Homework 1

Tianyi Li
PID: tianyili

February 8, 2018

Q 1.a. (4 points) Two vectors x and y have zero mean (given $\bar{x} = \bar{y} = 0$). The cosine measure and correlation between them will be equal.

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|} \tag{1}$$

$$\begin{aligned} corr(\vec{x}, \vec{y}) &= \frac{cov(\vec{x}, \vec{y})}{std(\vec{x}) \cdot std(\vec{y})} \\ &= \frac{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^2} \cdot \sqrt{\frac{1}{n-1} \cdot \frac{1}{n-1} \cdot \sum_{i=1}^{n} (y_i - \bar{y})^2}} \\ &= \frac{\sum_{i=1}^{n} (x_i \cdot y_i)}{|\vec{x}| \cdot |\vec{y}|} \\ &= cos(\vec{x}, \vec{y}) \end{aligned} \tag{2}$$

Q 1.b. the mathematical relationship between cosine similarity and Euclidean distance when each data object vector has an L2 length (magnitude) of 1 (given $|\vec{x}| = |\vec{y}| = 1$):

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|} = \vec{x} \cdot \vec{y} \tag{3}$$

$$\begin{aligned} dist(\vec{x}, \vec{y}) &= |\vec{x} - \vec{y}| \\ &= \sqrt{(\vec{x} - \vec{y})^2} \\ &= \sqrt{(|\vec{x}|)^2 + (|\vec{y}|)^2 - 2 \cdot \vec{x} \cdot \vec{y}} \\ &= \sqrt{2 - 2 \cdot cos(\vec{x}, \vec{y})} \end{aligned} \tag{4}$$

Such being the case, $dist(\vec{x}, \vec{y})^2 + 2 \cdot cos(\vec{x}, \vec{y}) = 2$

Q 2. To prevent a bias towards longer documents, we first compute the *relative word frequency* $rf(w, A)$ of word w in an article set A:

$$rf(w, A) = \frac{f(w, A)}{max\{f(w_i, A) : w_i \in A\}} \tag{5}$$

The domain weight $C_{w_i,D}$, in order to quantify the ability of word w_i in representing a targeted domain, should consider the word frequency in articles related to all events in a domain D, and compare its frequency in all articles.

Word frequency in articles related to all events in a domain D:

$$f_D(w_i) = \sum_{A \in A_D} rf(w_i, A) \tag{6}$$

Word frequency in all articles:

$$f_A(w_i) = \sum_{A \in A} rf(w_i, A) \tag{7}$$

$$\begin{aligned} C_{w_i,D} &= \frac{f_D(w_i)}{f_A(w_i)} \\ &= \frac{\sum_{A \in A_D} rf(w_i, A)}{\sum_{A \in A} rf(w_i, A)} \\ &= \frac{\sum_{A \in A_D} \frac{f(w_i,A)}{max\{f(w_j,A):w_j \in A\}}}{\sum_{A \in A} \frac{f(w_i,A)}{max\{f(w_j,A):w_j \in A\}}} \end{aligned} \tag{8}$$

The event weight $E_{w_i,e}$ quantifies the ability of word w_i in distinguishing event e from other events in the same domain.

Word frequency in articles related to an event e:

$$f_e(w_i) = \sum_{A \in A_e} rf(w_i, A) \tag{9}$$

Word frequency in all articles in the same domain D as event e:

$$f_{D_e}(w_i) = \sum_{A \in \{A_D : e \in D\}} rf(w_i, A) \tag{10}$$

$$\begin{aligned} E_{w_i,e} &= \frac{f_e(w_i)}{f_{D_e}(w_i)} \\ &= \frac{\sum_{A \in A_e} rf(w_i, A)}{\sum_{A \in \{A_D : e \in D\}} rf(w_i, A)} \\ &= \frac{\sum_{A \in A_e} \frac{f(w_i,A)}{max\{f(w_j,A):w_j \in A\}}}{\sum_{A \in \{A_D : e \in D\}} \frac{f(w_i,A)}{max\{f(w_j,A):w_j \in A\}}} \end{aligned} \tag{11}$$

Q 3. Set up a Cartesian coordinate plane on the ground with John being the origin. John's position is always $(0, 0)$, and Mike's position can be represented by $(x, y), x, y \in [0, 1]$

Manhattan Distance between Mike and John:

$$\begin{aligned} distance &= |x| + |y| \\ min &= 1 \\ max &= \sqrt{2} \end{aligned} \tag{12}$$

Euclidean Distance between Mike and John:

$$\begin{aligned} distance =& \sqrt{x^2 + y^2} = 1 \\ min =& 1 \\ max =& 1 \end{aligned} \tag{13}$$

Chebyshev Distance between Mike and John:

$$\begin{aligned} distance =& max(|x|, |y|) \\ min =& \frac{\sqrt{2}}{2} \\ max =& 1 \end{aligned} \tag{14}$$

Q 4.

1. Calculate Euclidean distance. Line:4 and 6. Formula for Euclidean distance:

$$\begin{aligned} EuclideanDist(\vec{x}, \vec{y}) =& |\vec{x} - \vec{y}| \\ =& \sqrt{(\vec{x} - \vec{y})^2} \\ =& \sqrt{\sum_i^n (x_i - y_i)^2} \end{aligned} \tag{15}$$

Neighborhood graphs:

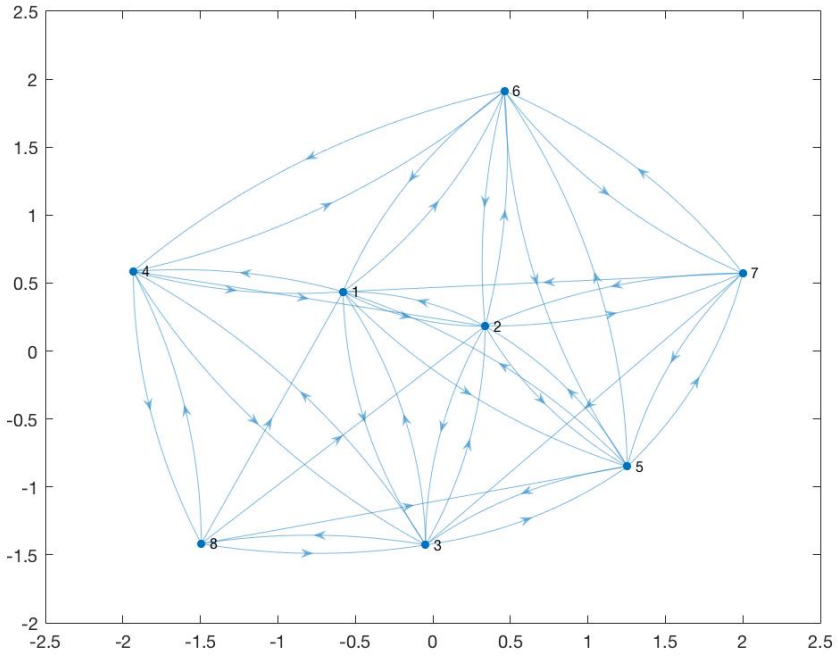

Figure 3: Q4.1a: Neighborhood graph connecting points xi and xj if xi is one of the 5 nearest neighbors of xj (8 data points)

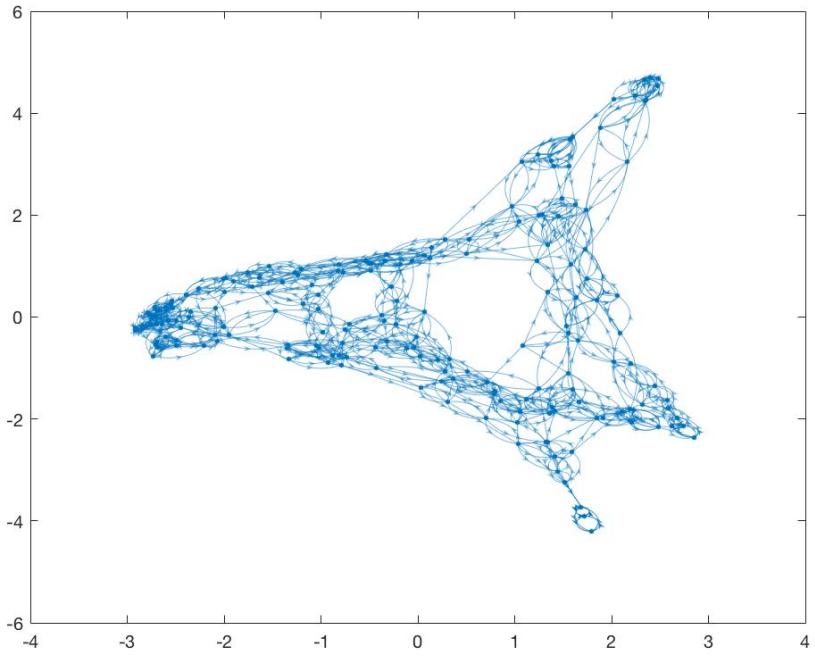

Figure 1: Q4.1a: Neighborhood graph connecting points xi and xj if xi is one of the 5 nearest neighbors of xj (200 data points)

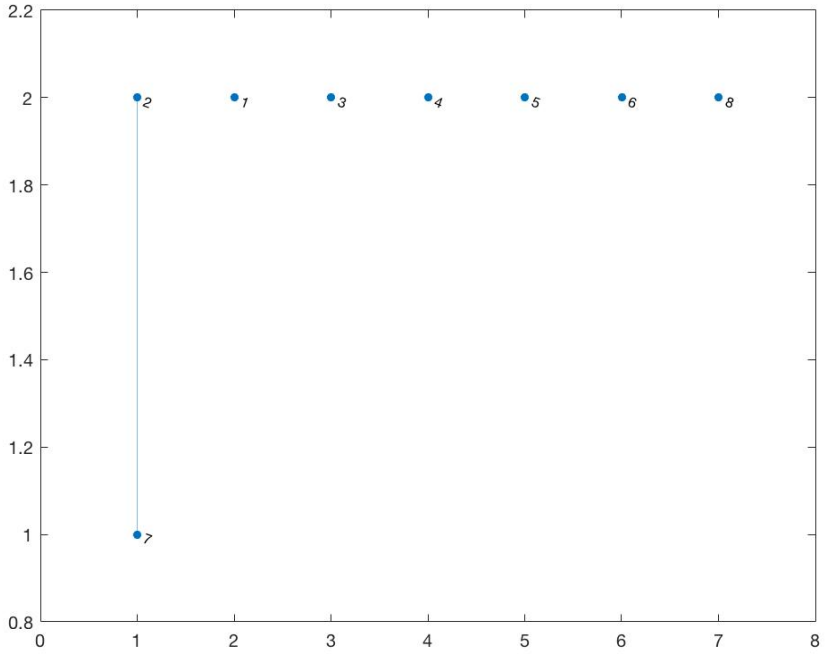

Figure 4: Q4.1b: Neighborhood graph connecting points xi and xj if their distance is less than 6 (8 data points)

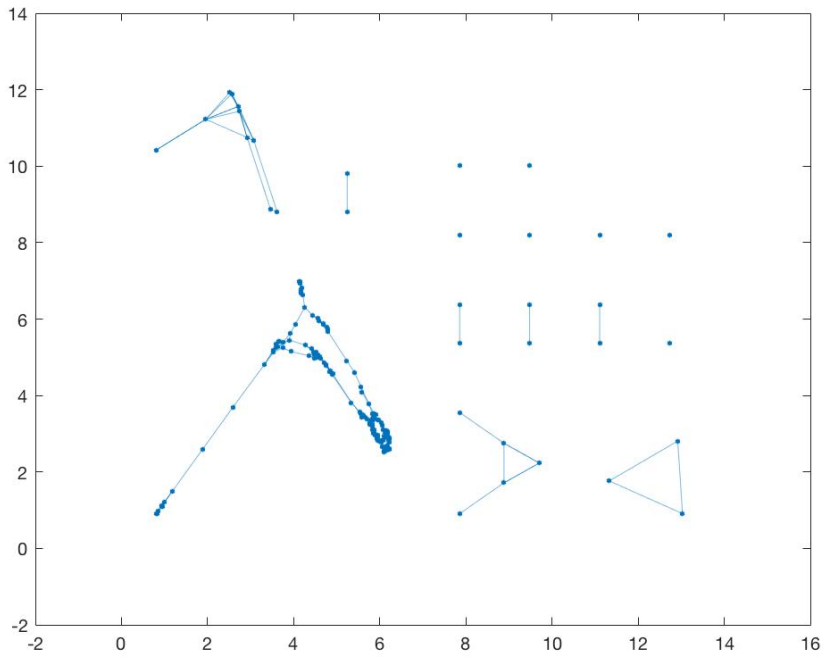

Figure 2: Q4.1b: Neighborhood graph connecting points xi and xj if their distance is less than 6 (200 data points)

geodestic(): line 52-63
8 by 8 distance matrices:

0	1	1	1	1	1	2	2
1	0	1	2	1	1	1	2
1	1	0	1	1	2	2	1
1	1	1	0	2	1	2	1
1	1	1	2	0	1	1	2
1	1	2	1	1	0	1	2
1	1	1	2	1	1	0	2
1	1	1	1	1	2	2	0

Table 1: Q4.2: Distance Matrix of 5 nearest neighbors graph

0	Inf	Inf	Inf	Inf	Inf	Inf	Inf
Inf	0	Inf	Inf	Inf	Inf	1	Inf
Inf	Inf	0	Inf	Inf	Inf	Inf	Inf
Inf	Inf	Inf	0	Inf	Inf	Inf	Inf
Inf	Inf	Inf	Inf	0	Inf	Inf	Inf
Inf	Inf	Inf	Inf	Inf	0	Inf	Inf
Inf	1	Inf	Inf	Inf	Inf	0	Inf
Inf	Inf	Inf	Inf	Inf	Inf	Inf	0

Table 2: Q4.2: Distance Matrix of 5 nearest neighbors graph

Code for Q4:

```
load data.mat
% the Euclidean distances between each pair of the data points (xi, xj)
%(a 200ÃŰ200 distance matrix)
EuclideanDistMat = dist(X_data', X_data);
%distances among the first 8 data points (a 8ÃŰ8 distance matrix)
First8 = EuclideanDistMat(1:8,1:8);

%construct neighborhood graphs
[adjMatless6_200, adjMat5nn_200, distGraphLess6_200, distGraph5nn_200] =
    ↪ plotGraph(EuclideanDistMat);
[adjMatless6_8, adjMat5nn_8, distGraphLess6_8, distGraph5nn_8] = plotGraph(
    ↪ First8);

% part 2: geodist
distMatLess6 = geodesic(distGraphLess6_8);
distMat5nn = geodesic(distGraph5nn_8);

function [adjMatless6, adjMat5nn, distGraphLess6, distGraph5nn] = plotGraph
    ↪ ( graphMat )
    size = length(graphMat);
    sorted = zeros(200,5);
    for i=1:size
        tmp = sort(graphMat(i,:));
        % minimal is oneself, counting from 2 to 6
        sorted(i,:) = tmp(2:6);
    end
    adjMatless6 = zeros(size);
    adjMat5nn = zeros(size);

    for i=1:size
        adjMat5nn(i,:) = ismember(graphMat(i,:),sorted(i,:));
        adjMat5nn(i,i) = 0;
        for j=1:size
```

```
            if graphMat(i,j) < 6
                if i == j
                    adjMatless6(i,j) = 0;
                else
                    adjMatless6(i,j) = 1;
                end
            end
        end
    end

    distGraphLess6 = graph(adjMatless6);
    distGraph5nn = digraph(adjMat5nn);
    f1 = figure;
    f2 = figure;
    figure(f1);
    plot(distGraphLess6);
    figure(f2);
    plot(distGraph5nn);

end

function geoDist = geodesic(distGraph)
    size = numnodes(distGraph);
    geoDist = zeros(size);
    for i=1:size
        for j=1:size
            geoDist(i,j) = length(shortestpath(distGraph,i,j,'Method','
                ↪ positive')) - 1;
            if geoDist(i,j) == -1
                geoDist(i,j) = Inf;
            end
        end
    end
end
```

Q 5.

(1) Boxplot of the first 100 rows of data:

Code:

```
thickness = csvread('thick.csv',1,1);
figure();
boxplot(thickness(1:100,:));
```

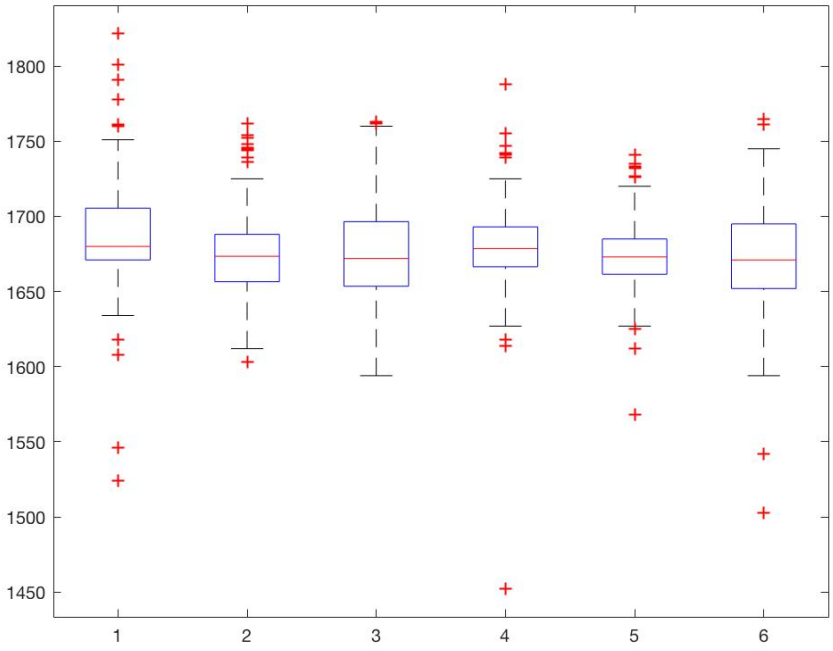

Figure 5: Q5.1: Boxplot of the first 100 rows of data

(2) Because the distribution of data values are not symmetrical. Only if there are same number of data values that are bigger than the lowest 25% of data and smaller than the highest 50%, and that bigger than the lowest 50% of data and smaller than the highest 75%.

Q 6.

1. Import: line 2

2. Suppose the whole data set $D = \{d\}$, where d is each data point. Formula for centering is $d' = d - \mu_D$, then scaling is $d'' = \frac{d'}{\sigma_D}$. Combined formula is

$$d'' = \frac{d - \mu_D}{\sigma_d}$$

. We apply this formula to each attribute (column) of the food data. (line 4-13)

3. The absolute value of normalized data points (z-score) represents the distance between the raw data value and the population mean in units of the standard deviation. z-score is negative when the raw data value is below the mean, positive when above.

4. Formula for correlation matrix in z-score (z_i^x is normalized x_i for each data point x_i,

$\vec{z_x}$ is normalized vector (column) $\vec{x}$) (implemented in line 15-24):

$$
\begin{aligned}
corr(\vec{x}, \vec{y}) &= \frac{cov(\vec{x}, \vec{y})}{std(\vec{x}) \cdot std(\vec{y})} \\
&= \frac{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \mu_x) \cdot (y_i - \mu_y)}{\sigma_{\vec{x}} \cdot \sigma_{\vec{y}}} \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^{n} (\frac{(x_i - \mu_x)}{\sigma_{\vec{x}}} \cdot \frac{(y_i - \mu_y)}{\sigma_{\vec{y}}}) \\
&= \frac{1}{n-1} \cdot \sum_{i=1}^{n} (z_i^x \cdot z_i^y) \\
&= \frac{1}{n-1} \cdot \vec{z_x} \cdot \vec{z_y}
\end{aligned}
\tag{16}
$$

5. line 27

6. Plot the percentage of variance captured by the individual components in decreasing order: line 29-43

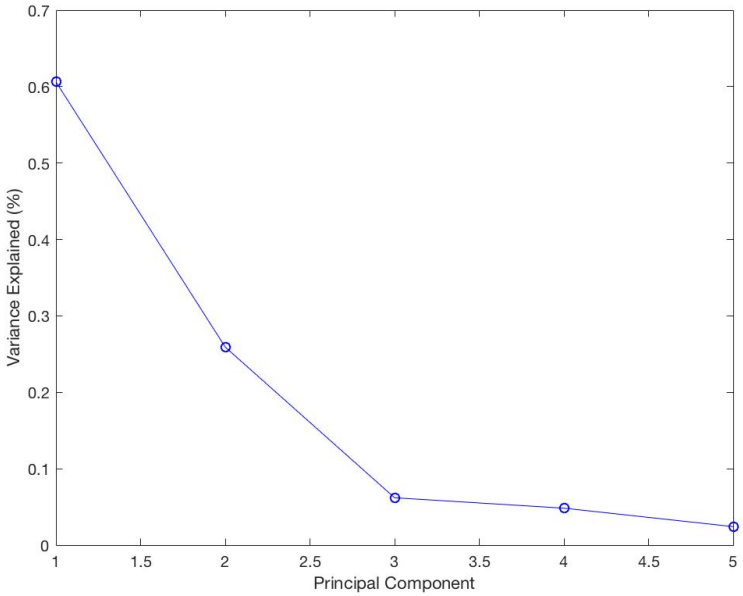

Figure 6: scree plot of the eigenvalues energies

7. I would use 2 component because the inflection point happens at the third component, and the first two components combined capture a reasonably high (close to 90%) percentage of variance. These steps are PCA algorithm, retaining $(\lambda_1^2 + \lambda_2^2)/(\lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2 + \lambda_5^2)10.8671/11.0364 = 98.47\%$ energy. ($\lambda_{1,2,3,4,5}$ are eigenvalues sorted in descending order)

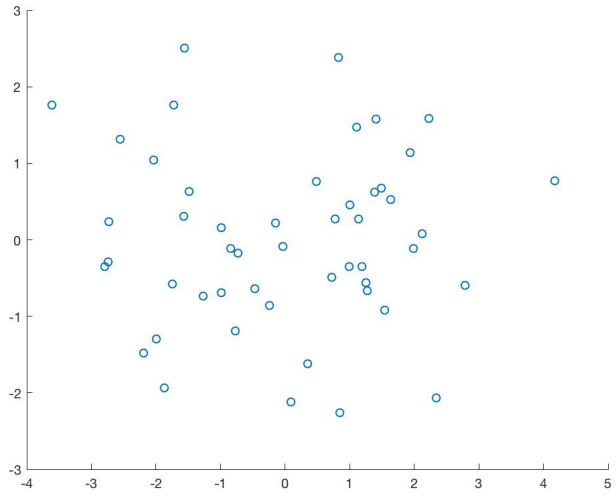

Figure 7: Projecting matrix to 2-D project with PCA

9. PCA is not recommended for high dimensional data, because it will be too expensive to compute too many eigenvalues and eigenvectors for each dimension.

10. SVD

11.2-D Scatter Plot for SVD

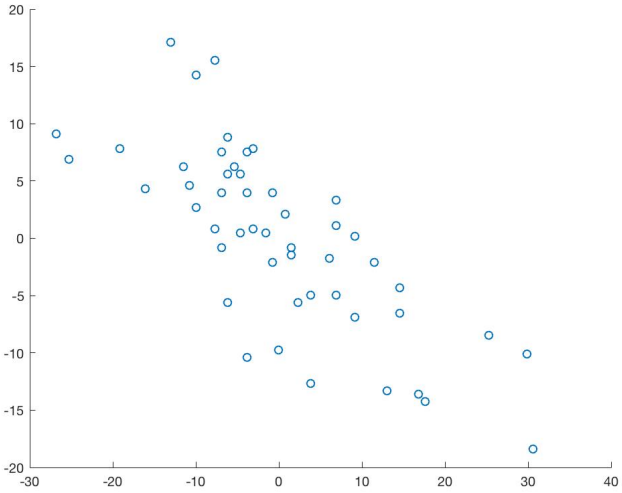

Figure 8: Projecting matrix to 2-D project with SVD

12. Yes.

the PCA algorithm computes the eigenvalues and eigenvectors of covariance matrix of the data, which is equivalently the correlation matrix of the standard normalized data. The covariance matrix of a data matrix X is $X \cdot X^T$ and is symmetric, the matrix is diagonalizable,

and the eigenvectors can be normalized such that they are orthonormal: $X \cdot X^T = W \cdot D \cdot W^T$, where W is the matrix of eigenvectors and D is the diagonal matrix of eigenvalues.

On the other hand, applying SVD to the data matrix X, we have $X = S \cdot \Sigma \cdot U^T$. The covariance matrix from this decomposition, given that $U^T \cdot U = I$

$$X \cdot X^T = S \cdot \Sigma \cdot U^T \cdot (S \cdot \Sigma \cdot U^T)^T = S \cdot \Sigma \cdot U^T \cdot U \cdot \Sigma \cdot S^T = S \cdot \Sigma^2 S^T$$

```
% 1. (2 points) Import: Import the data file food.csv in Matlab.
rawfood = csvread('food.csv',1,1);

% 2. Processing: Centering
food = rawfood;
meanFood = mean(food);
meanFood = repmat(meanFood,length(food(:,1)),1);
centeredFood = food-meanFood;

% 2. Processing: Scaling centered data
stdFood = std(food);
stdFood = repmat(stdFood,length(food(:,1)),1);
scaledFood = centeredFood./stdFood;

%4. customized correlation matrix
numDim = length(food(1,:));
numObs = length(food(:,1));
corrMat = zeros(numDim);
n = length(food(:,1));
for i=1:numDim
    for j=1:numDim
        corrMat(i,j) = (1/(numObs-1)) * scaledFood(:,i)'*scaledFood(:,j);
    end
end

% 5. eigenvalues and eigen vectors
[V,D] = eigs(corrMat);

% 6. sort eigen values from largest to smalelst
[c, ind]=sort(diag(D),'descend');
D2=diag(c);

% 6. accordingly update the order of the eigenvectors in matrix
V2 = V(:,ind);

% Plot the percentage of variance captured by the individual components
% in decreasing order
allDim = length(D2);
```

```
cumG = c/sum(c);
figure();
plot(cumG,'b-o');
xlabel('Principal Component')
ylabel('Variance Explained (%)')

numComponents = 2;
projMatPCA = V2(:,1:numComponents);
figure();
scatter(scaledFood*projMatPCA(:,1),scaledFood*projMatPCA(:,2));

% 10. SVD
[U,S,V] = svd(scaledFood);
% sort s matrix from largest to smalelst
[c, ind]=sort(diag(S),'descend');
S2=diag(c);
% project
projMatSVD = S2(1:numComponents,:)';
figure()
scatter(scaledFood*projMatSVD(:,1), scaledFood*projMatSVD(:, 2));
```