 页），则除了第一个非单一维度 （第 33-236 页）外，其他维度必须具有相同的大小。**ttest2** 基于第一个非单一维度进行检验。

数据类型： **single** | **double**

名称-值对组参数

指定可选的、以逗号分隔的 **Name,Value** 对组参数。**Name** 为参数名称，**Value** 为对应的值。**Name** 必须放在引号内。您可采用任意顺序指定多个名称-值对组参数，如 **Name1,Value1,...,NameN,ValueN** 所示。

示例： **'Tail','right','Alpha',0.01,'Vartype','unequal'** 指定在 1% 显著性水平上的右尾检验，并且不假设 **x** 和 **y** 具有相等的总体方差。

Alpha - 显著性水平
0.05 （默认） | 范围 (0,1) 内的标量值

假设检验的显著性水平，指定为以逗号分隔的对组，其中包含 **'Alpha'** 和范围 (0,1) 内的一个标量值。

示例： **'Alpha',0.01**

数据类型： **single** | **double**

Dim - 维度
第一个非单一维度 （默认） | 正整数值

用于检验均值的输入矩阵的维度，指定为以逗号分隔的对组，其中包含 **'Dim'** 和一个正整数值。例如，指定 **'Dim',1** 检验列均值，而 **'Dim',2** 检验行均值。

示例： **'Dim',2**

数据类型： **single** | **double**

Tail - 备择假设的类型
'both' （默认） | **'right'** | **'left'**

要计算的备择假设的类型，指定为以逗号分隔的对组，其中包含 **'Tail'** 和以下项之一：

- **'both'** - 检验总体均值不相等的备择假设。
- **'right'** - 检验 **x** 的总体均值大于 **y** 的总体均值的备择假设。
- **'left'** - 检验 **x** 的总体均值小于 **y** 的总体均值。

ttest2 根据指定的备择假设检验总体均值相等的原假设。

示例： **'Tail','right'**

Vartype - 方差类型
'equal' （默认） | **'unequal'**

方差类型，指定为以逗号分隔的对组，其中包含 **'Vartype'** 和以下项之一。

'equal'	假设 **x** 和 **y** 来自方差未知但具有齐性的正态分布，并对此假设进行检验。
'unequal'	假设 **x** 和 **y** 来自方差未知且不具有齐性的正态分布，并对此假设进行检验。这称为 Behrens-Fisher 问题。**ttest2** 使用 Satterthwaite 逼近计算有效自由度。

Vartype 必须为单一方差类型，即使 **x** 是矩阵或多维数组也是如此。

示例： **'Vartype','unequal'**

输出参数

h - 假设检验结果
1 | 0

假设检验结果，返回为 **1** 或 **0**。

- 如果 **h = 1**，这表明在 **Alpha** 显著性水平上拒绝原假设。
- 如果 **h = 0**，这表明未能在 **Alpha** 显著性水平上拒绝原假设。

p - p 值
范围 [0,1] 内的标量值

检验的 p 值，以 [0,1] 范围内的标量值形式返回。**p** 是观测到的检验统计量与原假设下观测到的值一样极端或更极端的概率。**p** 值较小会让人对原假设的有效性产生怀疑。

ci - 置信区间
向量

x 和 **y** 的总体均值差的置信区间，以二元素向量形式返回，其中包含 100 × (1 – **Alpha**)% 置信区间的上下边界。

stats - 检验统计量
结构体

双样本 t 检验的检验统计量，以包含以下内容的结构体形式返回：

- **tstat** - 检验统计量的值。
- **df** - 检验的自由度。
- **sd** - 总体标准差的合并估计（方差具有齐性时），或包含总体标准差的非合并估计的向量（方差具有非齐性时）。

详细信息

双样本 t 检验

双样本 t 检验是一种参数化检验，它比较两个独立数据样本的位置参数。

检验统计量的计算公式为：

$$t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_x^2}{n} + \frac{s_y^2}{m}}},$$

其中 $\bar{x}$ 和 $\bar{y}$ 是样本均值，s_x 和 s_y 是样本标准差，n 和 m 是样本大小。

在假设两个数据样本来自具有方差齐性的总体的情况下，原假设下的检验统计量具有自由度为 n + m – 2 的 Student t 分布，样本标准差被替换为池化标准差

$$s = \sqrt{\frac{(n-1)s_x^2 + (m-1)s_y^2}{n+m-2}}.$$

在不假设两个数据样本来自具有方差齐性的总体的情况下，原假设下的检验统计量具有近似 Student t 分布，其自由度的数目由 Satterthwaite 逼近给出。此检验有时称为 Welch t 检验。

多维数组

多维数组有两个以上的维度。例如，如果 **x** 是 1×3×4 数组，则 **x** 是三维数组。

第一个非单一维度

第一个非单一维度是其大小不等于 1 的数组的第一个维度。例如，如果 **x** 是 1×2×3×4 数组，则第二个维度是 **x** 的第一个非单一维度。

提示

- 使用 **sampsizepwr** 计算：
  - 对应于指定检验功效和参数值的样本大小；
  - 给定真实参数值时，特定样本大小的检验功效；
  - 可用指定的样本大小和检验功效检测的参数值。

扩展功能

GPU 数组
通过使用 Parallel Computing Toolbox™ 在图形处理单元 (GPU) 上运行来加快代码执行。

此函数完全支持 GPU 数组。有关详细信息，请参阅“Run MATLAB Functions on a GPU” (Parallel Computing Toolbox)。

另请参阅
ttest | **ztest** | **sampsizepwr**

在 R2006a 之前推出

分类学习器

使用有监督的机器学习训练模型以对数据进行分类

说明

分类学习器会训练模型，以对数据进行分类。使用此 App，您可以使用各种分类器来探索有监督机器学习。您可以探查数据，选择特征，指定验证方案，训练模型和评估结果。您可以执行自动训练来搜索最佳分类模型类型，包括决策树、判别分析、支持向量机、逻辑回归、最近邻、朴素贝叶斯、集成和神经网络分类。

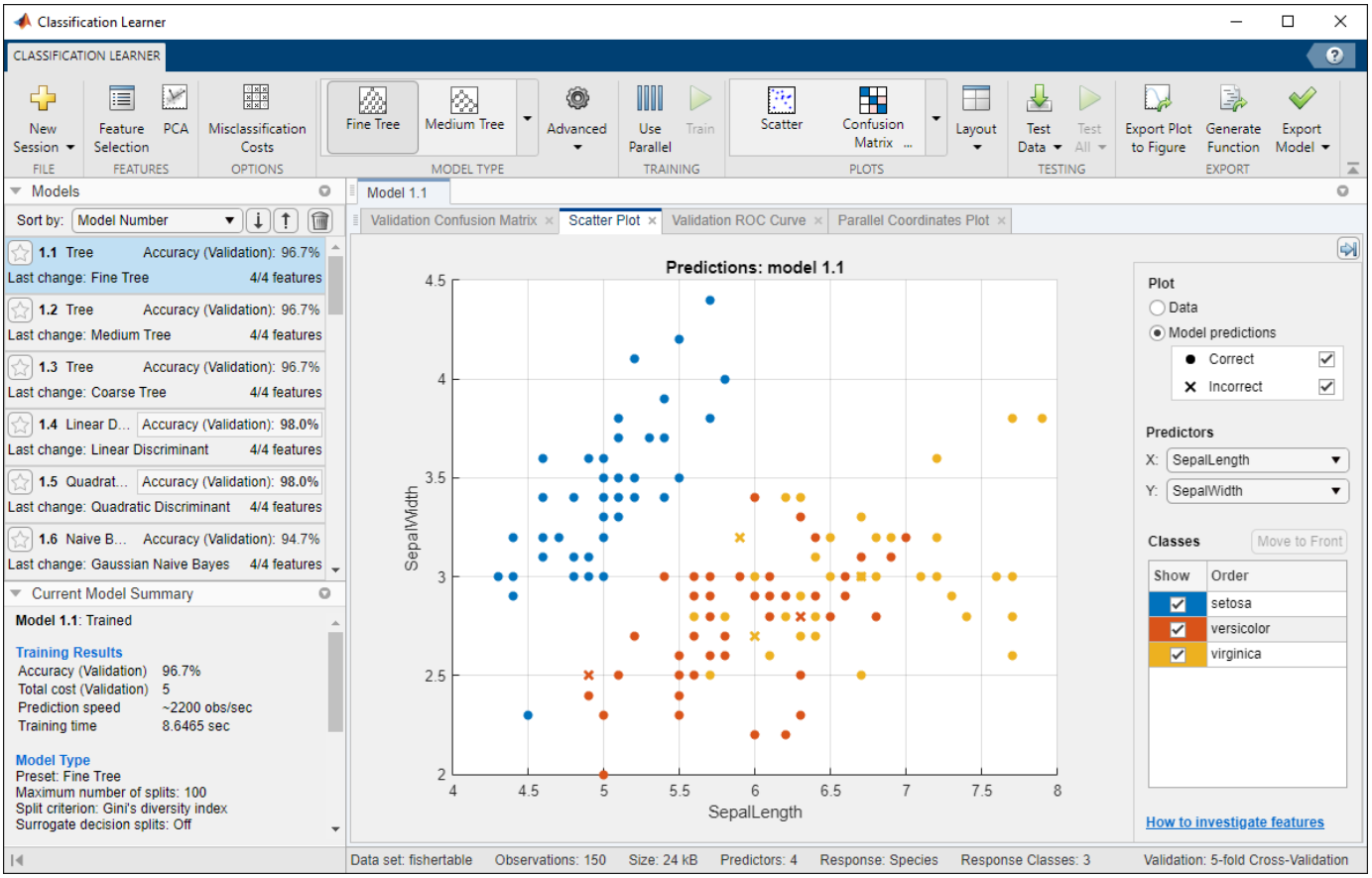
您可以通过提供一组已知的输入数据（观测值或示例）和对数据的已知响应（例如标签或类）来执行有监督的机器学习。您可以使用这些数据来训练模型，该模型可以为对新数据的响应生成预测。要将模型与新数据结合使用或要了解编程式分类，您可以将模型导出到工作区或生成 MATLAB 代码来重新创建经过训练的模型。

提示 要开始使用，请在分类器列表中，尝试用 **全部(快速训练)** 来训练所选模型。请参阅“Automated Classifier Training”。

需要的产品

- MATLAB
- Statistics and Machine Learning Toolbox

注意：当您在 MATLAB Online™ 中使用分类学习器时，您可以使用云中心集群并行训练模型（需要 Parallel Computing Toolbox）。有关详细信息，请参阅“Use Parallel Computing Toolbox with Cloud Center Cluster in MATLAB Online” (Parallel Computing Toolbox)。



打开 分类学习器 App

- MATLAB 工具条：在 **Apps** 选项卡上的 **Machine Learning** 下，点击该 App 的图标。
- MATLAB 命令提示符：输入 `classificationLearner`。

示例

- “Train Classification Models in Classification Learner App”
- “Select Data and Validation for Classification Problem”
- “Automated Classifier Training”
- “Feature Selection and Feature Transformation Using Classification Learner App”
- “Choose Classifier Options”
- “Assess Classifier Performance in Classification Learner”
- “Export Classification Model to Predict New Data”

编程用途

`classificationLearner` 打开分类学习器，或将焦点放在该 App 上（如果该 App 已打开）。

classificationLearner(Tbl,ResponseVarName) 打开分类学习器，并使用表 **Tbl** 中包含的数据填充“从参数新建会话”对话框。指定为字符向量或字符串标量的 **ResponseVarName** 参数是 **Tbl** 中包含类标签的响应变量的名称。**Tbl** 中的其余变量是预测变量。

classificationLearner(Tbl,Y) 打开分类学习器，并使用表 **Tbl** 中的预测变量和向量 **Y** 中的类标签填充“从参数新建会话”对话框。您可以将响应 **Y** 指定为分类数组、字符数组、字符串数组、逻辑向量、数值向量或字符向量元胞数组。

classificationLearner(X,Y) 打开分类学习器，并使用向量 **Y** 中的 n×p 预测变量矩阵 **X** 和 n 类标签填充“来自参数的新会话”对话框。**X** 的每行对应一个观测值，每列对应一个变量。**Y** 的长度和 **X** 的行数必须相等。

classificationLearner(___ ,Name,Value) 使用一个或多个名称-值参数以及上述语法中的任何输入参数组合来指定交叉验证选项。例如，您可以指定 **'KFold',10** 使用 10 折交叉验证方案。

- **'CrossVal'**，指定为 **'on'**（默认值）或 **'off'**，是交叉验证标志。如果您指定 **'on'**，则该 App 使用 5 折交叉验证。如果您指定 **'off'**，则该 App 将使用再代入验证。

  您可以使用 **'Holdout'** 或 **'KFold'** 名称-值参数来覆盖 **'CrossVal'** 交叉验证设置。一次只能指定其中一个参数。
- **'Holdout'**，指定为 [0.05,0.5] 范围内的数值标量，该值是用于留出法验证的数据的比例。该 App 使用其余数据进行训练。
- **'KFold'**，指定为 [2,50] 范围内的正整数，用于交叉验证的折数。

另请参阅

App
回归学习器

函数
fitctree | **fitcdiscr** | **fitcsvm** | **fitcecoc** | **fitcknn** | **fitcensemble** | **fitcnet** | **fitglm**

主题
“Train Classification Models in Classification Learner App”
“Select Data and Validation for Classification Problem”
“Automated Classifier Training”
“Feature Selection and Feature Transformation Using Classification Learner App”
“Choose Classifier Options”
“Assess Classifier Performance in Classification Learner”
“Export Classification Model to Predict New Data”

在 R2015a 中推出

A

样本数据集

样本数据集

Statistics and Machine Learning Toolbox 软件包括下表中的示例数据集。

要将数据集加载到 MATLAB 工作区中，请键入：

```
load filename
```

其中 **filename** 是表中列出的文件之一。

数据集包含单独的数据变量、具有引用的描述变量以及封装数据集及其描述的数据集数组（如果适用）。

文件	数据集的描述
acetylene.mat	具有相关预测变量的化学反应数据
arrhythmia.mat	来自 UCI 机器学习存储库的心律失常数据
carbig.mat	汽车的测量值，1970-1982
carsmall.mat	**carbig.mat** 的子集。汽车的测量值，1970、1976、1982
census1994.mat	来自 UCI 机器学习存储库的成人数据
cereal.mat	早餐谷物成分
cities.mat	美国大都市地区的生活质量评分
discrim.mat	用于判别分析的 **cities.mat** 版本
examgrades.mat	0-100 分的考试成绩
fisheriris.mat	Fisher 1936 年的鸢尾花数据
flu.mat	Google 流感趋势估计的美国不同地区的 ILI（流感样疾病）百分比，疾病预防控制中心根据哨点提供商报告对 ILI 百分比进行了加权
gas.mat	1993 年马萨诸塞州的汽油价格
hald.mat	水泥发热与原料混合
hogg.mat	牛奶的不同配送方式中的细菌数量
hospital.mat	仿真的医疗数据
humanactivity.mat	五种活动的人体活动识别数据：坐、站、走、跑和跳舞
imports-85.mat	1985 年来自 UCI 存储库的自动导入数据库
ionosphere.mat	来自 UCI 机器学习存储库的电离层数据集
kmeansdata.mat	四维聚类数据
lawdata.mat	15 所法学院的平均分数和 LSAT 分数
mileage.mat	两家工厂的三种汽车型号的里程数据
moore.mat	关于五个预测变量的生化需氧量
morse.mat	非编码人员对摩尔斯电码的识别情况
nlpdata.mat	从 MathWorks® 文档中提取的自然语言处理数据。
ovariancancer.mat	关于 4000 个预测变量的分组观测值 [1][2]
parts.mat	36 个圆形零件的大小偏差
polydata.mat	多项式拟合的样本数据

文件	数据集的描述
popcorn.mat	爆米花机型和品牌的爆米花产出
reaction.mat	Hougen-Watson 模型的反应动力学
spectra.mat	60 份汽油样本的近红外光谱和辛烷值
stockreturns.mat	仿真的股票回报

参考

[1] Conrads, Thomas P., Vincent A. Fusaro, Sally Ross, Don Johann, Vinodh Rajapakse, Ben A. Hitt, Seth M. Steinberg, et al. "High-Resolution Serum Proteomic Features for Ovarian Cancer Detection." *Endocrine-Related Cancer* 11 (2004): 163–78.

[2] Petricoin, Emanuel F., Ali M. Ardekani, Ben A. Hitt, Peter J. Levine, Vincent A. Fusaro, Seth M. Steinberg, Gordon B. Mills, et al. "Use of Proteomic Patterns in Serum to Identify Ovarian Cancer." *The Lancet* 359, no. 9306 (February 2002): 572–77.

B

概率分布

正态分布

本节内容

概述

正态分布，有时称为高斯分布，是双参数曲线族。使用正态分布建模的通常理由是中心极限定理，该定理（粗略地）指出，随着样本大小趋向无穷大，来自任何具有有限均值和方差的分布的独立样本总和会收敛为正态分布。

Statistics and Machine Learning Toolbox 提供了几种处理正态分布的方法。

- 通过对样本数据进行概率分布拟合 (**fitdist**) 或通过指定参数值 (**makedist**) 来创建概率分布对象 **NormalDistribution**。然后使用对象函数来计算分布、生成随机数等。
- 使用**分布拟合器**以交互方式处理正态分布。您可以从该 App 中导出对象并使用对象函数。
- 将分布特定的函数（**normcdf**、**normpdf**、**norminv**、**normlike**、**normstat**、**normfit**、**normrnd**）与指定的分布参数结合使用。分布特定的函数可以接受多个正态分布的参数。
- 将一般分布函数（**cdf**、**icdf**、**pdf**、**random**）与指定的分布名称 (**'Normal'**) 和参数结合使用。

参数

正态分布使用下列参数。

参数	说明	支持
mu (μ)	均值	$-\infty < \mu < \infty$
sigma (σ)	标准差	$\sigma \geq 0$

标准正态分布具有零均值和单位标准差。如果 z 是标准正态，则 σz + μ 也是正态，其均值为 μ，标准差为 σ。相反，如果 x 是均值为 μ、标准差为 σ 的正态，则 z = (x – μ) / σ 为标准正态。

参数估计

最大似然估计 (MLE) 是最大化似然函数的参数估计。正态分布的 μ 和 σ^2 的最大似然估计量分别是

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$$

和

$$s^2_{\mathrm{MLE}} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 .$$

$\bar{x}$ 是样本 $x_1, x_2, \ldots, x_n$ 的样本均值。样本均值是参数 μ 的无偏估计量。但是，s^2_{MLE} 是参数 σ^2 的有偏估计量，这意味着其预期值不等于参数。

最小方差无偏估计量 (MVUE) 通常用于估计正态分布的参数。MVUE 是参数的所有无偏估计量中方差最小的估计量。正态分布的参数 μ 和 σ^2 的 MVUE 分别是样本均值 $\bar{x}$ 和样本方差 s^2。

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

要对数据进行正态分布拟合并求出参数估计值，请使用 **normfit**、**fitdist** 或 **mle**。

- 对于未删失数据，**normfit** 和 **fitdist** 计算无偏估计值，**mle** 计算最大似然估计值。
- 对于删失数据，**normfit**、**fitdist**、**mle** 计算最大似然估计值。

与返回参数估计值的 **normfit** 和 **mle** 不同，**fitdist** 返回拟合的概率分布对象 **NormalDistribution**。对象属性 **mu** 和 **sigma** 存储参数估计值。

有关示例，请参阅“拟合正态分布对象” （第 B-4 页）。

概率密度函数

正态概率密度函数 (pdf) 是

$$y = f(x\,|\,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad \text{for } x \in \mathbb{R}.$$

似然函数是被视为参数函数的 pdf。最大似然估计 (MLE) 是最大化 **x** 的固定值的似然函数的参数估计。

有关示例，请参阅“计算并绘制正态分布 pdf” （第 B-4 页）。

累积分布函数

正态累积分布函数 (cdf) 表示为

$$p = F(x\,|\,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x} e^{\frac{-(t-\mu)^2}{2\sigma^2}}dt, \quad \text{for } x \in \mathbb{R}.$$

p 是参数为 μ 和 σ 的正态分布中的一个观测值落入 (-∞,x] 区间的概率。

标准正态累积分布函数 Φ(x) 在功能上与误差函数 **erf** 相关。

$$\Phi(x) = \frac{1}{2}\left(1 - \operatorname{erf}\left(-\frac{x}{\sqrt{2}}\right)\right)$$

其中

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt = 2\Phi(\sqrt{2}x) - 1.$$

有关示例，请参阅“绘制标准正态分布 cdf” （第 B-5 页）。

示例

拟合正态分布对象

加载样本数据并创建包含学生考试成绩数据的第一列的向量。

```
load examgrades
x = grades(:,1);
```

通过对数据进行正态分布拟合来创建正态分布对象。

```
pd = fitdist(x,'Normal')
```

```
pd =
  NormalDistribution

  Normal distribution
     mu = 75.0083   [73.4321, 76.5846]
  sigma =  8.7202   [7.7391, 9.98843]
```

参数估计值旁边的区间是分布参数的 95% 置信区间。

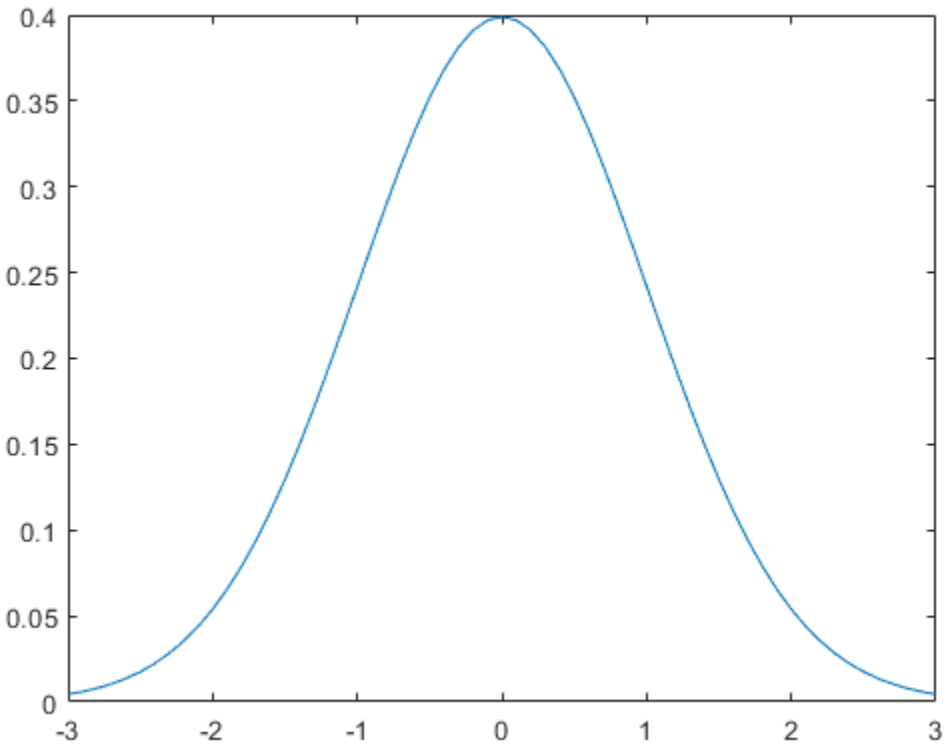
计算并绘制正态分布 pdf

计算参数 μ 等于 0、σ 等于 1 的标准正态分布的 pdf。

```
x = [-3:.1:3];
y = normpdf(x,0,1);
```

绘制 pdf。

```
plot(x,y)
```



绘制标准正态分布 cdf

创建一个标准正态分布对象。

```
pd = makedist('Normal')
```

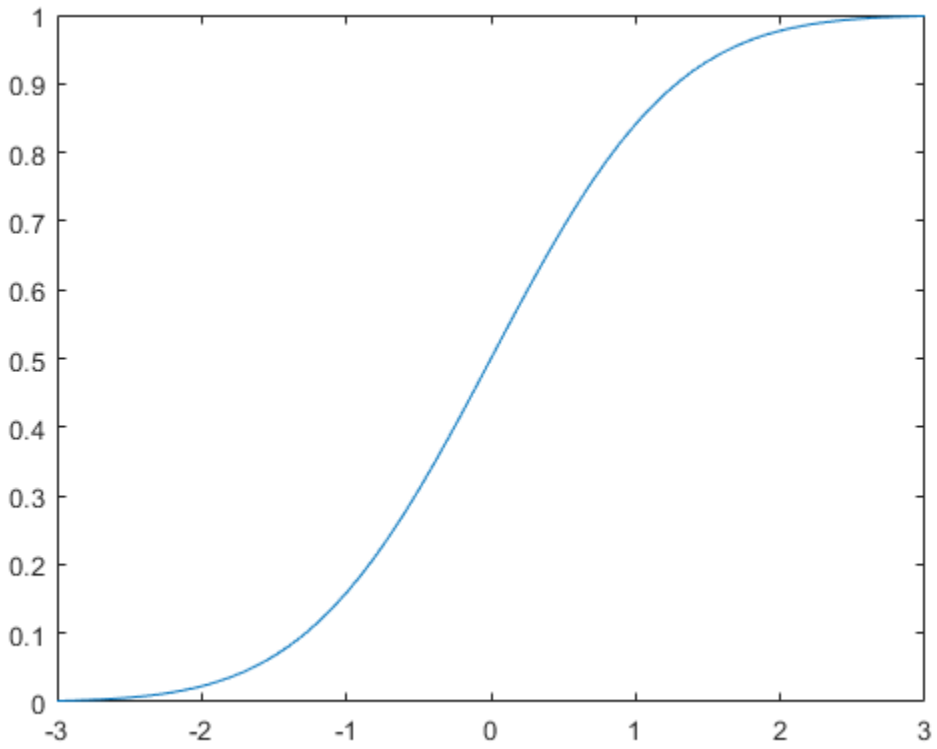
```
pd =
  NormalDistribution

  Normal distribution
       mu = 0
    sigma = 1
```

指定 x 值并计算 cdf。

```
x = -3:.1:3;
p = cdf(pd,x);
```

绘制标准正态分布的 cdf。

```
plot(x,p)
```



比较 gamma 和正态分布 pdf

gamma 分布具有形状参数 a 和尺度参数 b。如果 a 的值较大，gamma 分布非常接近均值 $\mu = ab$、方差 $\sigma^2 = ab^2$ 的正态分布。

计算参数 **a = 100** 和 **b = 5** 的 gamma 分布的 pdf。

```
a = 100;
b = 5;
x = 250:750;
y_gam = gampdf(x,a,b);
```

为了进行比较，计算基于 gamma 分布逼近的正态分布的均值、标准差和 pdf。

```
mu = a*b
```

```
mu = 500
```

```
sigma = sqrt(a*b^2)
```

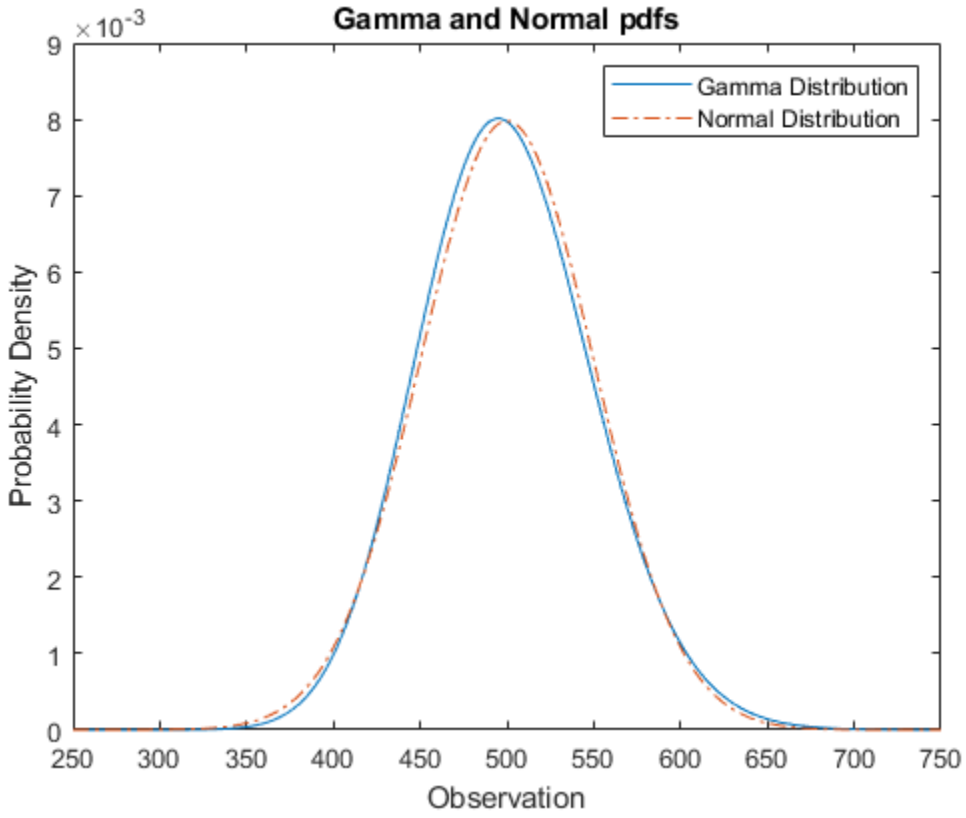
```
sigma = 50
```

```
y_norm = normpdf(x,mu,sigma);
```

将 gamma 分布和正态分布的 pdf 绘制在同一图窗上。

```
plot(x,y_gam,'-',x,y_norm,'-.')
title('Gamma and Normal pdfs')
```

```
xlabel('Observation')
ylabel('Probability Density')
legend('Gamma Distribution','Normal Distribution')
```



正态分布的 pdf 逼近 gamma 分布的 pdf。

正态分布和对数正态分布之间的关系

如果 X 遵循具有参数 μ 和 σ 的对数正态分布，则 log(X) 遵循具有均值 μ 和标准差 σ 的正态分布。使用分布对象检查正态分布和对数正态分布之间的关系。

通过指定参数值创建对数正态分布对象。

```
pd = makedist('Lognormal','mu',5,'sigma',2)
```

```
pd =
  LognormalDistribution

  Lognormal distribution
     mu = 5
    sigma = 2
```

计算对数正态分布的均值。

```
mean(pd)
```

```
ans = 1.0966e+03
```

对数正态分布的均值不等于 mu 参数。对数值的均值等于 mu。通过生成随机数来确认这种关系。

从对数正态分布中生成随机数，并计算其对数值。

```
rng('default');  % For reproducibility
x = random(pd,10000,1);
logx = log(x);
```

计算对数值的均值。

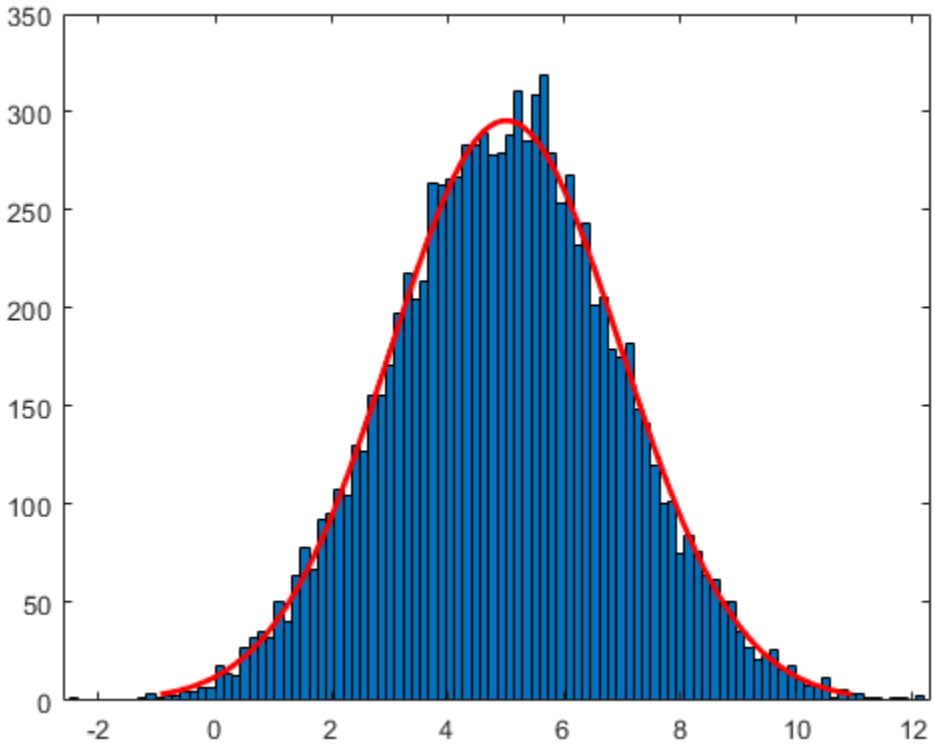
```
m = mean(logx)
```

```
m = 5.0033
```

x 的对数的均值接近 x 的 mu 参数，因为 x 具有对数正态分布。

用正态分布拟合构造 logx 的直方图。

```
histfit(logx)
```



该图显示 x 的对数值呈正态分布。

histfit 使用 fitdist 对数据进行分布拟合。使用 fitdist 获得在拟合中使用的参数。

```
pd_normal = fitdist(logx,'Normal')
```

```
pd_normal =
  NormalDistribution
```

```
Normal distribution
    mu = 5.00332   [4.96445, 5.04219]
 sigma = 1.98296   [1.95585, 2.01083]
```

估计的正态分布参数接近对数正态分布参数 5 和 2。

比较 Student t 和正态分布 pdf

Student t 分布是依赖于单参数 ν（自由度）的曲线族。随着自由度 ν 趋向无穷大，t 分布逼近标准正态分布。

计算参数 `nu = 5` 的 Student t 分布和参数 `nu = 15` 的 Student t 分布的 pdf。

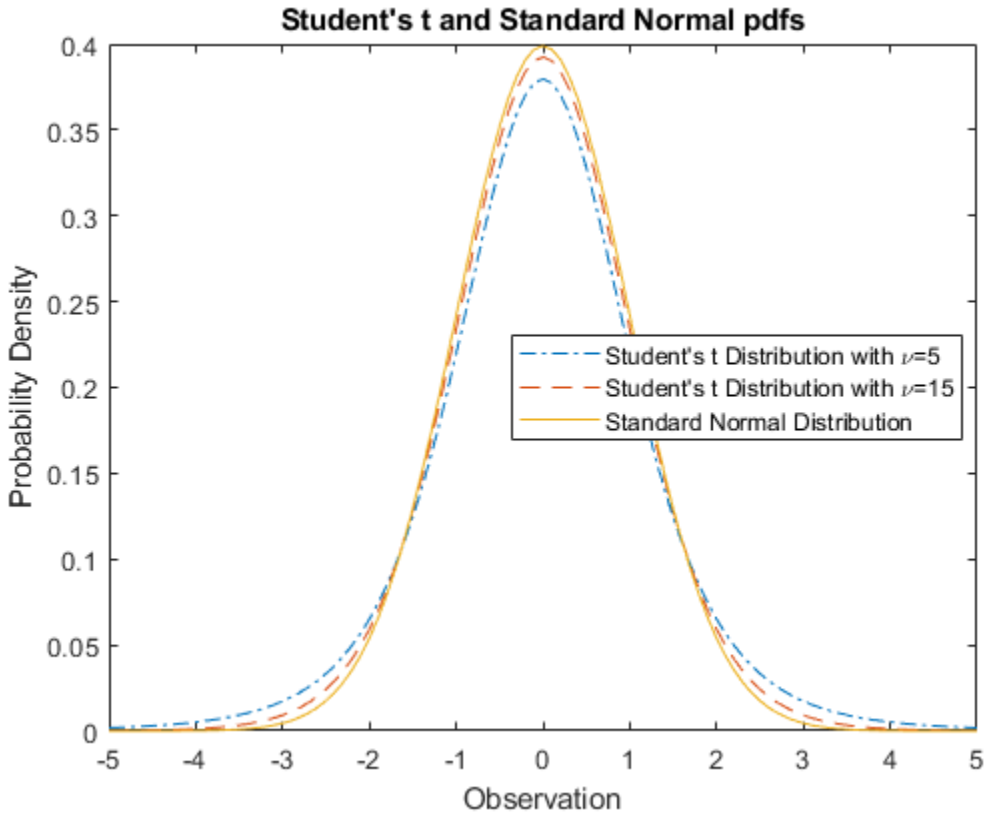
```
x = [-5:0.1:5];
y1 = tpdf(x,5);
y2 = tpdf(x,15);
```

计算标准正态分布的 pdf。

```
z = normpdf(x,0,1);
```

将 Student t pdf 和标准正态 pdf 绘制在同一图窗上。

```
plot(x,y1,'-.',x,y2,'--',x,z,'-')
legend('Student''s t Distribution with \nu=5', ...
   'Student''s t Distribution with \nu=15', ...
   'Standard Normal Distribution','Location','best')
xlabel('Observation')
ylabel('Probability Density')
title('Student''s t and Standard Normal pdfs')
```



标准正态 pdf 的尾部比 Student t pdf 短。

相关分布

- “Binomial Distribution” - 二项分布对 n 次重复试验的成功总数和成功概率 p 进行建模。随着 n 的增长，二项分布可以用 μ = np 和 $\sigma^2 = np(1-p)$ 的正态分布来逼近。请参阅“Compare Binomial and Normal Distribution pdfs”。
- “Birnbaum-Saunders Distribution” - 如果 x 具有参数为 β 和 γ 的 Birnbaum-Saunders 分布，则

  $$\frac{\left(\sqrt{x/\beta} - \sqrt{\beta/x}\right)}{\gamma}$$

  具有标准正态分布。
- “Chi-Square Distribution” - 卡方分布是平方和、独立、标准正态随机变量的分布。如果一组（包含 n 个）观测值呈正态分布，方差为 σ^2、样本方差为 s^2，则 $(n-1)s^2/\sigma^2$ 具有自由度为 n–1 的卡方分布。**normfit** 函数使用此关系来计算正态参数 σ^2 的估计的置信区间。
- “Extreme Value Distribution” - 极值分布适用于对尾部呈指数急剧衰减的分布（如正态分布）中的最小值或最大值建模。
- “Gamma Distribution” - gamma 分布具有形状参数 a 和尺度参数 b。如果 a 的值较大，gamma 分布非常接近均值 μ = ab、方差 $\sigma^2 = ab^2$ 的正态分布。gamma 分布仅对正实数才有密度。请参阅“比较 gamma 和正态分布 pdf” （第 B-6 页）。
- “Half-Normal Distribution” - 半正态分布是折叠正态分布和截断正态分布的特例。如果随机变量 **Z** 具有标准正态分布，则 $X = \mu + \sigma|Z|$ 为具有参数 μ 和 σ 的半正态分布。

- “Logistic Distribution” - 逻辑分布用于增长模型和逻辑回归。与正态分布相比，逻辑分布的尾部更长、峰度更高。
- “Lognormal Distribution” - 如果 X 遵循具有参数 μ 和 σ 的对数正态分布，则 log(X) 遵循具有均值 μ 和标准差 σ 的正态分布。请参阅“正态分布和对数正态分布之间的关系” （第 B-7 页）。
- “Multivariate Normal Distribution” - 多元正态分布是一元正态分布的双变量或多变量泛化。它是相关变量的随机向量的分布，其中每个元素都呈一元正态分布。在最简单的情形下，各变量之间没有相关性，向量的元素是独立的一元正态随机变量。
- “泊松分布” （第 B-13 页） - 泊松分布是接受非负整数值的单参数离散分布。参数 λ 既是分布的均值，也是分布的方差。随着 λ 的增大，泊松分布可以用 μ = λ 和 $\sigma^2 = \lambda$ 的正态分布来逼近。
- “Rayleigh Distribution” - 瑞利分布是 Weibull 分布的特例，应用于通信理论中。如果粒子在 x 和 y 方向上的分量速度是两个独立的正态随机变量（均值为零并具有方差齐性），则粒子在每单位时间行进的距离遵循瑞利分布。
- “Stable Distribution” - 正态分布是稳定分布的特例。第一个形状参数 α = 2 的稳定分布对应于正态分布。

  $$N\left(\mu,\sigma^2\right) = S\left(2,0,\frac{\sigma}{\sqrt{2}},\mu\right).$$

- “Student's t Distribution” - Student t 分布是一个依赖单参数 ν（自由度）的曲线族。随着自由度 ν 趋向无穷大，t 分布逼近标准正态分布。请参阅“比较 Student t 和正态分布 pdf” （第 B-9 页）。

  如果 x 是大小为 n 的随机样本，来自均值为 μ 的正态分布，则统计量

  $$t = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

  （其中 $\bar{x}$ 是样本均值，s 是样本标准差）具有包含 n–1 个自由度的 Student t 分布。
- “t Location-Scale Distribution” - t 位置尺度分布适用于对尾部比正态分布重（更容易出现离群值）的数据分布进行建模。当形状参数 ν 趋向无穷大时，它逼近正态分布。

参考

[1] Abramowitz, M., and I. A. Stegun. *Handbook of Mathematical Functions*. New York: Dover, 1964.

[2] Evans, M., N. Hastings, and B. Peacock. *Statistical Distributions*. 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc., 1993.

[3] Lawless, J. F. *Statistical Models and Methods for Lifetime Data*. Hoboken, NJ: Wiley-Interscience, 1982.

[4] Marsaglia, G., and W. W. Tsang. “A Fast, Easily Implemented Method for Sampling from Decreasing or Symmetric Unimodal Density Functions.” *SIAM Journal on Scientific and Statistical Computing*. Vol. 5, Number 2, 1984, pp. 349–359.

[5] Meeker, W. Q., and L. A. Escobar. *Statistical Methods for Reliability Data*. Hoboken, NJ: John Wiley & Sons, Inc., 1998.

另请参阅

NormalDistribution | **normcdf** | **normpdf** | **norminv** | **normlike** | **normstat** | **normfit** | **normrnd** | **erf**

详细信息

- “Working with Probability Distributions”
- “Supported Distributions”
- “Compare Multiple Distribution Fits”
- “Multivariate Distributions”

泊松分布

本节内容

概述

泊松分布是单参数曲线族，它对随机事件的发生次数进行建模。此分布适用于涉及计算在给定的时间段、距离、面积等范围内发生随机事件的次数的应用情形。应用泊松分布的例子包括盖革计数器每秒咔嗒的次数、每小时走入商店的人数，以及网络上每分钟的丢包数。

Statistics and Machine Learning Toolbox 提供了几种处理泊松分布的方法。

- 可通过对样本数据进行概率分布拟合或通过指定参数值来创建概率分布对象 **PoissonDistribution**。然后使用对象函数来计算分布、生成随机数等。
- 使用**分布拟合器**以交互方式处理泊松分布。您可以从该 App 中导出对象并使用对象函数。
- 将分布特定的函数（**poisscdf**、**poisspdf**、**poissinv**、**poisstat**、**poissfit**、**poissrnd**）与指定的分布参数结合使用。分布特定的函数可以接受多个泊松分布的参数。
- 将一般分布函数（**cdf**、**icdf**、**pdf**、**random**）与指定的分布名称 (**'Poisson'**) 和参数结合使用。

参数

泊松分布使用以下参数。

参数	说明	支持
lambda (λ)	均值	$\lambda \geq 0$

参数 λ 也等于泊松分布的方差。

参数为 λ_1 和 λ_2 的两个泊松随机变量之和是一个参数为 $\lambda = \lambda_1 + \lambda_2$ 的泊松随机变量。

概率密度函数

泊松分布的概率密度函数 (pdf) 是

$$f(x \mid \lambda) = \frac{\lambda^x}{x!}e^{-\lambda};\ x = 0, 1, 2, \ldots, \infty\ .$$

结果是随机事件发生的概率正好是 x。对于离散分布，pdf 也称为概率质量函数 (pmf)。

有关示例，请参阅 “计算泊松分布 pdf” （第 B-14 页）。

累积分布函数

泊松分布的累积分布函数 (cdf) 为

$$p = F(x \mid \lambda) = e^{-\lambda} \sum_{i=0}^{floor(x)} \frac{\lambda^i}{i!}.$$

结果是随机事件发生的概率最多为 x。

有关示例，请参阅“计算泊松分布 cdf” （第 B-15 页）。

示例

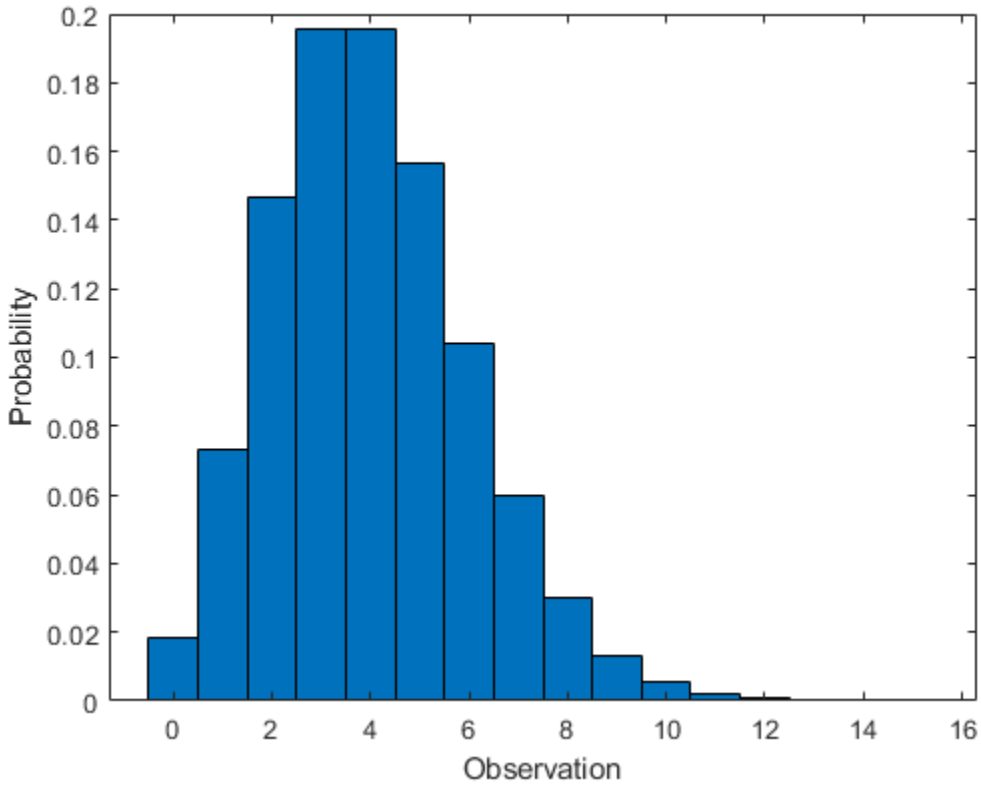
计算泊松分布 pdf

计算参数 **lambda** = 4 的泊松分布的 pdf。

```
x = 0:15;
y = poisspdf(x,4);
```

用宽度为 1 的条形绘制 pdf。

```
figure
bar(x,y,1)
xlabel('Observation')
ylabel('Probability')
```



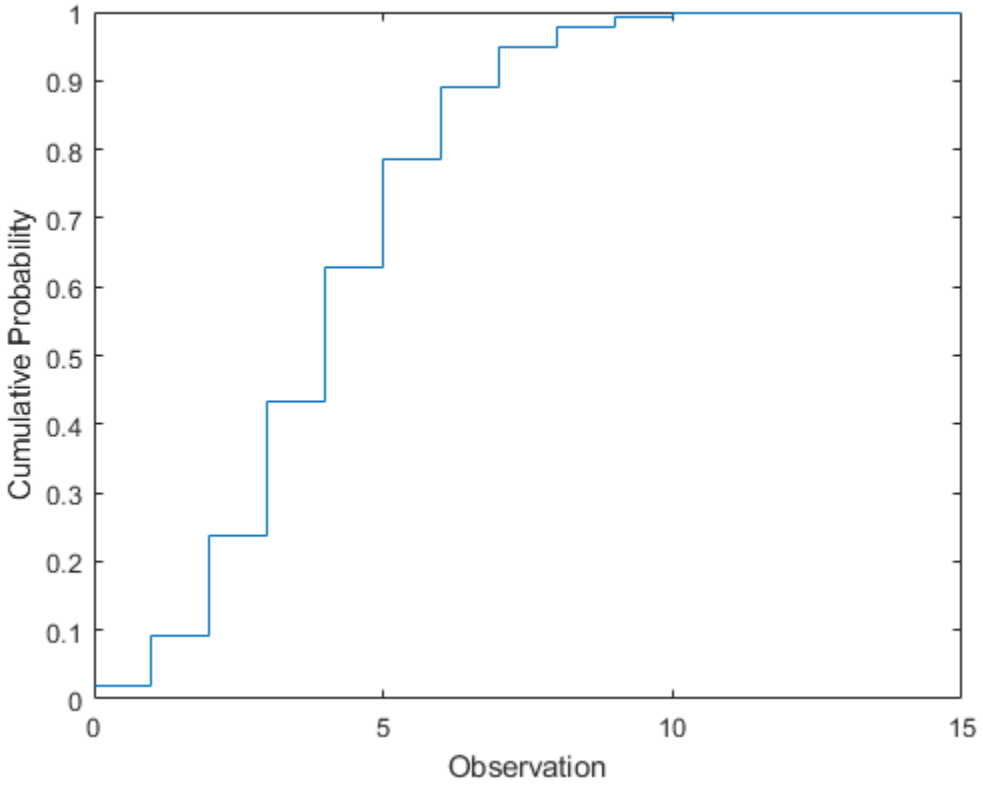
计算泊松分布 cdf

计算参数 **lambda = 4** 的泊松分布的 cdf。

```
x = 0:15;
y = poisscdf(x,4);
```

绘制 cdf。

```
figure
stairs(x,y)
xlabel('Observation')
ylabel('Cumulative Probability')
```



比较泊松分布和正态分布的 pdf

当 **lambda** 较大时，泊松分布可以用均值为 **lambda** 和方差为 **lambda** 的正态分布来逼近。

计算参数 **lambda = 50** 的泊松分布的 pdf。

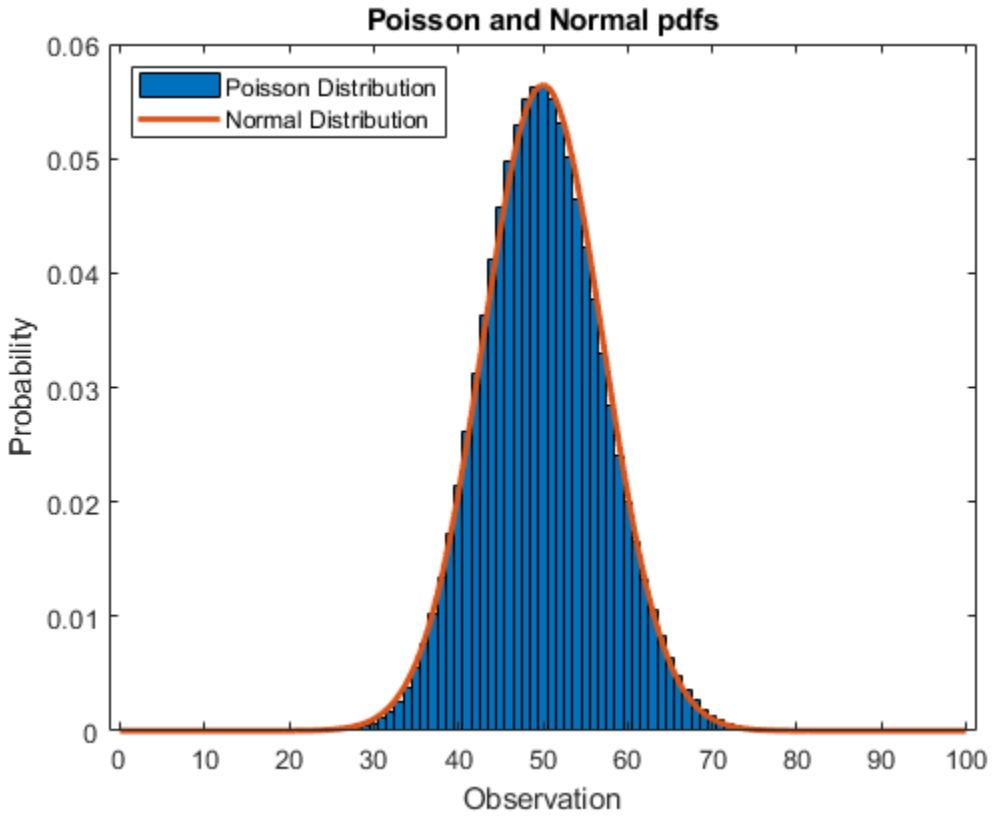
```
lambda = 50;
x1 = 0:100;
y1 = poisspdf(x1,lambda);
```

计算对应正态分布的 pdf。

```
mu = lambda;
sigma = sqrt(lambda);
x2 = 0:0.1:100;
y2 = normpdf(x2,mu,sigma);
```

在同一个轴上绘制这些 pdf。

```
figure
bar(x1,y1,1)
hold on
plot(x2,y2,'LineWidth',2)
xlabel('Observation')
ylabel('Probability')
title('Poisson and Normal pdfs')
legend('Poisson Distribution','Normal Distribution','location','northwest')
hold off
```



正态分布的 pdf 高度逼近泊松分布的 pdf。

相关分布

- “Binomial Distribution” - 二项分布是双参数离散分布，它对 N 次独立试验的成功次数进行计数，成功概率为 p。泊松分布是二项分布的极限情况，其中 N 趋向无穷大，p 趋向零，而 Np = λ。请参阅“Compare Binomial and Poisson Distribution pdfs”。

- “Exponential Distribution” - 指数分布是具有参数 μ（均值）的单参数连续分布。泊松分布对随机事件在给定时间内发生的次数计数进行建模。在这种模型中，事件的时间间隔服从均值为 $\frac{1}{\lambda}$ 的指数分布。
- “正态分布” （第 B-2 页） - 正态分布是双参数连续分布，具有参数 μ（均值）和 σ（标准差）。当 λ 较大时，泊松分布可以用具有 $\mu = \lambda$ 和 $\sigma^2 = \lambda$ 的正态分布来逼近。请参阅“比较泊松分布和正态分布的 pdf” （第 B-15 页）。

参考

[1] Abramowitz, Milton, and Irene A. Stegun, eds. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. 9. Dover print.; [Nachdr. der Ausg. von 1972]. Dover Books on Mathematics. New York, NY: Dover Publ, 2013.

[2] Devroye, Luc. *Non-Uniform Random Variate Generation*. New York, NY: Springer New York, 1986. https://doi.org/10.1007/978-1-4613-8643-8

[3] Evans, Merran, Nicholas Hastings, and Brian Peacock. *Statistical Distributions*. 2nd ed. New York: J. Wiley, 1993.

[4] Loader, Catherine. *Fast and Accurate Computation of Binomial Probabilities*. July 9, 2000.

另请参阅

PoissonDistribution | poisscdf | poisspdf | poissinv | poisstat | poissfit | poissrnd

详细信息

- “Working with Probability Distributions”
- “Supported Distributions”

参考书目

参考书目

[1] Aas, Kjersti, Martin. Jullum, and Anders Løland. "Explaining Individual Predictions When Features Are Dependent: More Accurate Approximations to Shapley Values." *arXiv:1903.10464* (2019).

[2] Atkinson, A. C., and A. N. Donev. *Optimum Experimental Designs*. New York: Oxford University Press, 1992.

[3] Bates, D. M., and D. G. Watts. *Nonlinear Regression Analysis and Its Applications*. Hoboken, NJ: John Wiley & Sons, Inc., 1988.

[4] Belsley, D. A., E. Kuh, and R. E. Welsch. *Regression Diagnostics*. Hoboken, NJ: John Wiley & Sons, Inc., 1980.

[5] Berry, M. W., et al. "Algorithms and Applications for Approximate Nonnegative Matrix Factorization." *Computational Statistics and Data Analysis*. Vol. 52, No. 1, 2007, pp. 155–173.

[6] Bookstein, Fred L. *Morphometric Tools for Landmark Data*. Cambridge, UK: Cambridge University Press, 1991.

[7] Bouye, E., V. Durrleman, A. Nikeghbali, G. Riboulet, and T. Roncalli. "Copulas for Finance: A Reading Guide and Some Applications." Working Paper. Groupe de Recherche Operationnelle, Credit Lyonnais, 2000.

[8] Bowman, A. W., and A. Azzalini. *Applied Smoothing Techniques for Data Analysis*. New York: Oxford University Press, 1997.

[9] Box, G. E. P., and N. R. Draper. *Empirical Model-Building and Response Surfaces*. Hoboken, NJ: John Wiley & Sons, Inc., 1987.

[10] Box, G. E. P., W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters*. Hoboken, NJ: Wiley-Interscience, 1978.

[11] Bratley, P., and B. L. Fox. "ALGORITHM 659 Implementing Sobol's Quasirandom Sequence Generator." *ACM Transactions on Mathematical Software*. Vol. 14, No. 1, 1988, pp. 88–100.

[12] Breiman, L. "Random Forests." *Machine Learning*. Vol. 4, 2001, pp. 5–32.

[13] Breiman, L., J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Boca Raton, FL: CRC Press, 1984.

[14] Bulmer, M. G. *Principles of Statistics*. Mineola, NY: Dover Publications, Inc., 1979.

[15] Bury, K.. *Statistical Distributions in Engineering*. Cambridge, UK: Cambridge University Press, 1999.

[16] Chatterjee, S., and A. S. Hadi. "Influential Observations, High Leverage Points, and Outliers in Linear Regression." *Statistical Science*. Vol. 1, 1986, pp. 379–416.

[17] Collett, D. *Modeling Binary Data*. New York: Chapman & Hall, 2002.

[18] Conover, W. J. *Practical Nonparametric Statistics*. Hoboken, NJ: John Wiley & Sons, Inc., 1980.

[19] Cook, R. D., and S. Weisberg. *Residuals and Influence in Regression*. New York: Chapman & Hall/CRC Press, 1983.

[20] Cox, D. R., and D. Oakes. *Analysis of Survival Data*. London: Chapman & Hall, 1984.

[21] Davidian, M., and D. M. Giltinan. *Nonlinear Models for Repeated Measurements Data*. New York: Chapman & Hall, 1995.

[22] Deb, P., and M. Sefton. "The Distribution of a Lagrange Multiplier Test of Normality." *Economics Letters*. Vol. 51, 1996, pp. 123–130.

[23] de Jong, S. "SIMPLS: An Alternative Approach to Partial Least Squares Regression." *Chemometrics and Intelligent Laboratory Systems*. Vol. 18, 1993, pp. 251–263.

[24] Demidenko, E. *Mixed Models: Theory and Applications*. Hoboken, NJ: John Wiley & Sons, Inc., 2004.

[25] Delyon, B., M. Lavielle, and E. Moulines, *Convergence of a stochastic approximation version of the EM algorithm*, Annals of Statistics, 27, 94-128, 1999.

[26] Dempster, A. P., N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society*. Series B, Vol. 39, No. 1, 1977, pp. 1–37.

[27] Devroye, L. *Non-Uniform Random Variate Generation*. New York: Springer-Verlag, 1986.

[28] Dobson, A. J. *An Introduction to Generalized Linear Models*. New York: Chapman & Hall, 1990.

[29] Dunn, O.J., and V.A. Clark. *Applied Statistics: Analysis of Variance and Regression*. New York: Wiley, 1974.

[30] Draper, N. R., and H. Smith. *Applied Regression Analysis*. Hoboken, NJ: Wiley-Interscience, 1998.

[31] Drezner, Z. "Computation of the Trivariate Normal Integral." *Mathematics of Computation*. Vol. 63, 1994, pp. 289–294.

[32] Drezner, Z., and G. O. Wesolowsky. "On the Computation of the Bivariate Normal Integral." *Journal of Statistical Computation and Simulation*. Vol. 35, 1989, pp. 101–107.

[33] DuMouchel, W. H., and F. L. O'Brien. "Integrating a Robust Option into a Multiple Regression Computing Environment." *Computer Science and Statistics: Proceedings of the 21st Symposium on the Interface*. Alexandria, VA: American Statistical Association, 1989.

[34] Durbin, R., S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge, UK: Cambridge University Press, 1998.

[35] Efron, B., and R. J. Tibshirani. *An Introduction to the Bootstrap*. New York: Chapman & Hall, 1993.

[36] Embrechts, P., C. Klüppelberg, and T. Mikosch. *Modelling Extremal Events for Insurance and Finance*. New York: Springer, 1997.

[37] Evans, M., N. Hastings, and B. Peacock. *Statistical Distributions*. 2nd ed., Hoboken, NJ: John Wiley & Sons, Inc., 1993, pp. 50–52, 73–74, 102–105, 147, 148.

[38] Friedman, J. H. "Greedy function approximation: a gradient boosting machine." *The Annals of Statistics*. Vol. 29, No. 5, 2001, pp. 1189-1232.

[39] Genz, A. "Numerical Computation of Rectangular Bivariate and Trivariate Normal and t Probabilities." *Statistics and Computing*. Vol. 14, No. 3, 2004, pp. 251–260.

[40] Genz, A., and F. Bretz. "Comparison of Methods for the Computation of Multivariate t Probabilities." *Journal of Computational and Graphical Statistics*. Vol. 11, No. 4, 2002, pp. 950–971.

[41] Genz, A., and F. Bretz. "Numerical Computation of Multivariate t Probabilities with Application to Power Calculation of Multiple Contrasts." *Journal of Statistical Computation and Simulation*. Vol. 63, 1999, pp. 361–378.

[42] Gibbons, J. D. *Nonparametric Statistical Inference*. New York: Marcel Dekker, 1985.

[43] Goldstein, A., A. Kapelner, J. Bleich, and E. Pitkin. "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation." *Journal of Computational and Graphical Statistics*. Vol. 24, No. 1, 2015, pp. 44-65.

[44] Goodall, C. R. "Computation Using the QR Decomposition." *Handbook in Statistics.* Vol. 9, Amsterdam: Elsevier/North-Holland, 1993.

[45] Goodnight, J.H., and F.M. Speed. *Computing Expected Mean Squares*. Cary, NC: SAS Institute, 1978.

[46] Hahn, Gerald J., and S. S. Shapiro. *Statistical Models in Engineering*. Hoboken, NJ: John Wiley & Sons, Inc., 1994, p. 95.

[47] Hald, A. *Statistical Theory with Engineering Applications*. Hoboken, NJ: John Wiley & Sons, Inc., 1960.

[48] Harman, H. H. *Modern Factor Analysis*. 3rd Ed. Chicago: University of Chicago Press, 1976.

[49] Hastie, T., R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. New York: Springer, 2001.

[50] Hill, P. D. "Kernel estimation of a distribution function." *Communications in Statistics – Theory and Methods*. Vol. 14, Issue 3, 1985, pp. 605-620.

[51] Hochberg, Y., and A. C. Tamhane. *Multiple Comparison Procedures*. Hoboken, NJ: John Wiley & Sons, 1987.

[52] Hoerl, A. E., and R. W. Kennard. "Ridge Regression: Applications to Nonorthogonal Problems." *Technometrics*. Vol. 12, No. 1, 1970, pp. 69–82.

[53] Hoerl, A. E., and R. W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics*. Vol. 12, No. 1, 1970, pp. 55–67.

[54] Hogg, R. V., and J. Ledolter. *Engineering Statistics*. New York: MacMillan, 1987.

[55] Holland, P. W., and R. E. Welsch. "Robust Regression Using Iteratively Reweighted Least-Squares." *Communications in Statistics: Theory and Methods*, *A6*, 1977, pp. 813–827.

[56] Hollander, M., and D. A. Wolfe. *Nonparametric Statistical Methods*. Hoboken, NJ: John Wiley & Sons, Inc., 1999.

[57] Hong, H. S., and F. J. Hickernell. "ALGORITHM 823: Implementing Scrambled Digital Sequences." *ACM Transactions on Mathematical Software*. Vol. 29, No. 2, 2003, pp. 95–109.

[58] Huang, P. S., H. Avron, and T. N. Sainath, V. Sindhwani, and B. Ramabhadran. "Kernel methods match Deep Neural Networks on TIMIT." *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2014, pp. 205–209.

[59] Huber, P. J. *Robust Statistics*. Hoboken, NJ: John Wiley & Sons, Inc., 1981.

[60] Jackson, J. E. *A User's Guide to Principal Components*. Hoboken, NJ: John Wiley and Sons, 1991.

[61] Jain, A., and R. Dubes. *Algorithms for Clustering Data*. Upper Saddle River, NJ: Prentice-Hall, 1988.

[62] Jarque, C. M., and A. K. Bera. "A test for normality of observations and regression residuals." *International Statistical Review*. Vol. 55, No. 2, 1987, pp. 163–172.

[63] Joe, S., and F. Y. Kuo. "Remark on Algorithm 659: Implementing Sobol's Quasirandom Sequence Generator." *ACM Transactions on Mathematical Software*. Vol. 29, No. 1, 2003, pp. 49–57.

[64] Johnson, N., and S. Kotz. *Distributions in Statistics: Continuous Univariate Distributions-2.* Hoboken, NJ: John Wiley & Sons, Inc., 1970, pp. 130–148, 189–200, 201–219.

[65] Johnson, N. L., N. Balakrishnan, and S. Kotz. *Continuous Multivariate Distributions*. Vol. 1. Hoboken, NJ: Wiley-Interscience, 2000.

[66] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Vol. 1, Hoboken, NJ: Wiley-Interscience, 1993.

[67] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Vol. 2, Hoboken, NJ: Wiley-Interscience, 1994.

[68] Johnson, N. L., S. Kotz, and N. Balakrishnan. *Discrete Multivariate Distributions*. Hoboken, NJ: Wiley-Interscience, 1997.

[69] Johnson, N. L., S. Kotz, and A. W. Kemp. *Univariate Discrete Distributions*. Hoboken, NJ: Wiley-Interscience, 1993.

[70] Jolliffe, I. T. *Principal Component Analysis*. 2nd ed., New York: Springer-Verlag, 2002.

[71] Jones, M.C. "Simple boundary correction for kernel density estimation." *Statistics and Computing*. Vol. 3, Issue 3, 1993, pp. 135-146.

[72] Jöreskog, K. G. "Some Contributions to Maximum Likelihood Factor Analysis." *Psychometrika*. Vol. 32, 1967, pp. 443–482.

[73] Kaufman L., and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Hoboken, NJ: John Wiley & Sons, Inc., 1990.

[74] Kempka, Michał, Wojciech Kotłowski, and Manfred K. Warmuth. "Adaptive Scale-Invariant Online Algorithms for Learning Linear Models." CoRR (February 2019). https://arxiv.org/abs/1902.07528.

[75] Kendall, David G. "A Survey of the Statistical Theory of Shape." *Statistical Science*. Vol. 4, No. 2, 1989, pp. 87–99.

[76] Klein, J. P., and M. L. Moeschberger. *Survival Analysis*. Statistics for Biology and Health. 2nd edition. Springer, 2003.

[77] Kleinbaum, D. G., and M. Klein. *Survival Analysis*. Statistics for Biology and Health. 2nd edition. Springer, 2005.

[78] Kocis, L., and W. J. Whiten. "Computational Investigations of Low-Discrepancy Sequences." *ACM Transactions on Mathematical Software*. Vol. 23, No. 2, 1997, pp. 266–294.

[79] Kotz, S., and S. Nadarajah. *Extreme Value Distributions: Theory and Applications*. London: Imperial College Press, 2000.

[80] Krzanowski, W. J. *Principles of Multivariate Analysis: A User's Perspective*. New York: Oxford University Press, 1988.

[81] Lawless, J. F. *Statistical Models and Methods for Lifetime Data*. Hoboken, NJ: Wiley-Interscience, 2002.

[82] Lawley, D. N., and A. E. Maxwell. *Factor Analysis as a Statistical Method*. 2nd ed. New York: American Elsevier Publishing, 1971.

[83] Le, Q., T. Sarlós, and A. Smola. "Fastfood — Approximating Kernel Expansions in Loglinear Time." *Proceedings of the 30th International Conference on Machine Learning*. Vol. 28, No. 3, 2013, pp. 244–252.

[84] Lilliefors, H. W. "On the Kolmogorov-Smirnov test for normality with mean and variance unknown." *Journal of the American Statistical Association*. Vol. 62, 1967, pp. 399–402.

[85] Lilliefors, H. W. "On the Kolmogorov-Smirnov test for the exponential distribution with mean unknown." *Journal of the American Statistical Association*. Vol. 64, 1969, pp. 387–389.

[86] Lindstrom, M. J., and D. M. Bates. "Nonlinear mixed-effects models for repeated measures data." *Biometrics*. Vol. 46, 1990, pp. 673–687.

[87] Little, Roderick J. A., and Donald B. Rubin. *Statistical Analysis with Missing Data*. 2nd ed., Hoboken, NJ: John Wiley & Sons, Inc., 2002.

[88] Lundberg, Scott M., and S. Lee. "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems* 30 (2017): 4765–774.

[89] Mardia, K. V., J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Burlington, MA: Academic Press, 1980.

[90] Marquardt, D.W. "Generalized Inverses, Ridge Regression, Biased Linear Estimation, and Nonlinear Estimation." *Technometrics*. Vol. 12, No. 3, 1970, pp. 591–612.

[91] Marquardt, D. W., and R.D. Snee. "Ridge Regression in Practice." *The American Statistician*. Vol. 29, No. 1, 1975, pp. 3–20.

[92] Marsaglia, G., and W. W. Tsang. "A Simple Method for Generating Gamma Variables." *ACM Transactions on Mathematical Software*. Vol. 26, 2000, pp. 363–372.

[93] Marsaglia, G., W. Tsang, and J. Wang. "Evaluating Kolmogorov' s Distribution." *Journal of Statistical Software*. Vol. 8, Issue 18, 2003.

[94] Martinez, W. L., and A. R. Martinez. *Computational Statistics with MATLAB*. New York: Chapman & Hall/CRC Press, 2002.

[95] Massey, F. J. "The Kolmogorov-Smirnov Test for Goodness of Fit." *Journal of the American Statistical Association*. Vol. 46, No. 253, 1951, pp. 68–78.

[96] Matousek, J. "On the L2-Discrepancy for Anchored Boxes." *Journal of Complexity*. Vol. 14, No. 4, 1998, pp. 527–556.

[97] McLachlan, G., and D. Peel. *Finite Mixture Models*. Hoboken, NJ: John Wiley & Sons, Inc., 2000.

[98] McCullagh, P., and J. A. Nelder. *Generalized Linear Models*. New York: Chapman & Hall, 1990.

[99] McGill, R., J. W. Tukey, and W. A. Larsen. "Variations of Boxplots." *The American Statistician*. Vol. 32, No. 1, 1978, pp. 12–16.

[100] Meeker, W. Q., and L. A. Escobar. *Statistical Methods for Reliability Data*. Hoboken, NJ: John Wiley & Sons, Inc., 1998.

[101] Meng, Xiao-Li, and Donald B. Rubin. "Maximum Likelihood Estimation via the ECM Algorithm." *Biometrika*. Vol. 80, No. 2, 1993, pp. 267–278.

[102] Meyers, R. H., and D.C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Hoboken, NJ: John Wiley & Sons, Inc., 1995.

[103] Miller, L. H. "Table of Percentage Points of Kolmogorov Statistics." *Journal of the American Statistical Association*. Vol. 51, No. 273, 1956, pp. 111–121.

[104] Milliken, G. A., and D. E. Johnson. *Analysis of Messy Data, Volume 1: Designed Experiments*. Boca Raton, FL: Chapman & Hall/CRC Press, 1992.

[105] Montgomery, D. *Introduction to Statistical Quality Control*. Hoboken, NJ: John Wiley & Sons, 1991, pp. 369–374.

[106] Montgomery, D. C. *Design and Analysis of Experiments*. Hoboken, NJ: John Wiley & Sons, Inc., 2001.

[107] Mood, A. M., F. A. Graybill, and D. C. Boes. *Introduction to the Theory of Statistics.* 3rd ed., New York: McGraw-Hill, 1974. pp. 540–541.

[108] Moore, J. *Total Biochemical Oxygen Demand of Dairy Manures*. Ph.D. thesis. University of Minnesota, Department of Agricultural Engineering, 1975.

[109] Mosteller, F., and J. Tukey. *Data Analysis and Regression*. Upper Saddle River, NJ: Addison-Wesley, 1977.

[110] Nelson, L. S. "Evaluating Overlapping Confidence Intervals." *Journal of Quality Technology*. Vol. 21, 1989, pp. 140–141.

[111] Patel, J. K., C. H. Kapadia, and D. B. Owen. *Handbook of Statistical Distributions*. New York: Marcel Dekker, 1976.

[112] Pinheiro, J. C., and D. M. Bates. "Approximations to the log-likelihood function in the nonlinear mixed-effects model." *Journal of Computational and Graphical Statistics*. Vol. 4, 1995, pp. 12–35.

[113] Rahimi, A., and B. Recht. "Random Features for Large-Scale Kernel Machines." *Advances in Neural Information Processing Systems*. Vol 20, 2008, pp. 1177–1184.

[114] Rice, J. A. *Mathematical Statistics and Data Analysis*. Pacific Grove, CA: Duxbury Press, 1994.

[115] Rosipal, R., and N. Kramer. "Overview and Recent Advances in Partial Least Squares." *Subspace, Latent Structure and Feature Selection: Statistical and Optimization Perspectives Workshop (SLSFS 2005), Revised Selected Papers (Lecture Notes in Computer Science 3940)*. Berlin, Germany: Springer-Verlag, 2006, pp. 34–51.

[116] Sachs, L. *Applied Statistics: A Handbook of Techniques*. New York: Springer-Verlag, 1984, p. 253.

[117] Scott, D. W. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.

[118] Searle, S. R., F. M. Speed, and G. A. Milliken. "Population marginal means in the linear model: an alternative to least-squares means." *American Statistician*. 1980, pp. 216–221.

[119] Seber, G. A. F. and A. J. Lee. *Linear Regression Analysis*. 2nd ed. Hoboken, NJ: Wiley-Interscience, 2003.

[120] Seber, G. A. F. *Multivariate Observations*. Hoboken, NJ: John Wiley & Sons, Inc., 1984.

[121] Seber, G. A. F., and C. J. Wild. *Nonlinear Regression*. Hoboken, NJ: Wiley-Interscience, 2003.

[122] Sexton, Joe, and A. R. Swensen. "ECM Algorithms that Converge at the Rate of EM." *Biometrika*. Vol. 87, No. 3, 2000, pp. 651–662.

[123] Silverman, B.W. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.

[124] Snedecor, G. W., and W. G. Cochran. *Statistical Methods*. Ames, IA: Iowa State Press, 1989.

[125] Spath, H. *Cluster Dissection and Analysis: Theory, FORTRAN Programs, Examples*. Translated by J. Goldschmidt. New York: Halsted Press, 1985.

[126] Stein, M. "Large sample properties of simulations using latin hypercube sampling." *Technometrics*. Vol. 29, No. 2, 1987, pp. 143–151. Correction, Vol. 32, p. 367.

[127] Stephens, M. A. "Use of the Kolmogorov-Smirnov, Cramer-Von Mises and Related Statistics Without Extensive Tables." *Journal of the Royal Statistical Society*. Series B, Vol. 32, No. 1, 1970, pp. 115–122.

[128] Street, J. O., R. J. Carroll, and D. Ruppert. "A Note on Computing Robust Regression Estimates via Iteratively Reweighted Least Squares." *The American Statistician*. Vol. 42, 1988, pp. 152–154.

[129] Student. "On the Probable Error of the Mean." *Biometrika*. Vol. 6, No. 1, 1908, pp. 1–25.

[130] Vellemen, P. F., and D. C. Hoaglin. *Application, Basics, and Computing of Exploratory Data Analysis*. Pacific Grove, CA: Duxbury Press, 1981.

[131] Weibull, W. “A Statistical Theory of the Strength of Materials.” *Ingeniors Vetenskaps Akademiens Handlingar*. Stockholm: Royal Swedish Institute for Engineering Research, No. 151, 1939.

[132] Zahn, C. T. “Graph-theoretical methods for detecting and describing Gestalt clusters.” *IEEE Transactions on Computers*. Vol. C-20, Issue 1, 1971, pp. 68–86.