# 第 2 回 レポート課題

## 課題１．SVM による Wine データのクラス分類

https://archive.ics.uci.edu/ml/datasets/Wine

上記の URL から wine.data，wine.names をダウンロードする．

wine.csv をダブルクリックして出力形式を数値行列として，インポートする．

ワークスペースに数値行列(178*14)の wine があることを確認する．

その後，「wine.mat」として保存する．

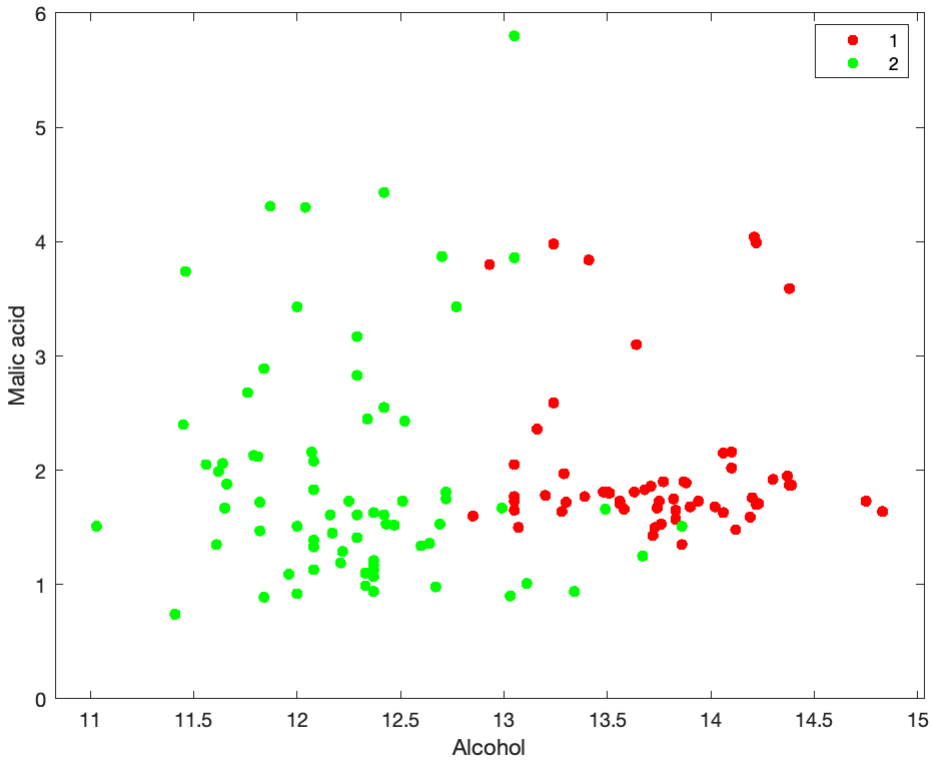
入力特徴量に Alcohol，Malic acid の２つの特徴量を用いる．

まずは，２クラスの分類を行う．

```
load('wine.mat');
indices = 1:130;
X = wine(indices,2:3);
Y = wine(indices,1);
```

可視化してみる．

```
gscatter(X(:,1), X(:,2), Y, 'rgb');
xlabel('Alcohol')
ylabel('Malic acid')
```




ここでは，アプリからエクスポートした学習済みの線形 SVM を用いて，1 行目のデータを予測してみる.

```
trainedModel.predictFcn(X(1,:))
Y(1,:)
```

```
ans = 1
```

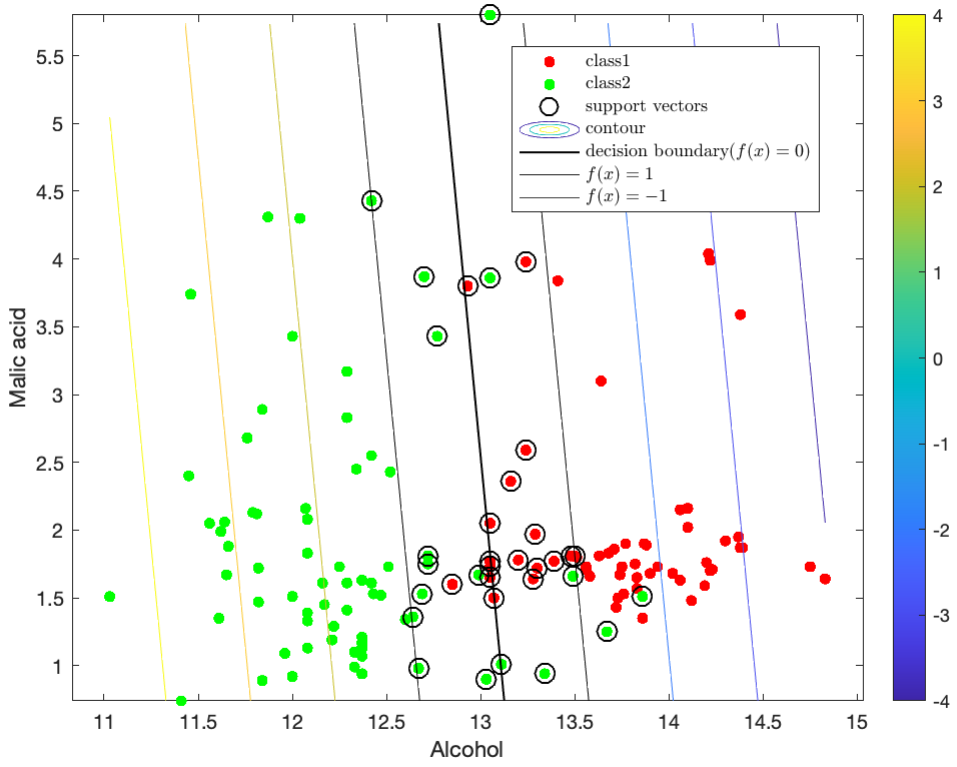
学習済みモデルが正解のラベルを出力していることがわかる.

ここからは，コマンドベースで SVM モデルの構築を行う.

まずは，ソフトマージン(C=1)の線形 SVM による分類を行う.

```
SVMModel = fitcsvm(X,Y, "BoxConstraint", 1);
svInd = SVMModel.IsSupportVector;
h = 0.1; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2))); % calculate g
[~,score] = SVMModel.predict([X1(:),X2(:)]);  % calculate scores at the each grid point
scoreGrid = reshape(score(:,2),size(X1,1),size(X2,2));
figure
gscatter(X(:,1), X(:,2), Y, 'rg');
hold on
plot(X(svInd,1),X(svInd,2),'ko','MarkerSize',10, "LineWidth",1) % Mark support vectors
contour(X1,X2,scoreGrid); % draw contour
contour(X1,X2,scoreGrid, [0 0],'k', "LineWidth", 1); %draw decision bounday
contour(X1,X2,scoreGrid, [1 1],'k'); %draw f(x)=1
```

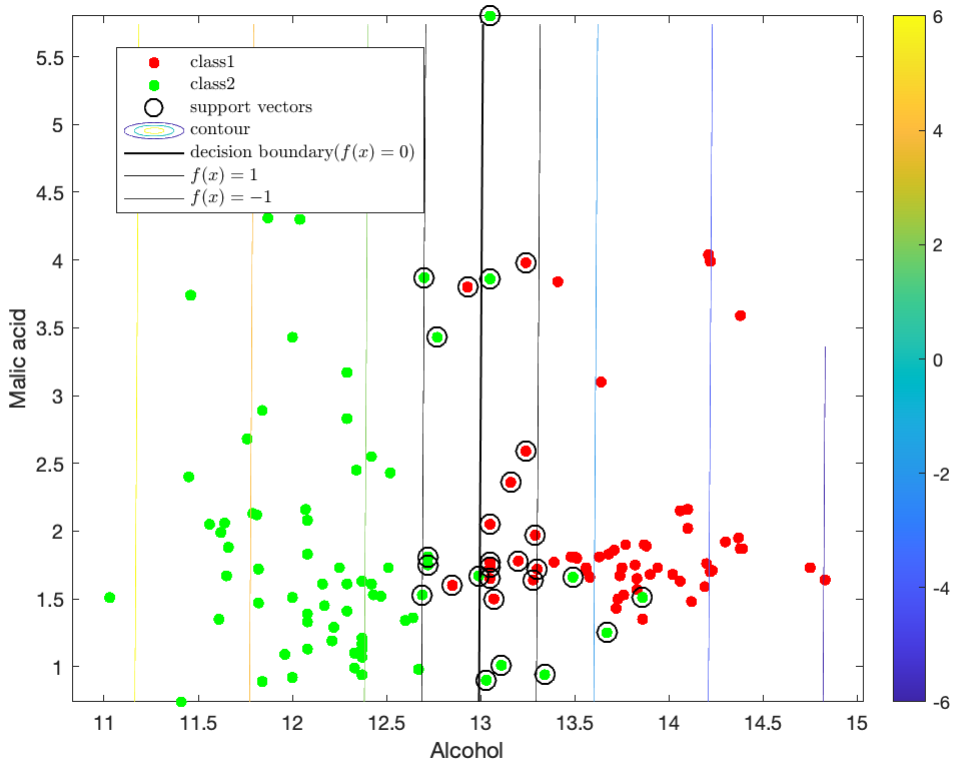
```
contour(X1,X2,scoreGrid, [-1 -1],'k'); %draw f(x)=-1
colorbar;
xlabel('Alcohol')
ylabel('Malic acid')
legend(["class1", "class2", "support vectors", "contour", "decision boundary($f(x)=0$)"
hold off
```




次に，C=1 から C=1000 と値を大きくして，ハードマージンによる分類を試してみる.

```
SVMModel = fitcsvm(X,Y, "BoxConstraint", 1000);
svInd = SVMModel.IsSupportVector;
h = 0.1; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2))); % calculate
[~,score] = SVMModel.predict([X1(:),X2(:)]);  % calculate scores at the each grid point
scoreGrid = reshape(score(:,2),size(X1,1),size(X2,2));
figure
gscatter(X(:,1), X(:,2), Y, 'rg');
hold on
plot(X(svInd,1),X(svInd,2),'ko','MarkerSize',10, "LineWidth",1) % Mark support vectors
contour(X1,X2,scoreGrid); % draw contour
contour(X1,X2,scoreGrid, [0 0],'k', "LineWidth", 1); %draw decision bounday
contour(X1,X2,scoreGrid, [1 1],'k'); %draw f(x)=1
contour(X1,X2,scoreGrid, [-1 -1],'k'); %draw f(x)=-1
colorbar;
xlabel('Alcohol')
ylabel('Malic acid')
```

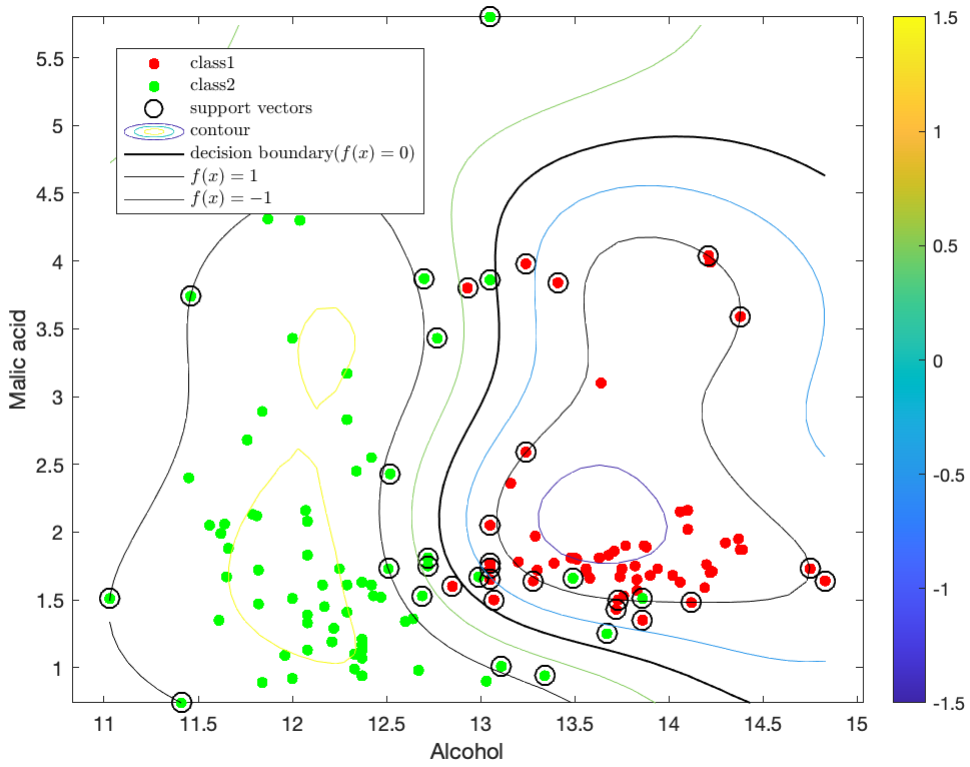
```
legend(["class1", "class2", "support vectors", "contour", "decision boundary($f(x)=0$)"
hold off
```




ハードマージンによる SVM を用いても，未だに誤分類が見られる.

線形 SVM による直線の境界線では，分類が難しいため，次に，RBF カーネルを用いた SVM を適用する.

```
SVMModel = fitcsvm(X,Y, "BoxConstraint", 1, "KernelFunction", "gaussian");
svInd = SVMModel.IsSupportVector;
h = 0.1; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2))); % calculate
[~,score] = SVMModel.predict([X1(:),X2(:)]);  % calculate scores at the each grid poin
scoreGrid = reshape(score(:,2),size(X1,1),size(X2,2));
figure
gscatter(X(:,1), X(:,2), Y, 'rg');
hold on
plot(X(svInd,1),X(svInd,2),'ko','MarkerSize',10, "LineWidth",1) % Mark support vectors
contour(X1,X2,scoreGrid); % draw contour
contour(X1,X2,scoreGrid, [0 0],'k', "LineWidth", 1); %draw decision bounday
contour(X1,X2,scoreGrid, [1 1],'k'); %draw f(x)=1
contour(X1,X2,scoreGrid, [-1 -1],'k'); %draw f(x)=-1
colorbar;
xlabel('Alcohol')
ylabel('Malic acid')
legend(["class1", "class2", "support vectors", "contour", "decision boundary($f(x)=0$)"
hold off
```

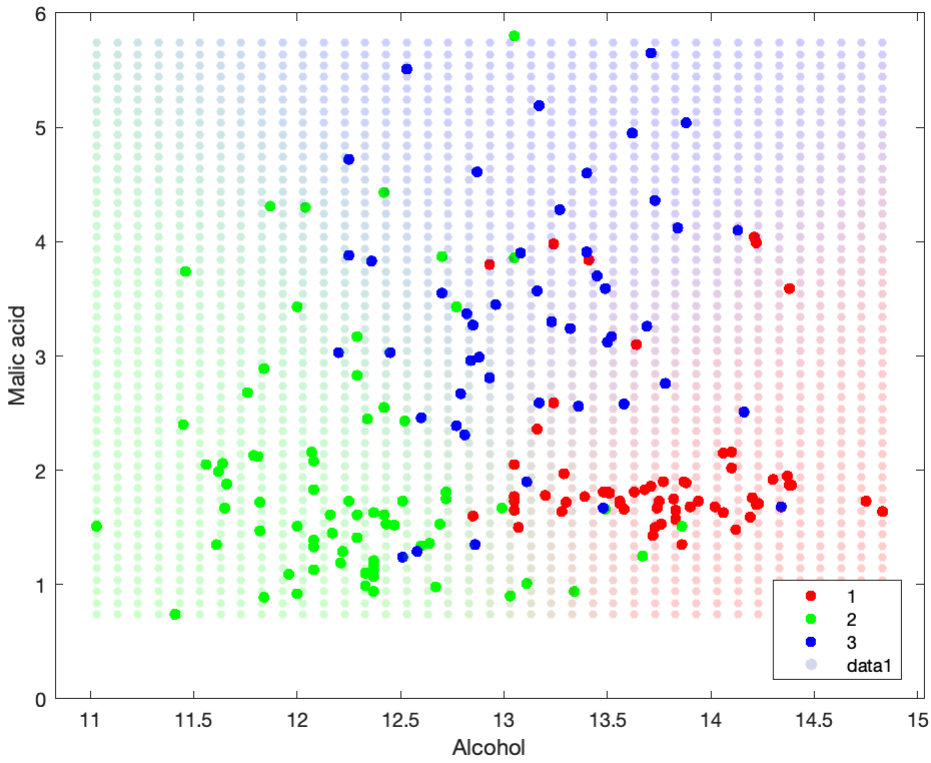



線形 SVM による分類より，RBF カーネルを用いた SVM の方が上手く分類できているが，多少の誤分類が含まれる結果となった．

今度は，２クラスではなく，多クラス（＝３クラス）の分類を行う．

```
clear;
load('wine.mat');
X = wine(:,2:3);
Y = wine(:,1);
```

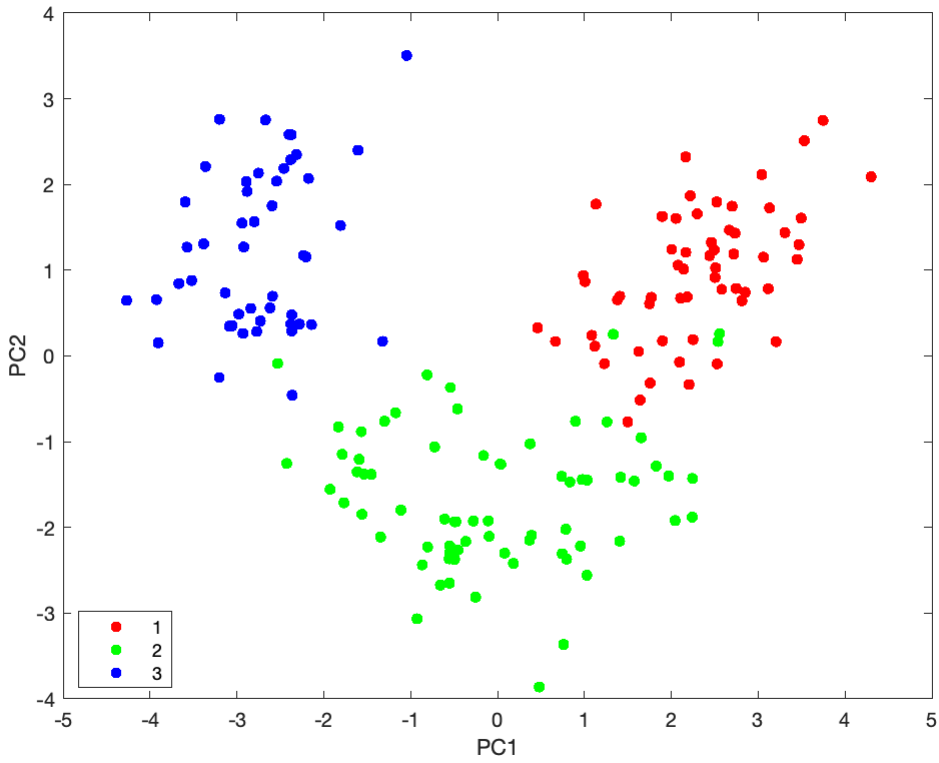
```
SVMModel = fitcecoc(X, Y, 'FitPosterior',true);
SVMModel.predict(X(1,:));
gscatter(X(:,1), X(:,2), Y, 'rgb');
hold on;
h=0.1;
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2)));
[~,~,~, posteriors]  = SVMModel.predict([X1(:), X2(:)]);  % calculate scores at the ea
scatter(X1(:), X2(:) , 20, posteriors, "filled", "MarkerFaceAlpha", 0.2);
xlabel('Alcohol')
ylabel('Malic acid')
hold off;
```




各クラスにおいて，誤分類が多く見られる．

そこで，前回の講義で扱った PCA を用いて，標準化した入力特徴量を次元削減した後，SVM による分類を試みる．

```
x = normalize(wine(:,2:end));
y = wine(:,1);
[coeff, score, latent, ~, explained, mu] = pca(x);
score;
figure;
gscatter(score(:,1), score(:,2), y, 'rgb');
xlabel('PC1');
ylabel('PC2');
```

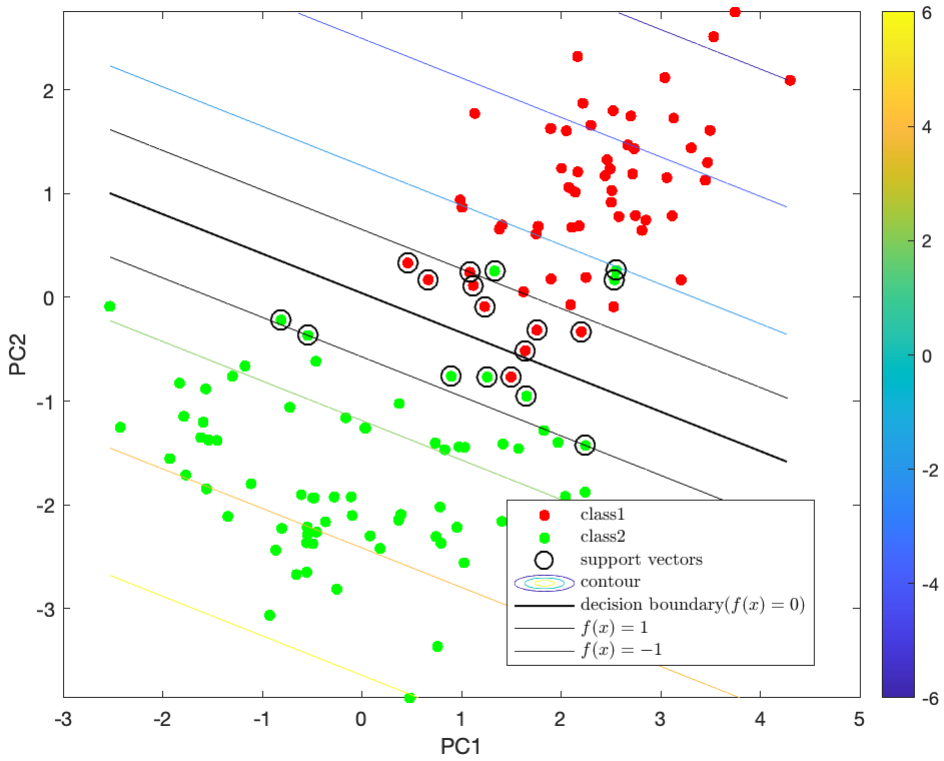



まずは，第一主成分，第二主成分を用いて，２クラスの分類を試してみる．

```
indices = 1:130;
X = score(indices,1:2);
Y = y(indices,1);
```

```
SVMModel = fitcsvm(X,Y, "BoxConstraint", 1);
svInd = SVMModel.IsSupportVector;
h = 0.1; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2))); % calculate
[~,score_] = SVMModel.predict([X1(:),X2(:)]);  % calculate scores at the each grid poi
scoreGrid = reshape(score_(:,2),size(X1,1),size(X2,2));
figure
gscatter(X(:,1), X(:,2), Y, 'rg');
hold on
plot(X(svInd,1),X(svInd,2),'ko','MarkerSize',10, "LineWidth",1) % Mark support vectors
contour(X1,X2,scoreGrid); % draw contour
contour(X1,X2,scoreGrid, [0 0],'k', "LineWidth", 1); %draw decision bounday
contour(X1,X2,scoreGrid, [1 1],'k'); %draw f(x)=1
contour(X1,X2,scoreGrid, [-1 -1],'k'); %draw f(x)=-1
colorbar;
xlabel('PC1')
ylabel('PC2')
legend(["class1", "class2", "support vectors", "contour", "decision boundary($f(x)=0$)"
```

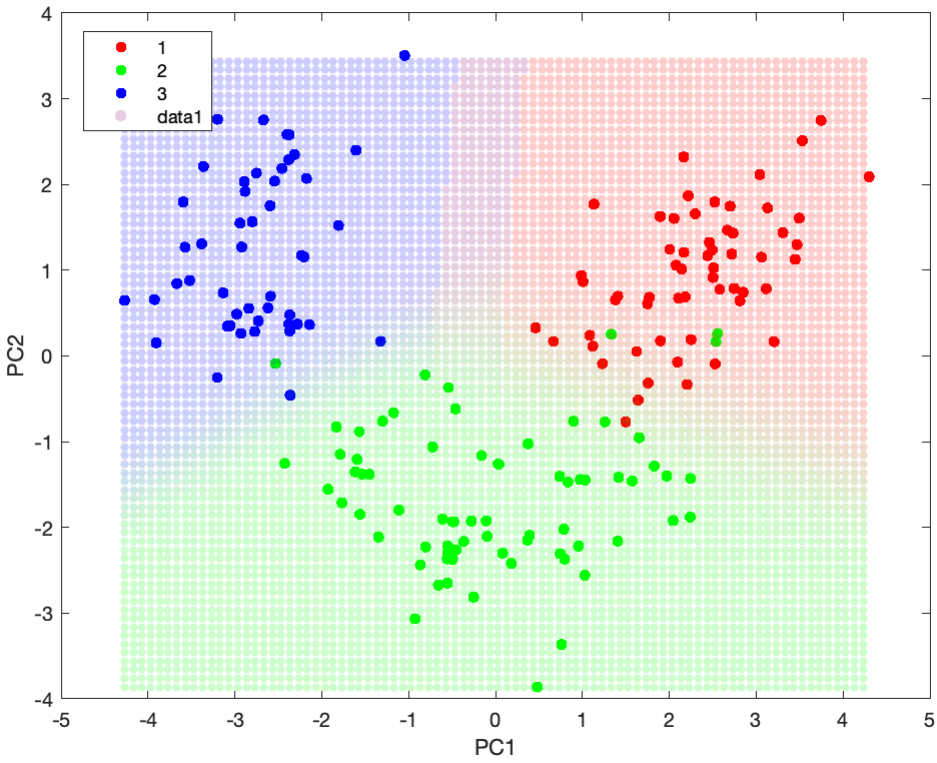
```
hold off
```




直線の境界線は引けないので，線形分類は難しそうである．

今度は，マルチクラスで行う．

```
X = score(:,1:2);
Y = y(:,1);
SVMModel = fitcecoc(X, Y, 'FitPosterior',true);
SVMModel.predict(X(1,:));
gscatter(X(:,1), X(:,2), Y, 'rgb');
hold on;
h=0.1;
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2)));
[~,~,~, posteriors]  = SVMModel.predict([X1(:), X2(:)]);  % calculate scores at the ea
scatter(X1(:), X2(:) , 20, posteriors, "filled", "MarkerFaceAlpha", 0.2);
xlabel('PC1')
ylabel('PC2')
hold off;
```




プロットした図をみると，全体的に上手く境界線が引けるように３クラスの分類ができている．特に，青色と赤色の点でプロットされたクラス１とクラス３の誤分類は，あまり見られない．

PCA を用いずに SVM で分類した結果と比較しても，PCA を用いて次元削減して，SVM で分類する方が良いことがわかる．標準化した入力特徴量を，PCA を用いて次元削減した後，SVM により分類することで，非常に上手く分類を行うことができた．

## 課題 2.　授業の感想

授業の演習・課題を通して，SVM の理論及び実践的な使い方まで習得することができた．特に，前回の講義であつかった PCA による次元削減を SVM と組み合わせることで，効率良く分類問題を扱えることが分かって楽しかった．講義で扱う内容が，次の項目と密接に繋がっており，順序良く講義の順番を組んでいただいていると思った．

現在の画像認識・自然言語処理などの多くの分類問題の解法の主流は，ニューラルネットワークによる深層学習だが，分類するのが安易そうなデータセットやデータ量が少ない場合に対しては，SVM による分類は十分有効だと思うので，CPU でも早く学習が可能な SVM は，一つの選択肢として今後も使えると思う．

## 課題３．　自身で用意したデータセットでの SVM による分類

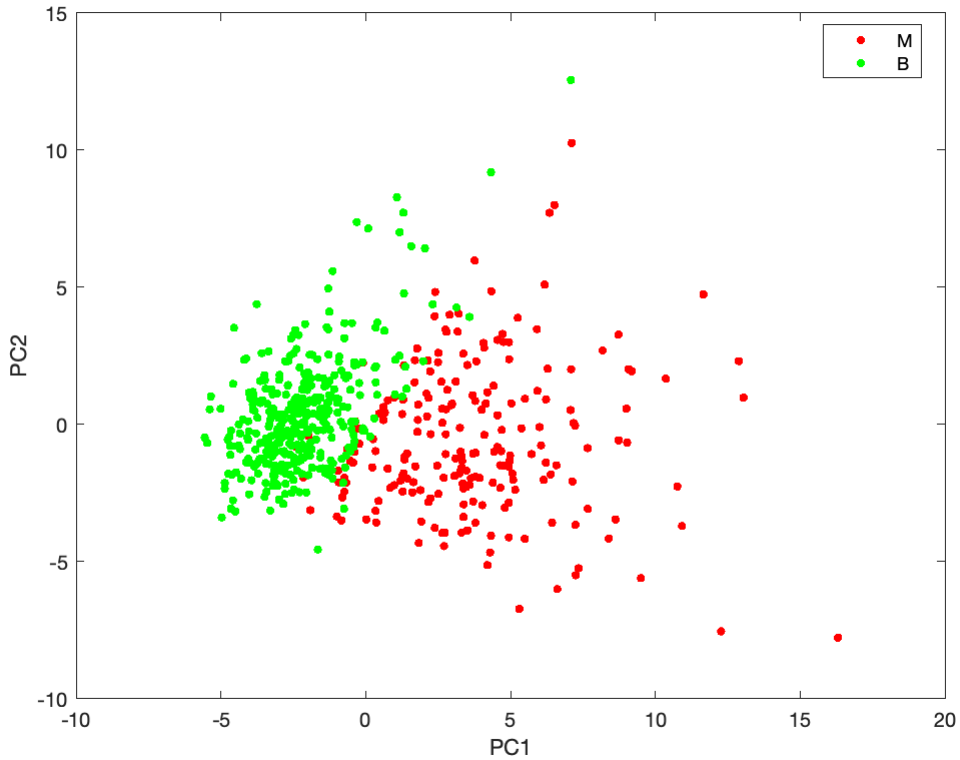
https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

上記 URL の乳がんのデータセットを使う.

５６９人の被験者に対して，１列目に ID，２列目に診断結果(良性:B/悪性:M)，３列目以降は様々な計測値が格納されている.

ここでは，標準化した入力特徴量を，PCA により次元削減し，第一主成分と第二主成分から，診断結果(良性:B/悪性:M)を SVM で分類することを試みる.

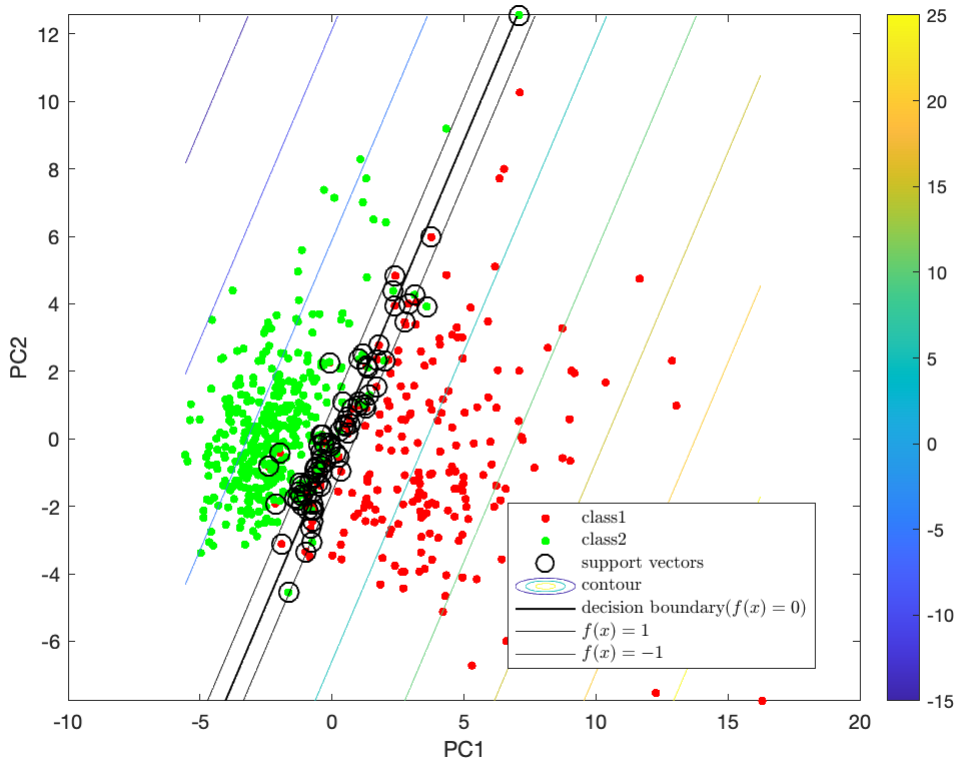
```
clear;
data = readmatrix('data.csv');
data(:,3:end);
X = normalize(data(:,3:end));
s = readtable('data.csv');
Y = string(s{:,2});
[coeff, score, latent, ~, explained, mu] = pca(X);
figure;
gscatter(score(:,1), score(:,2), Y, 'rg');
xlabel('PC1');
ylabel('PC2');
```




まずは，ソフトマージン(C=1)の線形 SVM による分類を行う.

```
X = score(:,1:2);
SVMModel = fitcsvm(X,Y, "BoxConstraint", 1);
svInd = SVMModel.IsSupportVector;
h = 0.1; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2))); % calculate
```

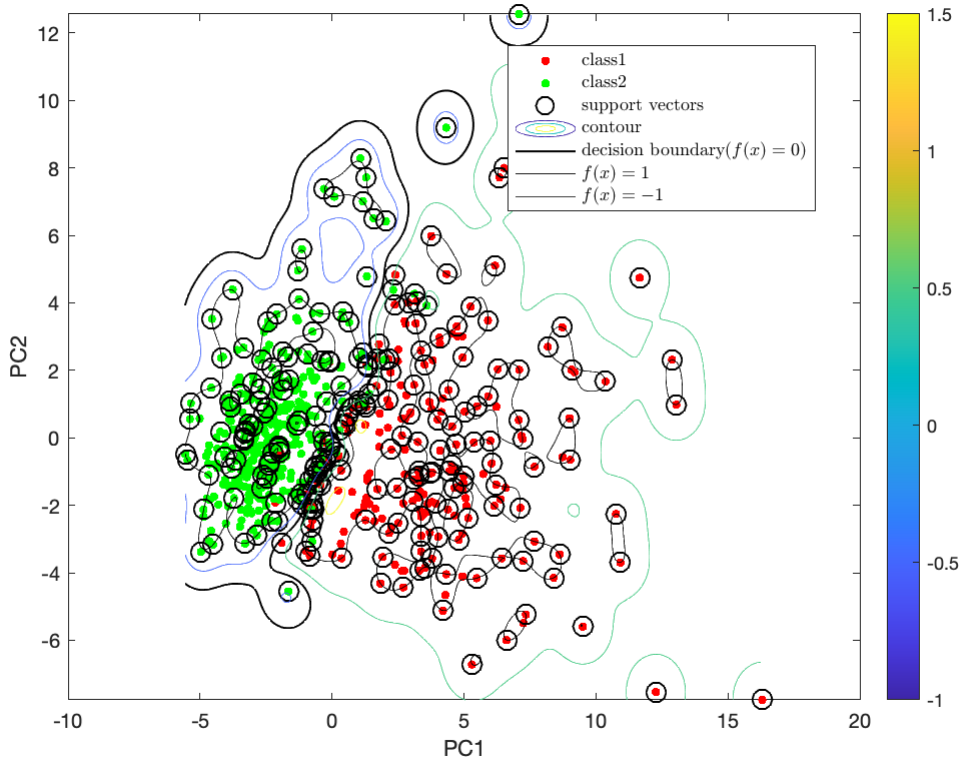
```
[~,score] = SVMModel.predict([X1(:),X2(:)]);  % calculate scores at the each grid point
scoreGrid = reshape(score(:,2),size(X1,1),size(X2,2));
figure
gscatter(X(:,1), X(:,2), Y, 'rg');
hold on
plot(X(svInd,1),X(svInd,2),'ko','MarkerSize',10, "LineWidth",1) % Mark support vectors
contour(X1,X2,scoreGrid); % draw contour
contour(X1,X2,scoreGrid, [0 0],'k', "LineWidth", 1); %draw decision bounday
contour(X1,X2,scoreGrid, [1 1],'k'); %draw f(x)=1
contour(X1,X2,scoreGrid, [-1 -1],'k'); %draw f(x)=-1
colorbar;
xlabel('PC1')
ylabel('PC2')
legend(["class1", "class2", "support vectors", "contour", "decision boundary($f(x)=0$)"
hold off
```




線形 SVM による直線の境界線では，分類が難しいため，次に，カーネルにガウスを採用した RBF カーネルを用いた SVM を適用する.

```
SVMModel = fitcsvm(X,Y, "BoxConstraint", 1, "KernelFunction", "gaussian");
svInd = SVMModel.IsSupportVector;
h = 0.1; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2))); % calculate
[~,score] = SVMModel.predict([X1(:),X2(:)]);  % calculate scores at the each grid point
scoreGrid = reshape(score(:,2),size(X1,1),size(X2,2));
figure
gscatter(X(:,1), X(:,2), Y, 'rg');
hold on
```

```
plot(X(svInd,1),X(svInd,2),'ko','MarkerSize',10, "LineWidth",1) % Mark support vectors
contour(X1,X2,scoreGrid); % draw contour
contour(X1,X2,scoreGrid, [0 0],'k', "LineWidth", 1); %draw decision bounday
contour(X1,X2,scoreGrid, [1 1],'k'); %draw f(x)=1
contour(X1,X2,scoreGrid, [-1 -1],'k'); %draw f(x)=-1
colorbar;
xlabel('PC1')
ylabel('PC2')
legend(["class1", "class2", "support vectors", "contour", "decision boundary($f(x)=0$)"
hold off
```




大まかに診断結果の良・悪を分類することができていることがわかる.

今度はグリッドサーチにより，ハイパーパラメータの最適化を試みる.

```
SVMModel = fitcsvm(X, Y, "KernelFunction", "gaussian", ...
    'OptimizeHyperparameters','auto',...
    'HyperparameterOptimizationOptions',struct('AcquisitionFunctionName',...
    'expected-improvement-plus', "Optimizer", "gridsearch"));
```

| Iter | Eval result | Objective | Objective runtime | BestSoFar (observed) | BoxConstraint | KernelScale |
|---|---|---|---|---|---|---|
| 1 | Best | 0.37258 | 0.25355 | 0.37258 | 2.1544 | 0.021544 |
| 2 | Best | 0.34271 | 0.090927 | 0.34271 | 2.1544 | 0.1 |
| 3 | Accept | 0.37258 | 0.091739 | 0.34271 | 215.44 | 0.001 |
| 4 | Accept | 0.37258 | 0.078593 | 0.34271 | 0.46416 | 1000 |
| 5 | Accept | 0.37258 | 0.095687 | 0.34271 | 215.44 | 0.021544 |

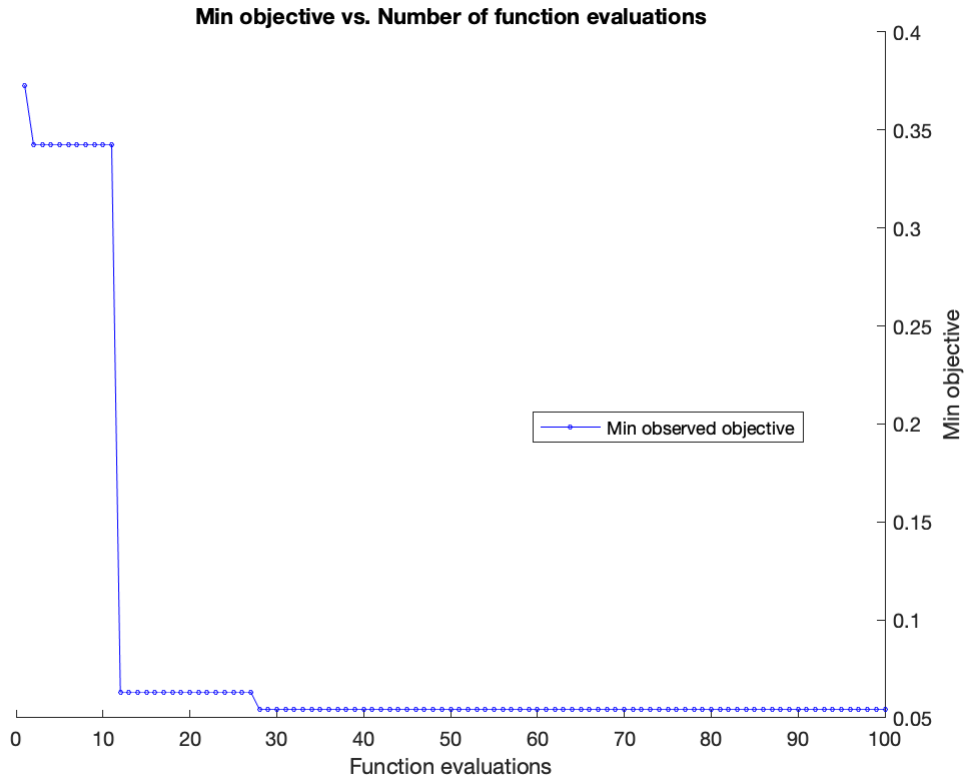
|    6 | Accept |     0.37258 |    0.073551 |     0.34271 |     0.021544 |         1000 |
|    7 | Accept |     0.37258 |    0.074726 |     0.34271 |           10 |     0.021544 |
|    8 | Accept |     0.37258 |    0.063377 |     0.34271 |    0.0046416 |       46.416 |
|    9 | Accept |     0.37258 |    0.074304 |     0.34271 |       2.1544 |    0.0046416 |
|   10 | Accept |     0.37258 |    0.058991 |     0.34271 |        0.001 |     0.021544 |
|   11 | Accept |     0.37258 |    0.076141 |     0.34271 |         1000 |        0.001 |
|   12 | Best   |    0.063269 |    0.053616 |    0.063269 |       46.416 |       46.416 |
|   13 | Accept |     0.37258 |    0.068756 |    0.063269 |        0.001 |       46.416 |
|   14 | Accept |     0.37258 |     0.06056 |    0.063269 |        0.001 |        0.001 |
|   15 | Accept |    0.072056 |    0.059053 |    0.063269 |         1000 |         1000 |
|   16 | Accept |     0.35852 |    0.065075 |    0.063269 |       46.416 |         1000 |
|   17 | Accept |     0.37258 |    0.070264 |    0.063269 |    0.0046416 |           10 |
|   18 | Accept |    0.065026 |    0.061513 |    0.063269 |          0.1 |       2.1544 |
|   19 | Accept |    0.065026 |    0.057489 |    0.063269 |         1000 |       215.44 |
|   20 | Accept |     0.37258 |    0.069481 |    0.063269 |     0.021544 |        0.001 |

| Iter | Eval result | Objective | Objective runtime | BestSoFar (observed) | BoxConstraint | KernelScale |
|---|---|---|---|---|---|---|
|   21 | Accept |     0.11951 |    0.067642 |    0.063269 |     0.021544 |           10 |
|   22 | Accept |    0.087873 |    0.088084 |    0.063269 |      0.46416 |      0.46416 |
|   23 | Accept |     0.34271 |    0.082348 |    0.063269 |       46.416 |          0.1 |
|   24 | Accept |    0.072056 |    0.058157 |    0.063269 |       215.44 |       215.44 |
|   25 | Accept |     0.37258 |    0.071403 |    0.063269 |          0.1 |       215.44 |
|   26 | Accept |     0.37258 |    0.060408 |    0.063269 |        0.001 |       215.44 |
|   27 | Accept |     0.37258 |     0.07782 |    0.063269 |      0.46416 |    0.0046416 |
|   28 | Best   |    0.054482 |    0.065795 |    0.054482 |         1000 |       46.416 |
|   29 | Accept |     0.37258 |    0.067994 |    0.054482 |    0.0046416 |     0.021544 |
|   30 | Accept |     0.37258 |    0.079544 |    0.054482 |      0.46416 |     0.021544 |
|   31 | Accept |     0.37258 |    0.077741 |    0.054482 |           10 |    0.0046416 |
|   32 | Accept |     0.37258 |    0.081075 |    0.054482 |       46.416 |     0.021544 |
|   33 | Accept |     0.37258 |     0.07728 |    0.054482 |         1000 |     0.021544 |
|   34 | Accept |     0.37258 |    0.078794 |    0.054482 |         1000 |    0.0046416 |
|   35 | Accept |    0.068541 |    0.056612 |    0.054482 |           10 |       46.416 |
|   36 | Accept |     0.37258 |     0.07625 |    0.054482 |    0.0046416 |          0.1 |
|   37 | Accept |    0.094903 |    0.087416 |    0.054482 |       46.416 |      0.46416 |
|   38 | Accept |     0.37258 |    0.067159 |    0.054482 |     0.021544 |       215.44 |
|   39 | Accept |    0.061511 |    0.055538 |    0.054482 |           10 |           10 |
|   40 | Accept |     0.37258 |    0.066148 |    0.054482 |        0.001 |           10 |

| Iter | Eval result | Objective | Objective runtime | BestSoFar (observed) | BoxConstraint | KernelScale |
|---|---|---|---|---|---|---|
|   41 | Accept |    0.065026 |    0.089365 |    0.054482 |       46.416 |       2.1544 |
|   42 | Accept |     0.12302 |    0.086941 |    0.054482 |         1000 |      0.46416 |
|   43 | Accept |     0.37258 |    0.068252 |    0.054482 |      0.46416 |       215.44 |
|   44 | Accept |    0.072056 |    0.064627 |    0.054482 |       46.416 |       215.44 |
|   45 | Accept |     0.37258 |    0.075335 |    0.054482 |      0.46416 |        0.001 |
|   46 | Accept |     0.37258 |    0.066453 |    0.054482 |        0.001 |       2.1544 |
|   47 | Accept |    0.072056 |    0.058567 |    0.054482 |       2.1544 |       46.416 |
|   48 | Accept |    0.057996 |    0.056281 |    0.054482 |       215.44 |       46.416 |
|   49 | Accept |     0.35852 |    0.064516 |    0.054482 |       2.1544 |       215.44 |
|   50 | Accept |     0.37258 |     0.06281 |    0.054482 |    0.0046416 |       2.1544 |
|   51 | Accept |     0.37258 |    0.066195 |    0.054482 |    0.0046416 |         1000 |
|   52 | Accept |     0.37258 |    0.068223 |    0.054482 |     0.021544 |    0.0046416 |
|   53 | Accept |     0.37258 |    0.079043 |    0.054482 |          0.1 |          0.1 |
|   54 | Accept |     0.37258 |    0.062284 |    0.054482 |        0.001 |    0.0046416 |
|   55 | Accept |    0.059754 |     0.05481 |    0.054482 |       2.1544 |           10 |
|   56 | Accept |     0.37258 |    0.073907 |    0.054482 |          0.1 |     0.021544 |
|   57 | Accept |     0.11775 |    0.080859 |    0.054482 |       215.44 |         1000 |
|   58 | Accept |    0.057996 |    0.063135 |    0.054482 |           10 |       2.1544 |
|   59 | Accept |    0.063269 |     0.06122 |    0.054482 |       46.416 |           10 |
|   60 | Accept |     0.37258 |    0.064552 |    0.054482 |          0.1 |         1000 |

| Iter | Eval result | Objective | Objective runtime | BestSoFar (observed) | BoxConstraint | KernelScale |
|---|---|---|---|---|---|---|
| 61 | Accept | 0.37258 | 0.073383 | 0.054482 | 0.021544 | 0.021544 |
| 62 | Accept | 0.37258 | 0.078466 | 0.054482 | 46.416 | 0.0046416 |
| 63 | Accept | 0.091388 | 0.087149 | 0.054482 | 10 | 0.46416 |
| 64 | Accept | 0.10193 | 0.073135 | 0.054482 | 0.021544 | 2.1544 |
| 65 | Accept | 0.056239 | 0.066931 | 0.054482 | 0.46416 | 2.1544 |
| 66 | Accept | 0.37258 | 0.066921 | 0.054482 | 2.1544 | 1000 |
| 67 | Accept | 0.11951 | 0.069017 | 0.054482 | 0.46416 | 46.416 |
| 68 | Accept | 0.10369 | 0.092663 | 0.054482 | 215.44 | 0.46416 |
| 69 | Accept | 0.37258 | 0.076385 | 0.054482 | 0.1 | 0.46416 |
| 70 | Accept | 0.063269 | 0.12673 | 0.054482 | 1000 | 10 |
| 71 | Accept | 0.37258 | 0.066727 | 0.054482 | 10 | 1000 |
| 72 | Accept | 0.37258 | 0.078322 | 0.054482 | 10 | 0.001 |
| 73 | Accept | 0.079086 | 0.085432 | 0.054482 | 2.1544 | 0.46416 |
| 74 | Accept | 0.11775 | 0.063181 | 0.054482 | 10 | 215.44 |
| 75 | Accept | 0.37258 | 0.06947 | 0.054482 | 0.0046416 | 0.46416 |
| 76 | Accept | 0.37258 | 0.077852 | 0.054482 | 46.416 | 0.001 |
| 77 | Accept | 0.37258 | 0.067626 | 0.054482 | 0.001 | 0.46416 |
| 78 | Accept | 0.070299 | 0.28587 | 0.054482 | 1000 | 2.1544 |
| 79 | Accept | 0.37258 | 0.067924 | 0.054482 | 0.0046416 | 0.001 |
| 80 | Accept | 0.37258 | 0.063714 | 0.054482 | 0.001 | 1000 |

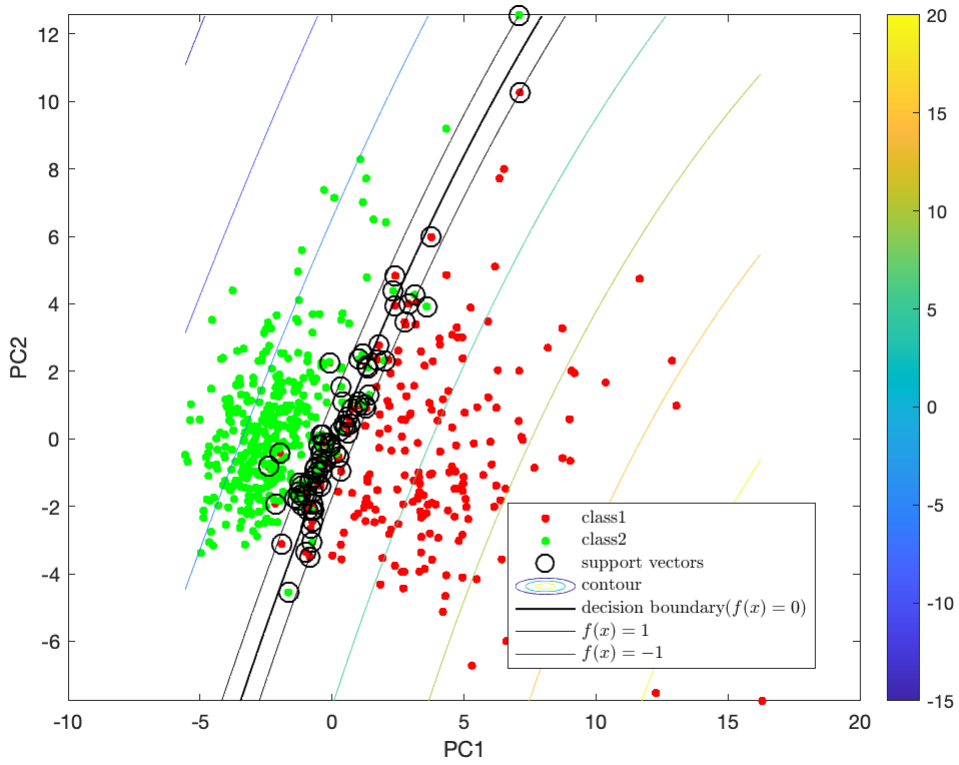
| Iter | Eval result | Objective | Objective runtime | BestSoFar (observed) | BoxConstraint | KernelScale |
|---|---|---|---|---|---|---|
| 81 | Accept | 0.065026 | 0.05884 | 0.054482 | 0.46416 | 10 |
| 82 | Accept | 0.34271 | 0.087684 | 0.054482 | 1000 | 0.1 |
| 83 | Accept | 0.063269 | 0.10008 | 0.054482 | 215.44 | 10 |
| 84 | Accept | 0.37258 | 0.081144 | 0.054482 | 0.021544 | 0.1 |
| 85 | Accept | 0.37258 | 0.069912 | 0.054482 | 0.021544 | 46.416 |
| 86 | Accept | 0.34271 | 0.090742 | 0.054482 | 10 | 0.1 |
| 87 | Accept | 0.36028 | 0.064777 | 0.054482 | 0.1 | 46.416 |
| 88 | Accept | 0.37258 | 0.078674 | 0.054482 | 215.44 | 0.0046416 |
| 89 | Accept | 0.37258 | 0.065952 | 0.054482 | 0.0046416 | 215.44 |
| 90 | Accept | 0.37258 | 0.073093 | 0.054482 | 0.1 | 0.0046416 |
| 91 | Accept | 0.37258 | 0.070399 | 0.054482 | 0.021544 | 0.46416 |
| 92 | Accept | 0.075571 | 0.058351 | 0.054482 | 0.1 | 10 |
| 93 | Accept | 0.37258 | 0.072923 | 0.054482 | 0.1 | 0.001 |
| 94 | Accept | 0.37258 | 0.067679 | 0.054482 | 2.1544 | 0.001 |
| 95 | Accept | 0.37258 | 0.071217 | 0.054482 | 0.0046416 | 0.0046416 |
| 96 | Accept | 0.37258 | 0.08439 | 0.054482 | 0.46416 | 0.1 |
| 97 | Accept | 0.37258 | 0.066328 | 0.054482 | 0.001 | 0.1 |
| 98 | Accept | 0.063269 | 0.06371 | 0.054482 | 2.1544 | 2.1544 |
| 99 | Accept | 0.34271 | 0.086128 | 0.054482 | 215.44 | 0.1 |
| 100 | Accept | 0.068541 | 0.12686 | 0.054482 | 215.44 | 2.1544 |




```
Optimization completed.
MaxObjectiveEvaluations of 100 reached.
Total function evaluations: 100
Total elapsed time: 21.6349 seconds
Total objective function evaluation time: 7.6838

Best observed feasible point:
    BoxConstraint    KernelScale
    _____________    ___________

        1000           46.416

Observed objective function value = 0.054482
Function evaluation time = 0.065795
```

```
svInd = SVMModel.IsSupportVector;
h = 0.1; % Mesh grid step size
[X1,X2] = meshgrid(min(X(:,1)):h:max(X(:,1)), min(X(:,2)):h:max(X(:,2))); % calculate g
[~,score] = SVMModel.predict([X1(:),X2(:)]);  % calculate scores at the each grid point
scoreGrid = reshape(score(:,2),size(X1,1),size(X2,2));
figure
gscatter(X(:,1), X(:,2), Y, 'rg');
hold on
plot(X(svInd,1),X(svInd,2),'ko','MarkerSize',10, "LineWidth",1) % Mark support vectors
contour(X1,X2,scoreGrid); % draw contour
contour(X1,X2,scoreGrid, [0 0],'k', "LineWidth", 1); %draw decision bounday
contour(X1,X2,scoreGrid, [1 1],'k'); %draw f(x)=1
contour(X1,X2,scoreGrid, [-1 -1],'k'); %draw f(x)=-1
colorbar;
```

```
xlabel('PC1')
ylabel('PC2')
legend(["class1", "class2", "support vectors", "contour", "decision boundary($f(x)=0$)"
hold off
```




パラメータを最適化することで，上記の図のようにデータから診断結果の２クラスに分類できていることがわかる.