Homework #08

# 1a

X=[ 4, 9, 8, 7, 1, 1, 1, 2, 1, 5, 4, 10, 9, 8, 2, 4;

5, 8, 8, 2, 9, 10, 1, 1, 2, 5, 6, 8, 7, 2, 9, 4];

Parameters:

% neighborhood kernel:

K\_nearest=0.8;

K\_other=0.1;

% learning rate:

eta=0.1;

Initial plot:

set(0,'defaulttextfontsize',18); set(0,'defaultaxesfontsize',18);

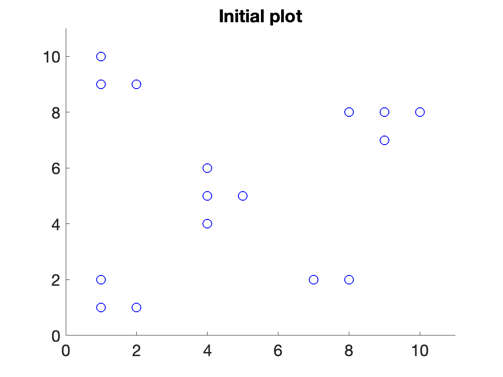
figure; clf; hold on; title('Initial plot')

xlim([0 11]); ylim([0 11]);

% plot datapoints

scatter(X(1,:),X(2,:),100,'bo');

hold off



Looks like 5ish clusters from the plot, we can try n=5

Reference pts:

N\_grid=5;

Xref=X(:, 6:6+N\_grid-1);

for j=1:1:length(Xref)

Xref(:,j)=Xref(:,j)+[1;-1];

end

% plot reference points:

figure; clf; hold on; title('Initial ref pts')

xlim([0 11]); ylim([0 11]);

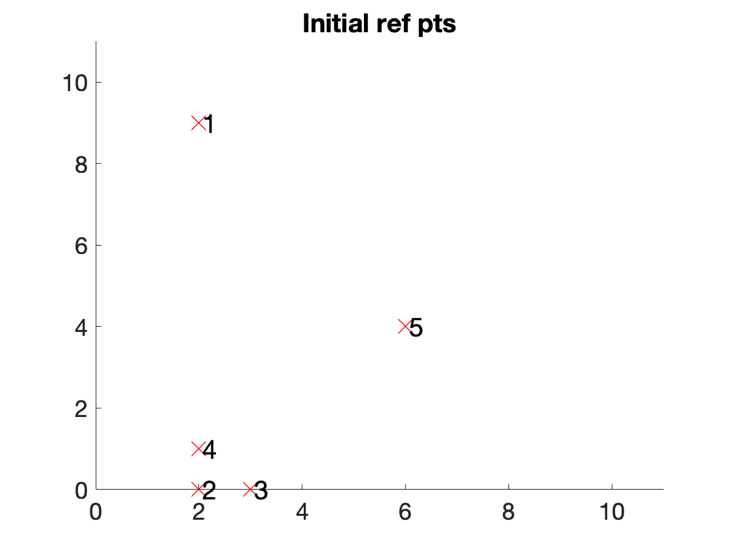
href=scatter(Xref(1,:),Xref(2,:),200,'rx');

for ii=1:N\_grid

h\_ref\_text(ii)=text(Xref(1,ii)+0.07,Xref(2,ii),num2str(ii),'FontSize',20);

end

hold off



First two iterations/datapoints in detail:

iepoch = 1 % TF said just 1st epoch only is ok

iepoch = 1

iter=0; % counter of data points being examined

for n=1:1:2

iter = iter + 1;

disp(['Iteration: ' num2str(iter)])

% initialize plot

set(0,'defaulttextfontsize',18); set(0,'defaultaxesfontsize',18);

figure; clf; hold on; title(['iteration: ' num2str(iter) ' before adjustment'])

xlim([0 11]); ylim([0 11]);

% plot datapoints

scatter(X(1,:),X(2,:),100,'bo');

% draw current data point in red:

h\_current\_data=scatter(X(1,n),X(2,n),100,'ro','MarkerFaceColor','r');

% plot reference points:

href=scatter(Xref(1,:),Xref(2,:),200,'rx');

for ii=1:N\_grid

h\_ref\_text(ii)=text(Xref(1,ii)+0.07,Xref(2,ii),num2str(ii),'FontSize',20);

end

hold off

% name current data pt in red

disp(['current datapoint in red is: ' num2str(n)])

% find which reference point is nearest to current data point:

% out of the 4 reference pts

[Dmin,I]=min([norm(X(:,n)-Xref(:,1)),norm(X(:,n)-Xref(:,2))...

,norm(X(:,n)-Xref(:,3)),norm(X(:,n)-Xref(:,4))]);

disp(['nearest ref point is: ' num2str(I)])

% adjust the nearest reference point using the learning rate:

Xref(:,I)=Xref(:,I)+eta\*K\_nearest\*(X(:,n)-Xref(:,I));

x0=Xref(:,I); dx=K\_nearest\*(X(:,n)-Xref(:,I)); dx=0.6\*dx/norm(dx);

disp(['nearest ref point adjusted by:'])

disp(eta\*K\_nearest\*(X(:,n)-Xref(:,I)))

% adjust the other reference points using the specified kernel:

nearest\_neighbors\_grid\_space=[I-1,I+1];

nearest\_neighbors\_grid\_space(nearest\_neighbors\_grid\_space==0)=N\_grid;

nearest\_neighbors\_grid\_space(nearest\_neighbors\_grid\_space==N\_grid+1)=1;

i1=0;

for nn=nearest\_neighbors\_grid\_space

i1=i1+1;

Xref(:,nn)=Xref(:,nn)+eta\*K\_other\*(X(:,n)-Xref(:,nn));

x0=Xref(:,nn); dx=K\_nearest\*(X(:,n)-Xref(:,nn)); dx=0.3\*dx/norm(dx);

disp(['neighbor ref point ' num2str(nn) ' adjusted by:'])

disp(eta\*K\_other\*(X(:,n)-Xref(:,nn)))

end

disp('output adjusted new ref pts:')

% initialize plot

set(0,'defaulttextfontsize',18); set(0,'defaultaxesfontsize',18);

figure; clf; hold on; title(['iteration: ' num2str(iter)])

xlim([0 11]); ylim([0 11]);

% plot datapoints

scatter(X(1,:),X(2,:),100,'bo');

% draw current data point in red:

h\_current\_data=scatter(X(1,n),X(2,n),100,'ro','MarkerFaceColor','r');

% plot reference points:

href=scatter(Xref(1,:),Xref(2,:),200,'rx');

for ii=1:N\_grid

h\_ref\_text(ii)=text(Xref(1,ii)+0.07,Xref(2,ii),num2str(ii),'FontSize',20);

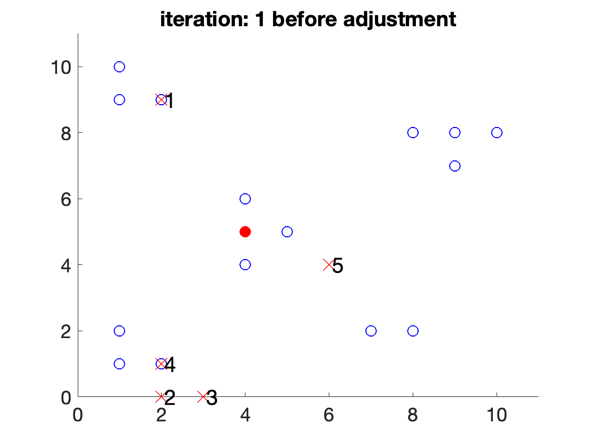
end

hold off

disp('')

end

Iteration: 1



current datapoint in red is: 1

nearest ref point is: 1

nearest ref point adjusted by:

0.1472

-0.2944

neighbor ref point 5 adjusted by:

-0.0198

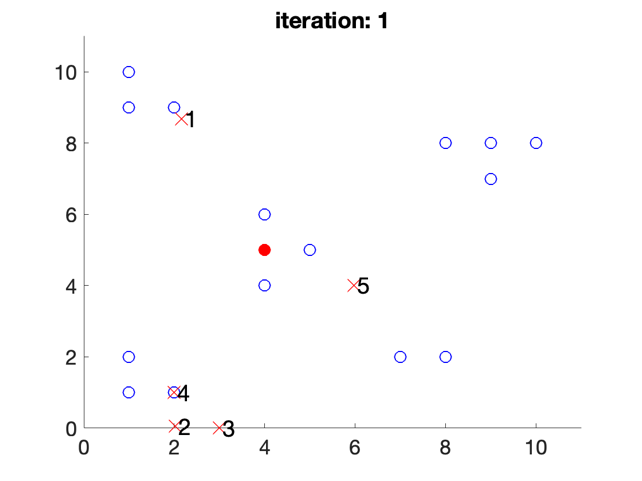
0.0099

neighbor ref point 2 adjusted by:

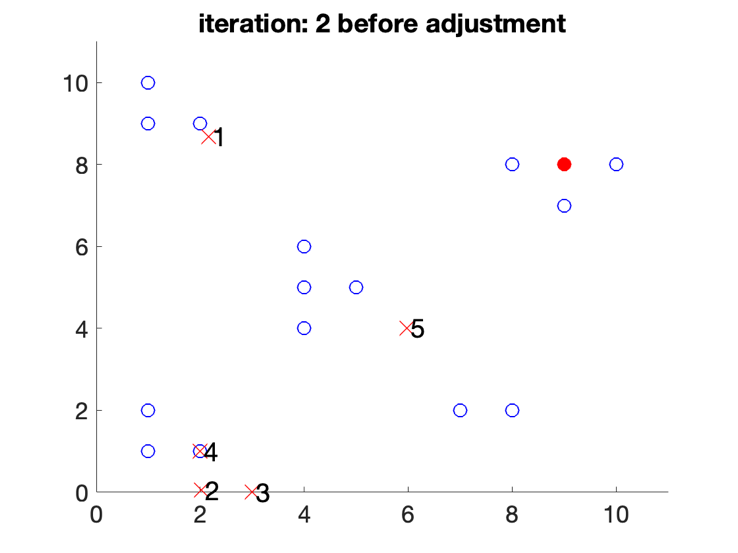
0.0198

0.0495

output adjusted new ref pts:



Iteration: 2



current datapoint in red is: 2

nearest ref point is: 1

nearest ref point adjusted by:

0.5034

-0.0500

neighbor ref point 5 adjusted by:

0.0299

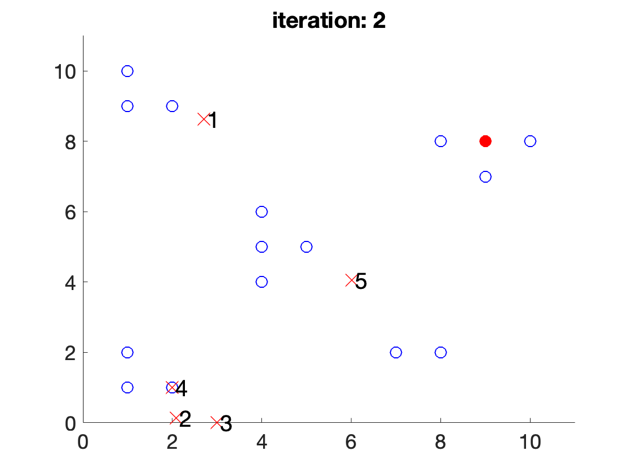
0.0395

neighbor ref point 2 adjusted by:

0.0691

0.0787

output adjusted new ref pts:



Iterating through all data pts:

for iepoch = 1:1:1 % TF said just 1st epoch only is ok

iter=0; % counter of data points being examined

for n=1:length(X(1,:))

iter = iter + 1;

% find which reference point is nearest to current data point:

% out of the 4 reference pts

[Dmin,I]=min([norm(X(:,n)-Xref(:,1)),norm(X(:,n)-Xref(:,2))...

,norm(X(:,n)-Xref(:,3)),norm(X(:,n)-Xref(:,4)),norm(X(:,n)-Xref(:,4))]);

% adjust the nearest reference point using the learning rate:

Xref(:,I)=Xref(:,I)+eta\*K\_nearest\*(X(:,n)-Xref(:,I));

x0=Xref(:,I); dx=K\_nearest\*(X(:,n)-Xref(:,I)); dx=0.6\*dx/norm(dx);

% adjust the other reference points using the specified kernel:

nearest\_neighbors\_grid\_space=[I-1,I+1];

nearest\_neighbors\_grid\_space(nearest\_neighbors\_grid\_space==0)=N\_grid;

nearest\_neighbors\_grid\_space(nearest\_neighbors\_grid\_space==N\_grid+1)=1;

i1=0;

for nn=nearest\_neighbors\_grid\_space

i1=i1+1;

Xref(:,nn)=Xref(:,nn)+eta\*K\_other\*(X(:,n)-Xref(:,nn));

x0=Xref(:,nn); dx=K\_nearest\*(X(:,n)-Xref(:,nn)); dx=0.3\*dx/norm(dx);

end

end

% initialize plot

set(0,'defaulttextfontsize',18); set(0,'defaultaxesfontsize',18);

figure; clf; hold on; title(['epoch: ' num2str(iepoch)])

xlim([0 11]); ylim([0 11]);

% plot datapoints

scatter(X(1,:),X(2,:),100,'bo');

% plot reference points:

href=scatter(Xref(1,:),Xref(2,:),200,'rx');

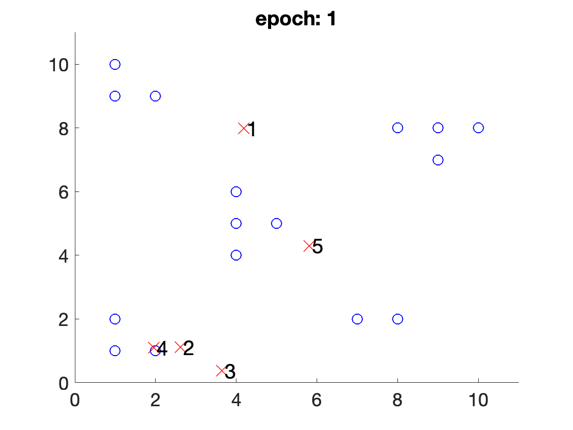
for ii=1:N\_grid

h\_ref\_text(ii)=text(Xref(1,ii)+0.07,Xref(2,ii),num2str(ii),'FontSize',20);

end

hold off

end



# 1b

% Matlab

rng(2021); N=250; X=zeros(2,4\*N);

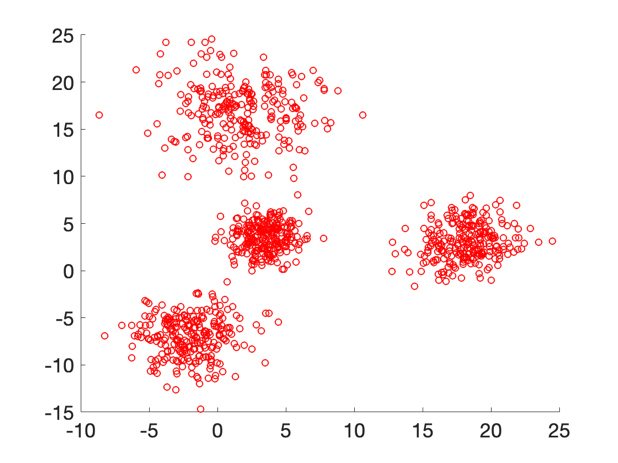
for i=1:N; X(:,i)=[5,5]' +(1.5\*(randn(2,1)-1)); end;

for i=N+1:2\*N; X(:,i)=[0,-5]'+(2.0\*(randn(2,1)-1));end;

for i=2\*N+1:3\*N; X(:,i)=[5,20]'+(3.0\*(randn(2,1)-1));end;

for i=3\*N+1:4\*N; X(:,i)=[20,5]'+(2.0\*(randn(2,1)-1));end;

p=randperm(4\*N); X=X(:,p);figure(1); clf; scatter(X(1,:),X(2,:),'r')



Adapted from "self\_organizing\_maps\_library\_function\_example.m"

nx=2; ny=3;

fprintf(1,' SOM geometry is (nx,ny)=(%d,%d).\n',nx,ny)

SOM geometry is (nx,ny)=(2,3).

% specify geometry of SOM:

net = selforgmap([nx ny]);

% train the network to find clusteroids:

net = train(net,X);

% examine network:

view(net);

% classify, Y is a set of vectors (y1,...yn), one per each data point,

% containing cluster assignments for all data points, e.g.,

% assuming four clusters, if y\_i=[0,1,0,0], then the

% corresponding i'th data point belongs to cluster 2:

Y = net(X);

% classes is a single vector contain the cluster assignments of all

% data points: if classes(i)=2, point i belongs to cluster 2:

classes = vec2ind(Y);

Plot clusters:

% plot clusters in different colors:

figure; clf; hold on

color ='rgbcmyrgbcmyrgbcmyrgbcmy';

marker='......xxxxxx++++++oooooo';

Nmaps=nx\*ny;

for i=1:1:Nmaps

plot(X(1,classes==i),X(2,classes==i),[color(i) marker(i)],'MarkerSize',12)

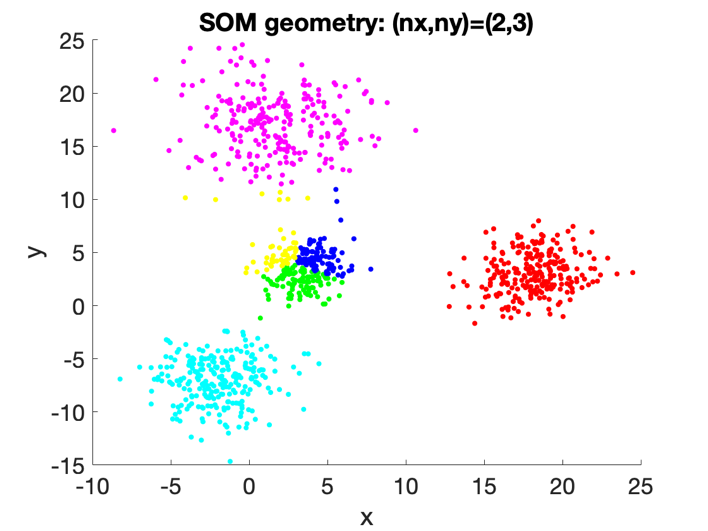
end

xlabel('x');

ylabel('y');

title(sprintf('SOM geometry: (nx,ny)=(%d,%d)',nx,ny));

hold off



# 2

X=[0.5,1.5,0.5,1,9,7.5,7,9.5,1.5,1,1,8.5,9.5,6.5,8,0.5,1,9,9.5,7,10,1.5,8.5,9.5;

9.5,8.5,0.5,1,0.5,5.5,4.5,9.5,9.5,0.5,1.5,0.5,1,5.5,4.5,9,9,9.5,9,5,9.5,0.5,1,8.5];

Initial plot:

set(0,'defaulttextfontsize',18); set(0,'defaultaxesfontsize',18);

figure; clf; hold on; title('Initial plot')

xlim([0 11]); ylim([0 11]);

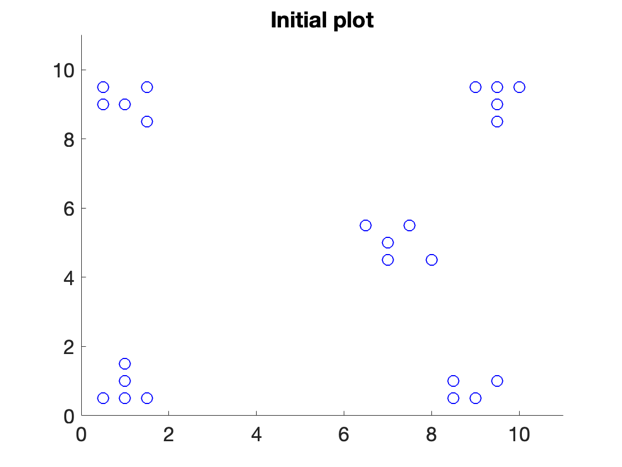
color ='rgbcmyrgbcmyrgbcmyrgbcmy';

marker='......xxxxxx++++++oooooo';

% plot datapoints

scatter(X(1,:),X(2,:),100,'bo');

hold off



Adapted from "spectral\_clustering\_example.m"

% Calculate distances between all pairs of points:

Y = pdist(X.','euclid'); % or could use, for example, Y = pdist(X,'minkowski',1);

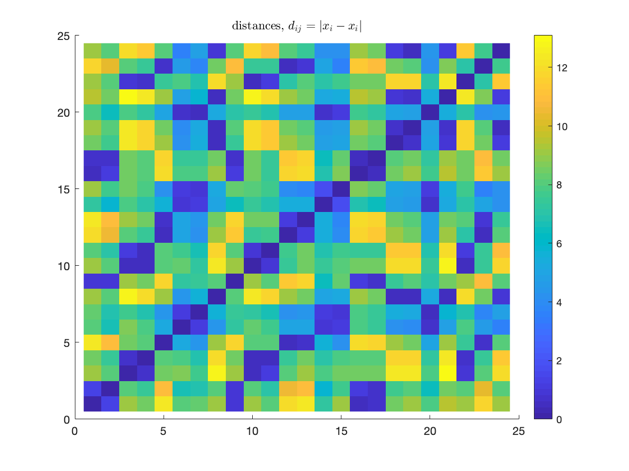
distances=squareform(Y);

figure; hold on

imagesc(distances); colorbar

title('distances, $d\_{ij}=|x\_i-x\_i|$','Interpreter','LaTeX')

hold off



% build similarity matrix from distances (Luxburg 2007):

dists\_std=std(distances(:)); % std of distances

W=exp(-distances.^2/dists\_std^2);

figure; hold on

imagesc(W); colorbar

title('similarity W: $\exp(-d\_{ij}^2/\sigma^2)$','Interpreter','LaTeX')

hold off

A picture containing qr code

Description automatically generated

% degree matrix is some over rows of adjacency matrix:

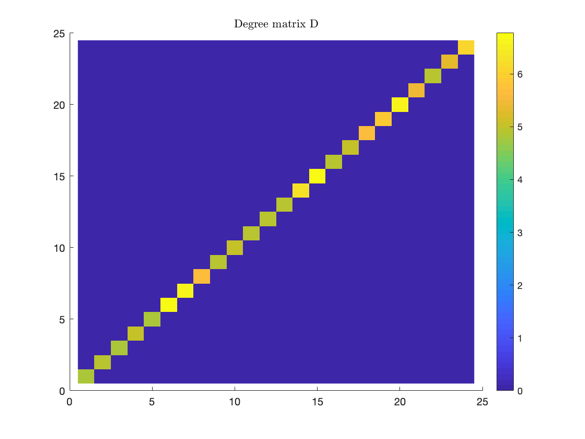
D=diag(sum(W));

figure; hold on

imagesc(D); colorbar

title('Degree matrix D','Interpreter','LaTeX')

hold off



% Laplacian matrix:

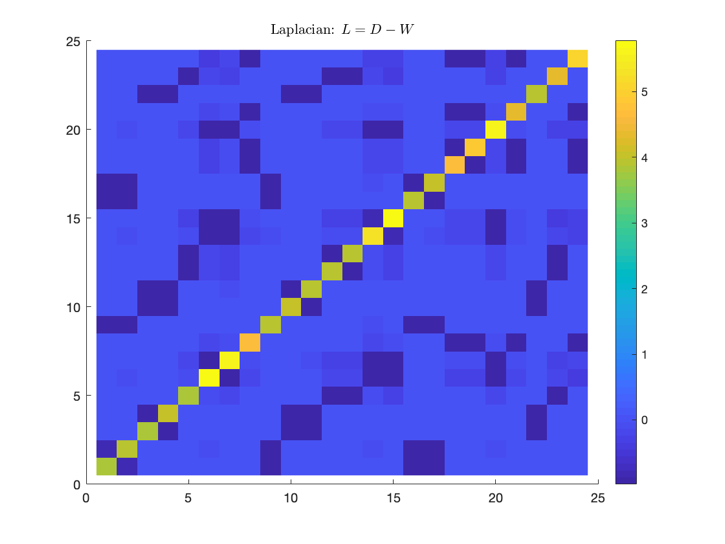
L=D-W;

figure; hold on

imagesc(L); colorbar

title('Laplacian: $L=D-W$','Interpreter','LaTeX')

hold off



% calculate eivenvectors of Laplacian matrix:

[V,D] = eig(L);

figure; hold on

d=diag(D);

plot(d,'-x');

% highlight second smallest eigenvalue:

plot(2,d(2),'ro','markersize',15);

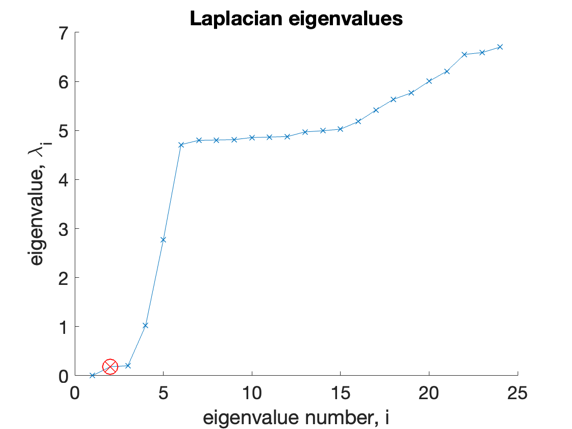
plot(2,d(2),'rx','markersize',15);

xlabel('eigenvalue number, i')

ylabel('eigenvalue, \lambda\_i')

title('Laplacian eigenvalues');

hold off



There seems to be a spectral gap around k=5, so we can try that

k=5

k = 5

figure; hold on

plot(0\*V(:,1),'--k','linewidth',0.5);

for ii=1:k

hl(ii)=plot(V(:,ii),[color(ii) 'x-']);

if ii>=2; set(hl,'linewidth',3,'markersize',15); end

legend\_text(ii)=sprintf('%d',ii);

end

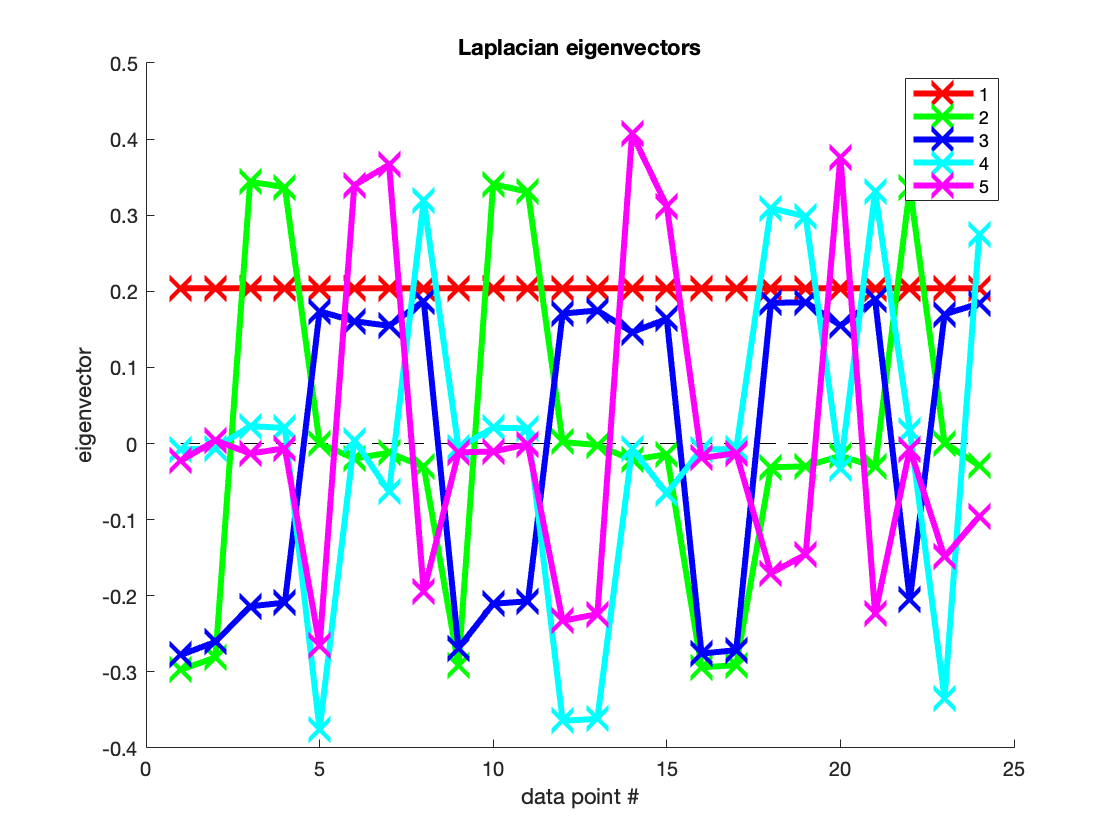
title(sprintf('Laplacian eigenvectors',k));

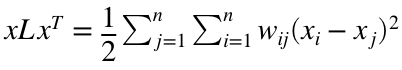
xlabel('data point #')

ylabel('eigenvector')

legend([hl(:)],legend\_text(:));

hold off



It turns out that  or the sum of corresponding W elements times the distances of points. So solving to minimize this expression of distances between points, we eventually get that  when it's minimized. So the minimum is with x eigenvectors of minimal lamdas. So to make k clusters, we can use the kth minimal lambda's eigenvector, ignoring the first one with eigenvector of just ones.

k-means Clustering:

opts = statset('Display','final');

[idx,C] = kmeans(V(:,2:k),k,'Distance','sqeuclidean' ...

,'Replicates',5,'Options',opts);

Replicate 1, 1 iterations, total sum of distances = 0.0302072.

Replicate 2, 2 iterations, total sum of distances = 0.0302072.

Replicate 3, 1 iterations, total sum of distances = 0.0302072.

Replicate 4, 1 iterations, total sum of distances = 0.0302072.

Replicate 5, 1 iterations, total sum of distances = 0.0302072.

Best total sum of distances = 0.0302072

Plot clusters in dataspace:

figure;hold on

for i=1:1:k

plot(X(1,idx==i),X(2,idx==i),[color(i) marker(i)],'MarkerSize',12)

end

% write legend:

for i=1:1:k

if i<10

legends(i,:)=['Cluster ' num2str(i)];

else

legends(i,:)=['Cluster' num2str(i)];

end

end

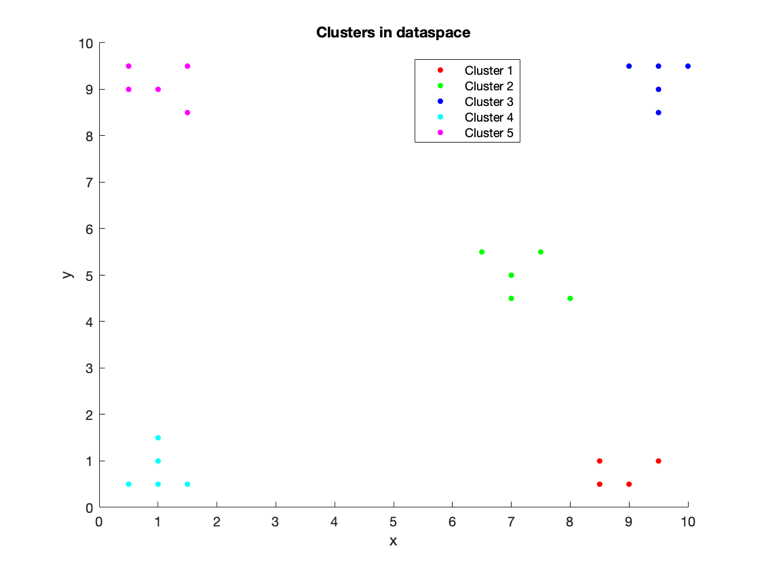
legend(legends,'Location','best');

title(sprintf('Clusters in dataspace'));

xlabel('x');

ylabel('y');

hold off



# 3a

clear all

load 'HW08\_CURE.mat'

Initial plot:

set(0,'defaulttextfontsize',18); set(0,'defaultaxesfontsize',18);

figure; clf; hold on; title('Initial plot')

xlim([0 11]); ylim([0 11]);

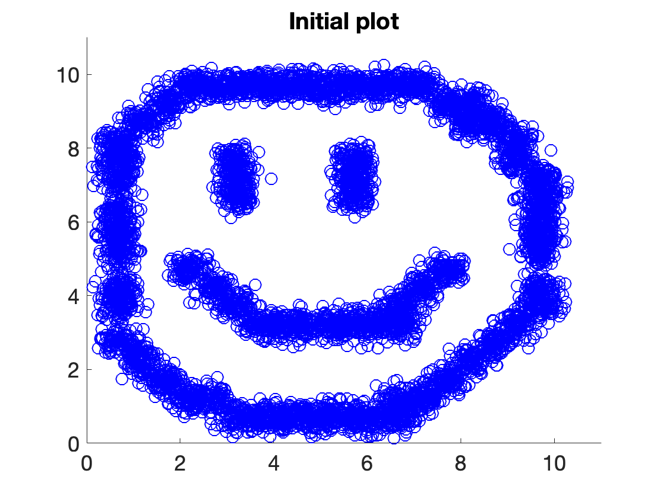
color ='rgbcmyrgbcmyrgbcmyrgbcmy';

marker='......xxxxxx++++++oooooo';

% plot datapoints

scatter(X(1,:),X(2,:),100,'bo');

hold off



Cute :)

Single:

Y=pdist(X');

distances=squareform(Y)

distances = 7200×7200

0 0.2556 0.2975 0.5334 0.4553 0.2999 0.5421 ⋯

0.2556 0 0.2673 0.5753 0.5156 0.2619 0.6627

0.2975 0.2673 0 0.3081 0.2528 0.0088 0.4161

0.5334 0.5753 0.3081 0 0.0807 0.3141 0.1938

0.4553 0.5156 0.2528 0.0807 0 0.2600 0.1811

0.2999 0.2619 0.0088 0.3141 0.2600 0 0.4243

0.5421 0.6627 0.4161 0.1938 0.1811 0.4243 0

0.5863 0.6366 0.3697 0.0617 0.1313 0.3757 0.1768

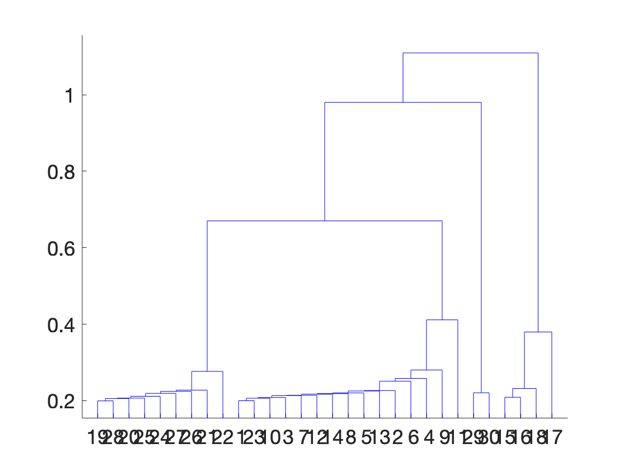
0.2061 0.3657 0.4973 0.7371 0.6579 0.4984 0.7266

0.7689 0.6407 0.4740 0.4494 0.4900 0.4704 0.6427

⋮

Z=linkage(Y,'single');

dendrogram(Z);



Seems like a pretty big jump @4 clusters

k=4

k = 4

idx=cluster(Z,'maxclust',k)';

figure;hold on

for i=1:1:k

plot(X(1,idx==i),X(2,idx==i),[color(i) marker(i)],'MarkerSize',12)

end

% write legend:

for i=1:1:k

if i<10

legends(i,:)=['Cluster ' num2str(i)];

else

legends(i,:)=['Cluster' num2str(i)];

end

end

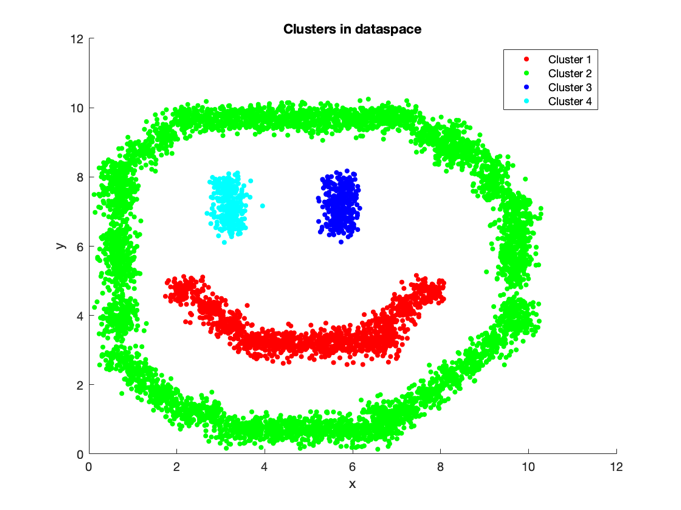
legend(legends,'Location','best');

title(sprintf('Clusters in dataspace'));

xlabel('x');

ylabel('y');

hold off

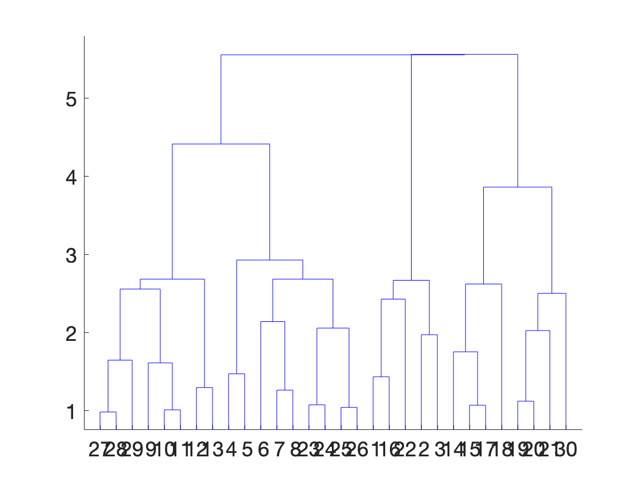


Centroid:

Z=linkage(Y,'centroid');

Warning: Non-monotonic cluster tree -- the centroid linkage is probably not appropriate.

dendrogram(Z);



Jump could be @k=4 still kind of

k=4

k = 4

idx=cluster(Z,'maxclust',k)';

figure;hold on

for i=1:1:k

plot(X(1,idx==i),X(2,idx==i),[color(i) marker(i)],'MarkerSize',12)

end

% write legend:

for i=1:1:k

if i<10

legends(i,:)=['Cluster ' num2str(i)];

else

legends(i,:)=['Cluster' num2str(i)];

end

end

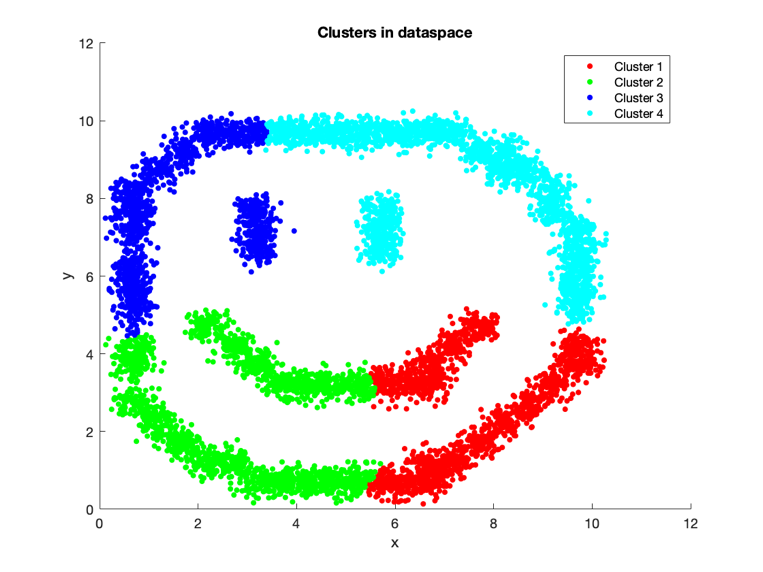
legend(legends,'Location','best');

title(sprintf('Clusters in dataspace'));

xlabel('x');

ylabel('y');

hold off



So centroid isn't going to be great here since clusters within clusters will have similar centroids that are not really differentiable.

But single is better for clusters within clusters as long as they don't have any points or parts touching/too close to each other. Single linkage is by the lowest distance between two points from the two clusters, which is significantly different between these smiley face clusters, and is not affected by centroids of clusters within clusters.

# 3b

clear all

load 'HW08\_CURE.mat'

Random initial representative points:

[M,N]=size(X);

n\_reps=300;

rand\_cols = randperm(N,n\_reps);

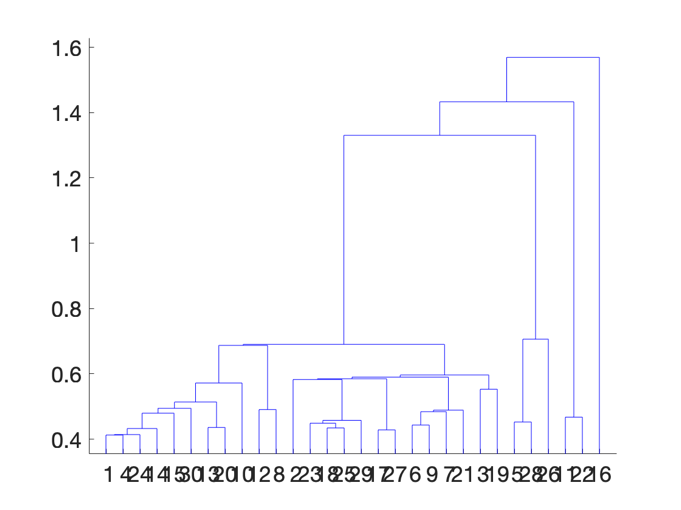
rand\_reps = X(:,rand\_cols);

Hierarchical clustering of rand\_reps:

Y=pdist(rand\_reps','euclid');

Z=linkage(Y,'single');

dendrogram(Z);



Jump around k=4

k=4;

idx\_rand\_reps = cluster(Z,'maxclust',k);

Everyone go find your rand\_rep point that's closest:

all\_dists=zeros([1 n\_reps]);

for i=1:N

for j=1:n\_reps

all\_dists(1,j)=norm(X(:,i)-rand\_reps(:,j));

end

[M,nearest\_rand\_rep] = min(all\_dists);

idx(i)=idx\_rand\_reps(nearest\_rand\_rep);

end

Plot:

figure;hold on

color ='rgbcmyrgbcmyrgbcmyrgbcmy';

marker='......xxxxxx++++++oooooo';

for i=1:1:k

plot(X(1,idx==i),X(2,idx==i),[color(i) marker(i)],'MarkerSize',12)

end

% write legend:

for i=1:1:k

if i<10

legends(i,:)=['Cluster ' num2str(i)];

else

legends(i,:)=['Cluster' num2str(i)];

end

end

legend(legends,'Location','best');

title(sprintf('Clusters in dataspace'));

xlabel('x');

ylabel('y');

Plot representatives:

for i=1:1:k

plot(rand\_reps(1,idx\_rand\_reps==i),rand\_reps(2,idx\_rand\_reps==i),'rx','MarkerSize',12)

end

hold off

