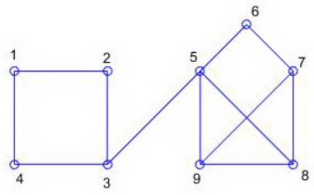
Lab Session 11

MA 453 : Matrix Computations Lab 2023 R. Alam

Many practical applications deal with large amounts of data. One of the standard problems in data science is to group the data in clusters containing similar items. In general, creating an algorithm which allows us to group data into clusters is a difficult task. There are several widely used algorithms which allow clustering of data by similarity. All these algorithms have their own advantages and disadvantages.

In this experiment, we will use the spectral clustering algorithm based on Fiedler vector. We will consider a graph representing an individual's profile on a social network. This friendship network can be represented as a graph with nodes being friends and edges showing the friendship connection between the individuals. The goal of clustering is to separate this graph into components which are tightly connected within themselves and have fewer connections to the outside.

The method we explore is based on a theorem by M. Fiedler, who has shown that for a connected graph (meaning that there is a path between any two vertices) the eigenvector corresponding to the second smallest eigenvalue of the Laplacian of the graph allows the graph to be split into two maximally intraconnected components. Consider the simple graph given below.



1. Create a code for the adjacency matrix of this graph into Matlab and save it in the matrix `AdjMatrix`.

   Find the row sums of the matrix `AdjMatrix` (use the sum function) and save it as `RowSums`.

2. Compute the Laplacian of the graph using the following command:

   ```
   LaplaceGraph = diag(RowSums)-AdjMatrix;
   ```

   Observe that the `diag(RowSums)` command creates a diagonal matrix with the vector `RowSums` on the main diagonal and zeros everywhere else. Observe also that the matrix `LaplaceGraph` is a symmetric positive semi-definite singular matrix. To check that it is singular, multiply the matrix `LaplaceGraph` with the column vector $[1, 1, 1, 1, 1, 1, 1, 1, 1]^\top$.

3. Find the eigenvalues and eigenvectors of the matrix `LaplaceGraph` using the `eig` function:

   ```
   [V,D]=eig(LaplaceGraph);
   ```

   Here the matrix `V` contains all the right eigenvectors of the matrix `LaplaceGraph` and the matrix `D` is a diagonal matrix with eigenvalues of the matrix `LaplaceGraph` on the

main diagonal. Since the Laplacian is a symmetric matrix, its eigenvalues are real. To simplify our task, let us sort the eigenvalues from smallest to largest. We will need to sort the eigenvectors along with them. Here is how to do that.

```
% sort eigenvalues and eigenvectors
[d,ind] = sort(diag(D));
D = D(ind,ind);
V = V(:,ind);
```

4. One of the eigenvalues of the matrix `LaplaceGraph` is zero (this is due to the fact that the matrix LaplaceGraph is singular) and all other eigenvalues of positive. Check your computed eigenvalues. Since we have sorted the eigenvalues, the smallest positive eigenvalue of the matrix `LaplaceGraph` is `D(2,2)` and `V(:, 2)` is the corresponding eigenvector. We normalize the eigenvector so that its first entry is positive.

```
V2 = V(:, 2);
% make sure the first entry is positive
if V2(1) < 0
V2 = -V2;
end
```

5. Separate the elements of the eigenvector V2 onto positive and negative:

```
pos=[];
neg=[];
  for j=1:9
    if V2(j)>0
      pos=[pos,j];
    else
     neg=[neg,j];
    end;
  end;
```

If everything is done correctly, the arrays `pos` and `neg` will contain correspondingly 1, 2, 3, 4 and 5, 6, 7, 8, 9 which would be an intuitive clustering choice.

6. Now apply the technique described above to a larger graph when the results cannot be visualized so easily. Choose a large graph (with vertices $N > 300$) representing a network (e.g., facebook friends) and call its adjacency matrix `Social`. The (i, j)-th element is equal to 1 if the i-th and j-th friends are also friends with each other and is equal to 0 otherwise. You may store the adjacency matrix in `social.mat`.

```
%% Load the data
clear;
load(social.mat,Social);
spy(Social);
```

The command `spy(Social)` will plot the sparsity pattern of the adjacency matrix `Social` on a figure, putting blue dots for non-zero elements of the matrix. Your task is to cluster this set of data into two maximally intraconnected groups.

7. Now, repeat the procedure we performed on a smaller matrix. Define the matrices `DiagSocial` and `LaplaceSocial` which are correspondingly the vector of row sums of the matrix `Social` and the Laplacian of this matrix. Compute the eigenvalues and the eigenvectors:

   ```
   [V,D]=eig(LaplaceSocial);
   ```

   Sort the eigenvalues from smallest to largest and normalize the eigenvector `V2` corresponding to the second smallest eigenvalue so that its first entry is positive.

8. Suppose that the social network has `N` friends (vertices). Permute the adjacency matrix `Social` using the permutation generated by the lists `pos, neg`:

   ```
   order=[pos,neg];
   [m,n]=size(Social);
   iden=eye(m,m);
    for j=1:N;
       for k=1:N;
         P(j,k)=iden(order(j),k);
       end;
    end;
   SocialOrdered=P*Social*P'
   ```

   Alternatively, a one line command that performs the above task is given by

   ```
   SocialOrdered = Social(order, order);
   ```

   What does the command `P*Social*P'` do to the rows and columns of the matrix `Social`?

9. Plot the matrix `SocialOrdered` using the function `spy` again. If you did everything correctly, you will see the matrix separating into two denser connected subsets.

10. What if we want more clusters? There are two ways to proceed. First, we can explore the eigenvector `V3` corresponding to the third smallest eigenvalue. We will group the nodes in the four clusters: the ones which have positive components in both eigenvectors `V2, V3 (++ group),` the ones which have a positive component in `V2` and a negative component in `V3(+- group),` the ones which have a negative component in `V2` and a positive component in `V3 (-+ group)`, and, finally, the ones with negative components in both vectors `(-- group)`. This generates 4 distinct groups and can be accomplished by the code below:

    ```
    %% Cluster in 4 groups
     V3=V(:,3);
        if V3(1) < 0
          V3 = -V3;
        end
     pp=[];
     pn=[];
     np=[];
    ```

```
nn=[];
  for j=1:m
    if (V2(j)>0)
      if (V3(j)>0)
        pp=[pp,j];
      else
        pn=[pn,j];
      end;
    else
      if (V3(j)>0)
        np=[np,j];
      else
         nn=[nn,j];
      end;
    end;
  end;

  order=[pp,pn,np,nn];
  iden=eye(m,m);
     for j=1:m;
        for k=1:m;
           P(j,k)=iden(order(j),k);
        end;
     end;

    SocialOrdered=P*Social*P'
    figure;
    spy(SocialOrdered)
```

11. An alternative way to obtain more clusters is to use the Fiedler vector procedure iteratively, meaning that we will apply it again to the clusters obtained in the previous step. This can be accomplished by running the following code:

```
%% Second order of Fiedler
SocialPos=Social(pos,pos);
SocialNeg=Social(neg,neg);
rowsumpos=sum(SocialPos,2);

DiagSocialPos=diag(rowsumpos);
LaplaceSocialPos=DiagSocialPos-SocialPos;
[VPos,DPos]=eig(LaplaceSocialPos);
% sort the eigenvalues from smallest to largest and intercahnge
% eigenvectors accordingly
V2Pos=VPos(:,2);
[mpos,npos]=size(V2Pos);
posp=[];
posn=[];
```

```
for j=1:mpos
    if (V2Pos(j)>0)
        posp=[posp,pos(j)];
    else
        posn=[posn,pos(j)];
    end;
end;

rowsumneg=sum(SocialNeg,2);
DiagSocialNeg=diag(rowsumneg);
LaplaceSocialNeg=DiagSocialNeg-SocialNeg;
[VNeg,DNeg]=eig(LaplaceSocialNeg);
  % sort the eigenvalues from smallest to largest and intercahnge
  % eigenvectors accordingly
V2Neg=VNeg(:,2);
[mneg,nneg]=size(V2Neg);
negp=[];
negn=[];

for j=1:mneg
    if (V2Neg(j)>0)
      negp=[negp,neg(j)];
    else
        negn=[negn,neg(j)];
    end;
end;

ordergen=[posp,posn,negp,negn];
iden=eye(m,m);
for j=1:m;
    for k=1:m;
       P(j,k)=iden(ordergen(j),k);
  end;
end;
SocialOrderedGen=P*Social*P'
figure;
spy(SocialOrderedGen)
```

The densely connected components obtained by the last two methods might be different. In the context of a social network, the dense clusters may represent groups of people who have common connections in real life.

*** End ***