Math 285 Homework 3

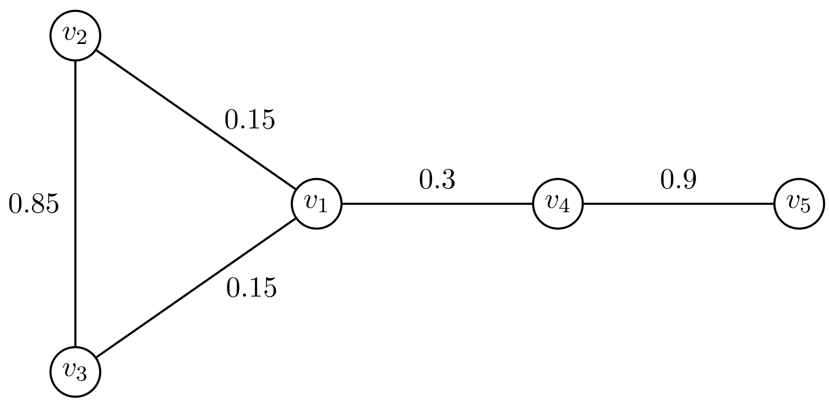
Ryan Shiroma

1. Graphs

Let $\mathbf{W}$ be a weight matrix for a graph.

$$\mathbf{W} = \begin{pmatrix} 0 & 0.15 & 0.15 & 0.3 & 0 \\ 0.15 & 0 & 0.85 & 0 & 0 \\ 0.15 & 0.85 & 0 & 0 & 0 \\ 0.3 & 0 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0.9 & 0 \end{pmatrix}$$

We can sketch out the graph of the weight matrix $\mathbf{W}$ using $w_{i,j}$ as the weight of the edge between nodes v_i and v_j.



Node v_1 has degree 0.6 (has edges to nodes v_2, v_3 and v_4)
Node v_2 has degree 1 (has edges to nodes v_1 and v_3)
Node v_3 has degree 1 (has edges to nodes v_1 and v_2)
Node v_4 has degree 1.2 (has edges to nodes v_1 and v_5)
Node v_5 has degree 0.9 (has an edge to node v_4)

Let's now make two separate cuts given given these partitions:
Partition 1: $V_1 = A_1 \cup B_1 = \{v_1, v_2, v_3\} \cup \{v_4, v_5\}$.

$$\begin{aligned}
\text{NCut}(A_1, B_1) &= \text{cut}(A_1, B_1)\left(\frac{1}{vol(A_1)} + \frac{1}{vol(B_1)}\right) \\
&= \left(\sum_{\substack{i \in A_1 \\ j \in B_1}} w_{i,j}\right)\left(\frac{1}{\sum\limits_{i \in A_1} d_i} + \frac{1}{\sum\limits_{i \in B_1} d_i}\right) \\
&= \left(0.3\right)\left(\frac{1}{0.6 + 1 + 1} + \frac{1}{1.2 + 0.9}\right) \\
&= \mathbf{0.25824} \\
\text{RatioCut}(A_1, B_1) &= \text{cut}(A_1, B_1)\left(\frac{1}{|A_1|} + \frac{1}{|B_1|}\right) \\
&= \left(0.3\right)\left(\frac{1}{3} + \frac{1}{2}\right) \\
&= \mathbf{0.25}
\end{aligned}$$

Partition 2: $A_2 \cup B_2 = \{v_1, v_4, v_5\} \cup \{v_2, v_3\}$.

$$\begin{aligned}
\text{NCut}(A_2, B_2) &= \text{cut}(A_2, B_2)\left(\frac{1}{vol(A_2)} + \frac{1}{vol(B_2)}\right) \\
&= \left(\sum_{\substack{i \in A_2 \\ j \in B_2}} w_{i,j}\right)\left(\frac{1}{\sum\limits_{i \in A_2} d_i} + \frac{1}{\sum\limits_{i \in B_2} d_i}\right) \\
&= \left(0.3\right)\left(\frac{1}{0.6 + 1.2 + 0.9} + \frac{1}{1 + 1}\right) \\
&= \mathbf{0.26111} \\
\text{RatioCut}(A_2, B_2) &= \text{cut}(A_2, B_2)\left(\frac{1}{|A_2|} + \frac{1}{|B_2|}\right) \\
&= \left(0.3\right)\left(\frac{1}{3} + \frac{1}{2}\right) \\
&= \mathbf{0.25}
\end{aligned}$$

The NCut for the first partition is smaller while the RatioCuts are the same for both partitions.

2. NCut Cluster Indicator Vector Definition

Let's show that $\text{NCut}(A, B) = \frac{\mathbf{x}^\intercal \mathbf{L}\mathbf{x}}{\mathbf{x}^\intercal \mathbf{D}\mathbf{x}}$ still holds if we change the definition of $\mathbf{x}$ to:

$$x_i = \begin{cases} \frac{1}{Vol(A)}, & i \in A \\ -\frac{1}{Vol(B)}, & i \in B \end{cases} \tag{1}$$

We'll first split this up by numerator and denominator. Lets start with the numerator:

$$\begin{aligned}
\mathbf{x}^\intercal \mathbf{L}\mathbf{x} &= \mathbf{x}^\intercal (\mathbf{D} - \mathbf{W})\mathbf{x} \\
&= \mathbf{x}^\intercal \mathbf{D}\mathbf{x} - \mathbf{x}^\intercal \mathbf{W}\mathbf{x} \\
&= \sum_i d_i x_i^2 - \sum_{i,j} w_{i,j} x_i x_j \\
&= \frac{1}{2}\left[2\sum_i d_i x_i^2 - 2\sum_{i,j} w_{i,j} x_i x_j\right] \\
&= \frac{1}{2}\left[\sum_i d_i x_i^2 + \sum_j d_j x_j^2 - 2\sum_{i,j} w_{i,j} x_i x_j\right] \\
&= \frac{1}{2}\left[\sum_i \left(\sum_j w_{i,j}\right) x_i^2 + \sum_j \left(\sum_i w_{i,j}\right) x_j^2 - 2\sum_{i,j} w_{i,j} x_i x_j\right] \\
&= \frac{1}{2}\left[\sum_{i,j} w_{i,j} x_i^2 + \sum_{i,j} w_{i,j} x_j^2 - 2\sum_{i,j} w_{i,j} x_i x_j\right] \\
&= \frac{1}{2}\left[\sum_{i,j} w_{i,j} \left(x_i^2 + x_j^2 - 2x_i x_j\right)\right] \\
&= \frac{1}{2}\left[\sum_{i,j} w_{i,j} \left(x_i - x_j\right)^2\right] \\
&= \frac{1}{2}\Bigg[\sum_{\substack{i\in A\\ j\in B}} w_{i,j}\left(\frac{1}{Vol(A)} - \frac{-1}{Vol(B)}\right)^2 + \sum_{\substack{i\in B\\ j\in A}} w_{i,j}\left(\frac{-1}{Vol(B)} - \frac{1}{Vol(A)}\right)^2 \\
&\qquad + \cancelto{0}{\sum_{i,j\in A} w_{i,j}\left(\frac{1}{Vol(A)} - \frac{1}{Vol(A)}\right)^2} + \cancelto{0}{\sum_{i,j\in A} w_{i,j}\left(\frac{-1}{Vol(B)} - \frac{-1}{Vol(B)}\right)^2}\Bigg] \\
&= \frac{1}{2}\left[\sum_{\substack{i\in A\\ j\in B}} w_{i,j}\left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right)^2 + \sum_{\substack{i\in B\\ j\in A}} w_{i,j}\left(\cancel{-}\left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right)\right)^2\right] \\
&= \frac{1}{2}\left[2\sum_{\substack{i\in A\\ j\in B}} w_{i,j}\left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right)^2\right] \\
&= \left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right)^2 \sum_{\substack{i\in A\\ j\in B}} w_{i,j} \\
&= \left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)}\right)^2 Cut(A, B)
\end{aligned}$$

Now let's take a look at the denominator:

$$
\begin{aligned}
\mathbf{x}^\intercal \mathbf{D}\mathbf{x} &= \sum_i d_i x_i^2 \\
&= \sum_{i \in A} d_i \left(\frac{1}{Vol(A)} \right)^2 + \sum_{i \in B} d_i \left(\frac{\cancel{-}1}{Vol(B)} \right)^2 \\
&= \left(\frac{1}{Vol(A)} \right)^2 \sum_{i \in A} d_i + \left(\frac{1}{Vol(B)} \right)^2 \sum_{i \in B} d_i \\
&= \left(\frac{1}{Vol(A)} \right)^2 Vol(A) + \left(\frac{1}{Vol(B)} \right)^2 Vol(B) \\
&= \frac{1}{Vol(A)} + \frac{1}{Vol(B)}
\end{aligned}
$$

Now that we've solved for both the numerator and the denominator we can plug them into the equation.

$$
\begin{aligned}
\frac{\mathbf{x}^\intercal \mathbf{L}\mathbf{x}}{\mathbf{x}^\intercal \mathbf{D}\mathbf{x}} &= \frac{\left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)} \right)^2 Cut(A, B)}{\frac{1}{Vol(A)} + \frac{1}{Vol(B)}} \\
&= \left(\frac{1}{Vol(A)} + \frac{1}{Vol(B)} \right) Cut(A, B) \\
&= \mathrm{NCut}(A, B) \qquad \square
\end{aligned}
$$

Therefore the NCut equation still holds when the cluster indicator vector, $\mathbf{x}$, is changed to equation (1). Of course, solving the system for $\mathbf{x}$ to minimize $\mathrm{NCut}(A, B)$ is very computationally expensive, so we relax the problem and let $\mathbf{x}$ can take on real values for the rest of the assignment.

3. NCut Matlab Implementation

The first thing we must do is find a suitable value for σ to compute our weighted similarity matrix $\mathbf{W}$. We'll first create a distance(L_2) matrix and find the average distance to the k^{th} nearest neighbor. Here I have connected the k nearest neighbors between all points in the data set. It seems that any value for k between 6 and 12 seem like a good fit. For this exercise we'll stick with the suggested value of k=7.

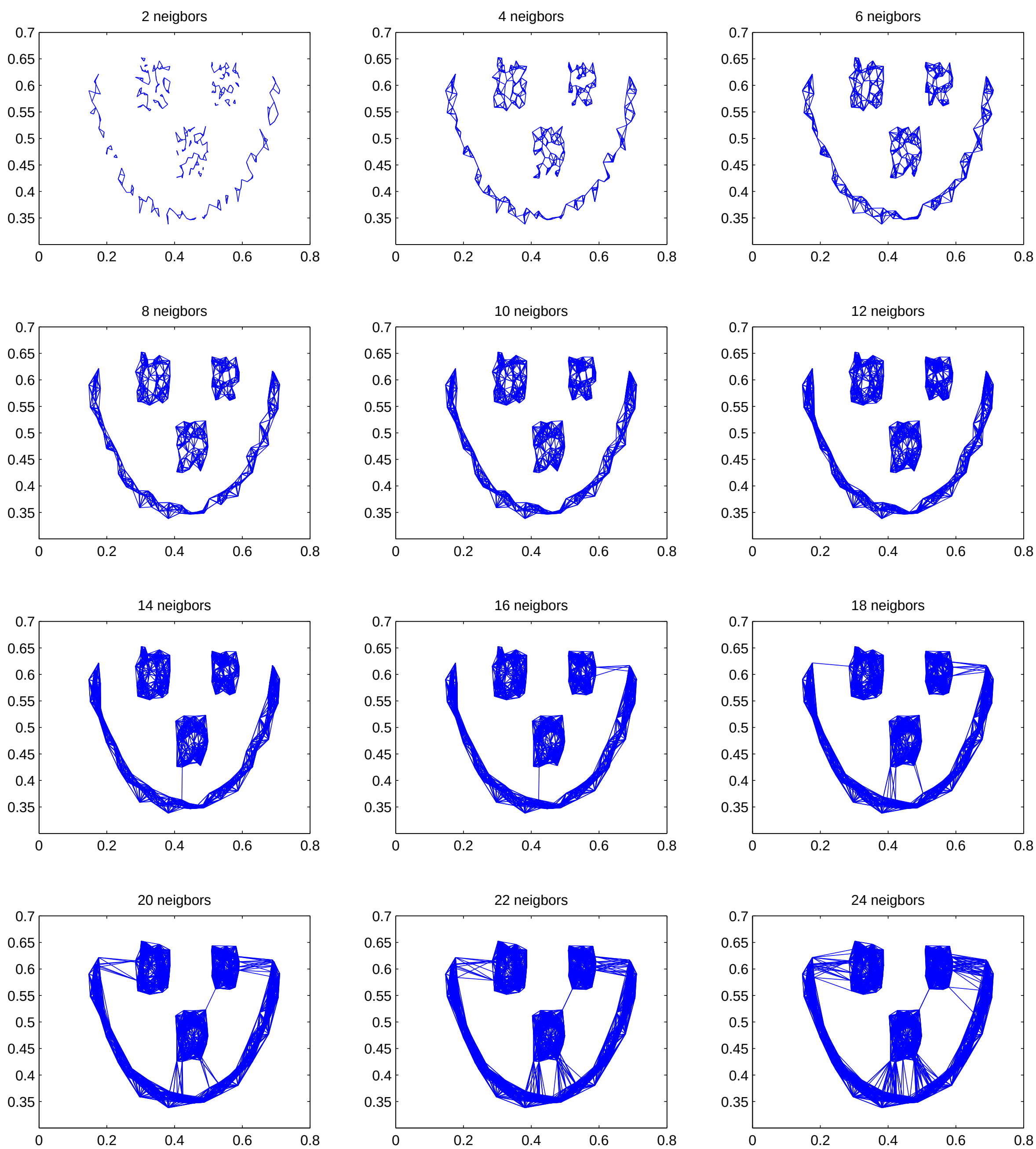


Figure 1: The k nearest neigbors have been connected to each point in the dataset

We'll then compute the weighted similarity matrix and its associated degree matrix, $\mathbf{W}$ and $\mathbf{D}$ respectively. Next we'll compute the eigenvalues and eigenvectors for the random walk Laplacian matrix $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$. We then run k-means on the eigenvectors corresponding the the second, third, and fourth smallest eigenvalues as points. The points are now tighly clustered and can be appropriately labeled using k-means.

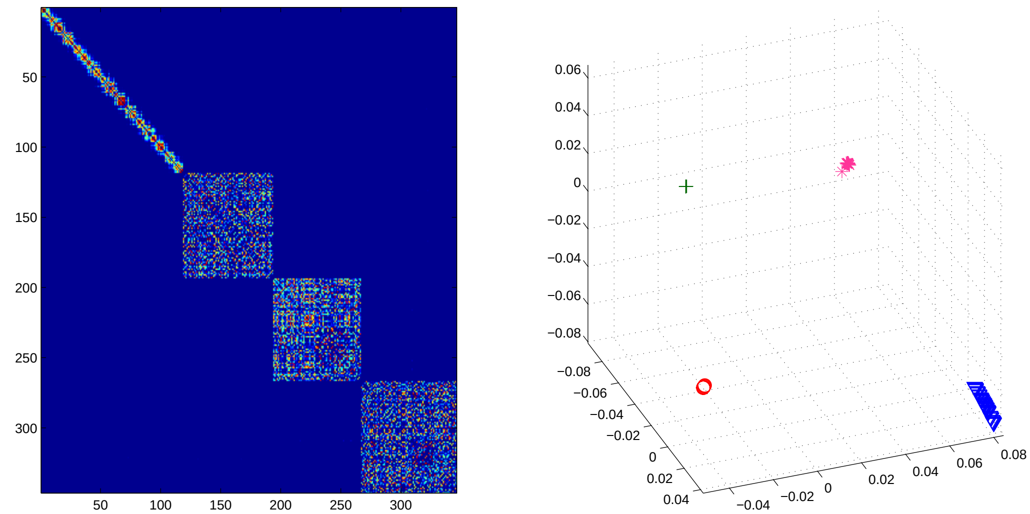


Figure 2: (Left)The W matrix, (Right) the plotted second through fourth eigenvectors

The labels are now applied to the original data points and zero percent error is achieved.

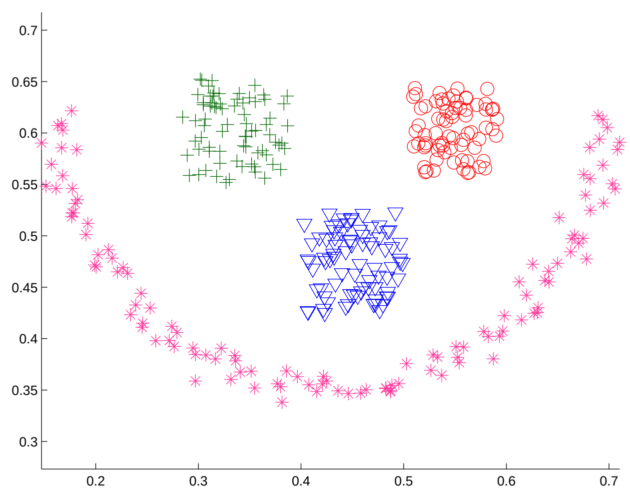


Figure 3: All of the points in the four clusters in the happy face dataset are correctly labeled

NCut Script

```
load fakeface.mat

[V,dist,W]=nCut(X);
n=size(X,1);
[sorteddist,idx]=sort(dist,2);
i=1;
for knn=2:2:24 %finding an ideal k nearest neighbors for sigma tuning
    y=reshape(idx(:,2:knn+1)',knn*n,1);
    x=reshape(reshape(repmat(1:n,1,knn),n,knn)',n*knn,1);
    subplot(4,3,i);
    plot([X(x,1)'; X(y,1)'],[X(x,2)' ;X(y,2)'],'b-');title([int2str(knn) ' neigbors']);
    i=i+1;
end

labels_kmeans = kmeans(V(:,2:4), 4, 'Replicates', 10); %run kmeans on the NCut eigenvectors
subplot(1,2,1);imagesc(W);
subplot(1,2,2);gcplot(V(:,2:4),labels_kmeans);grid on; %display the clusters by plotting v_2 x v_3 x v_3
figure; gcplot(X, labels_kmeans); axis equal; %display the cluster labels on the original data points
```

NCut Function Code

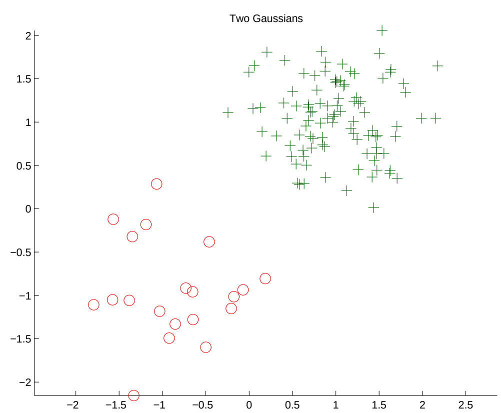
```
function [V,dist,W,D,sigma] = nCut(X,tuning_method,tuning_param)
% NCUT  runs the ncut algorithm on X
%   V = NCUT(X) median of the 7th nearest neighbor as sigma.
%   V = NCUT(X,'median',k) median of the kth nearest neighbors as sigma.
%   V = NCUT(X,'average',k) average of the 7th nearest neighbors as sigma.
%   V = NCUT(X,'custom',val) val as sigma.
%   V = NCUT(X,'self-tuning',k) kth nearest neighbor of each node as sigma.

knn=7; %default nearest neighbor setting
n=size(X,1);
if nargin ==1
    tuning_method = 'median';
elseif strcmp(tuning_method,'custom')
    sigma=ones(n,1)*tuning_param;
end

if ~strcmp(tuning_method,'custom')
%Lets first initialize sigma
    dist=zeros(n,n); % initialize the distance matrix
    for i=1:n
        for j=i+1:n
            dist(i,j)=sqrt(sum((X(i,:)-X(j,:)).^2));
        end
    end
    dist=dist + dist';
    sorted_dist=sort(dist,2);

    if strcmp(tuning_method,'average')
        sigma=ones(n,1)*mean(sorted_dist(:,knn+1));

    elseif strcmp(tuning_method,'median')
        sigma=ones(n,1)*median(sorted_dist(:,knn+1));

    elseif strcmp(tuning_method,'self-tuning')
        sigma=sorted_dist(:,knn+1);
    end
end

%now lets creat the W matrix
W=zeros(n,n);
for i=1:n
    % Since K is symmetric, we only need to compute an upper triangular
    % matrix just add the transpose to itself.
    for j=i:n
        W(i,j)= exp( -sum((X(i,:)-X(j,:)).^2)/(2*sigma(i)*sigma(j)));
    end
end
W=W + W'-2*diag(diag(W));% compute W from the upper triangular W matrix.

D=diag(sum(W)); %creates the D matrix
L_rw=eye(n)-inv(D)*W; %creates the L random walk matrix
[V,eigenvalues]=eig(L_rw); %finds the eigenvectors/values

[~,index]=sort(diag(eigenvalues)); %sorts the eigenvectors by eigenvalue.
V=V(:,index);
return;
```

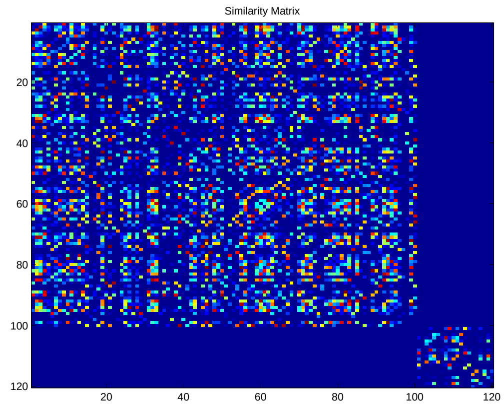
4. Ratio Cut

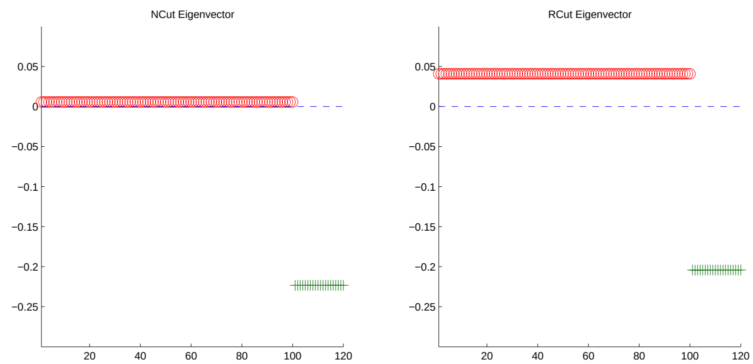
We will now modify the NCut function code to minimize the RatioCut instead of the NCut.

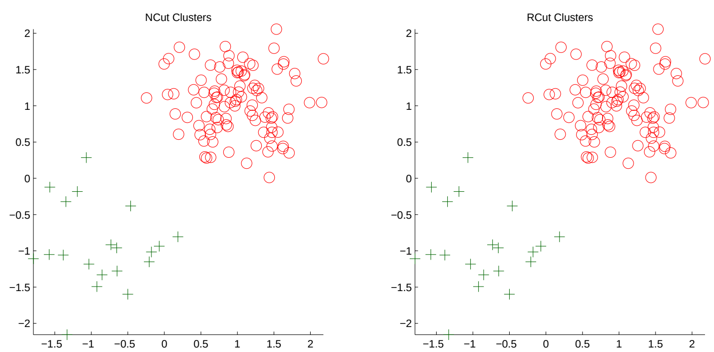
RatioCut Script

```
load twogaussians_1L1S.mat

%true labels
gcplot(X,labels);axis equal;title('Two Gaussians');

%Normalized Cut
V=nCut(X);
[labels_kmeans,~,scatter_NCut] = kmeans(V(:,2), 2, 'Replicates', 10);

%Ratio Cut
V2=RCut(X);
[labels2_kmeans,~,scatter_RCut] = kmeans(V2(:,2), 2, 'Replicates', 10);

%plot the respective eigenvectors
subplot(1,2,1);
gcplot(V(:,2),labels_kmeans);ylim([-.3,.1]);
title('NCut Eigenvector');hold on; plot([0,120],[0,0],'--');

subplot(1,2,2);
gcplot(V2(:,2),labels2_kmeans);ylim([-.3,.1]);
title('RCut Eigenvector');hold on; plot([0,120],[0,0],'--');

%plot the Normalized Cut and Ratio Cut clusters
subplot(1,2,1);
gcplot(X, labels_kmeans); axis equal;title('NCut Clusters');
subplot(1,2,2);
gcplot(X, labels2_kmeans); axis equal;title('RCut Clusters');
```









We cannot say which method performed better since both methods were 100% successful in labeling the clusters. Even though the same sigma is used both times, we cannot compare k-means scatter between them because we are using two different cut methods. It seems that even though one cluster has four times the number of nodes as the other cluster, the value of Cut(A,B) is small enough that it overcomes the fact that $\left(\frac{1}{|A|}+\frac{1}{|B|}\right)$ is minimized when $|A|=|B|=50$. This is because our choice of σ was succesful in forcing the between-cluster weights to be virtually 0's.

RatioCut Function Code

```
function [V,dist,W,D,sigma] = RCut(X,tuning_method,tuning_param)
% RCUT  runs the ratiocut algorithm on X
%   V = RCUT(X) median of the 7th nearest neighbor as sigma.
%   V = RCUT(X,'median',k) median of the kth nearest neighbors as sigma.
%   V = RCUT(X,'average',k) average of the 7th nearest neighbors as sigma.
%   V = RCUT(X,'custom',val) val as sigma.
%   V = RCUT(X,'self-tuning',k) kth nearest neighbor of each node as sigma.

knn=7; %default nearest neighbor setting
n=size(X,1);
if nargin ==1
    tuning_method = 'median';
elseif strcmp(tuning_method,'custom')
    sigma=ones(n,1)*tuning_param;
end

if ~strcmp(tuning_method,'custom')
%Lets first initialize sigma
    dist=zeros(n,n); % initialize the distance matrix
    for i=1:n
        for j=i+1:n
            dist(i,j)=sqrt(sum((X(i,:)-X(j,:)).^2));
        end
    end
    dist=dist + dist';
    sorted_dist=sort(dist,2);

    if strcmp(tuning_method,'average')
        sigma=ones(n,1)*mean(sorted_dist(:,knn+1));

    elseif strcmp(tuning_method,'median')
        sigma=ones(n,1)*median(sorted_dist(:,knn+1));

    elseif strcmp(tuning_method,'self-tuning')
        sigma=sorted_dist(:,knn+1);
    end
end

%now lets creat the W matrix
W=zeros(n,n);
for i=1:n
    % Since K is symmetric, we only need to compute an upper triangular
    % matrix just add the transpose to itself.
    for j=i:n
        W(i,j)= exp( -sum((X(i,:)-X(j,:)).^2)/(2*sigma(i)*sigma(j)));
    end
end
W=W + W'-2*diag(diag(W));% compute W from the upper triangular W matrix.

D=diag(sum(W)); %creates the D matrix
L=D-W; %creates the L matrix
[V,eigenvalues]=eig(L); %finds the eigenvectors/values

[~,index]=sort(diag(eigenvalues)); %sorts the eigenvectors by eigenvalue.
V=V(:,index);
return;
```

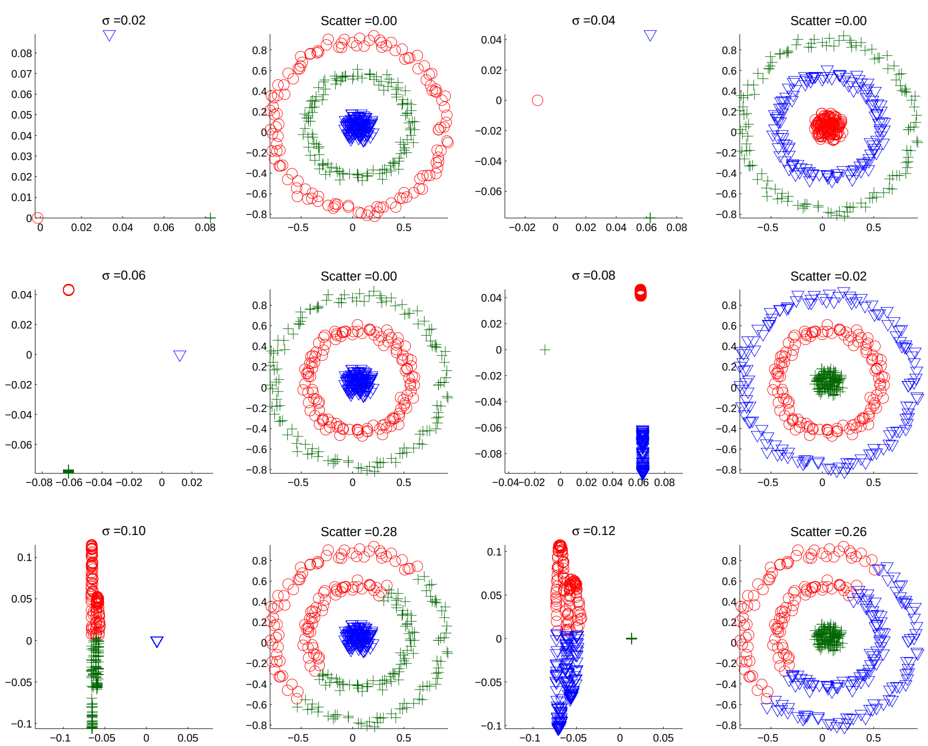
5. Choosing a Suitable σ

Let's now look at how adjusting σ affects the accuracy of the clustering.

Sigma Choice Code

```
load threecircles.mat

V=nCut(X);

sigma=0.02:0.02:0.12;
scatter=zeros(length(sigma),1);
for i=1:length(sigma)
    V=nCut(X,'custom',sigma(i));
    [labels_kmeans,~,sumd] = kmeans(V(:,2:3), 3, 'Replicates', 10);
    scatter(i)=sum(sumd);
    subplot(length(sigma)/2,4,2*i-1)
    gcplot(V(:,2:3), labels_kmeans); axis equal;title(strcat('\sigma = ',sprintf('%0.2f
        ',sigma(i))),'FontSize',12);
    subplot(length(sigma)/2,4,2*i)
    gcplot(X, labels_kmeans); axis equal;title(strcat('Scatter = ',sprintf('%0.2f',
        scatter(i))),'FontSize',12);
end
figure; plot(sigma,scatter);title('Scatter vs Sigma'),xlabel('Sigma');ylabel('Total
    Scatter');
set(gca,'xtick', [0.02:0.02:0.12]);
[~,idx]=sort(scatter);
```

The first three values of σ show very tight clusters in the eigenspace. After $\sigma > 0.08$, however, we see that the two outer rings become mislabeled.



The scatter plotted below shows a drastic increase in scatter with $\sigma = 0.1$ and $\sigma = 0.12$.

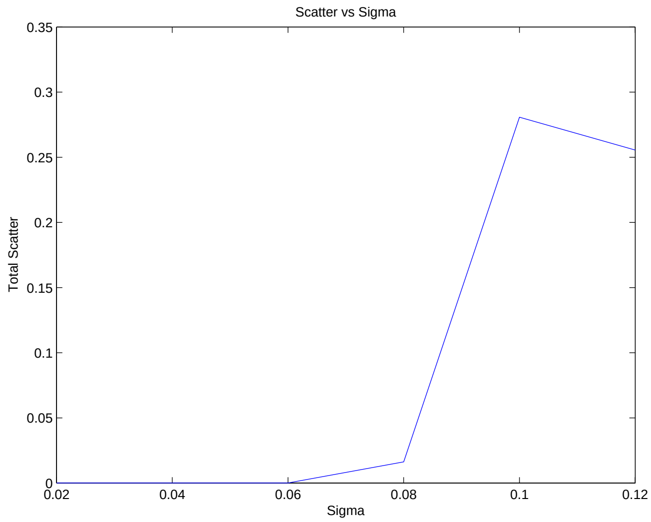


Figure 4: Total Scatter is sufficiently small for σ between 0.02 and 0.08