CS 8735: Report for assignment 4

Chanmann Lim

November 12, 2015

The Matlab code for all experiments is in the **Appendix** section.

Problem 1. In this task, we are going carry out spectral clustering on synthetic Circle.dat dataset. The first step in spectral clustering is to construct sparse graph by considering each data-point as a vertex of the Graph G(V,E) then connect two vertices that have the squared Euclidean distance smaller than $\epsilon = 1.5$. Creating an edge e_{ij} when

$$||x_i - x_j||^2 < \epsilon \quad \forall i, j \tag{1}$$

Then using this sparse graph to generate n-by-n W matrix where n is the size of the dataset and $w(i,j)$, the element at row i^{th} and column j^{th} is defined as

$$w(i,j) = e^{\frac{-||x_i - x_j||^2}{\sigma^2}} \quad \text{if } \mathrm{e_{ij}} \text{ exists and 0 otherwise} \tag{2}$$

where $\sigma^2 = 2$, then we compute diagonal D matrix where each diagonal element Dii is the significance of each vertex:

$$D_{ii} = \sum_{j \in V} w(i,j) \tag{3}$$

Next, we define the graph Laplacian matrix $L = DW$ and perform transformation on L to get normalized graph Laplacian matrix $\tilde{L} = D^{-1/2} L D^{-1/2}$ finally we can carry out eigen analysis on the normalized graph Laplacian matrix to obtain the eigenvalues and eigenvectors for our clustering task.

a) The smallest five eigenvalues we got are [0, 0.0002, 0.0024, 0.0083, 0.0158].

b) In minimizing the normalized cut spectral clustering pushes the smallest eigenvalue to zero hence it is no longer be appropriate for clustering task and we have to use the eigenvector z_1 that corresponding to the first non-zero eigenvalue to compute $y_1 = D^{-1/2} z_1$ then assign x_i to cluster C1 if $y_i < median(y)$ otherwise assign x_i to cluster C2.

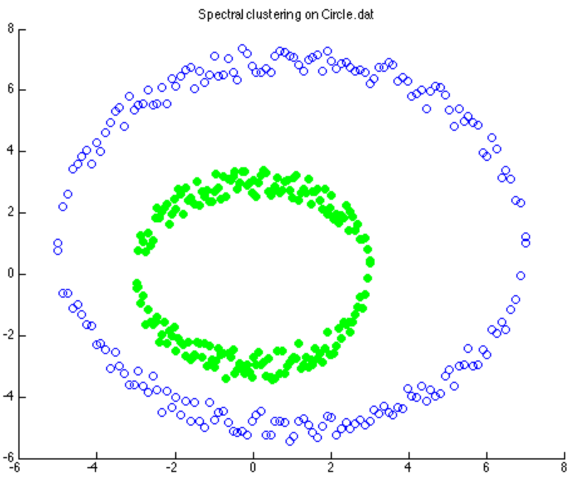

Figure 1: Plot of cluster assignment

Problem 2. In language modeling training, given four commands and a list of eight words we are going carry out latent semantic analysis by employing singular value decomposition (SVD) to transform the W (word by document) matrix into a smooth representation or concept of document namely "scaled document vectors" then we are going to merge a new test document and project its scaled document vector by mean of "fold-in" method and finally to compute the Euclidean distance between the new test command and the existing commands then rank them accordingly.

a) $W_{8\times4}$ matrix is constructed such that its element $W_{ij} = (1-\epsilon_i)\dfrac{C_{ij}}{n_j}$ where C_{ij} is the numbers of $word_i$ occurred in $document_j$, n_j is the numbers of words in $document_j$ and ϵ_i is the normalized entropy of $word_i$ in the training set.

$$\epsilon_i = \frac{-\sum_{j=1}^{N}\frac{C_{ij}}{t_i}\log\frac{C_{ij}}{t_i}}{-\sum_{j=1}^{N}\frac{1}{N}\log\frac{1}{N}} \tag{4}$$

$$= -\frac{1}{\log N}\sum_{j=1}^{N}\frac{C_{ij}}{t_i}\log\frac{C_{ij}}{t_i} \tag{5}$$

and $t_i = \sum_{j=1}^{N} C_{ij}$ with log being log base 2 when computing ϵ_i. We obtain:

$$W = \begin{bmatrix} 0.0519 & 0.0519 & 0.0415 & 0 \\ 0.0519 & 0.0519 & 0.0415 & 0 \\ 0 & 0 & 0 & 0 \\ 0.1250 & 0 & 0.1000 & 0 \\ 0 & 0.2500 & 0 & 0 \\ 0 & 0 & 0.1000 & 0.1667 \\ 0 & 0 & 0 & 0.3333 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

b) Next we decompose W by carry out SVD decomposition $W = U_R S_R V_R^T$ and we obtain:

$$U_R = \begin{bmatrix} 0.0200 & -0.2450 & 0.2798 & 0.1085 \\ 0.0200 & -0.2450 & 0.2798 & 0.1085 \\ 0.0000 & 0.0000 & -0.0000 & 0.0000 \\ 0.0449 & -0.1263 & 0.8121 & 0.2857 \\ 0.0066 & -0.9280 & -0.2760 & -0.0484 \\ 0.4768 & -0.0313 & 0.2636 & -0.8380 \\ 0.8774 & 0.0416 & -0.1955 & 0.4362 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

$$S_R = \begin{bmatrix} 0.3759 & 0 & 0 & 0 \\ 0 & 0.2633 & 0 & 0 \\ 0 & 0 & 0.1903 & 0 \\ 0 & 0 & 0 & 0.0662 \end{bmatrix} \tag{8}$$

$$V_R = \begin{bmatrix} 0.0204 & -0.1565 & 0.6862 & 0.7101 \\ 0.0099 & -0.9776 & -0.2100 & -0.0128 \\ 0.1432 & -0.1371 & 0.6875 & -0.6986 \\ 0.9894 & 0.0329 & -0.1116 & 0.0866 \end{bmatrix} \tag{9}$$

c) By keeping the eigenvectors corresponding to the largest tow eigenvalues and computing the scaled document vectors of the four documents with $\bar{V} = S_R V_R^T$ we obtain dimension reduction in document smooth representation.

$$\bar{V_2} = \begin{bmatrix} 0.0077 & 0.0037 & 0.0538 & 0.3719 \\ -0.0412 & -0.2574 & -0.0361 & 0.0087 \end{bmatrix} \tag{10}$$

d) For the new test document we use "fold-in" method to compute $\bar{v_2}(5) = U_{8\times 2}^T \tilde{d_5}$ where $\tilde{d_5} = (1-\epsilon_i)\dfrac{C_{i5}}{n_5}$ and we obtain:

$$\bar{v_2}(5) = \begin{bmatrix} 0.0606 \\ -0.0166 \end{bmatrix} \tag{11}$$

e) Finally we compute the Euclidean distance between the test command (d-5) and the training commands using their scaled document vector.

	d-1	d-2	d-3	d-4
d-5	0.0584	0.2474	0.0206	0.3123

Hence the closest neighbors of d-5 rank from d-3, d-1, d-2 and d-4 sequentially.

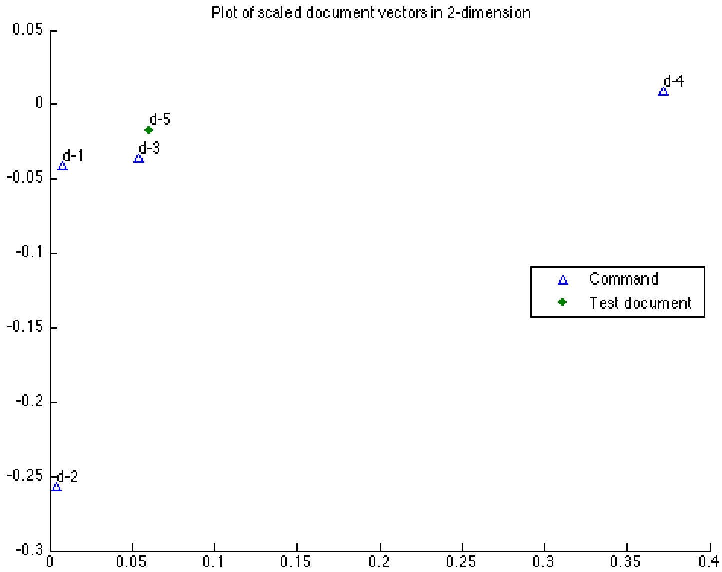

Figure 2: Plot of scaled document vector in 2-dimension

Problem 3. Given a Euclidean distance matrix of five data samples, we are going to use classical multidimensional scaling (MDS) algorithm to estimate the 2-D coordinates of the five data-points. In MDS we begin by squaring the elements of the Euclidean distance matrix then perform double centering on the squared distance matrix with $J = \mathbf{I}_n - \dfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$ to obtain $B_\Delta = -\dfrac{1}{2}JD^{(2)}J$ then we carry out eigen decomposition of $B_\Delta = V\Lambda V^T$. We finally choose the two largest positive eigenvalues and their corresponding eigenvectors to form approximation for the five data samples by computing $\hat{X} = V_+\Lambda_+^{\frac{1}{2}}$.

a) The result of eigenvalue matrix Λ_+, eigenvector matrix V_+ and the estimated coordinates $\hat{X}$ are:

$$\Lambda_+ = \begin{bmatrix} 4.0000 & 0 \\ 0 & 2.8000 \end{bmatrix} \tag{12}$$

$$V_+ = \begin{bmatrix} 0.5000 & -0.4781 \\ 0.5000 & 0.1195 \\ -0.0000 & 0.7171 \\ -0.5000 & 0.1195 \\ -0.5000 & -0.4781 \end{bmatrix} \tag{13}$$

$$\hat{X} = \begin{bmatrix} 1.0000 & -0.8000 \\ 1.0000 & 0.2000 \\ -0.0000 & 1.2000 \\ -1.0000 & 0.2000 \\ -1.0000 & -0.8000 \end{bmatrix} \tag{14}$$

b) To verify that the estimated data coordinates are centered we compute the mean of $\hat{X}$ and it in fact yield the values very close to zeros due to numerical computation precision bias. We also recover the Euclidean distance matrix from the projected data samples $\hat{X}$ and confirm the exact match with the original distance matrix.

$$\hat{D} = \begin{bmatrix} 0 & 1.0000 & 2.2361 & 2.2361 & 2.0000 \\ 1.0000 & 0 & 1.4142 & 2.0000 & 2.2361 \\ 2.2361 & 1.4142 & 0 & 1.4142 & 2.2361 \\ 2.2361 & 2.0000 & 1.4142 & 0 & 1.0000 \\ 2.0000 & 2.2361 & 2.2361 & 1.0000 & 0 \end{bmatrix} \tag{15}$$

Appendix:

assignment_4.m

```
clc;
clear all;
close all;

problem_1
problem_2
problem_3
problem_4
```

problem_1.m

```
% load data
X = load('Circles.dat');

% design parameters
sigma2 = 2;
e = 1.5;

% construct sparse graph and W
[~, N] = size(X);
% strength of edges(e_ij)
W = zeros(N,N);
for i=1:N
    for j=i:N
        d = (X(:,i)-X(:,j))' * (X(:,i)-X(:,j));
        if d < e
            W(i,j) = exp(-d/sigma2);
        end
    end
end
W = W + triu(W, 1)';
% significance of the vertice
D = diag( sum(W) );
% graph Laplacian matrix
L = D - W;
% normalized L
L_tilde = D^-0.5 * L * D^-0.5;

% eigen decomposition
[V, A] = eig(L_tilde);
a = sort( diag(A) );

% compute y
y = D^-0.5 * V(:,2);
I = y < median(y);
C1 = X(:,I==1);
C2 = X(:,I==0);

% plot
figure, hold on;
scatter(C1(1,:), C1(2,:));
scatter(C2(1,:), C2(2,:), 'filled', 'g');
title('Spectral clustering on Circle.dat');

% print smallest 5 eigenvalues
display('The smallest 5 eigenvalues =');
display(' ');
fprintf('\t %0.4f \n', a(1:5));
```

problem_2.m

```
% data
docs = {'what is the time' 'what is the day' 'what time is the meeting' 'cancel the meeting'};
words = {'what' 'is' 'the' 'time' 'day' 'meeting' 'cancel' 'when'};

M = length(words);
N = length(docs);

% numbers of word_i occurred in doc_j
```

```
C = zeros(M,N);
% normalized entropy
e = zeros(M,1);
for i=1:M
    for j=1:N
        C(i,j) = sum( strcmp(strsplit(docs{j}), words{i}) );
    end
    p = C(i,:)/sum(C(i,:));
    I = p.*log2(p);
    % cancel the effect of non-existing words
    e(i) = -1/log2(N) * sum(I(~isnan(I)));
end

% W matrix (words by documents)
W = zeros(M, N);
for j=1:N
    nj = length( strsplit(docs{j}) );
    for i=1:M
        W(i,j) =  (1-e(i)) * C(i,j)/nj;
    end
end
display(W);

% SVD decomposition
[U,S,V] = svd(W);

% scaled document vectors
SR = diag( diag(S) );
R = length(SR);
U = U(:,1:R);

display(U);
display(SR);
display(V);
% numbers of dimensions to keep
k = 2;
V_bar = SR*V';
V2 = V_bar(1:k,:);
display(V2);

% test with new document
d5 = 'when is the meeting';
w_d5 = strsplit(d5);
nj = length( w_d5 );
C_i5 = zeros(M,1);
for i=1:M
    C_i5(i) = sum( strcmp(w_d5, words{i}) );
end
w_i5 = (1-e).*(C_i5/nj);
v_bar_5 = U(:,1:k)' * w_i5;
display(v_bar_5);

% compute distance from d5
D = zeros(1,N);
for j=1:N
    distance = V2(:,j) - v_bar_5;
    D(j) = sqrt(distance' * distance);
end
display(D);

% plot
figure; hold on;
scatter(V2(1,:), V2(2,:), '^');
labels = num2str((1:N)','d-%d');
text(V2(1,:), V2(2,:), labels, 'horizontal','left', 'vertical','bottom');
scatter(v_bar_5(1), v_bar_5(2), 'filled', 'd');
text(v_bar_5(1), v_bar_5(2), 'd-5', 'horizontal','left', 'vertical','bottom');
hold off;
title('Plot of scaled document vectors in 2-dimension');
legend('Command', 'Test document', 'Location', 'east');
```

problem_3.m

```
% distance matrix
```

```
D = [0 1 sqrt(5) sqrt(5) 2
     1 0 sqrt(2) 2 sqrt(5)
     sqrt(5) sqrt(2) 0 sqrt(2) sqrt(5)
     sqrt(5) 2 sqrt(2) 0 1
     2 sqrt(5) sqrt(5) 1 0];

[n,~] = size(D);
% Classical multidimensional scaling algorithm (MDS)
D2 = D.^2;
% double centering
J = eye(n) - (1/n * ones(n,n));
B_delta = -0.5 * J*D2*J;
% eigen analysis
[V,A] = eig(B_delta);
a = diag(A);
[I,~] = find(a>0);
a = a(I);
% numbers of coordinate to keep
k = 2;
[~,J] = sort(a, 'descend');
A_plus = diag( a(J(1:k)) );
V_plus = V(:,I(J(1:k)));
display(A_plus);
display(V_plus);
% compute coordinate matrix
X_hat = V_plus * sqrt(A_plus);
display(X_hat);

% verify X_hat is centered
z = mean(X_hat);
display(z);
% recover distance matrix from estimated coordinate
D_hat = zeros(n,n);
for i=1:n
    for j=1:n
        d = X_hat(i,:) - X_hat(j,:);
        D_hat(i,j) = sqrt(d*d');
    end
end
display(D_hat);
```

problem_4.m

```
% class labels
Y = [1 1 1 1 1 2 2 2 2 2 2 2];
% clusters labels
a = [2 2 2 2 2 1 1 1 1 1 1 1];
b = [2 2 1 1 1 2 2 2 2 2 2 2];
c = [1 1 1 2 2 2 2 2 2 2 1 1];

% compute P_ij, Pi and Pj
[Pa_ij, Pa_j, Pi, I] = Prob(Y, a);
[Pb_ij, Pb_j] = Prob(Y, b);
[Pc_ij, Pc_j] = Prob(Y, c);

% compute mutual information (MI)
Qa = Pa_ij./(Pi*ones(1,2))./(Pa_j*ones(1,2))'; Qa(Qa==0) = 1;
Qb = Pb_ij./(Pi*ones(1,2))./(Pb_j*ones(1,2))'; Qb(Qb==0) = 1;
Qc = Pc_ij./(Pi*ones(1,2))./(Pc_j*ones(1,2))'; Qc(Qc==0) = 1;

MI_a = sum( sum(Pa_ij .* log2(Qa)) );
MI_b = sum( sum(Pb_ij .* log2(Qb)) );
MI_c = sum( sum(Pc_ij .* log2(Qc)) );

% compute normalized mutual information (NMI)
NMI_a = MI_a / sqrt(entropy(Pi)*entropy(Pa_j));
NMI_b = MI_b / sqrt(entropy(Pi)*entropy(Pb_j));
NMI_c = MI_c / sqrt(entropy(Pi)*entropy(Pc_j));
display(NMI_a);
display(NMI_b);
display(NMI_c);
```

problem_1.m

```
% load data
X = load('Circles.dat');

% design parameters
sigma2 = 2;
e = 1.5;

% construct sparse graph and W
[~, N] = size(X);
% strength of edges(e_ij)
W = zeros(N,N);
for i=1:N
    for j=i:N
        d = (X(:,i)-X(:,j))' * (X(:,i)-X(:,j));
        if d < e
            W(i,j) = exp(-d/sigma2);
        end
    end
end
W = W + triu(W, 1)';
% significance of the vertice
D = diag( sum(W) );
% graph Laplacian matrix
L = D - W;
% normalized L
L_tilde = D^-0.5 * L * D^-0.5;

% eigen decomposition
[V, A] = eig(L_tilde);
a = sort( diag(A) );

% compute y
y = D^-0.5 * V(:,2);
I = y < median(y);
C1 = X(:,I==1);
C2 = X(:,I==0);

% plot
figure, hold on;
scatter(C1(1,:), C1(2,:));
scatter(C2(1,:), C2(2,:), 'filled', 'g');
title('Spectral clustering on Circle.dat');

% print smallest 5 eigenvalues
display('The smallest 5 eigenvalues =');
display(' ');
fprintf('\t %0.4f \n', a(1:5));
```

Prob.m

```
function [ Pij, Pj, Pi, I ] = Prob( Y, C )
% Prob compute probabilities that
%       x is assigned to cluster j and it belongs to class i = Pij
%       x belongs to class i = Pi
%       x is assigned to cluster j = Pj

n = length(Y);
I = unique(Y);
l = length(I);

Ci = zeros(l, n);
Cj = zeros(l, n);
for i=1:l
    Ci(i,:) = Y == I(i);
    Cj(i,:) = C == I(i);
end
Pi = sum(Ci,2)/n;
Pj = sum(Cj,2)/n;

Pij = zeros(l,l);
for i=1:l
    for j=1:l
```

```
        Pij(i,j) = sum(Ci(i,:) .* Cj(j,:))/n;
    end
end
```

entropy.m

```
function [ H ] = entropy( P )
% ENTROPY compute entropy of distribution P

P = P(:);
H = sum( -P .* log2(P) );
```