Table of Contents

```
% Austin Welch
% EC503 HW8.1
% K-means vs. Spectral Clustering
clear all; clc; tic; %#ok<CLALL>
```

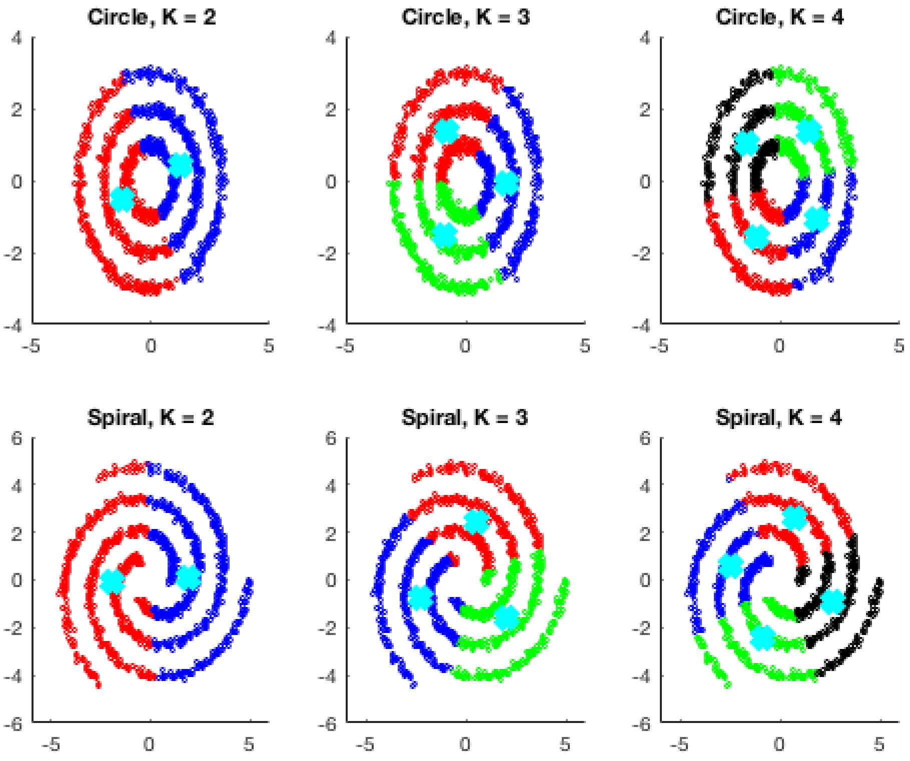
(a)

K-means Clustering, centroids, distance sums

```
% generate data
[D1, Label1] = sample_circle(3);
[D2, Label2] = sample_spiral(3);

figure(1);
K = [2 3 4];
colors = {'red','blue','green','black'};
% try different number of clusters
for i=K
    rng(2); % seed the random nubmer generator
    % K-means clustering

 [IDX1,C1,SUMD1]=kmeans(D1,i,'Replicates',20,'Distance','sqeuclidean');
    rng(2); % seed the random nubmer generator

 [IDX2,C2,SUMD2]=kmeans(D2,i,'Replicates',20,'Distance','sqeuclidean');

    fprintf(['Within cluster sums of points-to-cluster-centroid\n',...
        '(Euclidean) l_2 squared distances for K = %d (CIRCLE):
\n'],i);

    % loop through labels and plot
    subplot(2,3,i-1);
    for j=1:i
        scatter(D1(IDX1==j,1),D1(IDX1==j,2),5,colors{j});
        xlim([-5 5]);
        ylim([-4 4]);
        hold on
        scatter(C1(j,1),C1(j,2),100,'X','Cyan','LineWidth',10);
        title(sprintf('Circle, K = %d', i));
        fprintf('cluster #%d sum: %0.4f\n', j,SUMD1(j));
    end

    fprintf(['\n\nWithin cluster sums of points-to-cluster-centroid
\n',...
        '(Euclidean) l_2 squared distances for K = %d (SPIRAL):
\n'],i);

    subplot(2,3,i+2);
    for j=1:i
        scatter(D2(IDX2==j,1),D2(IDX2==j,2),5,colors{j});
        xlim([-6 6]);
        ylim([-6 6]);
        hold on;
        scatter(C2(j,1),C2(j,2),100,'X','Cyan','LineWidth',10);
        title(sprintf('Spiral, K = %d', i));
        fprintf('cluster #%d sum: %0.4f\n', j,SUMD2(j));
    end
    fprintf('\n\n');
end

% K-means performs poorly on the circle and spiral datasets because
 they
% are not linearly separable.
```

```
Within cluster sums of points-to-cluster-centroid
(Euclidean) l_2 squared distances for K = 2 (CIRCLE):
cluster #1 sum: 2275.9689
cluster #2 sum: 2213.8406


Within cluster sums of points-to-cluster-centroid
(Euclidean) l_2 squared distances for K = 2 (SPIRAL):
cluster #1 sum: 4714.1095
cluster #2 sum: 4896.8117


Within cluster sums of points-to-cluster-centroid
(Euclidean) l_2 squared distances for K = 3 (CIRCLE):
cluster #1 sum: 909.7201
cluster #2 sum: 1040.0791
cluster #3 sum: 892.0526


Within cluster sums of points-to-cluster-centroid
(Euclidean) l_2 squared distances for K = 3 (SPIRAL):
cluster #1 sum: 1830.5437
cluster #2 sum: 2101.8777
cluster #3 sum: 1974.2920
```

```
Within cluster sums of points-to-cluster-centroid
(Euclidean) l_2 squared distances for K = 4 (CIRCLE):
cluster #1 sum: 567.7306
cluster #2 sum: 444.7208
cluster #3 sum: 547.7926
cluster #4 sum: 536.2319


Within cluster sums of points-to-cluster-centroid
(Euclidean) l_2 squared distances for K = 4 (SPIRAL):
cluster #1 sum: 1231.3936
cluster #2 sum: 956.5762
cluster #3 sum: 1047.8670
cluster #4 sum: 1163.6454
```



(b)

Three variants of spectral clustering: one unnormalized, two normalized

```
% Step 1: Weighted adjacency matrices

% fully-connected graph, W = S (similarity score)
W1 = zeros(length(D1),length(D1));
W2 = zeros(length(D2),length(D2));
```

```
sigma = 0.2; % given std
for i=1:length(D1) % == length(D2)
    for j=1:length(D1)
        % calculate gaussian similarity scores S(xi,xj)
        W1(i,j) = exp(-((D1(i,1)-D1(j,1))^2 + (D1(i,2)-
D1(j,2))^2)/ ...
            (2*sigma.^2));
        W2(i,j) = exp(-((D2(i,1)-D2(j,1))^2 + (D2(i,2)-
D2(j,2))^2)/ ...
            (2*sigma.^2));
    end
end

% Step 2: Degree matrices D
DM1 = diag(sum(W1,2));
DM2 = diag(sum(W2,2));
```

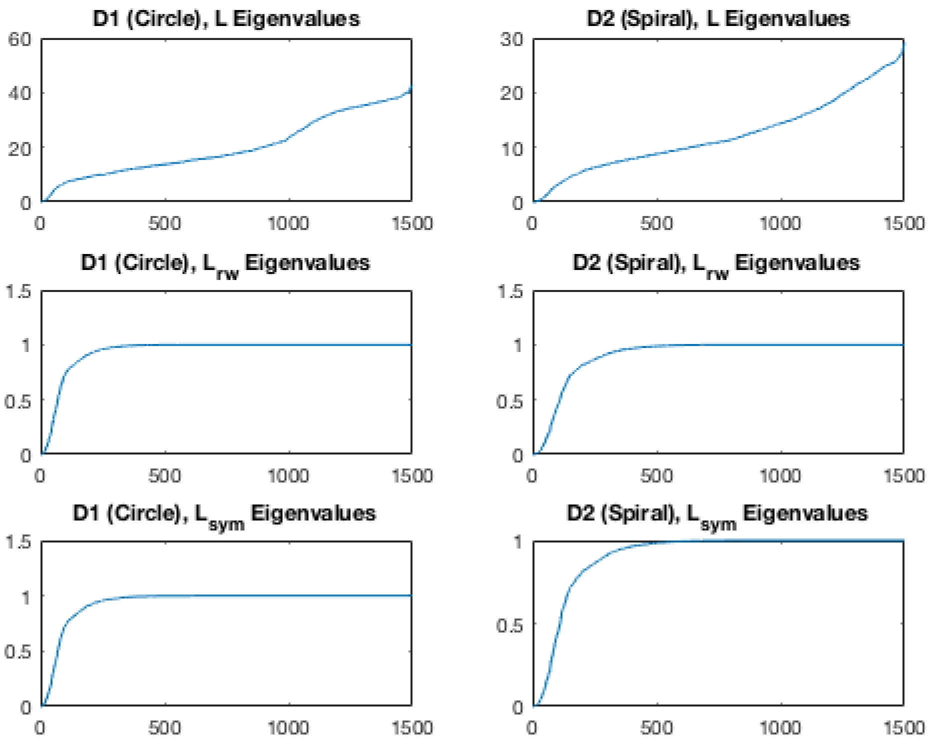
Step 3: Graph Laplacians

```
% Compute the un-normalized graph Laplacian L = D - W
L1 = DM1 - W1;
L2 = DM2 - W2;

% Compute the normalized graph Laplacian L_rw = D^{-1}*L
L1_rw = inv(DM1)*L1; %#ok<*MINV>
L2_rw = inv(DM2)*L2;

% Compute the normalized graph Laplacian L_sym = D^{-1/2}*L*D^{-1/2}
L1_sy = inv(sqrt(DM1))*L1*inv(sqrt(DM1));
L2_sy = inv(sqrt(DM2))*L2*inv(sqrt(DM2));

% Step 4: First K eigenvectors of L, L_rw, L_sym

% full eigenvectors/values for each L
[V1_un,G1_un] = svd(L1);
[V2_un,G2_un] = svd(L2);
[V1_rw,G1_rw] = svd(L1_rw);
[V2_rw,G2_rw] = svd(L2_rw);
[V1_sy,G1_sy] = svd(L1_sy);
[V2_sy,G2_sy] = svd(L2_sy);
```

(i) Plot the eigenvalues

```
figure(2);

subplot(3,2,1);
plot(flipud(diag(G1_un)));
title('D1 (Circle), L Eigenvalues');

subplot(3,2,2);
plot(flipud(diag(G2_un)));
title('D2 (Spiral), L Eigenvalues');
```

```
subplot(3,2,3);
plot(flipud(diag(G1_rw)));
title('D1 (Circle), L_{rw} Eigenvalues');

subplot(3,2,4);
plot(flipud(diag(G2_rw)));
title('D2 (Spiral), L_{rw} Eigenvalues');

subplot(3,2,5);
plot(flipud(diag(G1_sy)));
title('D1 (Circle), L_{sym} Eigenvalues');

subplot(3,2,6);
plot(flipud(diag(G2_sy)));
title('D2 (Spiral), L_{sym} Eigenvalues');
```



D1 (Circle), L Eigenvalues

D2 (Spiral), L Eigenvalues

D1 (Circle), L_{rw} Eigenvalues

D2 (Spiral), L_{rw} Eigenvalues

D1 (Circle), L_{sym} Eigenvalues

D2 (Spiral), L_{sym} Eigenvalues

First K eigenvectors for each L

```
V1cell = cell(3,3);
V2cell = cell(3,3);
for i=K-1
  % V1cell{L,K}
  V1cell{1,i} = V1_un(:,end-i:end);
  V1cell{2,i} = V1_rw(:,end-i:end);
  V1cell{3,i} = V1_sy(:,end-i:end);
  % V2cell{L,K}
  V2cell{1,i} = V2_un(:,end-i:end);
  V2cell{2,i} = V2_rw(:,end-i:end);
  V2cell{3,i} = V2_sy(:,end-i:end);
end

% normalize V_sy rows so that l-2 norms are 1
for i=K-1 % implicit expansion: Matlab 2016b+
    V1cell{3,i} = V1cell{3,i} ./ sqrt(sum(V1cell{3,i}.^2,2));
    V2cell{3,i} = V2cell{3,i} ./ sqrt(sum(V2cell{3,i}.^2,2));
end
```

Step 5: Clustering

Cluster n rows of V with k-means into k clusters, for each L

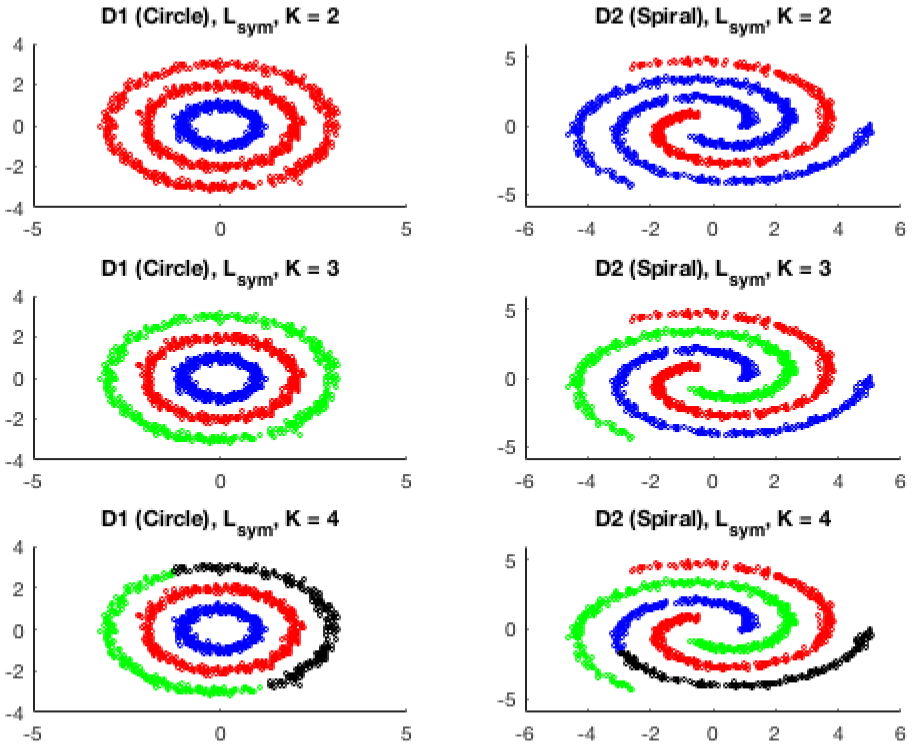
```
% Spectral clustering predictions, {L,K}
SID1 = cell(3,3);
SID2 = cell(3,3);

for i=1:3 % L's
    for j=K % K's
        rng(2); % seed random number generator
        SID1{i,j-1} = kmeans(V1cell{i,j-1},j);
        rng(2);
        SID2{i,j-1} = kmeans(V2cell{i,j-1},j);
    end
end
```

(b)(ii)

For SC-3 (L_sym), Plot predictions for D1,D2, K=2,3,4

```
figure(3);
i1 = [1 3 5]; i2 = [2 4 6];
for i=K
    for j=1:i
        subplot(3,2,i1(i-1));
        hold on;

 scatter(D1(SID1{3,i-1}==j,1),D1(SID1{3,i-1}==j,2),5,colors{j});
        xlim([-5 5]);
        ylim([-4 4]);
        title(sprintf('D1 (Circle), L_{sym}, K = %d', i));

        subplot(3,2,i2(i-1));
        hold on;

 scatter(D2(SID2{3,i-1}==j,1),D2(SID2{3,i-1}==j,2),5,colors{j});
        xlim([-6 6]);
        ylim([-6 6]);
        title(sprintf('D2 (Spiral), L_{sym}, K = %d', i));
    end
```

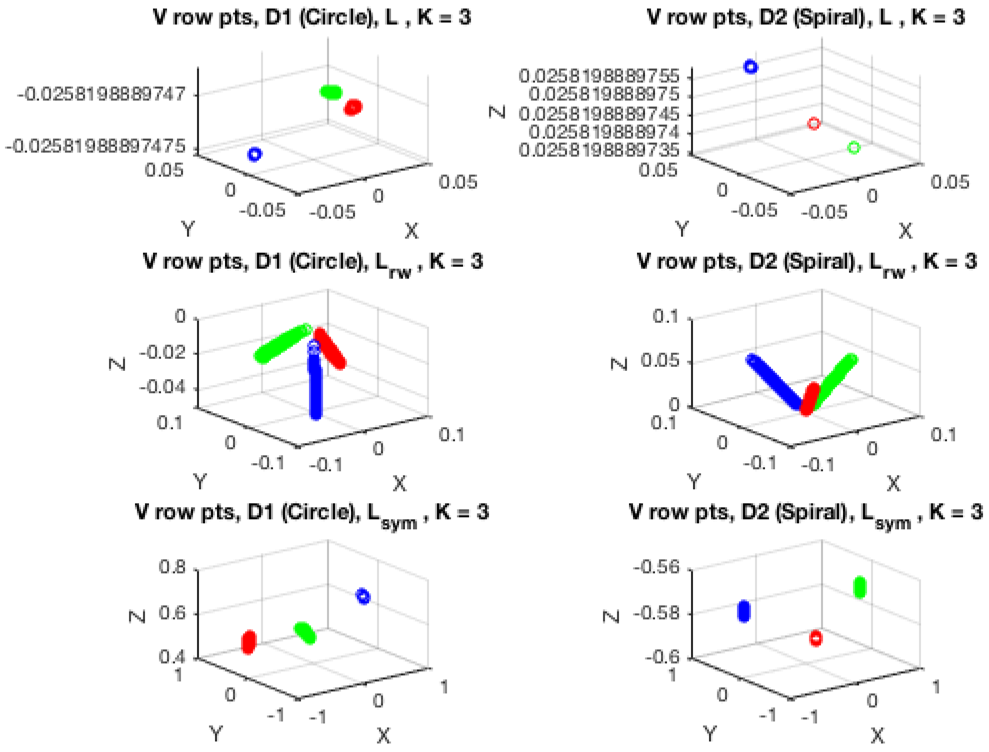
```
end
```



(b)(iii)

For K=3, plot the rows of the V matrices in SC-1,2,3 Normalize the rows to l-2 unit norm before plotting

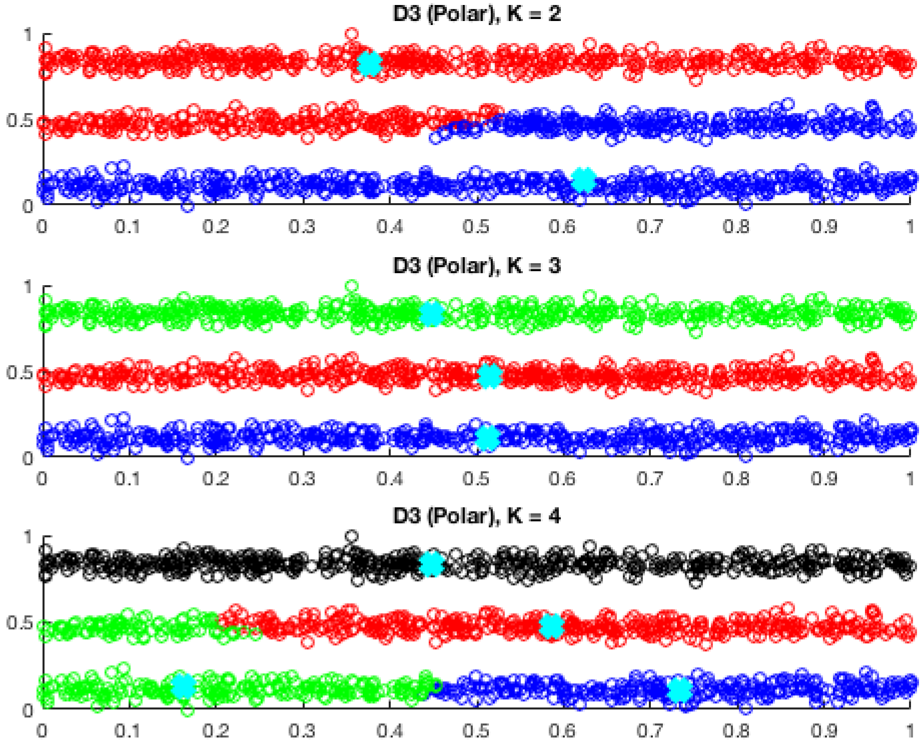
```
%{
% V_sym is already normalized, so do the same for V_un, V_rw
for i=1:2 % {i,2} corresponds to {L(i),K=3}
    V1cell{i,2} = V1cell{i,2} ./ sqrt(sum(V1cell{i,2}.^2,2));
    V2cell{i,2} = V2cell{i,2} ./ sqrt(sum(V2cell{i,2}.^2,2));
end
%}

% Plot
subsc = {'','rw','sym'};
figure(4);
for i=1:3 % L's
        % D1 (Circle)
        subplot(3,2,i1(i));
        scatter3(V1cell{i,2}(:,1),V1cell{i,2}(:,2),V1cell{i,2}
(:,3),...
            30,colormat(SID1{i,2}));
        title(sprintf('V row pts, D1 (Circle), L_{%s} , K =
 3',subsc{i}));
        xlabel('X'); ylabel('Y'); zlabel('Z');
        % D2 (Spiral)
        subplot(3,2,i2(i));
        scatter3(V2cell{i,2}(:,1),V2cell{i,2}(:,2),V2cell{i,2}
(:,3),...
            30,colormat(SID2{i,2}));
        title(sprintf('V row pts, D2 (Spiral), L_{%s} , K = 3',
 subsc{i}));
        xlabel('X'); ylabel('Y'); zlabel('Z');
end
```



(c) Polar coordinates

```
% Transform D1 from cartesian to polar coordinates
% theta,rho  <= x,y
[D3(:,1), D3(:,2)] = cart2pol(D1(:,1),D1(:,2));

% normalize angle and radius each to 0:1
D3(:,1) = (D3(:,1)-min(D3(:,1)))/(max(D3(:,1))-min(D3(:,1)));
D3(:,2) = (D3(:,2)-min(D3(:,2)))/(max(D3(:,2))-min(D3(:,2)));
PIDX = cell(1,3); PCTR = cell(1,3); PSUM = cell(1,3);
for i=K
    rng(2);
    % calculate polar k-means with l-1 distance
    [PIDX{i-1}, PCTR{i-1}, PSUM{i-1}] =
 kmeans(D3,i,'Replicate',20',...
        'Distance','cityblock');
end

% (i) Plot polar k-means clusters with centroids
figure(5)
for i=K
    subplot(3,1,i-1); hold on;
    for j=1:i
        scatter(D3(PIDX{i-1}==j,1),D3(PIDX{i-1}==j,2),colors{j});
        scatter(PCTR{i-1}(:,1),PCTR{i-1}(:,2),100,'X','Cyan',...
            'LineWidth',10);
    end
    title(sprintf('D3 (Polar), K = %d', i));
    % (ii) Report within-cluster sums of point-to-centroid distances
    fprintf(['\n\nWithin-cluster sums of point-to-centroid distances
\n',...
        '(cityblock) for K = %d (ordered from 1,...,k):\n'],i);
    disp(PSUM{i-1});
end

toc
```

```


Within-cluster sums of point-to-centroid distances
(cityblock) for K = 2 (ordered from 1,...,k):
  258.1402
  274.1630



Within-cluster sums of point-to-centroid distances
(cityblock) for K = 3 (ordered from 1,...,k):
  130.7241
  141.2037
  135.2861



Within-cluster sums of point-to-centroid distances
(cityblock) for K = 4 (ordered from 1,...,k):
   85.0166
   44.9184
   73.1753
  135.2861

Elapsed time is 16.222578 seconds.
```



Published with MATLAB® R2017a

Table of Contents

```
% Austin Welch
% EC503 HW8.2
% Spectral Clustering on Airbnb Data
% http://insideairbnb.com/get-the-data.html

clear all; clc; %#ok<*CLALL>
Data = load('BostonListing.mat');
X = [Data.latitude, Data.longitude];
Y = Data.neighbourhood;
```

Spectral clustering

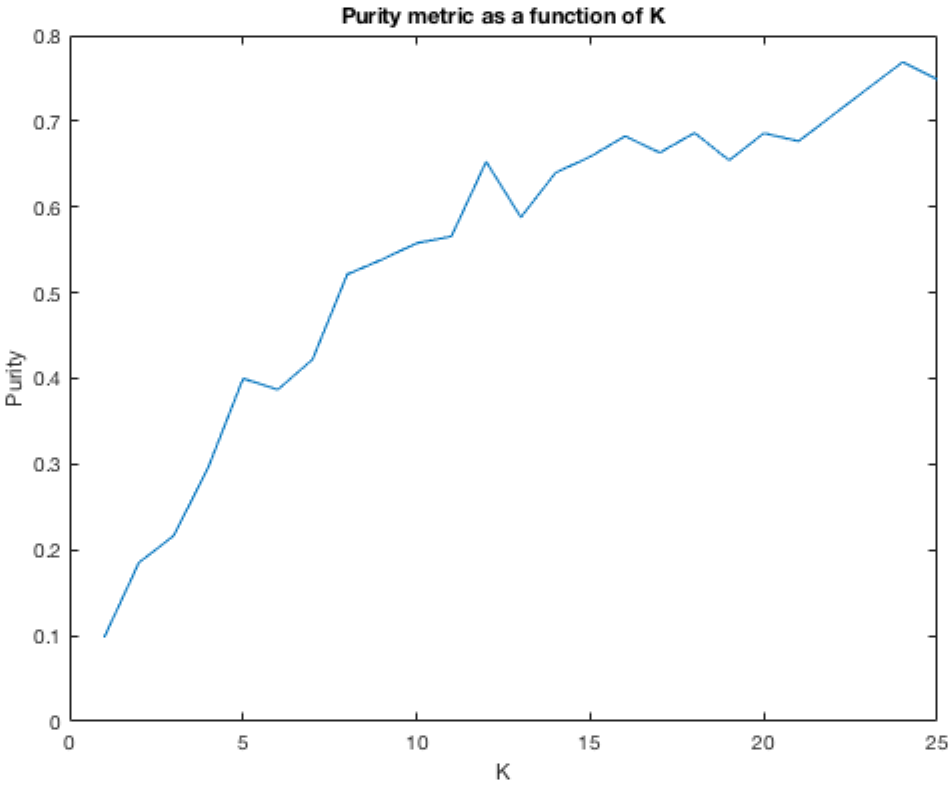
Gaussian similarity distance, fully-connected graph, std = 0.01, symmetrically normalized graph Laplacian for spectral clustering for K=1,2,...,25 calculate the "purity" metric of the obtained cluster by treating the neighborhood label as the ground truth. Plot the purity metric (y-axis) as a function of k (x-axis)

```
% Step 1: Weighted adjacency matrices

% fully-connected graph, W = S (similarity score)
W = zeros(length(X),length(X));
sigma = 0.01; % given std


for i=1:length(X)
    for j=1:length(X)
        % calculate gaussian similarity scores S(xi,xj)
        W(i,j) = exp(-((X(i,1)-X(j,1))^2 + (X(i,2)-X(j,2))^2)/ ...
            (2*sigma.^2));
    end
end

% Step 2: Degree matrix D
D = diag(sum(W,2));

% Compute the un-normalized graph Laplacian L = D - W
L = D - W;

% Compute the normalized graph Laplacian L_sym = D^{-1/2}*L*D^{-1/2}
L_sym = inv(sqrt(D))*L*inv(sqrt(D)); %#ok<MINV>

[V,G] = svd(L_sym);
```

```
K = 1:25;
Vmats = cell(length(K),1);

for i=K
  Vmats{i} = V(:,end-i+1:end);
end

% normalize V rows so that l-2 norms are 1
for i=K % implicit expansion: Matlab 2016b+
    Vmats{i} = Vmats{i} ./ sqrt(sum(Vmats{i}.^2,2));
end

% perform K-means on V mats
P = cell(25,1);
for i=K
    rng(2);
    P{i} = kmeans(Vmats{i},i);
end
```

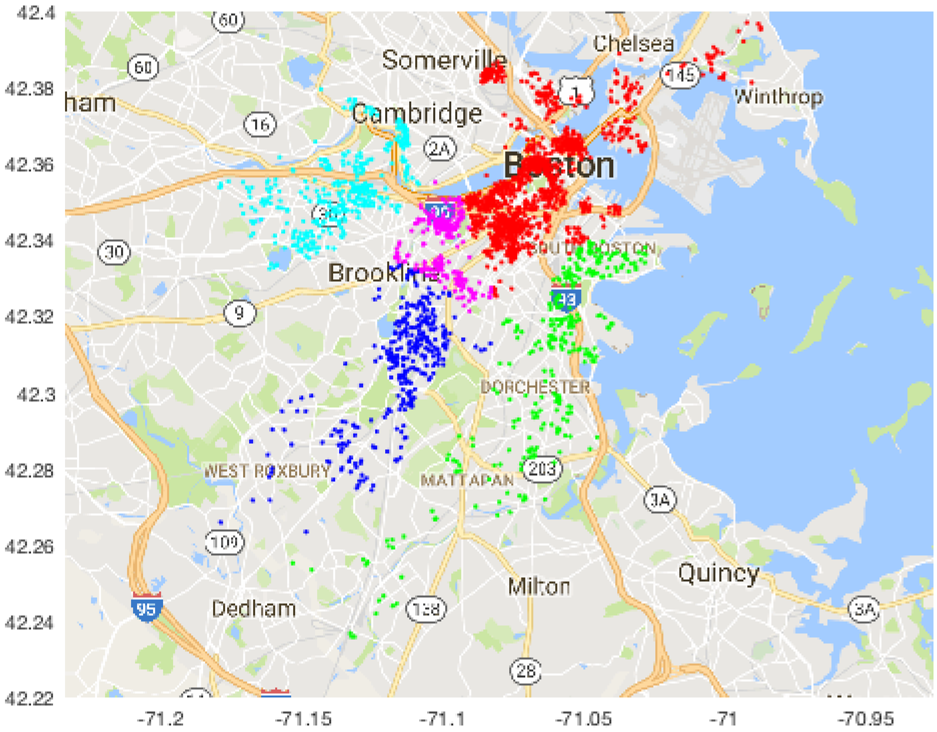
(a) Purity metric

```
purity = zeros(25,1);
for i=K
    votes = cell(i,1);
    num = 0;
    denom = 0;
    for j=1:i
        Ysub = Y(P{i}==j);
        [unique_strings,~,string_map] = unique(Ysub);
        most_common_string = unique_strings(mode(string_map));
        votes{j} = most_common_string;

        num = num + sum(strcmp(most_common_string,Ysub));
        denom = denom + length(Ysub);
    end
    purity(i) = num/denom;
end
figure(1);
plot(K,purity);
title('Purity metric as a function of K');
xlabel('K');
ylabel('Purity');
```



(b) Plot clusters for K = 5

```
figure(2); hold on;
lat = X(:,1); lon = X(:,2);
plot(lon(P{5}==1),lat(P{5}==1),'.r','MarkerSize',6)
plot(lon(P{5}==2),lat(P{5}==2),'.b','MarkerSize',6)
plot(lon(P{5}==3),lat(P{5}==3),'.g','MarkerSize',6)
plot(lon(P{5}==4),lat(P{5}==4),'.m','MarkerSize',6)
plot(lon(P{5}==5),lat(P{5}==5),'.c','MarkerSize',6)
plot_google_map
```



Published with MATLAB® R2017a