Contents

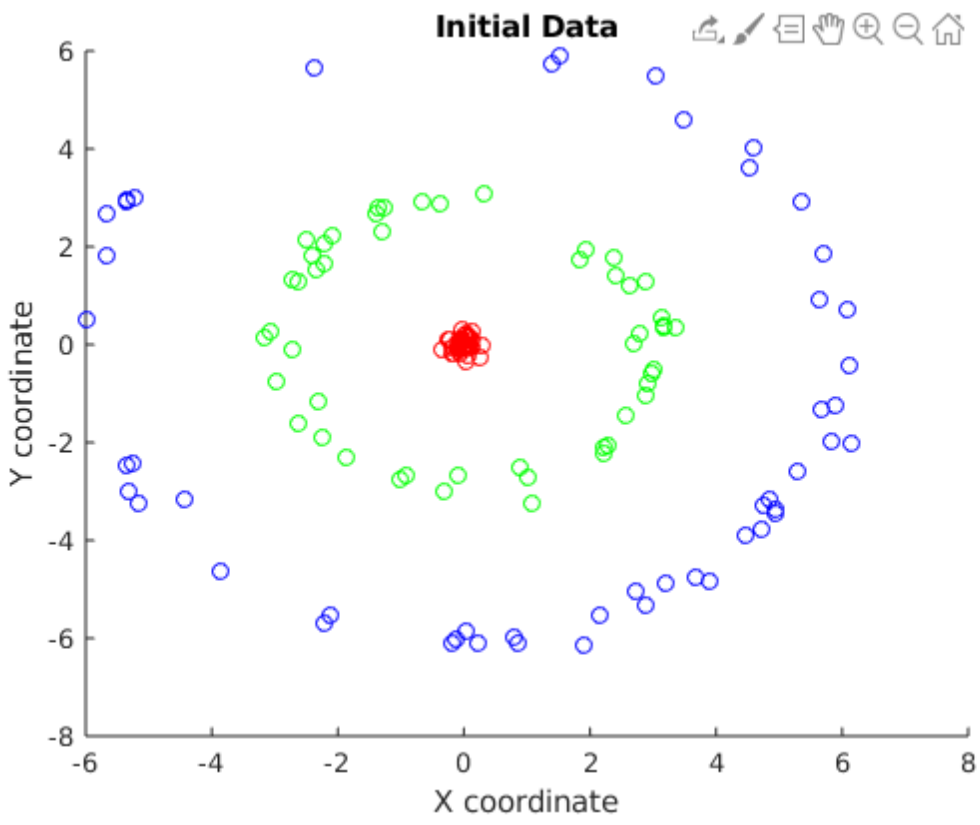

```
clear;clc; close all; rng(1);
sampleSize = 50;
limit = 2;
```

Kernel PCA on circular DATA

In this code, we have generated some circular data with some noise (2D data) We will perform kernel PCA on this data using gaussian and polynomial kernels

Generate Data

```
dataset = dataGenerator(true, sampleSize);
```



Create the Kernel Matrix from uncentered Data

```
KG = kernelMatrixCalculator(dataset, 'gauss');
[N, ~] = size(dataset);
```

```
oneN=ones(N)/N;
K_gaussian=KG-oneN*KG-KG*oneN+oneN*KG*oneN;

K_gaussian = (K_gaussian + K_gaussian')/2;

KP = kernelMatrixCalculator(dataset, 'poly');
K_polynomial=KP-oneN*KP-KP*oneN+oneN*KP*oneN;

K_polynomial = (K_polynomial + K_polynomial')/2;
```

Perform Normalisation on matrices

```
K_polynomial = K_polynomial / N;
K_gaussian = K_gaussian / N;

K_polynomial = K_polynomial ./ repmat(std(K_polynomial), N, 1);
K_gaussian = K_gaussian ./ repmat(std(K_gaussian), N, 1);
```

Perform SVD

```
[VG, EG] = eig(K_gaussian);
[VP, EP] = eig(K_polynomial);

EigvalsG=diag(EG);
[~,IX]=sort(EigvalsG,'descend');
VecG=VG(:,IX);
EigvalsG = EigvalsG(IX);
VecG = VecG(:, 1:limit);
EigvalsG = EigvalsG(1:limit);

EigvalsP=diag(EP);
[~,IX]=sort(EigvalsP,'descend');
VecP=VP(:,IX);
EigvalsP = EigvalsP(IX);
VecP = VecP(:, 1:limit);
EigvalsP = EigvalsP(1:limit);
```

Perform Normalization

For gaussian kernel:

```
div=sqrt(EigvalsG);
VecG=VecG./(div*ones(1,N))';
% For polynomial kernel
div=sqrt(EigvalsP);
VecP=VecP./(div*ones(1,N))';
%
```

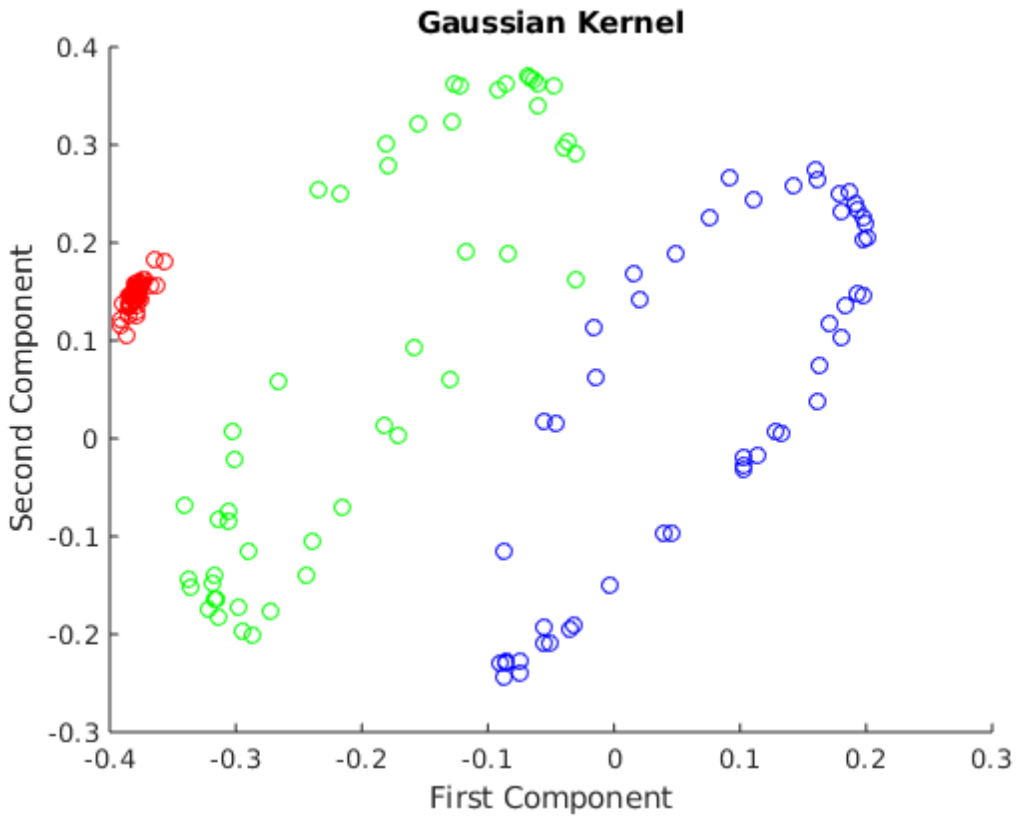
Transform Data into new dimensions (along the best d eigenvectors)

```
YG = projectData(VecG, KG, 2);
YP = projectData(VecP, KP, 2);
```

Plot the Gaussian kernal data

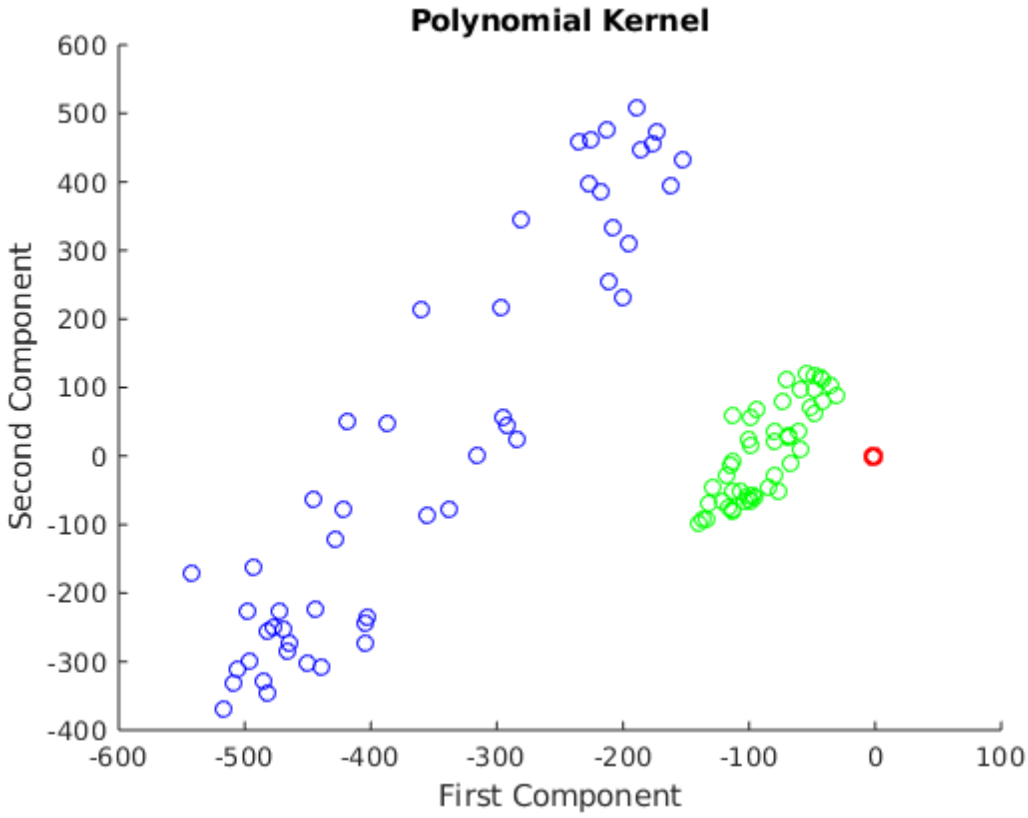
```
figure(2);
a = sampleSize;
b = 2 * a;
```

```
c = 3 * a;
scatter(YG(1:a,1), YG(1:a,2), 'r');
hold on;
scatter(YG(a+1:b,1), YG(a+1:b,2), 'g');
hold on;
scatter(YG(b+1:c,1), YG(b+1:c,2), 'b');
hold on;
xlabel('First Component');
ylabel('Second Component');
title('Gaussian Kernel');
hold off;
```



Plot the Polynomial kernal data

```
figure(3);
scatter(YP(1:a,1), YP(1:a,2), 'r');
hold on;
scatter(YP(a+1:b,1), YP(a+1:b,2), 'g');
hold on;
scatter(YP(b+1:c,1), YP(b+1:c,2), 'b');
hold on;
xlabel('First Component');
ylabel('Second Component');
title('Polynomial Kernel');
hold off;
```



Published with MATLAB® R2019b