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

6. Hello

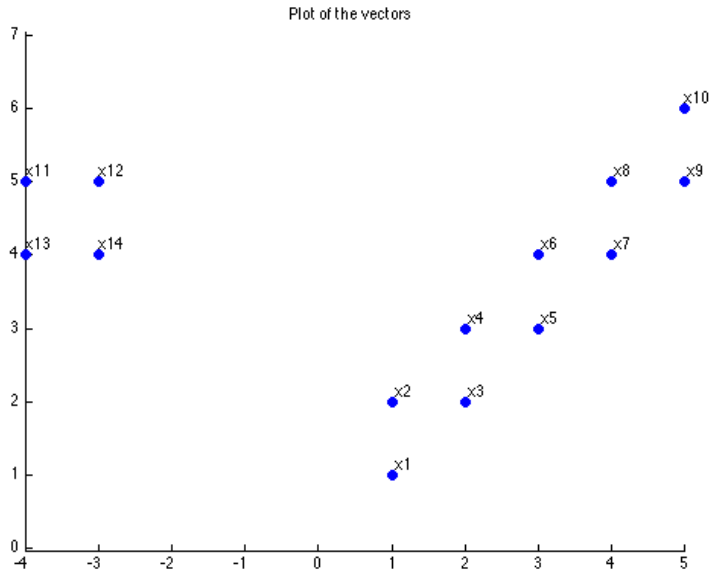

Figure 1: Plot of vectors

Appendix:

assignment_2.m

```
clc;
clear all;
close all;

problem_6
problem_7
```

problem_6.m

```
% load data
X = [[1  1]; [1  2]; [2  2]; [2  3]; [3  3]; [3  4]; [4  4]
    [4  5]; [5  5]; [5  6]; [-4  5]; [-3  5]; [-4  4]; [-3  4]];

% clustering parameters
q = 14;
Theta = sqrt(2);

% run clustering algorithm
% a.
seq = 1:14;
A = X(seq,:);
A_bsas = bsas(A, Theta, q);
A_mbsas = mbsas(A, Theta, q);
display(seq);
print_cluster(A_bsas, seq);
print_cluster(A_mbsas, seq);

% b.
seq = [1 10 2 3 4 11 12 5 6 7 13 8 14 9];
B = X(seq,:);
B_bsas = bsas(B, Theta, q);
B_mbsas = mbsas(B, Theta, q);
display(seq);
print_cluster(B_bsas, seq);
print_cluster(B_mbsas, seq);

% c.
seq = [1 10 5 2 3 11 12 4 6 7 13 14 8 9];
C = X(seq,:);
C_bsas = bsas(C, Theta, q);
C_mbsas = mbsas(C, Theta, q);
display(seq);
print_cluster(C_bsas, seq);
print_cluster(C_mbsas, seq);

% d. Plot the vectors
figure;
scatter(X(:,1), X(:,2), 'filled');
axis equal;
title('Plot of the vectors');

labels = num2str((1:size(X,1))','x%d');
text(X(:,1), X(:,2), labels, 'horizontal','left', 'vertical','bottom');
```

bsas.m

```
function [ C ] = bsas( X, Theta, q )
% bsas - Basic Sequential Clustering Algorithm
%
%       X       : sequential dataset
%       Theta   : clustering threshold
%       q       : maximum number of allowed clusters

[N,~] = size(X);

% numbers of cluster
m = 1;
C = {1};
% clusters representative
```

```
R = X(C{m},:);
for i=2:N
    [d, ck] = min_distance(X(i,:), R);
    if d > Theta && m < q
        m = m + 1;
        C{m} = i;
        R(m,:) = X(i,:);
    else
        nc = length(C{ck});
        C{ck} = [C{ck} i];
        % update representative
        % mc_new = (nc*mc+x) / (nc+1)
        R(ck,:) = (nc*R(ck,:) + X(i,:)) / (nc+1);
    end
end
```

mbsas.m

```
function [ C ] = mbsas( X, Theta, q )
% mbsas - Modified Basic Sequential Clustering Algorithm
%
%       X       : sequential dataset
%       Theta   : clustering threshold
%       q       : maximum number of allowed clusters

[N,~] = size(X);

% numbers of cluster
m = 1;
C = {1};
% clusters representative
R = X(C{m},:);

% class determination
for i=2:N
    [d,~] = min_distance(X(i,:), R);
    if d > Theta && m < q
        m = m + 1;
        C{m} = i;
        R(m,:) = X(i,:);
    end
end

% pattern classification
classified = cell2mat(C);
for i=1:N
   if ~nnz(classified == i)
        [~, ck] = min_distance(X(i,:), R);
        nc = length(C{ck});
        C{ck} = [C{ck} i];
        % update representative
        % mc_new = (nc*mc+x) / (nc+1)
        R(ck,:) = (nc*R(ck,:) + X(i,:)) / (nc+1);
   end
end
```

min_distance.m

```
function [ d, k ] = min_distance( x, C )
% distance - Compute min distance from x to all clusters = min d(x, C)
%   C - clusters representative

    [m,~] = size(C);
    D = zeros(1, m);
    for i=1:m
        D(i) = d2(x, C(i,:));
    end
    [d, k] = min(D);
end

% Euclidean distance between x and y
function [ d ] = d2(x, y)
    x_tilte = x - y;
```

```
    d = sqrt(x_tilte * x_tilte');
end
```

print_cluster.m

```
function print_cluster(C, seq)
% print_cluster - print cluster values
%   C   - set of clusters (cell array)
%   seq - sequence of x

disp([inputname(1) ' ='']);
disp(' ');
m = length(C);
for i=1:m
    fprintf(['     {' sprintf(' x%d ', seq(C{i})) '}']);
end
disp(' ');
disp(' ');
```