实验五 matlab中的数据挖掘，分类、聚类和关联等操作

1. 实验目的
2. 熟悉分类、聚类和关联等操作原理
3. 了解matlab实现分类、聚类和关联操作方法
4. 编写并尝试matlab中的svm算法
5. 实验平台

MATLAB 2022b

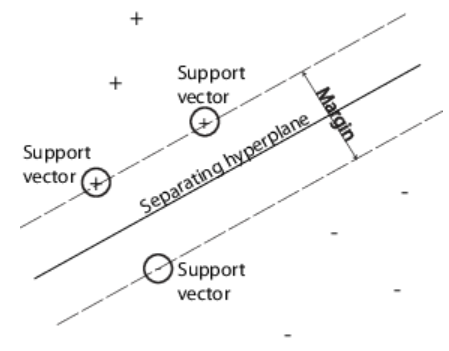
Statistics and Machine Learning Toolbox库需自行安装

三、实验步骤

1. 了解支持向量机svm
2. 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。



1. 不可分离的数据

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。

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。

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。

1. 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

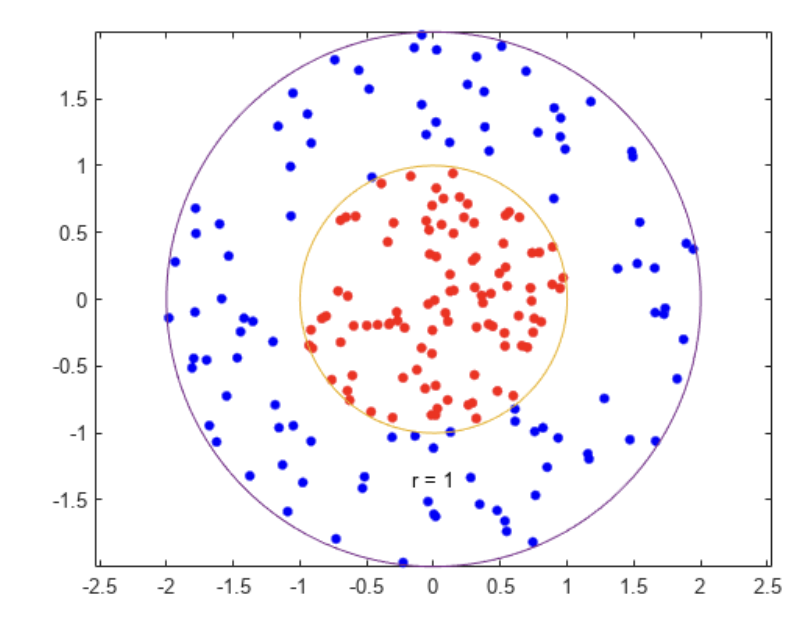
plot(data2(:,1),data2(:,2),"b.","MarkerSize",15)

fplot(@(t)sin(t),@(t)cos(t))

fplot(@(t)2\*sin(t),@(t)2\*cos(t))

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

fplot(@(t)sin(t),@(t)cos(t))

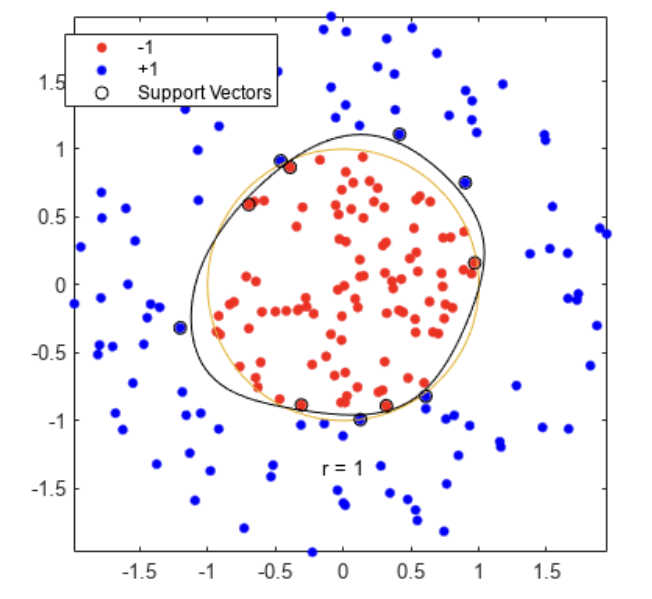
h(3) = plot(data3(cl.IsSupportVector,1),data3(cl.IsSupportVector,2),"ko");

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

fplot(@(t)sin(t),@(t)cos(t))

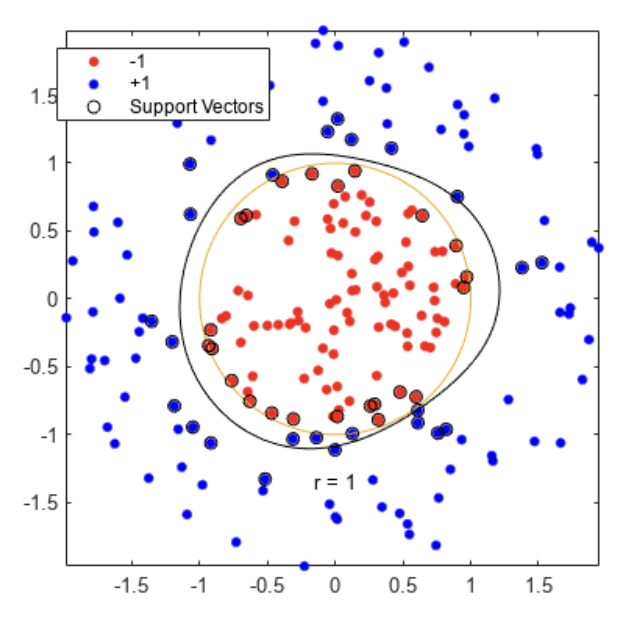
h(3) = plot(data3(cl2.IsSupportVector,1),data3(cl2.IsSupportVector,2),"ko");

contour(x1Grid,x2Grid,reshape(scores2(:,2),size(x1Grid)),[0 0],"k")

legend(h,["-1","+1","Support Vectors"])

axis equal

hold off



1. 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);

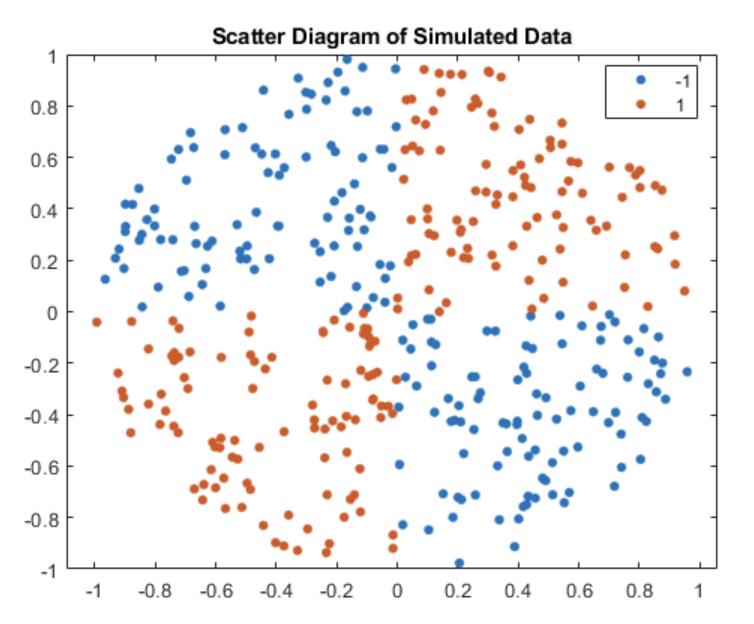
Y(2\*n + 1:end) = -1; % Labels

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 核将它们转换为格拉姆矩阵。

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

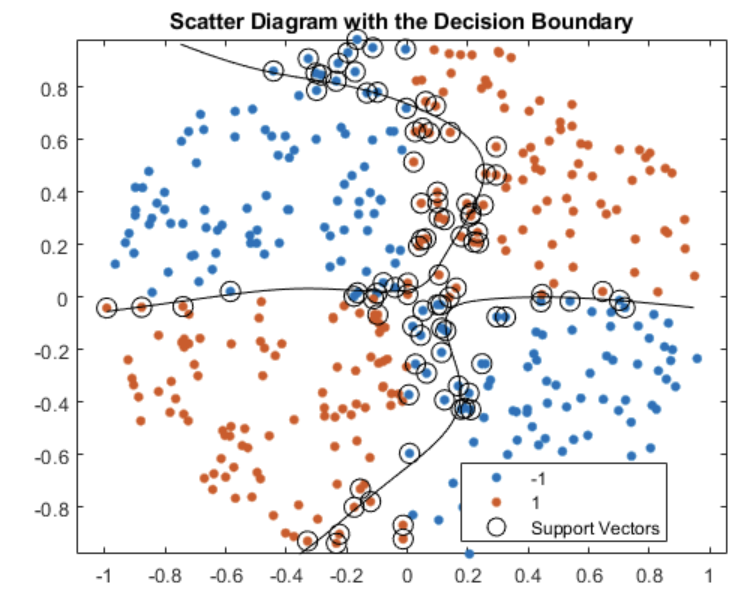
contour(x1Grid,x2Grid,reshape(scores1(:,2),size(x1Grid)),[0 0],'k');

% Decision boundary

title('Scatter Diagram with the Decision Boundary')

legend({'-1','1','Support Vectors'},'Location','Best');

hold off



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

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');

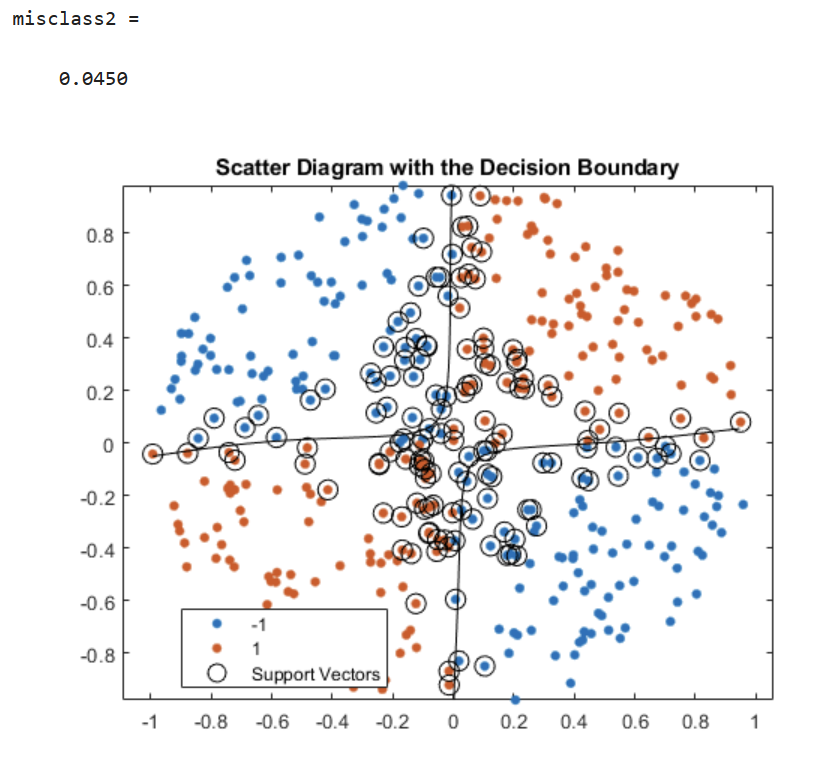
legend({'-1','1','Support Vectors'},'Location','Best');

hold off

CVMdl2 = crossval(Mdl2);

misclass2 = kfoldLoss(CVMdl2);

misclass2



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。

1. 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。

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。

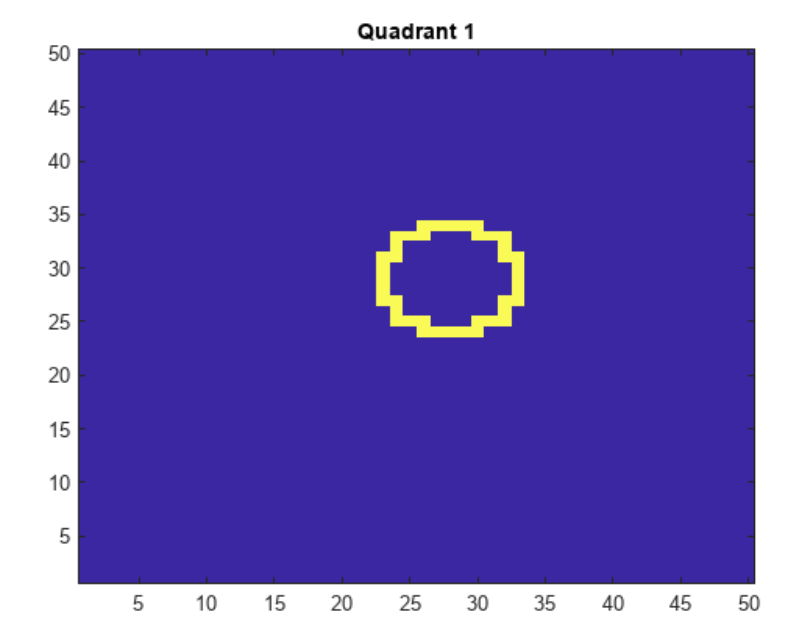
figure

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)

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

该模型消耗 763.6 MB。

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

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

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))

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]))

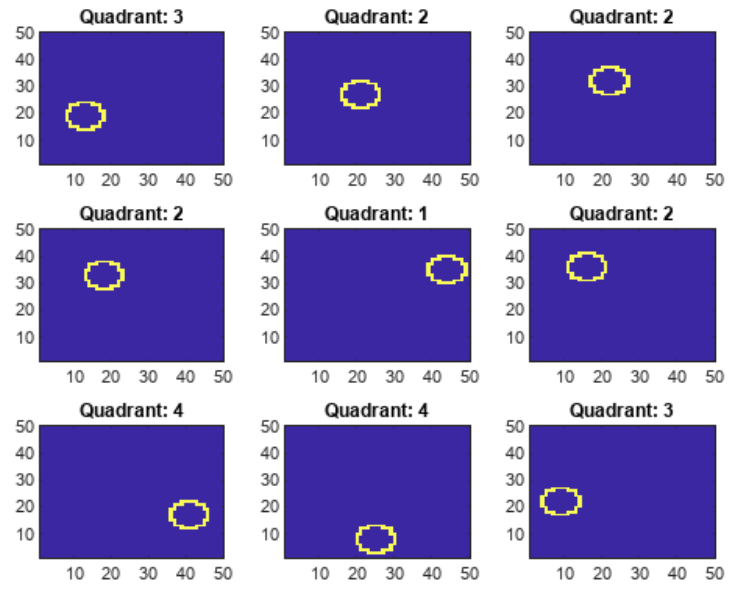
h = gca;

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。